Capítulo 1

HTML Y XHTML: PÁGINAS WEB CON CONTENIDO ESTÁTICO

En 1990 aparece un nuevo servicio vía Web llamado HTTP (Hypertext Transfer Protocol, Protocolo de Transferencia de Hipertexto), con la finalidad de poder intercambiar información haciendo uso de texto e imágenes, a través de la entonces novedosa[1] red de Internet, y así complementar al resto de servicios que ya se encontraban en funcionamiento: FTP, servicio utilizado para la transferencia de archivos, TELNET, servicio utilizado para el control remoto de equipos, o SMTP, servicio utilizado para intercambiar correos electrónicos.

Con la finalidad de estructurar la información transferida a través del servicio HTTP, la Organización Europea para la Investigación Nuclear (CERN) en 1991 pública un novedoso lenguaje de etiquetas que pretende dar la posibilidad de maquetar de una manera muy sencilla los datos recibidos por un cliente HTTP llamado lenguaje HTML (*HyperText Markup Language*, Lenguaje de Marcas de Hipertexto). De esta forma, haciendo uso de un intérprete para estas etiquetas HTML en el cliente HTTP (por ejemplo Internet Explorer o Mozilla Firefox), es posible presentar la información recibida desde un servidor HTTP de manera organizada. Según esto, la información almacenada en el servidor HTTP deja de ser texto plano con imágenes, para pasar a ser una información etiquetada sin ningún criterio establecido, más que el del diseñador que se haya encargado de ello, el cual decide el aspecto resultante de la maquetación que será visualizado en la aplicación cliente tras su interpretación.

Figura 1.1. Pasos seguidos en la comunicación HTTP entre un cliente y un servidor

1 Aunque en 1972 ya se hizo la primera demostración pública de lo que posteriormente sería Internet, ARPANET, hasta la implementación del protocolo HTTP y surgimiento del mundo WWW, Internet era una desconocida.

Con la finalidad de establecer unas recomendaciones sobre el número y tipo de etiquetas a utilizar surge el W3C (*World Wide Web Consortium*). Este es un consorcio internacional fundado y dirigido por Tim Berners-Lee, padre del protocolo HTTP y del lenguaje de etiquetado HTML, que se encarga de producir estándares en base a unas recomendaciones, especificaciones y directrices previas, asegurando un acceso universal con una multimedia más atractiva.

Otro aspecto importante que es tratado por el W3C y que merece mención a parte, es lo concerniente al diseño Web pensando en aquellos usuarios con discapacidades y edad avanzada. En este sentido el W3C ha creado el estándar WCAG (Web Content Accesibility Guidelines, Pautas de Accesibilidad al Contenido en la Web), que actualmente se encuentra en su versión 2.0.

1. CONTENIDOS DEL CAPÍTULO

Los contenidos tratados en este capítulo podrían resumirse de la siguiente forma:

— En primer lugar se informará del software que puede utilizarse en el equipo cliente HTTP para probar todos los aspectos, ejemplos y ejercicios presentados a lo largo del capítulo.

— Características generales del lenguaje de etiquetas HTML. Se clasificarán las distintas etiquetas existentes, y se presentarán las diferencias más importantes entre los documentos HTML y XHTML.

— A continuación, en base a la clasificación anterior, se presentan los distintos tipos de etiqueta junto con ejemplos de su utilización, y ejercicios prácticos que puedan resultar de útiles.

2. SOFTWARE DE DESARROLLO

Una característica importante que se va a mantener durante los tres primeros capítulos, es que nos vamos a limitar de momento al ámbito del cliente HTTP, dejando para los capítulos posteriores todo lo referente a la instalación y configuración de un servidor HTTP. Esto implica, que el único software que es necesario tener instalado en el equipo para poder practicar y comprobar todo lo que aparece en el capítulo es un navegador Web o cliente HTTP, como pueden ser "Internet Explorer", "Mozilla Firefox" u "Opera", ya que los documentos HTML/XHTML pueden generarse a partir de cualquier editor de textos.

Con la finalidad de facilitar la edición de los documentos HTML/XHTML es posible instalar también algún entorno de desarrollo como son "*Microsoft FrontPage*", "*Dreamweaver*" en entornos Microsoft Windows, o "*Nvu*", "*Quanta Plus*", "*kate*" en entornos GNU/Linux. Señalar que los dos programas indicados para entornos de Microsoft Windows son los más afamados, pero presentan el inconveniente que son de pago, pudiendo encontrar alternativas gratuitas a estos en sitios Web como *softonic* (*http://www.softonic.com*).

Por tanto, serán necesarios dos pasos para poder probar los documentos HTML/XHTML que generemos a lo largo del capítulo:

1. Mediante la ayuda de un editor HTML/XHTML generamos el documento.
2. Visualizamos el resultado anterior mediante la ayuda de un navegador o cliente Web.

3. CARACTERÍSTICAS DEL LENGUAJE DE ETIQUETAS HTML. DOCUMENTOS HTML Y XHTML

Desde que en 1991 el CERN propusiera una primera versión del lenguaje HTML, hasta la actual versión 4.01 se han sucedido diversas versiones con sus correspondientes modificaciones y mejoras, dando lugar a un estándar que presenta las siguientes características más destacables:

A) **HTML es un lenguaje de marcado o etiquetado de la información**. Mediante el uso de etiquetas es posible marcar e identificar los distintos elementos de información que forman parte de un documento con la finalidad de personalizar la presentación de la información en un cliente Web o HTTP (por ejemplo, Internet Explorer o Mozilla Firefox). Es decir, estas etiquetas HTML son utilizadas por el cliente Web, en base a las cuales, decide el aspecto con que será mostrada la información al usuario. Por esta razón a todo cliente Web se le denomina también intérprete HTML, ya que su función primordial es interpretar el significado de las etiquetas, y actuar en consecuencia. El número de etiquetas HTML disponibles es un conjunto finito, totalmente definido por el estándar[2], entre las que podemos encontrar para etiquetar a los títulos, a los párrafos de texto, a las imágenes o a los objetos Flash (sonido, imágenes animadas, películas, etc.) entre otros.

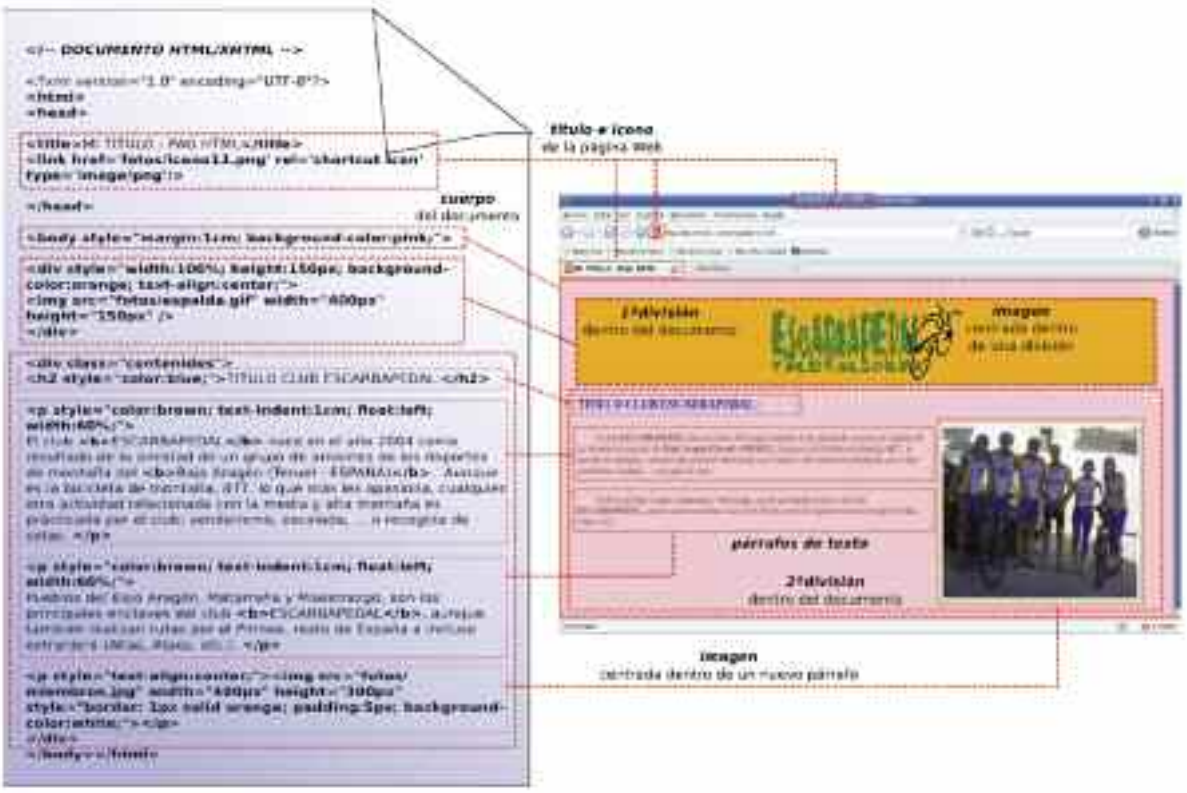


Figura 1.2. Un documento HTML/XHTML está compuesto por información etiquetada, cuyas etiquetas son utilizadas por el cliente Web o intérprete HTML a la hora de la presentación de dicha información

2 La lista de todas las etiquetas HTML disponibles según el estándar podemos consultarla en Web del W3C, "http://www.w3.org/TR/html4/index/elements.html".

Como veremos en un apartado posterior, las etiquetas HTML pueden ser simples o dobles, pero ambas se caracterizan por estar encerradas entre los signos mayor y menor "<>", diferenciándose entre sí por el nombre de la etiqueta, y el conjunto de atributos disponibles:

```
<nombre-etiqueta-HTML [atributo1=valor1] [atributo2=valor2] ...> Texto / Imagenes </nombre-etiqueta-HTML>
```

Para identificar que el contenido de un fichero ha sido marcado haciendo uso de etiquetas HTML, y así poder diferenciarlo del resto de ficheros, se hace uso de una extensión reservada a tal efecto: ".html" o ".htm"[3].

B) **HTML es un un lenguaje interpretado sin compilación previa.** A diferencia de la edición de programas en lenguajes de programación como C o Java, HTML no requiere de una compilación previa a la interpretación y ejecución del documento etiquetado. Es decir, un cliente Web tras recibir un documento HTML no realiza ningún tipo de análisis léxico, sintáctico o semántico sobre las etiquetas HTML utilizadas en él, limitándose a leer su contenido de manera secuencial, de tal forma que cada vez que encuentra una etiqueta, comprueba si esta definida en el estándar, y en caso afirmativo, es interpretada y ejecutada, mostrando su efecto sobre la información que etiqueta en la pantalla de nuestro navegador.

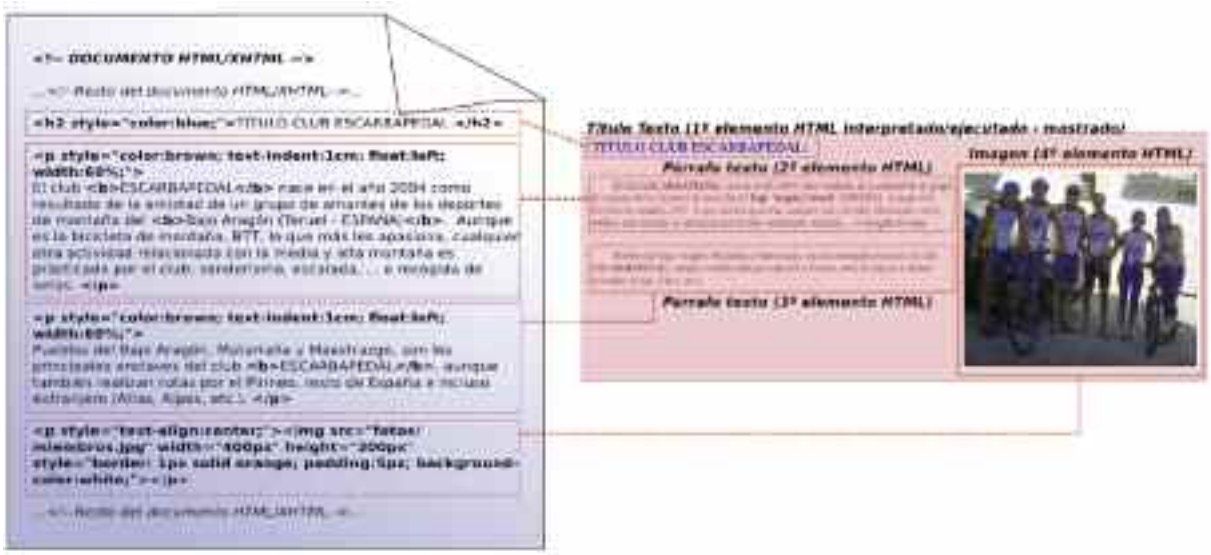


Figura 1.3. El cliente o navegador Web lee el documento HTML e interpreta secuencialmente las etiquetas HTML que se va encontrando, mostrando el resultado en pantalla

C) **HTML es un lenguaje anárquico**. Relacionada con la característica anterior, un intérprete HTML omite cualquier error léxico, sintáctico o semántico producido en la edición del documento HTML, pasando desapercibo tanto para el editor del documento como para el que visualiza en el navegador Web su resultado. Al contrario que los lenguajes de programación que se verán en capítulos posteriores, el lenguaje de etiquetas HTML es muy permisivo, y es posible cometer multitud de incongruencias sintácticas, y visualizarse perfectamente el resultado de su maquetación en un cliente Web. Por ejemplo, es posible abrir una etiqueta doble HTML, y olvidarse de la cierre, y no mostrarnos ningún mensaje erróneo. Esto implica que si al leer el intérprete el

3 "*.html" y "*.htm" son extensiones equivalentes. Mientras que ".html" es la extensión por defecto, ".htm" surgió en los inicios del estándar HTML a raíz de que ciertos sistemas operativos como MS-Dos o Windows 3.11, hoy obsoletos, no manejaban extensiones con más de tres caracteres.

documento localiza una etiqueta errónea, desde el punto de vista semántico, al no corresponderse con ninguna etiqueta especificada en el estándar, o sintácticamente, al no respetar las reglas de declaración de la etiqueta en cuestión, el navegador los omitirá, mostrando el resultado en base a dichas omisiones. Es decir, si una etiqueta no se reconoce es omitida en la presentación de la información, sin influir en el resto de etiquetas, y si por el contrario, si es reconocida, pero no esta editada de una manera correcta, se asumirán los errores, y se mostrará el resultado sin que esto influya. Por ejemplo, en la siguiente figura (figura 1.4.), se muestra un documento HTML que contiene dos errores:

1) Se ha declarado una etiqueta HTML llamada "**y**" que no esta definida en el entandar W3C HTML. El efecto de este error semántico es nulo, al ser omitida por parte del intérprete HTML.

2) Se ha declarado incorrectamente una etiqueta "img" definida en el estándar, al no cerrarse esta con un signo ">". A pesar de este error sintáctico, el intérprete asume que deseamos visualizar una imagen, y la muestra.

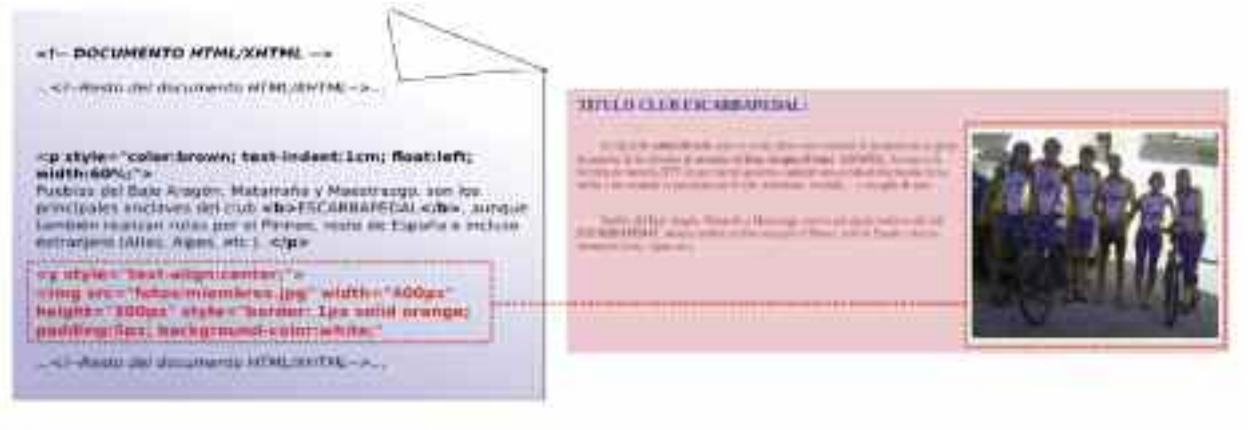


Figura 1.4. Las errores sintácticos y semánticos en el etiquetado son omitidos por el intérprete HTML

Con la finalidad de restringir esta permisividad en los errores y hacer de HTML un lenguaje más estricto, surge la recomendación W3C XHTML (*e**X**tensible **H**yper**t**ext **M**arkup **L**anguage*, Lenguaje Extensible de Marcado de Hipertexto), resultado de aglutinar los estándares HTML y XML, convirtiéndose en lo que se supone será el sustituto natural de HTML. Para distinguir un documento HTML de un XHTML se hace uso de una extensión diferente. Mientras que los documentos HTML hacen uso de extensiones "*.html" o "*.htm", los documentos XHTML utilizan "*.xhtml". A lo largo del capítulo hablaremos indistintamente de HTML y XHTML, al ser esté último una recomendación casi idéntica a la versión 4.01 del estándar HTML en la que nos vamos a centrar. En el caso de que existan diferencias entre ambos, serán indicadas explícitamente, y resumidas al final del capítulo.

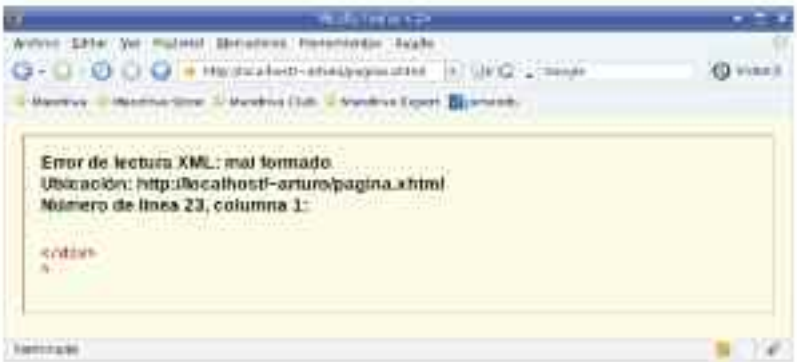


Figura 1.5. El intérprete de documentos XHTML es mucho más estricto, no dejando pasar errores sintácticos o semánticos, informando de ello por pantalla

D) **HTML no es un lenguaje de programación**. En HTML no es posible definir variables y asignarles un valor en tiempo de ejecución, o introducir instrucciones de control del flujo en la interpretación de las etiquetas HTML, tal como sucede en cualquier lenguaje de programación. HTML no es un lenguaje de programación, simplemente es un lenguaje de marcado compuesto por un conjunto de etiquetas preestablecidas por el W3C, que serán interpretadas por el cliente Web en el mismo orden en que son declaradas en el documento HTML, y que irán confeccionando el aspecto final del documento visualizado en el cliente Web.

E) **HTML etiqueta información estática**. La información presentada por un cliente Web a través de un documento HTML se caracteriza por ser estática. Esto significa que el resultado de la presentación de sus contenidos en el navegador es siempre igual independientemente de la hora y fecha en que se acceda al documento, de quien sea el que acceda a él, o desde donde se acceda. Como veremos en capítulos posteriores para conseguir un dinamismo en los contenidos visualizados en función de quien, cuando y desde donde se acceda será necesario una interacción con el servidor Web que nos ha suministrado el documento Web. Un ejemplo de este tipo de sitios Web dinámicos sería un servicio WebMail, donde tras una autenticación[4] los contenidos que se muestran, son dependientes del usuario que accede.

En relación con este aspecto también cabría resaltar los documentos DHTML (*Dinamic HTML*, HTML Dinámico), donde sin necesidad de interactuar con el servidor Web, es posible ofrecer un cierto dinamismo en los contenidos mostrados por el navegador. Para ello, combinando el lenguaje de etiquetado HTML con hojas de estilo CSS, y un lenguaje interpretado como JavaScript, nos permiten controlar tanto la presentación de los contenidos como su comportamiento. En el caso de hacer uso del lenguaje de programación JavaScript, como veremos en el capitulo X es posible crear y ejecutar programas en el cliente Web, permitiéndonos controlar el comportamiento del sitio Web.

F) **HTML es indiferente a las mayúsculas, saltos de línea y espaciados en blanco**. Al contrario que otros lenguajes, HTML no distingue entre mayúsculas y minúsculas en el etiquetado, permitiéndonos escribir el nombre de las etiquetas HTML y de sus atributos de la manera que se nos antoje. Además, HTML omite los saltos de línea que provoquemos al editar los documentos HTML, tanto en la edición de sus etiquetas como en su contenido, lo que nos permitirá hacer documentos más legibles. En el caso de querer forzar uno o varios saltos de línea en el contenido del documento, será necesario utilizar una etiqueta que lo indique explícitamente, "**
", haciendo uso de ella tantas veces como saltos requiramos. De igual forma, con espaciados en blanco superiores a un espacio, serán omitidos. Si queremos forzar dentro del contenido un número de espacios mayor, será necesario utilizar el carácter especial " **" tantas veces como espacios requiramos. Según lo anterior, los dos trozos de código HTML semánticamente son equivalentes:

```
<nombre-etiqueta-HTML [atributo1=valor1] [atributo2=valor2] ...> Texto e Imagenes </nombre-etiqueta-HTML>
```

```
<NOMBRE-etiqueta-HTML
    [ATributo1=Valor1]
    [atRIBUto2=VALOR2]
    ...>
```

4 En la actualidad son muy populares los servicios Web, donde tras una autenticación, los contenidos se personalizan: correo Web (WebMail), álbumes de fotos Web, gestores de contenidos, etc.

```
        Texto e
        Imagenes
    </nombre-etiqueta-HTML>
```

En relación a esta discriminación entre mayúsculas y minúsculas ejercida por el intérprete HTML, es importante señalar que no es extensible al intérprete XHTML. Este último es sensible a ello, y en caso de utilizar mayúsculas, las etiquetas HTML no serán reconocidas por él. Según esto, con la finalidad de que las documentos HTML sean compatibles con ambos intérpretes, todas las etiquetas que se utilicen en este capítulo se escribirán en minúsculas.

G) Con la finalidad de poder documentar los documentos HTML y favorecer su posterior mantenimiento, HTML nos permite introducir comentarios. Para que estos comentarios sean omitidos por el intérprete de etiquetas HTML y no afecten en la presentación de los contenidos del documento en el navegador Web se hace uso de un etiquetado especial que advertirá de ello: "*<!— Comentarios HTML —>*".

4. ETIQUETAS HTML

Para poder crear documentos HTML/XHTML será necesario conocer en primer lugar las distintas etiquetas HTML disponibles. En el presente apartado se presentarán las etiquetas HTML más importantes del estándar HTML 4.01, y XHTML 1.1 junto con sus atributos más significativos. Estas etiquetas se diferencian de la información o contenido del documento HTML/XHTML al encontrarse acotadas por los símbolos reservados menor y mayor, "<>":

```
<nombre-etiqueta-HTML [atributo1=valor1] [atributo2=valor2] ...>
    <!—Información / Contenido Documento HTML/XHTML—>
</nombre-etiqueta-HTML>
```

En una primera clasificación, la lista de etiquetas disponibles en HTML/XHTML podría distinguirse entre etiquetas simples, dobles o múltiples:

1. Etiquetas simples: su efecto en la presentación de los contenidos en el navegador se produce en el mismo momento en que son leídas e interpretadas. Ejemplos de este tipo de etiquetas, son "**
", cuando queremos provocar un salto de linea, o "<hr>**" si queremos que se visualice en el una línea separadora. En el caso de que el documento sea XHTML sus restricciones no permiten hacer uso de etiquetas simples sin que sean cerradas. Para ello estas etiquetas deberán acabar en "/>" en lugar de un signo mayor ">". De esta forma, las etiquetas anteriores pasarían a ser: "**<hr />**" y "**
**". Las etiquetas simples más comunes son las siguientes:

ETIQUETAS SIMPLES EN HTML	ETIQUETAS SIMPLES EN XHTML	SIGNIFICADO
<**br** *[atributo1=valor1] ...*>	<**br** *[atributo1=valor1] ... />*	Salto de línea
<**hr** *[atributo1=valor1] ...*>	<**hr** *[atributo1=valor1] ... />*	Línea separadora de contenidos
<**img** *[atributo1=valor1] ...*>	<**img** *[atributo1=valor1] ... />*	Imagen
<**link** *[atributo1=valor1] ...*>	<**link** *[atributo1=valor1] ... />*	Enlace
<**meta** *[atributo1=valor1] ...*>	<**meta** *[atributo1=valor1] ... />*	Propiedades del documento
<**param** *[atributo1=valor1] ...*>	<**param** *[atributo1=valor1] ... />*	Parámetros de un objeto

2. Etiquetas dobles: están compuestas por una etiqueta de inicio y otra de final bajo el mimo nombre, *<nombre-etiqueta-HTML [atributo1=valor1] [atributo2=valor2] ...> </nombre-etiqueta-HTML>*, distinguibles por estar encerradas entre los símbolos menor y mayor, "< >" la de inicio, y menor con barra invertida (*slash*), y mayor, "</ >", la final. Entre ambas etiquetas se colocarán los contenidos y otras etiquetas HTML que queramos que se vean afectadas por estas.

```
<nombre-etiqueta-HTML [atributo1=valor1] [atributo2=valor2] ...>
    <!—Información / Contenido Documento HTML/XHTML—>
</nombre-etiqueta-HTML>
```

Los efectos que pueden provocar sobre la información encerrada puede ser muy variada como veremos en la exposición de las distintas etiquetas HTML disponibles.

3. Etiquetas múltiples: se corresponden con etiquetas simples o dobles que suelen repetirse en el etiquetado de los contenidos de los documentos HTML/XHML múltiples veces de manera seguida, supeditadas a su vez a unas etiquetas dobles de nivel superior. Un ejemplo típico sería una lista ordenada, "*** elementos de la lista ***" (Ordered List), donde los elementos de la lista, "***<li>***" (Item List), es una cantidad variable indeterminada. Por ejemplo, si quisiéramos presentar en el navegador un documento Web correspondiente a los ingredientes de una supuesta receta de cocina el resultado del etiquetado de sus contenidos sería el siguiente:

Lista Ordenada en HTML	Lista Ordenada en XHTML
<**ol** *[atributo1=valor1] ...*>	</**ol**> <**ol** *[atributo1=valor1] [atributo2=valor2]*>
<**li** *[atributo1=valor1] ...*> 100 grs. Cebolla.	<**li** *[atributo1=valor1] ...*> 100 grs. Cebolla. </**li**>
<**li** *[atributo1=valor1] ...*> 200 grs. sal.	<**li** *[atributo1=valor1] ...*> 200 grs. sal. </**li**>
...	...
<**li** *[atributo1=valor1] ...*> 1 litro agua.	<**li** *[atributo1=valor1] ...*> 1 litro agua. </**li**>
</**ol**>	

Como puede advertirse en el ejemplo anterior, al igual que sucede con las etiquetas simples, las restricciones de XHTML no permiten declarar una etiqueta HTML sin que esta sea cerrada. En el caso de tratarse de los elementos de una lista, las etiquetas pasarán a ser dobles.

En base a la función que tiene asignada cada etiqueta dentro del documento HTML/XHTML también pueden clasificarse en diferentes grupos:

— Etiquetas HTML globales. Se encargan de organizar el resto de etiquetas HTML dentro del documento, estructurando de esta forma sus contenidos.

— Etiquetas HTML encargadas de dar el aspecto deseado al texto que forma parte del documento.

— Etiquetas HTML utilizadas en la creación de listas ordenadas y desordenadas.

— Etiquetas HTML para la inserción de imágenes, videos y sonidos en el documento.

— Etiquetas HTML para la creación de tablas.

— Etiquetas HTML utilizadas en la organización de los contenidos.

— Etiquetas HTML para la creación de formularios.

Todas estas etiquetas HTML pueden tener de manera opcional acompañando al nombre de la etiqueta una serie de atributos que nos permiten personalizar su comportamiento junto con su valor asignado. Aunque algunos de los atributos pueden asociarse de manera global al conjunto de las etiquetas, como "id", "class" o "style", hay otros que son específicos de determinadas etiquetas HTML. Por ejemplo, para colocar una imagen en una página Web será necesario utilizar la etiqueta simple HTML "<img />" a la cual habrá que añadirle unos atributos para informar al intérprete HTML de al menos la ruta relativa donde se localiza la imagen en relación al documento HTML (***src="ruta"***), y en el caso de que queramos ajustar sus dimensiones, su altura y anchura (***height="tamaño"*** y ***width="tamaño"***). Suponiendo un caso práctico de una imagen llamada "foto1.jpg", localizada en una subcarpeta denominada "imagenes" dentro de la carpeta donde se encuentra el archivo de etiquetado HTML que se este editando, con un tamaño de presentación de 300 pixels de altura y 400 pixels de anchura, la etiqueta quedaría de la siguiente forma:

```
<img src="imagenes/foto1.jpg" height="300px" width="400px">
```

¡¡Advertencia!! En el estándar HTML el valor asignado a los atributos asociados a una etiqueta HTML tan sólo será necesario encerrarlos entre comillas dobles, *"valor-atributo",* o simples, *'valor-atributo'*, en el caso de que contenga espacios en blanco. En XHTML será necesario en cualquier caso.

4.1. Etiquetas HTML globales

Las etiquetas HTML globales son de utilización común en todo documento HTML/XHTML que se quiera crear. Se encargan de definir características genéricas del documento y organizar la información etiquetada dentro del documento.

ETIQUETA HTML y ATRIBUTOS	EJEMPLO
<!DOCTYPE *tipo_documento*> <**html** ***lang|xml:lang***> </html> <**head** ***lang|xml:lang***> </head> <**title** ***lang|xml:lang***> </title> <**meta** *name http-equiv content* ***lang|xml:lang*** /> <**link** *rel type href title* /> <**style** ***type media title***> </style> <**script** ***src type***> </script> <**body** ***id class style*** ***lang|xml:lang title***> </body>	<?xml version="1.0" encoding="utf-8"?> <!**DOCTYPE** HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd"> <**html**> <**head**> <**title**> Mi primer sitio Web </**title**> <**meta** name="generator" content="Quanta Plus" /> <**meta** name="author" content="Arturo Martin Romero" /> <**meta** name="organizacion" content="IES Rio Arba" /> <**meta** name="description" content="Ciclos Formativos Informatica" /> <**meta** name="keywords" lang="es" content="ESO, bachillerato, FP" /> <**meta** http-equiv="Content-Type" content="text/html; charset=utf-8" /> <link href='fotos/icono.png' rel='shortcut icon' type='image/png' /> <**style** type="text/css"**>** */* Estilos CSS */* **</style>** <**script** type="text/javascript"**>** ***/* Scripts JavaScript*/*** **</script>** </**head**> <**body** style="background:pink"> *Etiquetas HTML encargadas de maquetar el cuerpo del documento* </body> </html>

<!**DOCTYPE** *tipo_documento*>: Etiqueta encargada de definir el tipo de documento HTML/XHTML que se va a crear. Junto con la extensión del documento (**.html, *.htm, *.xhtml*), informa al navegador o cliente Web del estándar utilizado en la creación del documento, su versión, y las reglas seguidas en su elaboración. Para indicar cuales son estas reglas, junto a la declaración del tipo de documento se indicará la ruta del documento DTD elaborado por el W3C que permitirá a nuestro navegador su comprobación y validación del documento. Los tipos de documentos serían los siguientes:

1. Documento HTML 4.01 o XHTML 1.0 "**transitional**": como bien indica su nombre, es un tipo de documento transitorio. Esta pensado para ser utilizado mientras se produce la transición del HTML clásico al HTML/XHTML estricto, tanto por parte de los diseñadores de páginas Web, como por parte de los desarrolladores de los intérpretes HTML o clientes Web. Con el tiempo se supone que este tipo de documento ira sustituyéndose por el "strict".

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

2. Documento HTML 4.01 o XHTML 1.0 "**strict**": se corresponde con un tipo de documento estricto donde se intenta separar completamente por un lado los contenidos del documento y por otro la presentación de estos presentación en el cliente Web, respetando simultáneamente un conjunto de reglas sintácticas y semánticas. En el HTML clásico ambos aspectos estaban mezclados, donde al mismo tiempo que se etiquetaba un elemento del documento mediante un conjunto de atributos se controlaba su aspecto en su presentación. Mediante este tipo de documentos, por un lado encontramos los elementos que forman el contenido del documento (por ejemplo, un título, un párrafo, dos imágenes, otro parráfo, etc.), y por otro el estilo asignado a estos elementos que controlarán su aspecto visual. La forma de conseguir esta separación la veremos en el siguiente capítulo mediante la utilización de hojas de estilo CSS, haciendo que los documentos sean más modulares, compactos y más legibles de cara a su posterior mantenimiento.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Según lo anterior, determinadas etiquetas HTML y atributos que eran utilizados clásicamente en HTML y que pueden ser utilizados en el modo "**transitional**", en modo "**strict**" han sido desaprobados y no es aconsejable su utilización. Etiquetas comúnmente utilizadas como "* Contenidos HTML *" para centrar los contenidos del documento en su presentación, o "* Texto *" usada para dar un aspecto agradable al texto en su presentación, ya no son aconsejados. De igual forma, atributos como "*background*" utilizado para personalizar el fondo del cuerpo del documento, "*bgcolor*" para asignar un color de fondo a una tabla o al propio documento, o "*border*" usado para asignar un borde a una imagen, tampoco podrán ser utilizados de manera estricta. La lista de estas etiquetas y atributos desaprobados son los siguientes:

ETIQUETAS DESAPROBADAS EN XHTML STRICT

** , , <applet> </applet>, <basefont>, <dir> </dir>, <isindex>, <menu> </menu>, <s> </s>, <strike> </strike>, <u> </u>, <iframe> </iframe>, <noframes> </noframes>**

ATRIBUTOS DESAPROBADOS EN XHTML STRICT
alink, background, link, text, vlink en la etiqueta HTML <**body** ***atributos***> </**body**>
bgcolor en las etiquetas HTML <**body|table|td|th|tr** ***atributos***> </**body|table|td|th|tr**>
nowrap en las etiquetas HTML <td|th ***atributos***> </**td|th**>
clear en la etiqueta HTML <**br** ***atributos*** />
language en la etiqueta HTML <**script** ***atributos***> </**script**>
noshade en la etiqueta HTML <**hr** ***atributos*** />
start en la etiqueta HTML <**ol** ***atributos***> </**ol**>
target en las etiquetas HTML <**area|base|link** ***atributos*** /> **y** <**a|form** ***atributos***> </**a|form**>
border en las etiquetas HTML <**img** ***atributos*** /> y <**object** ***atributos***> </**object**>, permitiéndose aún en <**table** ***atributos***> </**table**>
name en las etiquetas HTML <**img** ***atributos*** /> y <**form** ***atributos***> </**form**>, permitiéndose aún en <**input|meta|**
param ***atributos*** /> y <**a|button|map|object|select|textarea** ***atributos***> </**a|button|map|object|select|textarea**>
type en las etiquetas HTML <**li|ol|ul** ***atributos***> </**li|ol|ul**>, permitiéndose aún en <**input|link|param** ***atributos*** /> y <**a|button|object|script|style** ***atributos***> </**a|button|object|script|style**>
value en la etiqueta HTML <**li**> </**li**>, permitiéndose aún en <**input|param** ***atributos*** /> y <**button|option** ***atributos***> </**button|option**>

3. Documento HTML 4.01 o XHTML 1.0 "**frameset**": equivalente al tipo "**transitional**", añadiendo a este la posibilidad de utilizar marcos o frames. Esta opción nos permite dividir la pantalla del navegador Web en diferentes zonas o marcos, asignando a cada una de ellas un documento diferente, organizando de una forma muy sencilla la presentación de los contenidos.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

Figura 1.6. Ejemplo de pagina Web dividida en marcos

Las nuevas versiones de XHML desaconsejan la utilización de estos marcos por varias razones entre las cuales podrían destacarse las siguientes:

— La página Web ya no se corresponde con un único documento HTML/XHTML, sino con tantos documentos como marcos se establezcan. Esto provoca que el tiempo invertido por el navegador Web en presentar los contenidos sea mayor.

— Origina problemas de impresión de los contenidos visualizados, ya que al corresponderse estos con diferentes documentos, el navegador Web puede no saber exactamente que contenidos imprimir.

— Presentan problemas de navegación al navegador Web cuando queremos acceder a documentos anteriores o siguientes.

— Cuando queremos guardar la referencia del contenido de uno de los frames mediante el uso de "Añadir página a marcadores", no suele funcionar correctamente.

— La facilidad de organización en la presentación de los contenidos no justifica su utilización, ya que como veremos en el capítulo siguiente, mediante el uso de páginas de estilo CSS es muy fácil conseguir resultados similares.

4. Documento XHTML versión 1.1/2: se corresponden con la evolución de la recomendación XHTML 1.0 asociada al documento de tipo "**strict**". Las nuevas versiones desaconsejan la utilización del modo "transitional" o "frameset" y centran sus esfuerzos para de manera global los sitios Web sean estrictos en su desarrollo.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/
TR/xhtml11/DTD/xhtml11.dtd">
```

Todos los tipos anteriores persiguen evitar incongruencias sintácticas y semánticas en la elaboración del documento HTML/XHTML. Sus diferencias radican en lo estrictos que son en la separación existente en el documento entre lo que son los contenidos que forman el documentos y la forma en que se presentan en el navegador Web. Es decir, mientras el tipo "transitional" permite la utilización de determinados atributos asociados a etiquetas HTML relacionados con su presentación, el tipo "strict" obliga a declarar todos los aspectos relacionados con dicha presentación en una sección del documento especialmente pensado para ello como veremos en el siguiente capítulo relacionado con las hojas de estilo CSS.

En el caso de hacer uso de una herramienta de desarrollo de sitios Web, estas etiquetas serán generadas directamente por la propia herramienta software, no siendo necesario su conocimiento.

— **<html *lang*|*xml:lang*> </html>**: etiqueta HTML doble que indican el comienzo y final del marcado del documento. Entre estas etiquetas se encuentra el resto del documento etiquetado, organizado este a su vez en un encabezado, **<head> </head>**, y un cuerpo, **<body> </body>**. Opcionalmente podemos informar al navegador del lenguaje utilizado para redactar el documento mediante el atributo "***lang***". En el caso de que el documento sea XHTML, su versión 1.1 recomienda sustituirlo por "***xml:lang***". Por ejemplo, **<html xml:lang="es"> </html>**.

— **<head *lang*|*xml:lang*> </head>**: encabezado del documento HTML. Alberga etiquetas que no influyen en la presentación de los contenidos del documento en el navegador, pero se encargan de definir aspectos genéricos del documento, como son el titulo del documento, el autor, la herramienta de desarrollo utilizada para su creación, una descripción de los contenidos, palabras clave que serán utilizadas por los buscadores para conocer los temas tratados en el documento, o el tipo de alfabeto utilizado en su redacción. Para todo ello, dentro de la cabecera, entre las etiquetas **<head> </head>**, se colocarán etiquetas tales como **<title> </title>** o **<meta>**. También suele localizarse dentro de la cabeza del docu-

mento las etiquetas **<link />**, **<style> </style>** y **<script> </script>**, encargadas de asociar un icono representativo al documento Web, unos estilos que personalizarán el aspecto que presentarán los contenidos del documento en el navegador, o scripts encargados de dinamizar los contenidos. Opcionalmente mediante el atributo "***lang***" ("***xml:lang***" en XHTML) podemos indicar el lenguaje utilizado en la descripción de la cabecera.

— **<title *lang|xml:lang*> </title>**: encierran el texto que hará de titulo del documento. Este será visible tanto en la barra de título del documento como en la pestaña del navegador en caso de tener abiertos más de un documento simultáneamente. Opcionalmente mediante el atributo "***lang***" ("***xml:lang***" en XHTML) podemos indicar el lenguaje utilizado en la redacción del título. Por ejemplo, **<title *lang="es"*>MI TUTULO – PAG HTML</title>.**

Figura 1.7. Efecto de la etiqueta "title" en el navegador

— **<meta *name http-equiv lang|xml:lang content* />**: Etiqueta encargada de aportar metainformación relacionada con el documento. Entendida esta como información de la información, corresponde con aquella información que no es parte del contenido del documento, pero que nos sirve para conocer aspectos tales como el autor del documento, una breve descripción de su contenido, palabras clave sobre los temas tratados en él que podrán ser utilizados por los buscadores de Internet, sus condiciones de uso o la herramienta software con que fue desarrollado, entre otras. Sus atributos más importantes son: "***name***" y "***http-equiv***", asignan un nombre identificador a la etiqueta, "***content***" informa sobre aspectos del contenido del documento indicados por "***name***" o "***http-equiv***" y "***lang***" especifica del lenguaje utilizado en el atributo "***content***".

Por ejemplo, si queremos informar en la cabecera de nuestro documento de que autor es "Arturo Martin Romero", perteneciente a la organización "IES Rio Arba", incluyendo una breve descripción del contenido, "Institulo IES Rio Arba. Ciclos Formativos Informatica.", y resaltando como palabras clave que serán utilizadas por los buscadores de Internet "ESO, bachillerato, FP, Ciclos Formativos, ESI, ASI, DAI":

```
<meta name="generator" content="Quanta Plus" />
<meta name="author" content="Arturo Martin Romero" />
<meta name="organizacion" content="IES Rio Arba" />
<meta name="description" lang="es" content="Institulo IES Rio Arba. Ciclos Formativos
    Informatica." />
<meta name="keywords" lang="es" content="ESO, bachillerato, FP, Ciclos Formativos, ESI,
    ASI, DAI" />
```

Además de la información anterior, una de las funciones más importantes de la etiqueta **<meta />** es informar al cliente o navegador Web de cual es el conjunto de caracteres o alfabeto

que se esta utilizando en el documento, lo que hará que puedan visualizarse sin problemas acentos, la letra "ñ", u otro tipo de caracteres propios del lenguaje latino y español utilizados en la elaboración del documento. Para no tener problemas en este aspecto, podemos indicar como sistema de codificación compatible con nuestro alfabeto a "utf-8".

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

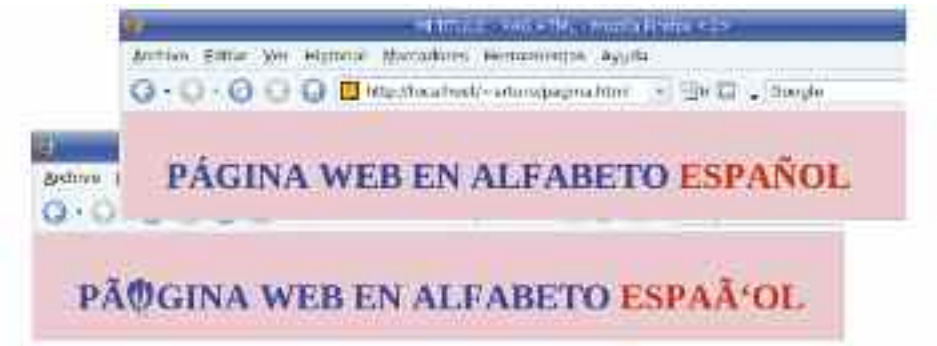


Figura 1.8. La etiqueta ***<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />*** *nos permite hacer uso de caracteres especiales como la letra "ñ" o los acentos*

En el caso de que el documento sea XHTML, existe una alternativa a la etiqueta **<meta />** anterior, que nos permite igualmente informar al navegador del sistema de codificación utilizado, "***<?xml version="1.0" encoding="UTF-8"?>***", junto con la versión del estándar XHTML utilizada.

Otro uso que le podemos dar a la etiqueta HTML **<meta />** es para refrescar nuestra página web cada cierto tiempo. Esto nos permite periódicamente refrescar los contenidos que esta visualizando el cliente Web manteniendo de manera actualizada la información mostrada. También puede ser utilizada para que al cabo de cierto tiempo automáticamente se refresque cargandose otra página Web diferente. Ambos casos se muestran a continuación:

```
<meta http-equiv="refresh" content="5">
<meta http-equiv="refresh" content="5; URL=otra-pagina.html">
```

— **<link *href rel type media* />:** Etiqueta utilizada en el encabezado del documento para informar de documentos externos que están relacionados de una manera u otra con el documento. Aunque puede utilizarse para indicar el conjunto de páginas Web junto con las cuales conforman el sitio Web, comúnmente es utilizada para informar de la hoja de estilos CSS de la que hace uso el documento HTML/XHTML para adecuar su presentación, o el fichero asociado al icono (png/ico) representativo del sitio Web que deseamos visualizar en el navegador. En relación a sus atributos más importantes, "***href***" informa de la ruta relativa del documento al que hace referencia, "***rel***" indica el tipo de relación entre el documento HTML/XHML y el documento referenciado, "***type***" el tipo de documento del que se trata, "***media***" informa del medio al que va destinado el documento y "***title***" asigna una breve descripción. Ejemplos del uso de esta etiqueta podrían ser los siguientes:

```
<link rel="index" href="http://www.escarbapedal.com/index.html" type="text/html"
      title="Indice del sitio Web" />
<link rel="shortcut icon" href="fotos/icono.png" type="image/png" />
```

```
<link rel="stylesheet" type="text/css" href="css/hoja-estilo-documento.css" media="screen" />
<link rel="stylesheet" type="text/css" href="css/hoja-estilo-impresion.css" media="print" />
```

*Figura 1.9. La etiqueta <**link** /> nos permite especificar un icono representativo para nuestro documento Web que será visualizado en la barra de direcciones del navegador y en la pestaña*

— **<style *type media title*> </style>**: Es utilizada para definir los estilos asociados a las distintas etiquetas utilizadas en el marcado del documento. Estos estilos se corresponden con un conjunto de propiedades que permiten personalizar el aspecto que presentarán los contenidos que forman parte del documento al ser visualizados desde un navegador o cliente Web. La lista de todas estas propiedades de estilo serán presentadas en el capítulo siguiente referente a las hojas de estilo CSS.

Entre sus atributos destacar a "***type***", el cual nos permite indicar el formato en que van a definirse los estilos (text/css), "***media***" informa del medio al que van destinados los estilos que en esta sección se definan (por defecto, la pantalla, "screen"), y "***title***" para asignar una breve descripción.

Para hacer referencia desde esta sección del documento a aquella etiqueta a la que le queremos asignar un estilo tenemos tres opciones:

1) A través del propio nombre de la etiqueta. El estilo asignado afectará de manera global a todas las etiquetas que tengan ese nombre dentro del documento sin distinción. Por ejemplo, en HTML/XHML para insertar una imagen en un documento se hace uso de la etiqueta **<img />**. Si quisiéramos que todas las imágenes del documento presentarán el mismo aspecto en su presentación lo definiríamos en esta sección:

```
<style type="text/css" media="screen">
    img { /* Propiedades de estilo asociadas a todas las imágenes del documento. */ }
</style>
```

2) Haciendo referencia al valor asignado al atributo "***class***" de la etiqueta a la que queremos asignar el estilo. Este atributo esta especialmente pensado para ello y nos permite clasificar de manera univoca a cada una de las etiquetas utilizadas en el documento. En el caso de que el valor asignado a este atributo sea compartido por varias etiquetas, el estilo afectará a todas ellas. Para informar al navegador Web de que estamos haciendo referencia al valor de este atributo, y que no se trata del nombre de una etiqueta, lo precederemos del símbolo almohadilla, "**#**".

```
<style type="text/css" media="screen">
```

```
    img#imagen1 { /* Propiedades de estilo asociadas a la imagenes con atributo class=
    "imagen1" */ }
    #estilo1 { /* Propiedades de estilo asociadas a toda etiqueta con atributo class=
    "estilo1" */ }
</style>
```

En el caso en que a un mismo elemento queramos asignarle más de un estilo de los definidos en la sección de estilos, es posible asignarle al atributo "***class***" más de un valor separados por espacios en blanco: *class="estilo1 estilo2 ..."*.

3) Haciendo referencia al valor asignado al atributo "***id***" de la etiqueta a la que queremos asignar el estilo. Al igual que el atributo "***class***", este atributo permite identificar de manera univoca cada uno de los elementos del documento. Aunque esta especialmente pensado para poder referenciar desde JavaScript a una etiqueta, puede ser aprovechado igualmente en las hojas de estilo CSS. Para informar al navegador Web de que estamos haciendo referencia al valor de este atributo, y que no se trata del nombre de una etiqueta, lo precederemos de un punto, "**.**".

```
<style type="text/css" media="screen">
    img.imagen1 { /* Propiedades de estilo asociadas a la imagenes con atributo id=
    "imagen1" */ }
    .estilo1 { /* Propiedades de estilo asociadas a toda etiqueta con atributo id="estilo1" */ }
</style>
```

— **<script *type src*> </script>**: Nos permite hacer uso de lenguajes de script dentro del documento HTML/XHTML con la finalidad de aumentar la potencia del sitio Web aportándole dinamismo. Estos scripts son un bloque de instrucciones redactado normalmente mediante la utilización de un lenguaje de programación interpretado o comandos del sistema, al cual se le asigna una determinada función. Sin su utilización los contenidos visualizados en el navegador como resultado de interpretar el documento HTML/XHTML son siempre los mismos independientemente de quien, cuando, como y desde donde se consulte. En tal situación, se dice que las páginas Web son estáticas. Mediante la utilización de scripts tenemos control sobre la hora y fecha en que se accede a la información, desde donde y como se accede, pudiendo personalizar en cada caso los contenidos que se muestran, haciendo que las páginas Web sean dinámicas.

Estos scripts pueden escribirse embebidos en el propio documento o editarse en un fichero independiente e importarlo, haciendo uso de algún lenguaje de programación de scripts. Entre estos destacar a "JavaScript" y "VBScript" por ser los más famosos, aunque solo el primero de ellos guarda un compatibilidad total con la mayoría de los clientes o navegadores Web actuales. "VBScript" (Visual Basic Script) presenta el inconveniente de ser un lenguaje desarrollado por Microsoft para su software, por lo que únicamente garantiza un correcto funcionamiento si los scripts escritos mediante este lenguaje es interpretado mediante su navegador Web "Internet Explorer". En relación a "JavaScript", no confundir con "JScript", siendo este último una adaptación por parte de Microsoft del primero para sus entornos, lo que tampoco garantiza que los scripts realizados con "JScript" puedan ser interpretados correctamente por otros navegadores diferentes a "Internet Explorer".

Mediate sus atributos "type" y "src" indicamos el tipo de lenguaje utilizado en el script, y la ruta relativa del fichero externo a importar que contiene el script en el caso de que no se embeba en el propio documento.

```
<script type="text/javascript">
    /* Lineas de código del script embebido en el documento. */
</script>
<script type="text/vbscript">
    /* Lineas de código del script embebido en el documento. */
</script>
<script type="text/javascript" src="scripts/script-externo.js"> </script>
```

A diferencia de las anteriores, la etiqueta <script></script> puede utilizarse tanto en el encabezado como en el cuerpo del documento. En el capítulo correspondiente a JavaScript veremos en más profundidad todos estos aspectos, y nos familiarizaremos con este lenguaje esencial hoy en día en la implementación de sitios Web.

— **<body *id class style lang|xml:lang title*> </body>**: El cuerpo del documento, es la zona donde se definen los elementos que van a componerlo. Entre ellos cabría destacar párrafos de texto, imágenes u otro tipo de objetos como películas flash, o archivos de sonido. Aquí va a encontrarse todos los contenidos del documento etiquetado mediante el uso de las etiquetas HTML que veremos en los apartados siguientes.

En cuanto a sus atributos, "***id***" nos permite identificar el cuerpo del documento para que pueda ser referenciado desde los scripts que se hayan definido en el documento (*<script></script>*), "***class***" clasifica al cuerpo del documento para poder asignarle un estilo definido en alguna de las hojas de estilo CSS que tenga asociadas el documento, "***style***" nos permite asignar una lista de propiedades de estilo CSS separadas estas por ";" que personalice el cuerpo del documento, "***lang|xml:lang***" establece el lenguaje utilizado para elaborar el texto, y "***title***" le asigna una breve descripción que será visualiza al situar el ratón sobre cualquier zona del cuerpo del documento.

```
<body id="cuerpo" class="estilocuerpo" style="background-color: pink;" lang="es"
title="Descripción sitio Web">
    /* Contenido del documento etiquetado. */
</body>
```

4.2. Etiquetas HTML para el formateado del texto

Con la finalidad de dar forma al texto que contiene el documento, HTML dispone de un conjunto de etiquetas de formateado. Estas se presentan a continuación:

ETIQUETAS HTML PARA EL FORMATEADO DEL TEXTO	
ETIQUETAS HTML Y ATRIBUTOS	**EJEMPLO**
<h*n* ***id class style lang|xml:lang title***> </h*n*> <p ***id class style lang|xml:lang title***> <blockquote ***id class style lang|xml:lang title***></blockquote> <pre ***id class style lang|xml:lang title***> </pre> <big ***id class style lang|xml:lang title***> </big>	<body style="background:#FF0000"> MI TITULO Un libro recomendado para configurar <u>ServiciosInternet</u> es el libro editado por

<small *id class style lang|xml:lang title*> </small>

<b *id class style lang|xml:lang title*>
<i *id class style lang|xml:lang title*>

<code *id class style lang|xml:lang title*> </code>
<samp *id class style lang|xml:lang title*> </samp>
<cite *id class style lang|xml:lang title*> </cite>
<q *id class style lang|xml:lang title*> </q>

<span *id class style lang|xml:lang title*>

<a *id class style title*>

<br *id class style* />
<hr *id class style* />
  &|a|e|i|o|u|acute; &|n|N|tilde; ...
<!— comentarios —>

<big>Ed. Mira</big> y autores <big>Arturo y Juanjo Martín Romero</big> <small>(precio recomendado: 30 euros)</small>.
</p>
</body>

— <h*n* *id class style lang|xml:lang title*> </h*n*>: Convierte al texto encerrado entre ambas etiquetas en un título dentro del documento, donde "n", con un valor entre 1 y 6, establece el tamaño del texto de mayor a menor respectivamente.

En cuanto a sus atributos, "***id***" nos permite identificar el titulo para que pueda ser referenciado desde los scripts que se hayan definido en el documento (<*script*> </*script*>), "***class***" clasifica al titulo del documento para poder asignarle un estilo definido en alguna de las hojas de estilo CSS que tenga asociadas el documento, "***style***" nos permite asignar una lista de propiedades de estilo CSS separadas estas por ";", "***lang|xml:lang***" establece el lenguaje utilizado para elaborar el texto, y "***title***" le asigna una breve descripción que será visualiza al situar el ratón sobre el titulo.

Una característica implícita de la utilización de esta etiqueta, es que provoca un salto de línea antes y después de la presentación del título en el navegador. En el caso de que deseemos evitar estos saltos de línea, será necesario asignar a la propiedad de estilo "display" su alvor "inline".

```
<H1 id="titulo1" class="titulo" style="color: red; display: inline;" lang="es"
title="Este es mi Titulo">
<!— Texto del título —>
</H1>
```

Otras propiedades de estilo que pueden ser asignadas a esta etiqueta a través de su atributo "**style**" pueden ser consultadas en el capítulo siguiente relativo a las hojas de estilo CSS. Entre ellas podrían considerarse las de mayor interés: "**color**" asigna un color al texto, "**font**" permite ajustar el tamaño, el grosor, el estilo cursiva, o la familia del texto, "**text-decoration**", lo remarca, "**text-align**" lo justifica, "**border**" le asigna un borde, "**padding**" establece la distancia ente el texto del título y su borde, "**margin**" determina sus margenes, y "**background-color**" asigna un color de fondo.

Figura 1.10. La etiqueta HTML <hn></hn> inserta títulos en el documento, cuyo tamaño depende del valor de n=1..6

— **<p *id class style lang|xml:lang title*> </p>**: Genera un párrafo con el texto que encierra. Al igual que la etiqueta **<h***n***> </h***n***>**, provoca un salto de línea implícito antes y después de mostrar el párrafo de texto en la navegador Web. Para evitar estos saltos de línea puede hacerse uso de la propiedad de estilo "*display: inline;*".

En cuanto a sus atributos, "***id***" nos permite identificar el parrafo para que pueda ser referenciado desde los scripts definidos en el documento (<*script*> </*script*>), "***class***" clasifica el párrafo para poder asignarle un estilo definido en alguna de las hojas de estilo CSS que tenga asociadas el documento, "***style***" nos permite asignar una lista de propiedades de estilo CSS separadas estas por ";", "***lang|xml:lang***" establece el lenguaje utilizado para elaborar el texto, y "***title***" le asigna una breve descripción que será visualiza al situar el ratón sobre el párrafo.

De todas sus atributos cabría destacar el atributo "***style***" el cual nos permite personalizar la aspecto del texto en su presentación. La lista de todas las propiedades de estilo disponibles asignables a este atributo puede ser consultada en el capítulo correspondiente a las hojas de estilo CSS, entre las que sería interesante consultar, las asociadas a los apartados del capítulo 2, 2.4 y 2.6, entre las que destacaríamos por su interés: "**color**" asigna un color al texto, "**font**" permite ajustar el tamaño, el grosor, el estilo cursiva, o la familia del texto, "**text-decoration**", lo remarca, "**text-align**" lo justifica y "**text-indent**" sangra la primera línea del párrafo.

— **<blockquote *id class style lang|xml:lang title*> </blockquote>**: Al igual que la etiqueta anterior, estructura el texto en párrafos, pero diferenciando el párrafo de texto encerrado entre estas etiquetas sangrándolo a izquierda y derecha. Sus atributos tienen el mismo efecto viso para **<*p*> </*p*>** (ver figura 1.11).

— **<pre *id class style lang|xml:lang title*> </pre>**: Nos permite crear un bloque de texto preformateado de manera libre. Se caracteriza por respetar todos los espacios en blanco, tabulaciones y saltos de línea provocados por el desarrollador en el texto que encierra, al contrario de lo que ocurre en en resto del documento etiquetado, ya que como ya se comentó en el apartado X.X referente a las características de HTML, los saltos de línea y tabulaciones introducidos en el texto del documento son omitidos, y las ráfagas de espacios en blanco son sustituidos por un único espacio de manera implícita por el intérprete HTML. Además, esta etiqueta omite igualmente los saltos de línea producidos por el intérprete HTML de forma implícita, cuando detecta que una línea de texto no cabe en el ancho del área de contenidos reservada por el navegador para mostrar el documento HTML/ XHTML.

En relación a las etiquetas anteriores, esta comparte con ellas únicamente que el intérprete al detectarla provoca un salto de línea antes y después de mostrar el texto preformateado que encierra. Como ya se ha indicado en los casos anteriores, para evitar estos saltos de línea puede hacerse uso de la propiedad de estilo "*display: inline;*".

En cuanto a sus atributos, "***id***" nos permite identificar el texto preformateado para que pueda ser referenciado desde los scripts definidos en el documento (<*script*> </*script*>), "***class***" clasifica el la etiqueta para poder asignarle un estilo definido en alguna de las hojas de estilo CSS que tenga asociadas el documento, "***style***" nos permite asignar una

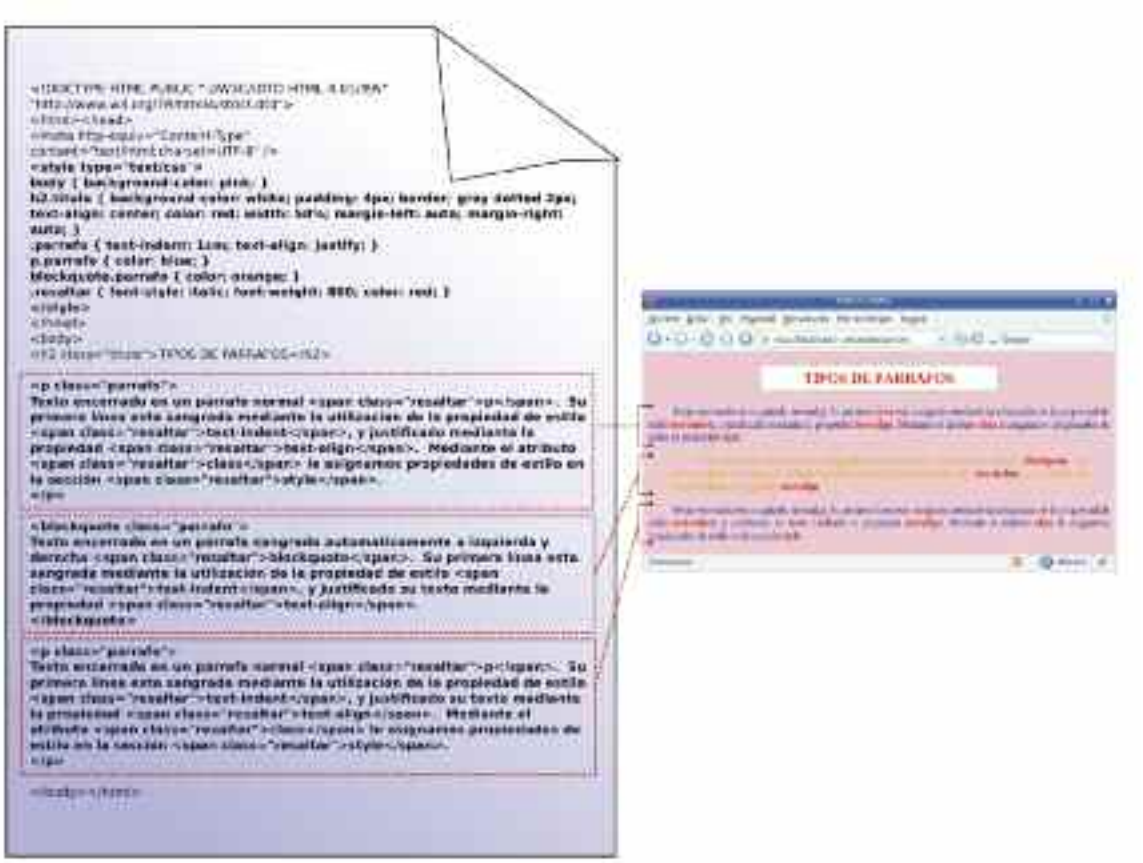


*Figura 1.11. Las etiquetas HTML <**p**> </**p**> y <**blockquote**> </**blockquote**> estructuran el texto del documento en párrafos*

lista de propiedades de estilo CSS separadas estas por ";", "***lang*|*xml:lang***" establece el lenguaje utilizado para elaborar el texto, y "***title***" le asigna una breve descripción que será visualiza al situar el ratón sobre el texto.

De todos sus atributos cabría destacar "***style***" el cual nos permite personalizar el aspecto del texto en su presentación. La lista de todas las propiedades de estilo disponibles asignables a este atributo puede ser consultada en el capítulo correspondiente a las hojas de estilo CSS (capítulo n.º 2), entre las que sería interesante consultar, las asociadas a los apartados 2.4 y 2.6.

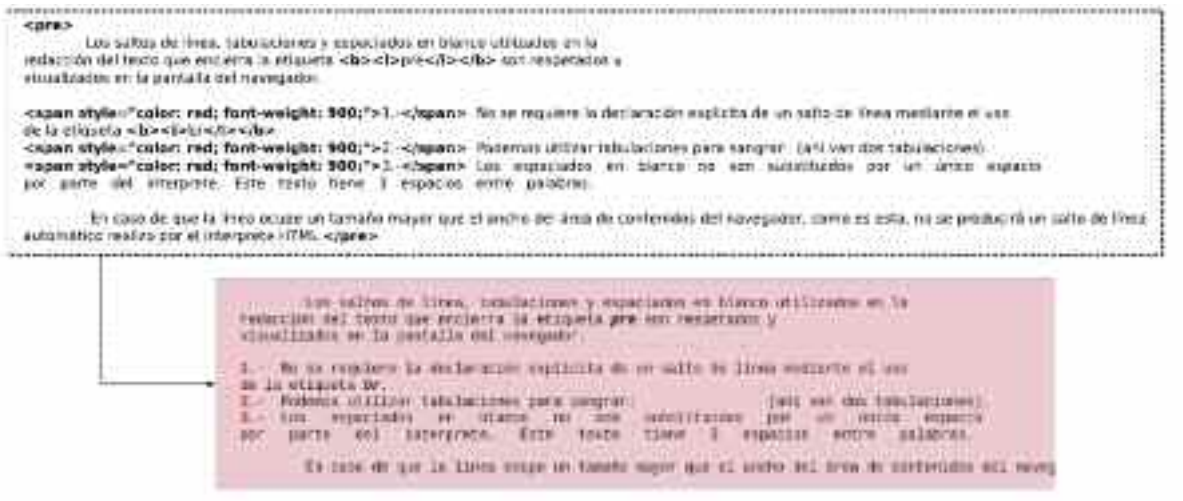


Figura 1.12. La etiqueta <pre> </pre> nos permite generar un párrafo de texto performateado de manera libre. Los saltos de línea, tabulaciones y espaciados en blanco no son omitidos por el intérprete HTML

— <**big|small|sub|sup|b|i|cite|q|code|samp** ***id class style lang|xml:lang title***> </**big|small|sub|sup|b|i|cite|q|code|samp**>: Dan un formato específico al texto que encierran. Mientras ***<big> </big>*** nos permite mostrar el texto en tamaño grande, ***<small> </small>*** lo hace en tamaño pequeño. ****** y ****** muestran el texto como subíndice o superíndice respectivamente. *** *** pone el texto en negrita e *** *** en cursiva. ***<cite> </cite>*** y ***<q> </q>*** permiten intercalar citas en el texto dándoles un formato diferenciador, poniendo el texto en cursiva la primera y encerrando el texto entre comillas dobles la segunda. ***<code> </code>*** y ***<samp> </samp>*** transforman el texto asociándole un formato similar al asociado a una máquina de escribir. Todas ellas se caracterizan por no introducir saltos de línea implícitos tal como sucedía con las etiquetas HTML anteriores.

En cuanto a sus atributos, "***id***" nos permite identificar estas etiquetas para que puedan ser referenciadas desde los scripts definidos en el documento (<*script*> </*script*>), "***class***" las clasifica para poder asignarle alguno de los estilos definidos en alguna de las hojas de estilo CSS que tenga asociadas el documento, "***style***" nos permite asignar una lista de propiedades de estilo CSS separadas estas por ";", "***lang|xml:lang***" establece el lenguaje utilizado para escribir el texto, y "***title***" les asigna una breve descripción que será visualiza al situar el ratón sobre el texto.

De todos sus atributos cabría destacar "***style***" el cual nos permite personalizar el aspecto del texto en su presentación. La lista de todas las propiedades de estilo disponibles asignables a este atributo puede ser consultada en el capítulo correspondiente a las hojas de estilo CSS (capítulo n.º 2), entre las que sería interesante consultar, las asociadas a los apartados 2.4 y 2.6.

— <**span** ***id class style lang|xml:lang title***> </**span**>: Etiqueta utilizada para asociar un estilo personalizado al texto que encierra mediante las propiedades de estilo CSS asignadas a su atributo "***style***". La lista de todas las propiedades de estilo disponibles asignables a este atributo puede ser consultada en el capítulo correspondiente a las hojas de estilo CSS (capítulo n.º 2). Por tanto, cuando el formato ofrecido por las etiquetas anteriores no se ajusta al aspecto que se desea que presente el texto en la pantalla del navegador, lo normal es hacer uso de esta etiqueta HTML. En cuanto al resto de sus atributos, "***id***", "***class***", "***lang|xml:lang***", y "***title***", tienen el mismo efecto que en las etiquetas HTML anteriores.

— <**a** ***href name target id class style title***> </**a**>: Establece un vínculo con un elemento que puede encontrarse dentro del propio documento HTML/XHTML, o en un documento externo. También es utilizado para establecer enlaces con direcciones de correo electrónico y descargas de archivos. El texto, imagen u objeto que encierren estas etiquetas, harán de enlace, de tal forma que al pinchar con el ratón sobre ello, nos lleve al elemento vinculado.

Su atributo "***href***" establece la ruta del elemento al que enlaza, "***name***" marca el elemento para que pueda ser referenciado, "***target***" indica en nombre del marco o frame en que será cargado el documento asignado a "href" en el caso de que hayamos dividido la página en frames (ver apartado X) y el resto de atributos con efecto similar al de las etiquetas anteriores.

Distinguiendo entre los distintos tipo de vínculos o enlaces que pueden establecerse destacar:

1) Vínculos dentro del propio documento. Nos permiten enlazar con cualquier parte del documento donde se haya definido una marca haciendo referencia a su identificador precedido del carácter almohadilla (**#**), *** ***. Estas marcas se insertan en el documento mediante la utilización de esta misma etiqueta, identificándola a través de su atributo "name": ******. Entre sus utilidades destacar la de crear un índice de contenidos en la página Web, desde la cual se puede enlazar con cada una de las partes del documento.

2) Vínculos con documentos externos de nuestra Web. Nos permite establecer enlaces entre las distintas páginas Web que componen un sitio Web. A través del atributo "href" indicaremos el documento o marca externa con el cual se establece el enlace.

```
<a href="documento-externo.html"> </a> <!— Establece un enlace sobre un documento externo —>
<a href="documento-externo.html#nombre_marca"> </a> <!— Establece un enlace sobre una marca definida en un documento externo —>
```

En el caso de que el enlace definido no apunte a un documento Web, sino a un archivo *pdf*, *jpg*, *zip*, *rar*, etc. puede utilizarse como enlace de descarga:

```
<a href="documento-externo.pdf|jpg|zip|..."> </a> <!— Enlace de descarga sobre un documento externo —>
```

3) Vínculos sobre documentos externos a nuestra Web. Nos permite establecer enlaces haciendo uso del identificador uniforme del recurso externo, URI (*Uniform Resource Identifier*), enlazando de esta forma con páginas Web externas, o partes de ellas haciendo referencia a sus marcas, u otro tipo de archivos, *pdf*, *jpg*, *zip*, *rar*, ... , utilizándose en este último caso en un enlace de descarga:

```
<a href="URI-documento-externo.html"> </a> <!— Establece un enlace sobre un documento externo —>
<a href="URI-documento-externo.html#nombre_marca"> </a> <!— Establece un enlace sobre una marca definida en un documento externo —>
<a href="URI-documento-externo.pdf|jpg|zip|..."> </a> <!— Enlace de descarga sobre un documento externo —>
```

4) Vínculo sobre una dirección email. En ultima instancia, si el equipo donde se ejecuta el navegador o cliente Web, esta configurado como cliente de correo electrónico sobre algún servidor SMTP, puede establecerse un enlace de tal forma que al pinchar sobre él, se ejecute este cliente email para enviar un correo electrónico a la dirección que indiquemos. Para ello se hará uso de la directiva "**mailto**", seguida de la dirección o direcciones email separadas por comas, a las que se desea enviarse el mensaje. Puede indicarse en la misma declaración del enlace, el asunto y el cuerpo del mensaje, haciendo uso de las palabras reservadas **subject**, **body** y los caracteres "**?**" y "**&**":

```
<a href="mailto:direccion1@dominio,direccion2@dominio?subject=Este es el asunto&body=
Este es el cuerpo"> </a>
```

Estos vínculos podrán comprobarse mediante la realización de ejercicios prácticos a lo largo del capítulo.

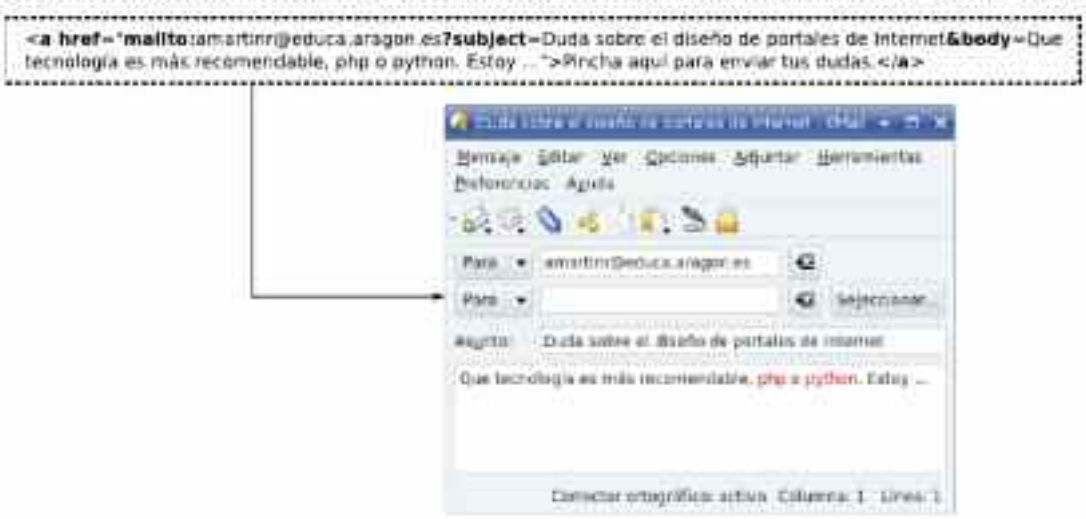


Figura 1.13. El enlace a una dirección email ejecuta un cliente de correo electrónico para el envío del correo electrónico a la dirección indicada

— **<br *id class style* />**: Es la etiqueta HTML que tendremos que utilizar en documentos HTML/XHTML para provocar de manera explicita un salto de línea en la presentación de sus contenidos en el navegador. Recordar que por defecto, a excepción de la etiqueta ***<pre> </pre>***, de manera implícita, el navegador a través de su intérprete HTML omite todo aquel salto de línea que se haya realizado al editar el documento.

En cuanto a sus atributos, "***id***" nos permite identificarla para que pueda ser referenciada desde los scripts definidos en el documento (<*script*> </*script*>), "***class***" la clasifica para poder asignarle alguno de los estilos definidos en alguna de las hojas de estilo CSS que tenga asociadas el documento y "***style***" nos permite asignar una lista de propiedades de estilo CSS separadas estas por ";", aunque al tratarse de un simple salto de línea, la asignación de propiedades tiene poca utilidad.

— **<hr *id class style* />**: Muestra una línea horizontal que nos permite separar dos áreas de textos, provocando de manera implícita un salto de linea antes y después de su representación. Sus atributos tienen el mismo efecto que las etiquetas anteriores, destacando a "***style***" al ser este el encargado de permitirnos personalizar el aspecto de la línea separadora. Entre las propiedades de estilo disponibles, cuya lista completa puede consultarse en el capitulo nº2 referente a hojas de estilo CSS, cabría señalar a "**width**", al ser la que controla su anchura, "**height**" su altura, "**background-color**" su color y "**border**" el aspecto de su borde.

— "** &|a|A|e|E|i|I|o|O|u|U|acute; &|u|U|uml; &|n|N|tilde; < >** ..." son los caracteres especiales HTML. Como ya se ha explicado a lo largo del capítulo, las ráfagas de espacios en blanco realizadas en la edición del documento son substituidos por un único espacio en blanco por el intérprete HTML, a excepción de si estos aparecen en la etiqueta ***<pre> </pre>***. En el caso de que queramos mostrar de un espacio en blanco HTML dispone del carácter especial "** **", el cual habrá que repetir tantas veces como espacios en blanco queramos provocar. En relación a los acentos, en caso de no hacer uso de la etiqueta ***<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />*** o ***<?xml version="1.0" encoding="UTF-8"?>***, será necesario indicar explicitamente al intérprete HTML la utilización de estos mediante el carácter especial "**&|a|A|e|E|i|I|o|O|u|U|acute;**" (Á -> Á). De forma similar ocurre con las diéresis, "**&|u|U|uml;**" (&uuml -> ü), la ñ, "**&|n|N|tilde;**" (ñ -> ñ), el signo menor (< -> <) o mayor (> -> >), un símbolo de copyright (**© ->** ©), unas comillas dobles (**"** -> ") o cualquier otro tipo de símbolo. En la dirección Web "***http://www.w3schools.com/tags/ref_symbols.asp***" podemos encontrar un lista más detallada.

— <*!— comentarios —*>: Etiqueta HTML que nos permite introducir comentarios dentro del documento. Estos permiten documentarlo y facilitar su posterior mantenimiento.

Por último, recordar que etiquetas ampliamente utilizadaspara dar formato al texto en versiones HTML anteriores a la 4.1 como *** ***, ***<u> </u>***, ***<s> </s>*** o *** ***, se encuentran desaprobados en la actualidad, supliendo su funcionalidad los parámetros de estilo CSS.

4.3. Etiquetas HTML utilizadas en la creación de listas

HTML dispone de un conjunto de etiquetas para la creación de listas ordenadas o numeradas, *** *** (ol, Ordered Lists), y no ordenadas, *** *** (ul, Unordered Lists). Cada uno de los elementos que componen la lista se etiquetará mediante *** ***.

ETIQUETAS HTML PARA LA CREACIÓN DE LISTAS

<ul *type* ***id class style lang|xml:lang title***>
  <li *type* ***id class style lang|xml:lang title***>
  <li *type* ***id class style lang|xml:lang title***>
  ...

<ol *type* ***id class style lang|xml:lang title***>
  <li *type* ***id class style lang|xml:lang title***>
  <li *type* ***id class style lang|xml:lang title***>
  ...

<dl ***id class style lang|xml:lang title***>
  <dt ***id class style lang|xml:lang title***> </dt>
  <dd ***id class style lang|xml:lang title***> </dd>
  ...
</dl>

```
<html><head>
<meta http-equiv="Content-Type" content="text/html;charset=UTF-8" />
...
</head><body>
<hr><h2 align=center>PAELLA VALENCIANA </h2><hr>
Para poder preparar una riquísima paella serán necesarios los siguientes ingredientes (6 personas):
<ul type="square" style="font-style:italic">
  <li> 1Kg Arroz. </li>
  <li> 150 grs. Judias Verdes. </li>
  ...
</ul>
Para su preparación deberemos seguir los siguientes pasos:
<ol type="I" style="font-style:italic">
  <li> Freir los pimientos a fuego lento durante 5 minutos. </li>
  <li> Añadir al sofrito la cebolla. </li>
  ...
</ol>
</body>
</html>
```

— <**ol|ul|li** *type* ***id class style lang|xml:lang title***> </**ul|ol|li**>: Etiquetas HTML utilizadas en documentos HTML/XHML para la creación de listas ordenadas o numeradas, *** *** y desordenadas, *** ***. En el caso de las primeras, precediendo a cada uno de los elementos de la lista, *** ***, se añadirá un carácter alfanumérico que se irá autoincrementando. En las segundas, este carácter será sustituido por un símbolo. El formato de éste carácter o símbolo se establece mediante el atributo "**type**" o a través de la propiedad de estilo "**list-style-type**". Los valores que se les puede asignar pueden ser consultados en el apartado X.X (3.5) del capítulo 2 asociado a las hojas de estilo CSS (por ejemplo, "***decimal|decimal-leading-zero|lower-roman|upper-roman|lower-alpha|upper-alpha***" para las ordenadas y "***disc|circle|square***" para las no ordenadas).

En cuanto al resto de sus atributos, "***id***" nos permite identificar a estas listas para que puedan ser referenciadas desde los scripts definidos en el documento (<*script*> </*script*>), "***class***" las clasifica para poder asignarle alguno de los estilos definidos en alguna de las hojas de estilo CSS que tenga asociadas el documento, "***style***" nos permite asignar una lista de propiedades de estilo CSS separadas estas por ";", "***lang|xml:lang***" establece el lenguaje utilizado para escribir el texto, y "***title***" les asigna una breve descripción que será visualiza al situar el ratón sobre el texto.

```
<h3 style="font-family: sans-serif; color: brown; text-decoration: underline;">Rutas BTT y sus caracteristicas:</h3>
<ol type="upper-roman">
  <li style="font-style: italic; font-weight: 600; color: blue;">Camino de Santiago Francés.</li>
    <ul type="square" style="color: red;">
      <li><span style="font-weight: 800;">Origen y destino:</span> Roncesvalles a Santiago Compostela.</li>
      <li><span style="font-weight: 800;">Distancia:</span> 750Km.</li>
      <li><span style="font-weight: 800;">Descripción:</span> Espectacular, Inolvidable y Gente entrañable.</li>
    </ul>
  <li style="font-style: italic; font-weight: 600; color: blue;">Camino de Santiago del Norte.</li>
    <ul type="square" style="color: red;">
      <li><span style="font-weight: 800;">Origen y destino:</span> Fuenterrabia (Hondarribia) a Santiago Compostela.</li>
      <li><span style="font-weight: 800;">Distancia:</span> 500Km.</li>
      <li><span style="font-weight: 800;">Descripción:</span> Montañas legendarias, paisajes de postal, olor a mar.</li>
    </ul>
  <li style="font-style: italic; font-weight: 600; color: blue;">Vía de la Plata.</li>
    <ul type="square" style="color: red;">
      <li><span style="font-weight: 800;">Origen y destino:</span> Sevilla a Gijón.</li>
      <li><span style="font-weight: 800;">Distancia:</span> 840Km.</li>
      <li><span style="font-weight: 800;">Descripción:</span> Impresionantes Dehesas, un paso atrás en el tiempo..... sin palabras.</li>
    </ul>
</ol>
```

— **<dl|dt|dd *id class style lang|xml:lang title*> </dl|dt|dd>**: Aunque de utilización menos habitual que las anteriores, nos permiten igualmente crear listas. Las etiquetas ***<dl>*** ***</dl>*** (Definition List), nos permiten definir la lista, ***<dt> </dt>*** (Definition Term), definen los terminos de la lista que la forman y ***<dd> </dd>*** (Definition Data), definen los datos. En cuanto a sus atributos tienen el mismo sentido que el descrito en las etiquetas anteriores.

4.4. Etiquetas HTML para la inserción de imágenes, vídeos y sonidos

En este apartado se presentarán las etiquetas HTML que nos permiten introducir en un documento algo más que texto: imágenes, películas y sonido.

ETIQUETAS HTML PARA INSERCIÓN DE IMÁGENES, VÍDEOS Y SONIDO

<**img** *src height width id class style lang|xml:lang title alt name* />

<**object** *type height width id class style lang|xml: lang title classid codebase name*>
  <**param** *name value*>
  < **param** *name value*>
  ...
</**object**>

```
<html><head>
<meta http-equiv="Content-Type" content="text /html;charset=UTF-8" />
</head><body>
<hr><h2 align=center>MUSICA HEAVY ÉPICO </h2><hr>
<object type="application/x-shockwave-flash" width="0" height="0">
    <param name="movie" value="musica/heavy-rock.swf" />
    <param name="menu" value="false" />
    <param name="quality" value="high" />
    <param name="loop" value="true" />
</object>
<p> Ahora estarás escuchando de fondo de manera ininterrumpida uno de los mejores temas del heavy metal épico nacional del grupo Tierra Santa, Legendario.
...
</p>
<p style="text-align: center;">
    <img src="imagenes/tierra-santa.jpg" width ="300" height="200" />
</p>
</body></html>
```

— <**img** *src height width id class style title alt name* />: Inserta la imagen especificada a través de su atributo "***src***" en el documento. El valor asignado a este atributo deberá corresponderse con la ruta relativa de la imagén en relación a la ubicación del documento HMTL/XHTML. En cuanto al resto de sus atributos, "height" y "width" especifican la altura y anchura de la imagen, "***id***" nos permite identificarla para que pueda ser referenciada desde los scripts definidos en el documento (<*script*> </*script*>), "***class***" la clasifica para poder asignarle alguno de los estilos definidos en alguna de las hojas de estilo CSS que tenga asociadas el documento, "***style***" nos permite asignar una lista de propiedades de estilo CSS separadas estas por ";", "***title***" le asigna una breve descripción que será visualiza al situar el ratón sobre el texto, "***alt***" establece el texto alternativo que será mostrado en caso de que no pueda cargarse la imagen, y "name" asigna un nombre a la imagen para que pueda ser referenciada desde las hojas de estilo, los scripts o ser utilizada como marca para un enlace HTML (**<*a*> </*a*>**).

  De todos sus atributos cabría destacar "***style***" el cual nos permite personalizar el aspecto de la imagen en su presentación. La lista de todas las propiedades de estilo disponibles asignables a este atributo puede ser consultada en el capítulo correspondiente a las hojas de estilo CSS (capítulo nº2), entre las que cabría mencionar a "**border**", la cual ajusta su borde, "**padding**" la distancia del relleno o marco entre la imagen y su borde, "**back-**

ground-color", color de ese relleno o marco, o "**margin**" establece los margenes respecto al espacio que la contiene. Para poder justificar la imagen, puede hacerse uso de la propiedad "**margin**", o introducir esta en un elemento HTML contenedor (*** ***, *** ***, etc.) donde mediante su propiedad de estilo "**text-align**" se establezca un alineaneamiento.

Figura 1.14. La etiqueta HTML ***<img />*** *nos permite insertar imágenes en nuestros documentos*

Otra posibilidad de posicionar la imagen sería haciendo uso de las propiedades de estilo CSS como "**position**" o "**float**", como pueden consultarse en el apartado X del capitulo X.

— **<object** *type data classid codetype codebase height width standby id class style lang|xml:lang title name*> **</object>**: Embebe objetos de tipo imagen *(type="**image/jpg|image/gif|image/png**")*, películas de Adobe Flash *(type="**application/x-shockwave-flash**")*, audio (*type="**audio/x-mp3|application/ogg|audio/mp4|audio/mp4a-latm|audio/mpeg**"*), video (*type="**video/x-flv|video/mp4|video/x-m4v|audio/mp4a-latm|video/3gpp|video/quicktime**"*) o applets de Java (*type="**application/x-java-applet**"*) en nuestros documentos HTML/XHTML. Para que estos objetos sean visibles en el navegador, será necesario tener instalados los complementos o plugin correspondientes[5]. Las últimas versiones de los navegadores más extendidos tan sólo incluyen los plugin asociados a los objetos más estandarizados, como son las imágenes y películas flash (**.swf*).

En cuanto a sus atributos, "***type***" informa del tipo de objeto, "***data***" especifica la ruta relativa donde se encuentra el objeto en relación al documento HTML/XHTML desde el que se invoca, "***classid***" identifica la clase del objeto, "***codetype***" indica el tipo de contenido indicado en "***classid***", "***codebase***" establece la ruta base en relación a la cual se pueden establecer rutas relativas por parte del resto de atributos, "***height***" y "***width***" establecen las dimensiones con que se mostrará el objeto en el navegador, "***standby***" determina el texto

5 Un plugin es un modulo software que al agregarlo a un programa lo complementa asignándole nuevas funciones específicas. Por ejemplo, un navegador, por defecto, no sabe reproducir vídeo, pero tras instalar el plugin adecuado, ya será reproducible.

que se verá en el navegador mientras se carga el objeto, y el resto de atributos, tienen el mismo sentido que para las etiquetas HTML anteriores. Según esto, las dos etiquetas HTML siguientes proporcionarían un resultado equivalente, donde "imagenes" se corresponde con una subcarpeta que contiene las imágenes del documento:

```
<img src="imagenes/archivo-imagen.jpg" width="400px" height="300px" style="border: brown; padding: 6px; background-color: orange;" />
<object type="image/jpg" data="imagenes/archivo-imagen.jpg" width="400px" height="300px" style="border: brown; padding: 6px; background-color: orange;"></object>
```

En el caso de querer insertar un objeto flash la etiqueta HTML sería la siguiente, donde "objetos" se corresponde con una subcarpeta que contiene las películas flash del documento:

```
<object type="application/x-shockwave-flash" data="objetos/archivo-flash.swf" width="400px" height="300px"> </object>
```

Otra característica importante de esta etiqueta es que nos permite reutilizar páginas HTML como si se tratase de otro objeto más, informando al intérprete HTML de ello mediante la indicación del tipo de objeto "**text/html**":

```
<object type="text/html" data="archivo-html.html" width="100%" height="400px"></object>
```

En cuanto al resto de atributos, mencionar que tienen un efecto similar al del resto de elemento HTML.

— **<param *name type value typevalue id* />**: Personaliza el comportamiento de la reproducción de un objeto, ***<object> </object>***, en nuestro navegador, modificando las propiedades de éste que no son editables a través del atributo "***style***". Permite controlar aspectos como son la calidad de la película (*quality*), el color de fondo (*bgcolor*), la repetición de la reproducción (*loop*) o la visualización del menú (*menu*) de control al pinchar con el botón derecho del ratón sobre el objeto. Su atributo "***name***" indica el aspecto a personalizar, y mediante "***value***" se le asigna un valor. "***valuetype***" establece si el "***value***" es de tipo "*data*", "*ref*" (*URI*) o "*object*" (*identificador asociado a otro objeto*), y "***type***", en el caso de que "***valuetype=ref***", indica el tipo. "***id***" le identifica para poder ser referenciado desde los scripts definidos en el documento.

Su utilización es exclusiva para la parametrización de objetos, lo que implica que siempre se encontrará encerrada por la etiqueta HTML doble ***<object> </object>***.

```
<object type="application/x-shockwave-flash" data="objetos/archivo-flash.swf" width="400px" height="300px">
    <param name="menu" value="false" />
    <param name="quality" value="high" />
    <param name="loop" value="false" />
</object>
```

— **<embed *src type quality autostart loop hidden width height pluginspage*/>**: Etiqueta equivalente a ***<object></object>*** nos permite embeber objetos en nuestros docu-

mentos HTML/XHTML. Mediante "***src***" indicamos la ruta relativa del objeto a insertar, "***type***" indica el tipo de objeto, "***quality***" la calidad de la reproducción, "***autostart***" indica si se autoreproducirá al cargarse la página, "***loop***" si esta reproducción se repetirá al terminar, "***hidden***" por si se quiere ocultar, "***width***" y "***height***" para ajustar sus dimensiones, y "***pluginspage***" para especificar la dirección Web donde el navegador podrá encontrar los complementos necesarios para su reproducción en caso de que el tipo de objeto no sea reconocido. Aunque es una etiqueta muy extendida y reconocida por la mayor parte de los navegadores Web, presenta el problema de que no es reconocida por el W3C, lo que nos imposibilita la validación del documento por parte de esta organización.

```
<embed type="application/x-shockwave-flash" src="objetos/objeto-flash.swf" quality=
"high" width="500px" height="450px" pluginspage="URI(adobe-flash)" loop="false" />
```

¡¡Ayuda!! Para la generación de objetos flash (*.swf) cabría destacar dos posibles alternativas:

1) El paquete software Adobe Flash nos permite generar de cero una animación *swf* con la máxima personalización. Ejemplos de ello se mostrarán en el capítulo X.

2) Generar una animación *swf* a partir de un archivo en otro formato. Para ello existe una herramienta software portable disponible tanto para entornos Microsoft Windows como GNU/Linux denominfición en formato "wav" o una presentación en formato "pdf". Ejemplos de ello serían los siguientes:

```
gif2swf img1.gif [img2.gif ...] [-o objeto.swf -X anchura-pixels -Y altura-pixels -r frames/segundo]
jpeg2swf img1.jpg [img2.jpg ...] [-o objeto.swf -X anchura-pixels -Y altura-pixels -r frames/segundo -j calidad]
png2swf img1.png [img2.png ...] [-o objeto.swf -X anchura-pixels -Y altura-pixels -r frames/segundo -q calidad]
wav2swf cancion.wav [-o objeto.swf -s muestras/segundo -r frames/segundo]
pdf2swf fichero.pdf [-o objeto.swf -p rango-paginas -j calidad]
avi2swf pelicula.avi [-o objeto.swf -n numero-frames-covertir -m mp3-bitrate -r muestras-mp3/segundo -q calidad -s escala]

[usuario@equipo]$ jpeg2swf|gif2swf|png2swf img1.jpg|gif|png img2.jpg|gif|png ... -o animacion.swf -X 500 -Y 400 -r 1
[usuario@equipo]$ wav2swf cancion.wav -o cancion.swf -s 22050
[usuario@equipo]$ pdf2swf presentacion.pdf -o presentacion.swf -j 90
[usuario@equipo]$ avi2swf pelicula.avi -o pelicula.swf -r 44100 -q 50 -s 80
```

En el caso de querer convertir un archivo de música en formato *mp3* a *swf* será necesaria una conversión intermedia a *wav*. De manera similar ocurre si queremos convertir un archivo de video en formato *mov*, *flv*, *mpeg*, ... a *swf*, al ser necesaria una conversión intermedia a formato *avi*. Para ello es posible hacer uso del software "ffmpeg", el cual permite hacer hacer conversiones entre diferentes formatos de audio y video (*http://www.ffmpeg.org*).

```
[usuario@equipo]$ ffmpeg -i cancion.mp3 cancion.wav
[usuario@equipo]$ ffmpeg -i pelicula.mpg pelicula.avi
```

Ejercicio práctico n.º 1

Diseñar un sitio Web relacionado con la temática que desees, cuya página principal "ejercicio1.html", presente las siguientes características:

a) El documento presentará un color de fondo verde claro (*background-color: GreenYellow;*)[6], y dos centímetros de margen lateral e inferior, y un centímetro en su margen superior (*margin: 2cm; margin-top: 1cm;*).

b) Mostrará un título <*h1*></*h1*> centrado (*margin-left: auto; margin-right: auto;*) haciendo alusión a la temática de la Web. Presentará un borde de color gris en línea continua con 2px de grosor (*border: gray solid 2px;*), un color de fondo blanco (*background-color: white;*), un color de letra rojo tomate (*color: tomato;*), 10px de distancia entre el borde del título y su contenido (*padding: 10px;*), con el texto centrado (*text-align: center;*), ocupando un ancho del 70% del área de contenidos del navegador (*width: 70%;*).

c) Tras el título, el documento presentará un índice de contenidos vinculado con cada una de las partes del documento <*a href="#marca"*></*a*>, donde cada uno de los puntos de éste índice se corresponde con los elementos de una lista HTML ordenada <*ol*></*ol*>. Al pinchar sobre cualquiera de sus elementos, se enlazará con la zona de la página Web donde se informa del punto en cuestión. Los elementos de la lista <*li*> </*li*> se mostrarán en negrita (*font-weight: 700;*) y color azul (*color: blue;*), eliminando el subrayado implícito que presenta todo enlace (*text-decoration: none;*). Para ello será necesario poner marcas al comienzo de cada una de las secciones del documento <*a name="marca"*></*a*>.

d) Junto al índice, a su derecha (*float: right;*), se mostrará una animación flash formada por un conjunto de imágenes relacionadas con la temática del sitio Web. Para generar este objeto, haremos uso de alguna de las aplicaciones "*jpeg2swf|gif2swf|png2swf*" del paquete software *swftools*, haciendo uso de la sintaxis mostrada en la ayuda anterior.

e) A continuación del índice se desglosará cada uno de sus puntos. Mediante una nueva lista ordenada se hará referencia a ellos, acompañados de un párrafo de texto explicativo <*p*></*p*>. Este texto presentará el texto justificado (*text-align: justify;*), en color rojo (*color: red;*), con su primera línea tabulada 1.5cm (*text-indent: 1.5cm;*). Tras este párrafo, una lista desordenada <*ul*></*ul*> resumirá las características más importantes del texto informativo, junto con un enlace de retorno al índice de contenidos <*a href="#marca-inicio"*></*a*>, a una página externa relacionada con el tema <*a href="URI"*></*a*>, a Google <*a href="www.google.com"*></*a*>, y a una dirección email para poder preguntar al autor cualquier duda <*a href="mailto:email"*></*a*>.

f) Se definirá un icono identificativo para el sitio Web <*link rel="shortcut icon"* /> que será mostrado en la barra de direcciones y pestañas del navegador, además de un titulo <*title*></*title*> que será visible en la barra del título.

g) Como música de fondo haremos que se escuche nuestra canción preferida. Para ello, deberemos convertir previamente dicha canción en un objeto flash reproducible por nuestro navegador mediante la aplicación "*wav2swf*" integrada en el paquete software *swftools*, haciendo uso de la sintaxis mostrada en la ayuda anterior.

h) Todos los estilos anteriores podrán asignarse a cada uno de los elementos que forman parte del documento HTML, bien mediante su atributo *style* (*style="propiedad-estilo:valor;"*) o definiéndolos en la sección <*style type="text/css"*></*style*> del documen-

6 La lista de todos los colores disponibles puede consultarse en "http://www.w3schools.com/tags/ref_colornames.asp".

to, desde la cual se hará referencia a ellos a través del valor del atributo *class* que se les haya asignado.

Figura 1.15. Aspecto de la página de inicio del sitio Web del ejercicio práctico n.º 1

Solución ejercicio n.º 1

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
<!— En la sección <head></head> definimos un título, un icono, una lista de estilos CSS y una música de fondo —>
<head><title>Rutas BTT España - Ejercicio 1</title>
<meta http-equiv="Content-Type" content="text/html;charset=UTF-8" />
<link href='logos/icono.png' rel='shortcut icon' type='image/png'/>
<style type="text/css">
<!— Las propiedades de estilo se asignan a los diferentes elementos HTML del documento haciendo referencia al elemento en cuestión mediante su nombre, o a través del valor de su atributo "class". —>
body { background-color: GreenYellow; margin: 2cm; margin-top: 1cm; }
h1.titulo { border: gray solid 2px; background-color: white; padding: 10px; text-align: center; color: tomato; width: 70%; margin-left: auto; margin-right: auto; }
object.animacion1 { float: right; margin-left: 0.8cm; width: 300px; height: 280px; }
li.lista1 { color: Blue; font-weight: 700; }
li.lista2 { font-style: italic; font-weight: 600; color: blue; }
a.enlace1 { text-decoration: none; color: Blue; }
p { text-align: justify; text-indent: 1.5cm; }
</style>
<!— Insertamos el objeto flash cancion.swf ubicado en la subcarpeta musica para que se escuche música de fondo. —>
<object type="application/x-shockwave-flash" data="musica/cancion.swf"></object>
</head>
<body>
```

```
<!— Creamos una marca al comienzo del documento para poder regresar a este punto
    desde cualquier parte del documento mediante un enlace <a href="#inicio"></a>. —>
<a name="inicio"></a><h1 class="titulo">GRANDES RUTAS BTT POR ESPAÑA</h1>
<!— Insertamos el objeto flash rutas.swf ubicado en la subcarpeta imágenes flotando a
    la derecha del documento. —>
<object type="application/x-shockwave-flash" data="imagenes/rutas.swf" class="anima-
    cion1"></object>
<!— Creamos una lista formada por enlaces HTML a distintas zonas del documento
    mediante marcas. —>
<ol>
<li class="lista1"><a href="#frances" class="enlace1">Camino de Santiago Francés.</a></li>
<li class="lista1"><a href="#norte" class="enlace1">Camino de Santiago del Norte.</a></li>
<li class="lista1"><a href="#plata" class="enlace1">Via de la Plata.</a></li>
<li class="lista1"><a href="#cid" class="enlace1">Ruta del Cid.</a></li>
... <!— Resto de elementos de la lista. —>
</ol>
<hr />
<h3 style="font-family: sans-serif; color: brown; text-decoration: underline;">Rutas BTT y sus
    características:</h3>
<!— Una nueva lista desglosará cada uno de los puntos del índice anterior. —>
<ol type="upper-roman">
<!— Cada elemento de la lista estará acompañado de una marca para poder acceder
    desde los enlaces del indice. —>
<a name="frances"></a><li class="lista2">Camino de Santiago Francés.</li>
<p style="color:red;">En España se inicia en los puertos de Somport (vía tolosana) o de
    Roncesvalles (Navarra). Los viajeros se dirigen a Puente la Reina (Navarra), pasando, en
    el primer caso, por Jaca (Huesca), Sangüesa (Navarra) y Monreal (Navarra); y por
    Pamplona, en el segundo ...<!— Resto del párrafo. —></p>
    <ul type="square" style="color: red;">
    <li><span style="font-weight: 800;">Origen y destino:</span> Roncesvalles a Santiago
    Compostela.</li>
    <li><span style="font-weight: 800;">Distancia:</span> 750Km.</li>
    <li><span style="font-weight: 800;">Descripción:</span> Espectacular, Inolvidable y
    Gente entrañable.
    </ul>
<br />
<!— Al final de cada elemento de la lista, habrá un enlace para poder regresar al índice
    de contenidos, haciendo referencia a la marca creada al comiendo del cuerpo del
    documento, junto con enlaces a una página externa relacionada con el tema tratado,
    a Google, y la cuenta email del autor para poder preguntar dudas. —>
<a href="#inicio" style="margin-right: auto; margin-left: auto;">Volver Indice</a>
<a href="http://www.escarbapedal.com#caminofrances">Escarbapedal - Camino Frances</a>
<a href="http://www.google.com">Google</a>
<a  href="mailto:amartinr@educa.aragon.es,juanjo@cossio.net?subject=Pregunta relacionada
    con rutas BTT por España&body=Escribe aquí tu pregunta...">Contactar por Email</a>
<hr />
<!— Siguiendo la misma estructura anterior se crearán el resto de elementos de la lista.—>
<a name="norte"></a><li class="lista2">Camino de Santiago del Norte.</li>
<p style="color:red;">Otra de las rutas existentes para peregrinar a Santiago es ...<!— Resto
    del párrafo. —></p>
    <ul type="square" style="color: red;">
    <li><span style="font-weight: 800;">Origen y destino:</span> Fuenterrabia (Hondarribia) a
    Santiago Compostela.</li>
```

```
        <li><span style="font-weight: 800;">Distancia:</span> 500Km.</li>
        <li><span style="font-weight: 800;">Descripción:</span> Montañas legendarias, paisajes
        de postal, olor a mar.
        </ul>
<a href="#inicio" style="margin-right: auto; margin-left: auto;">Volver Indice</a>
<a href="http://www.escarbapedal.com#caminonorte">Escarbapedal - Camino Frances</a>
... <!— Resto de enlaces a Google y a la dirección email. —>
<hr />
<a name="plata"><li class="lista2">Via de la Plata.</li>
    ... <!— Resto de contenidos del documento, haciendo uso de la estructura anterior. —>
</body>
</html>
```

4.5. Etiquetas HTML para la creación de tablas

Con la finalidad de organizar de una manera sencilla la información que se presenta en un documento HTML/XHTML, el lenguaje de marcado HTML dispone de un conjunto de etiquetas útiles para la creación de tablas. Estas son utilizadas tanto para la presentación de datos de manera organizada en el navegador, como para organizar los distintos elementos que componen el documento.

ETIQUETAS HTML PARA LA FORMACIÓN DE TABLAS	
<table ***width border frames rules cellpadding cellspacing id class style lang|xml:lang title*****>** **</table>** **<caption** ***id class style lang|xml:lang title*****>** **</caption>** **<thead|tbody|tfoot** ***align=left|center|right|justify valign=top|middle|bottom|baseline id class style lang|xml:lang title*****></thead|tbody|tfoot>** **<tr** ***align=left|center|right|justify valign=top|middle|bottom|baseline id class style lang|xml:lang title*****></tr>** **<th|td** ***align=left|center|right|justify valign=top|middle|bottom|baseline id class style lang|xml:lang title*** colspan rowspan**></th|td>**	<html><head>...</head><body> <table> <caption>Titulo de la tabla HTML</caption> <tr><th>PRIMERA COLUMNA – 1ª FILA</th> <th>SEGUNDA COLUMNA – 1ª FILA</th> <th>...</th></tr> <tr><th>PRIMERA COLUMNA – 2ª FILA</th> <th>SEGUNDA COLUMNA – 2ª FILA</th> <th>...</th></tr> ... </table> </body></html>

— **<table** ***width border frames rules cellpadding cellspacing id class style lang|xml:lang title*****></table>**: Inserta una tabla en un documento HTML/XHTML. Mediante su atributo "***width***" establecemos el ancho de la tabla, "***border***" le asigna un borde personalizado, "***frames***" especifica que lados de la tabla serán visibles (frames="void | above | below | hsides | vsides | lhs | rhs | box | border"), "***rules***" establece que líneas o reglas se mostrarán entre las celdas de una tabla (rules="none | groups | rows | cols | all"), "cellpadding" y "cellspacing" determinan la distancia entre el contenido de las celdas de la tabla y su borde de celda, y entre las celdas y el borde de la tabla respectivamente, "***id***" nos permite identificarla para que pueda ser referenciada desde los scripts definidos en el documento (<*script*> </*script*>), "***class***" la clasifica para poder asig-

narle alguno de los estilos definidos en alguna de las hojas de estilo CSS que tenga asociadas el documento, "***style***" nos permite asignar una lista de propiedades de estilo CSS separadas estas por ";" y "***title***" le asigna una breve descripción que será visualiza al situar el ratón sobre el texto.

El resto de etiquetas encargadas de definir la estructura de la tabla quedarán encerradas por estas.

— **<caption *id class style lang|xml:lang title*></caption>**: Establece un título para la tabla. Por defecto, este se mostrará encima de la tabla, centrado. Sus atributos tienen el mismo efecto que en el resto de etiquetas HTML.

— **<thead|tbody|tfoot *align valign id class style lang|xml:lang title*></thead|tbody|tfoot>**: Estas etiquetas permiten definir el conjunto de etiquetas que formarán parte de la cabecera, cuerpo y pie de la tabla. Mediante sus atributo "align" y "valign" indicamos el alineamiento horizontal y vertical deseado del contenido de las celdas (***align=left|center|right|justify***, ***valign=top|middle|bottom|baseline***). El resto de sus atributos tienen el mismo efecto que en el resto de etiquetas.

— **<tr *align valign id class style lang|xml:lang title*></tr>**: Define una fila en una tabla. Será necesario insertar esta etiqueta HTML tantas veces como filas deseamos que contenga la tabla. Mediante sus atributo "align" y "valign" indicamos el alineamiento horizontal y vertical deseado del contenido de las celdas que formarán parte de cada una de estas filas (***align=left|center|right|justify***, ***valign=top|middle|bottom|baseline***). El resto de sus atributos tienen el mismo efecto que en el resto de etiquetas.

— **<th|td** colspan rowspan ***align valign id class style lang|xml:lang title*></th|td>**: Etiquetas utilizadas para definir columnas dentro de cada una de las filas de la tabla. La diferencia entre ambas, es que <th></th> se suele utilizar en la declaración de las columnas de la primera fila, poniendo el texto que encierra en negrita y de manera centrada dentro de su celda. Sus atributos "colspan" y "rowspan" nos permiten agrupar celdas de una tabla, por columnas o filas respectivamente, "align" y "valign" asignan un alineamiento horizontal y vertical al contenido de las celdas (***align=left|center|right|justify***, ***valign=top|middle|bottom|baseline***), y el resto de sus atributos tienen el mismo efecto que en cualquier otra etiqueta HTML ya explicada anteriormente.

Ejemplo tabla HTML

A continuación se muestra un ejemplo de tabla correspondiente a la nueva estructuración del ciclo formativo de grado medio de explotación de sistemas informáticos (ESI) que entrará en vigor supuestamente a partir del curso 2009-2010.

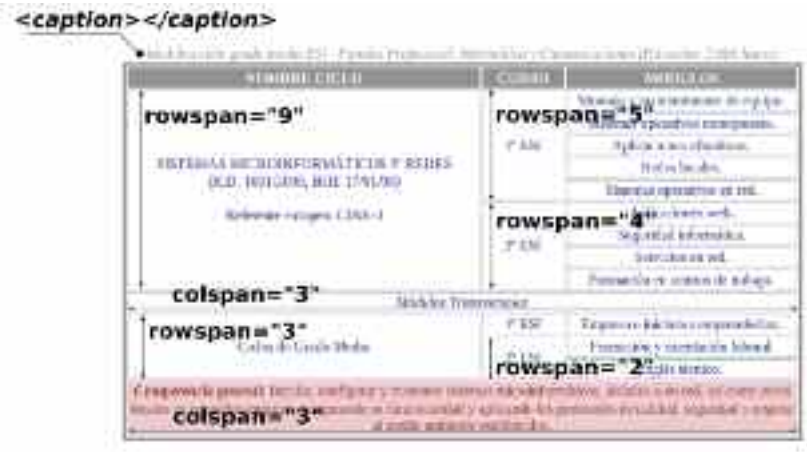


Figura 1.16. Tabla HTML de ejemplo

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html><head><meta http-equiv="Content-Type" content="text/html;charset=UTF-8" />
<style> td {text-align: center; color: blue; } </style>
</head>
<body>
<table width="60%" border="1" style="border: gray solid 4px; ">
<caption style="color: gray; font-style: italic;">Modulos ciclo grado medio ESI - Familia Profesional: Informática y Comunicaciones (Duración:  2.000 horas)</caption>
<tr style="background-color: gray; color: white;"><th> NOMBRE CICLO </th><th> CURSO </th><th> MODULOS </th></tr>
<tr><td rowspan="9">SISTEMAS MICROINFORMÁTICOS Y REDES<br />(R.D. 1691/2008, BOE 17/01/08)<br /><br />Referente europeo: CINE-3</td><td rowspan="5"> 1º ESI </td><td>Montaje y mantenimiento de equipo.</td></tr>
<tr><td>Sistemas operativos monopuesto.</td></tr>
<tr><td>Aplicaciones ofimáticas.</td></tr>
<tr><td>Redes locales.</td></tr>
<tr><td>Sistemas operativos en red.</td></tr>
<tr><td rowspan="4"> 2º ESI </td><td>Aplicaciones web.</td></tr>
<tr><td>Seguridad informática.</td></tr>
<tr><td>Servicios en red.</td></tr>
<tr><td>Formación en centros de trabajo.</td></tr>
<tr><td colspan="3" style="text-align: center; font-style: italic;">Módulos Transversales </td></tr>
<tr><td rowspan="3">Ciclos de Grado Medio</td><td rowspan="1"> 1º ESI </td><td>Empresa e iniciativa emprendedora.</td></tr>
<tr><td rowspan="2">2º ESI</td><td>Formación y orientación laboral.</td></tr>
<tr><td>Inglés técnico.</td></tr>
<tr><td colspan="3" style="background-color: pink; color: brown;">
<b>Competencia general:</b> Instalar, configurar y mantener sistemas microinformáticos, aislados o en red, así como redes locales en pequeños entornos, asegurando su funcionalidad y aplicando los protocolos de calidad, seguridad y respeto al medio ambiente establecidos.
</td></tr>
</table>
</body></html>
```

4.6. Etiquetas HTML utilizadas en la organización de contenidos

Aunque la forma más socorrida de estructurar los contenidos que forman parte de un documento HTML/XHTML es mediante el uso de tablas, el lenguaje de marcado HTML dispone de un conjunto de etiquetas que permiten organizarlos de una manera sencilla.

ETIQUETAS HTML ENCARGADAS DE DAR ESTILO A LA PÁGINA

<frameset *rows cols id class style title*>
    <frame *src name noresize scrolling="auto|yes|no" frameborder="0|1" marginwidth marginheight id class style title*> </frame>
    <frame *src name noresize scrolling="auto|yes|no" frameborder="0|1" marginwidth marginheight id class style title*> </frame>
...
</frameset>

<div *id class style lang|xml:lang title*>

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>
<title> EJEMPLO DE PAGINA DIVIDIDA EN MARCOS </title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link href='../fotos/icono13.png' rel='shortcut icon' type='image/x-icon' /></head>
<frameset rows="15%,85%">
    <frame src="arriba.html" noresize name="arriba">
    <frameset cols="20%,80%">
        <frame src="izquierda.html" noresize name="izquierda">
        <frame src="derecha.html" name="derecha">
    </frameset>
</frameset>
</html>
```

— <**frameset** ***rows cols id class style title***></**frameset**>: Divide el área reservada por el navegador para mostrar los contenidos de la página Web en regiones rectangulares, marcos o frames. Sus atributos "***rows***" y "***cols***" permiten decidir en cuantas filas y columnas se establecerá esta división, y su tamaño, "***id***" nos permite identificarlo para que pueda ser referenciado desde los scripts definidos en el documento (<*script*> </*script*>), "***class***" lo clasifica para poder asignarle alguno de los estilos definidos en alguna de las hojas de estilo CSS que tenga asociadas el documento, "***style***" nos permite asignar una lista de propiedades de estilo CSS separadas estas por ";" y "***title***" le asigna una breve descripción que será visualiza al situar el ratón sobre él.

— <**frame** ***src name noresize scrolling frameborder marginwidth marginheight id class style title***></**frame**>: Define un marco dentro de la pagina Web declarado previamente a través de la etiqueta ***<frameset></frameset>***. Su atributo "***src***" indica la ruta relativa del documento HTML/XHTML que será mostrado en el interior del marco. "***name***" identifica al marco para poder hacer referencia a él mediante el atributo "***target***" en enlaces y formularios (<*a href="URL-pagina-Web" target="name-frame"*></*a*>), "***noresize***" establece que no se va a poderse cambiar el tamaño del marco desde el navegador, "***scrolling***" determina si serán visibles las barras de desplazamiento cuando los contenidos que alberga un marco no quepan en el área visible que tiene asignada (*scrolling="auto|yes|no"*), "***frameborder***" indica si será visible o no el borde del marco (***frameborder="0|1"***), "marginwidth" y "marginheight" asignan los margenes laterales, superior e inferior respectivamente, que afectará a los contenidos del marco, "***id***" nos permite identificarlo para que pueda ser referenciado desde los scripts definidos en el documento (<*script*> </*script*>), "***class***" lo clasifica para poder asignarle alguno de los estilos definidos en alguna de las hojas de estilo CSS que tenga asociadas el documento, "***style***" nos permite asignar una lista de propiedades de estilo CSS separadas estas por ";" y "***title***" le asigna una breve descripción que será visualiza al situar el ratón sobre él.

Ejemplo frames HTML

Una característica importante de este tipo de organización de contenidos es que la página de inicio del sitio Web simplemente sirve para dividir el área de contenidos en marcos, sin cuerpo de contenidos <*body*></*body*>. Los contenidos visualizados serán los asociados a los documentos indicados a través del atributo "***src***" asignado a cada frame.

En el siguiente ejemplo, se divide el documento en tres frames, en dos filas (*rows="15%,85%"*), dividida a su vez la segunda fila en dos columnas (*cols="20%,80%"*).

Figura 1.17. Ejemplo de división de la página en marcos

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>
<title> EJEMPLO DE PAGINA DIVIDIDA EN MARCOS </title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link href='../fotos/icono.png' rel='shortcut icon' type='image/x-icon' /></head>
<frameset rows="15%,85%">
        <frame src="arriba.html" noresize name="arriba">
        <frameset cols="20%,80%">
            <frame src="izquierda.html" noresize name="izquierda">
            <frame src="derecha.html"  name="derecha">
        </frameset>
</frameset>
</html>
```

— <div ***id class style lang|xml:lang title***> : Etiqueta HTML que nos permite dividir el área de contenidos del navegador en regiones, permiéndonos estructurar de una manera muy sencilla sus contenidos. En cuanto a sus atributos, "***id***" nos permite identificar a las distintas divisiones formadas para que puedan ser referenciadas desde los scripts definidos en el documento (<*script*> </*script*>), "***class***" las clasifica para poder asignarle alguno de los estilos definidos en alguna de las hojas de estilo CSS que tenga asociadas el documento, "***style***" nos permite asignar una lista de propiedades de estilo CSS separadas estas por ";", "***lang|xml:lang***" establece el lenguaje utilizado en la redacción del texto que alberga, y "***title***" le asigna una breve descripción que será visualiza al situar el ratón sobre la división establecida.

Ejemplo divisiones HTML

A continuación se mostrará como estructurar los contenidos de una página Web de manera similar a como se a hecho en el ejemplo anterior mediante frames.

*Figura 1.18. Ejemplo de división de la página mediante la etiqueta HTML *

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head><title> EJEMPLO DE PAGINA DIVIDIDA EN DIVISIONES </title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link href='../fotos/icono.png' rel='shortcut icon' type='image/x-icon' />
</head>
<body>
<div style="width: 100%; height: 15%; text-align: center; background-color: pink;">
        <span style="font-size: 1.8cm; color: blue;">DIVISION Nº1</span>
</div>
<div style="width: 80%; height: 85%; text-align: center; background-color: #CC99FF; float:
right;">
        <span style="font-size: 1.8cm; color: white;"><br /><br />DIVISION Nº3</span>
</div>
<div style="width: 20%; height: 85%; text-align: center; background-color: blue;">
        <span style="font-size: 1.8cm; color: white;"><br /><br />DIV<br />Nº2</span>
</div>
</body></html>
```

Ejercicio práctico n.º 2

Diseñar la página de inicio de un sitio Web (*ejercicio2.xhtml*) relacionado con la temática que desees presentando las siguientes características:

a) Una tabla HTML se encargara de organizar los contenidos de la página Web, tal como se muestra en la siguiente figura. A modo de ejemplo, estará formada por dos columnas y nueve filas. En el capítulo siguiente, veremos como hacer lo mismo mediante el uso de divisiones, * *.

Figura 1.19. Una tabla HTML se encargará de organizar los contenidos del sitio Web

b) La fila superior de la tabla, con sus dos columnas agrupadas (*colspan="2"*), contendrá la cabecera del documento compuesta por el logotipo del sitio Web, **. Presentará un ancho del 100%, y un alto de 150 pixels del área reservada por el navegador para los contenidos, con la imagen centrada y un color de fondo naranja (*width: 100%; height: 150px; text-align: center; background: orange;*). Al situar el ratón sobre la imagen deberá mostrarse un texto informativo (*title="texto informativo"*).

c) Bajo la fila anterior, la columna de la izquierda, con un ancho del 15% (*width: 15%;*), desempeñará la función de menú del sitio Web, donde cada una de sus filas se corresponderá con un ítem del menú o enlace a las distintas páginas que forman parte del sitio Web, *ítem*. Para destacar a estos ítem y que no se vean muy abultados, se pondrán en negrita, centrados en la celda, y con una distancia entre estos y el borde de la celda superior e inferior de 10 pixels (*font-weight: bold; text-align: center; padding-top: 10px; padding-bottom: 10px;*), diferenciandose entre sí los ítem pares de los impares al presentar unos un color de letra y fondo, blanco y verde respectivamente (*color: white; background-color: green;*), y los otros, azul y naranja (*color: blue; background-color: orange;*). Además se deberá eliminar el subrayado que de manera implícita se produce al declarar un enlace (*text-decoration: none;*). La última celda o fila de esta columna quedará vacía, con un color de fondo gris y un alto que se ajustará automáticamente en función de las dimensiones del área de contenidos de la página Web (*background-color: gray; height: auto;*).

d) La columna de la derecha, con todas sus filas agrupadas (*rowspan="8"*) y un ancho del 85% (*width: 85%;*), albergará los contenidos de la página Web. Presentará un color de fondo amarillo (*background-color: yellow;*) y un altura que se ajustará en función de la cantidad de contenidos que contenga (*height:auto;*). Para que los párrafos de texto se puedan leer de una manera clara, y no queden muy pegados al borde de la celda de la tabla, se establecerá unos margenes laterales de 30 pixels y superior de 20 pixels (*margin-left: 30px; margin-top: 20px; margin-right: 30px;*). Este texto se mostrará justificado, con su primera línea sangrada 1 cm (*text-align: justify; text-indent: 1cm;*). En cuanto a las imágenes se mostrarán centradas (*text-align: center;*), con un borde de 1pixel de grosor y color azul (*border: 1px solid blue;*), con una separación entre este borde y la imagen de 5px con fondo azul claro, que hará las veces de marco (*padding: 5px; background-color: SkyBlue;*).

e) Todos los estilos anteriores se definirán en la sección *<style type="text/css"></style>* del documento, desde la cual se hará referencia a cada uno de los elementos que forman parte del documento XHTML a través del valor del atributo *class* que se les haya asignado.

f) Mediante las etiquetas *<meta />* se aportará una información que podría ser utilizada por los buscadores en el caso de que la página estuviera colgada en Internet, como puede ser el autor, y un conjunto de palabras clave que den información sobre los contenidos del sitio Web.

Figura 1.20. Aspecto de la página Web de inicio del ejercicio práctico n.º 2

Solución ejercicio n.º 2

```
<!— Las dos primeras líneas indican el juego de caracteres utilizado y el tipo de documento XHTML. —>
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Ejercicio 2 - Web Escarbapedal</title>
    <meta name="AUTHOR" content="Arturo y Juanjo Martín Romero" />
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <meta name="KEYWORDS" content="escarbapedal, BTT, montaña, senderismo, setas" />
</head>
<!— En la sección "style" se definen todos los estilos que se asignan a los elementos del documento. —>
<style type="text/css">
    td.cabecera { background: orange; width: 100%; height: 150px; text-align: center; }
```

```
<!— La altura ”height: 5%;” del enlace es orientativa, ya que esta la determina el tamaño de la letra y el padding. —>
    td.enlace { font-weight: bold; padding-top: 10px; padding-bottom: 10px; text-align: center; width: 15%; height: 5%;}
    .tipo1 { background-color: green; color: white; }
    .tipo2 { background-color: orange; color: blue; }
    a { text-decoration: none; }
    td.ultimo { background-color: gray; height: auto;}
    td.contenidos { width: 85%; height:auto; background-color: yellow;}
    p.texto{ margin-left: 30px; margin-top: 20px; margin-right: 30px; text-align: justify; text-indent: 1cm; }
    p.imagen { text-align: center; }
    p.imagen img { border: 1px solid blue; background-color: SkyBlue; padding: 5px; }
</style>
<link href='fotos/icono.png' rel='shortcut icon' type='image/x-icon'>
<body>
<!— Una tabla se encarga de estructurar todos los contenidos de la página Web. —>
<table align=”center” border=”1” cellpadding=”0” cellspacing=”0” width=”100%”>
<!— La primera fila de la tabla contiene el logotipo de la Web —>
<tr><td class=”cabecera” colspan=”2”>
    <img src=”fotos/espalda.gif” width=”400px” height=”150px” title=”Logotipo club ESCARBAPEDAL” />
</td></tr>
<!— Bajo la fila anterior, la columna de la izquierda estará compuesta por enlaces a otras páginas Web. —>
<tr>
<!— El primer enlace del menú es de retorno a la página de inicio del sitio Web. —>
<td class=”enlace tipo1”><a href=”ejercicio2.html” class=”tipo1”>INICIO</a></td>
<!— La columna de la derecha alberga los contenidos de la página Web: texto e imágenes. —>
<td class=”contenidos” rowspan=”8”>
    <p class=”texto”>El club <b>ESCARBAPEDAL</b> esta compuesto por un grupo de aficionados a las actividades de montaña. Entre estas actividades cabría señalar las rutas en bicicleta BTT, aunque también se desarrollan otras actividades como pueden ser senderismo, barranquismo, etc.</p>
    <p class=”imagen”><img src=”fotos/miembros.jpg” width=”400px” height=”300px” />
    <br />Club Escarbapedal. Beceite.</p>
</td>
</tr>
    <tr><td class=”enlace tipo2”><a href=”rutas.html” class=”tipo2”>RUTAS BTT</a></td></tr>
    <tr><td class=”enlace tipo1”><a href=”senderismo.html” class=”tipo1”>SENDERISMO</a></td></tr>
    <tr><td class=”enlace tipo2”><a href=”barrancos.html” class=”tipo2”>BARRANCOS</a></td></tr>
    <tr><td class=”enlace tipo1”><a href=”setas.html” class=”tipo1”>SETAS</a></td></tr>
    <tr><td class=”enlace tipo2”><a href=”alpinismo.html” class=”tipo2”>ALPINISMO</a></td></tr>
    <tr><td class=”enlace tipo1”><a href=”otros.html” class=”tipo1”>OTROS</a></td></tr>
    <tr><td class=”ultimo”></td>
</tr></table></body></html>
```

4.7. Etiquetas HTML para la creación de formularios

Un formulario HTML es la opción más básica utilizada en el diseño Web para poder establecer una comunciación entre el cliente y el servidor HTTP. Normalmente son utilizados para solicitar información al usuario que esta visualizando los contenidos Web desde su navegador, y en base a ella, el servidor realizar una tarea encomendada: personalizar los contenidos Web en base a la información introducida, comprobar o insertar datos en una base de datos, etc. Estos formularios son utilizados en sitios Web tan extendidos como Google, para que el usuario pueda indicar las palabras clave de busqueda de información, en el registro e ingreso de usuarios a multitud de servicios Web (correo WebMail, albumes de fotos Web, etc.), o para la introducción de textos de dialogo en chats, entre otros muchos ejemplos. Teniendo en cuenta que el presente capítulo no requiere de ningún servidor Web, al centrarse únicamente en el desarrollo de documentos Web HTML/XHTML que son interpretados mediante un simple cliente Web, posponemos la introducción de estos formularios a la segunda parte del libro, donde se muestra cómo instalar y configurar un servidor Web.

Capítulo 2

HOJAS DE ESTILO CSS

Las hojas de estilo en cascada (CSS, Cascading Style Sheets) son la herramienta utilizada en la actualidad por los desarrolladores de sitios Web a la hora de controlar el aspecto y presentación de los contenidos de un documento HTML. Como vimos en el capítulo anterior, las propiedades de estilo asociadas a cada elemento del documento HTML pueden asignarseles haciendo uso de los correspondientes atributos HTML que tenga definidos, y aunque ésta es una opción correcta, no se recomienda al presentar dos problemas destacables:

1) A medida que aumentan el número de atributos en los elementos HTML que componen el documento HTML, hacen que éste sea más ilegible, pudiéndose convertir en un documento demasiado enfarragoso.

2) Comúnmente suele darse el caso de que los atributos y propiedades de estilo asignadas a un elemento HTML del documento suelen aplicarse a otros elementos del mismo documento, traduciéndose en una excesiva redundancia dentro del documento.

Con la finalidad de solventar los dos problemas anteriores surgen las hojas de estilo CSS. Éstas nos van a permitir independizar los contenidos que forman parte de un sitio Web, con la presentación de estos en pantalla (*screen*) o en su impresión (*print*). Es decir, los documentos HTML se deberían limitar a indicar de que elementos esta compuesta la página Web a la que hace referencia, y por otro lado, sus hojas de estilo asociadas nos informan del aspecto que presentarán. Los estilos definidos en CSS estarán asociados a los elementos del documento HTML, permitiéndonos reutilizar su declaración, por varios elementos HTML simultáneamente, o incluso por varios documentos HTML, evitando redundancias en la información, y dejando un aspecto más modular y legible, dando solución a los dos problemas anteriores.

Figura 2.1. En la estructura de un sitio Web actual están separados los documentos relacionados con el contenido de los encargados de su presentación

En resumen, CSS es la mejor manera de separar los contenidos (HTML/XHTML) de su presentación (CSS), imprescindible hoy en día para la creación de sitios Web de mediana y alta complejidad.

1. CONTENIDOS DEL CAPÍTULO

Los contenidos del presente capítulo pueden dividirse en dos grandes partes:

— El capítulo comienza presentando a CSS como el estándar actual utilizado por los principales navegadores Web en la presentación de los contenidos de documentos Web, indicando las diferentes alternativas que se recomiendan en el estándar a la hora de definir las hojas de estilo CSS asociadas a los documentos HTML/XHTML, destacando las características más importantes de CSS.

— A continuación se presenta, de manera clasificada, la lista de las principales propiedades de estilo existentes en el estándar CSS, que nos permiten controlar la forma en que se presentan los contenidos de un documento HTML/XHTML a través de un navegador Web.

2. CARACTERÍSTICAS DE CSS

En 1996 el organismo "World Wide Web (WWW) Consortium", también llamado "W3C", encargado de crear todos los estándares que están relacionados con la Web, elabora la primera recomendación oficial de CSS, conocida como versión CSS1, con la finalidad de llegar a un consenso entre las diferentes propuestas que se habían realizado a esta ese momento. Dos años después, en 1998, "W3C" presenta la segunda versión de esta recomendación como ampliación y mejora de la anterior, CSS2. Poco a poco, las distintas empresas desarrolladoras de navegadores Web, han ido adoptando estas recomendaciones, acabando en lo que hoy es una herramienta software estándar utilizada en la maquetación y presentación de documentos HTML y XHML. Actualmente se encuentra en construcción lo que será la tercera versión CSS3.

Es estándar CSS establece que las propiedades de estilo a asignar a los distintos elementos que componen el las distintas páginas Web o documentos HTML o XHML de un sitio Web puede hacerse de tres formas:

1. Incluyendo las propiedades de estilo en la misma declaración del elemento HTML haciendo uso del atributo "**style**". Visto ya en el capítulo anterior, dentro de la etiqueta HTML asociada al elemento, se asigna la lista de propiedades de estilo al atributo "style":

```
<div id="division1" style="background-color: pink; border: 2px solid red; padding: 2cm; color: blue;"></div>
```

Esta es la opción menos recomendable de todas, ya que no permite la reutilización de los estilos asignados. Es decir, si las mismas propiedades de estilo, van a ser utilizadas por otra división del documento, o cualquier otro elemento del documento, será necesario volver a declararlo:

```
<div id="division1" style="background-color: pink; border: 2px solid red; padding: 2cm; color: blue;"></div>
<div id="division2" style="background-color: pink; border: 2px solid red; padding: 2cm; color: blue;"></div>
```

2. Haciendo uso de la etiqueta doble HTML, <style type="text/css"></style>, dentro de la cabecera del documento HTML/XHTML, <head></head>. Esta etiqueta nos permite centralizar dentro del propio documento todos los estilos utilizados por los distintos elementos que lo forman. Para asociar los estilos definidos a los elementos HTML, puede hacerse de tres formas:

   a) Haciendo referencia al elemento HTML a través del nombre de la etiqueta HTML. Para ello, indicaremos el nombre de la etiqueta a la que queramos asignar las propiedades de estilo, colocando estas entre llaves separadas por un punto y coma, "**;**", de igual forma que se asignaban en la opción anterior a través atributo "**style**".

```
<style type="text/css">
    etiqueta_HTML1, etiqueta_HTML2, ... { propiedad1: valor; propiedad2: valor; propiedad3:
    valor; ...}
</style>
```

Esta primera posibilidad, nos permite afectar en una sola declaración de estilo a todos aquellos elementos que compartan el mismo nombre de etiqueta:

```
<head>
<style type="text/css">
    div { background-color: pink; border: 2px solid red; padding: 2cm; color: blue; }
</style>
</head>
<body>
    <div id="division1">...</div>
    <div id="division2">...</div>
</body>
```

En el caso de que queramos asignarle el mismo estilo a elementos HTML con distinta nombre de etiqueta, pondremos la lista de elementos afectados, separados por comas:

```
<head>
<style type="text/css">
    div, h2, p { background-color: pink; border: 2px solid red; padding: 2cm; color: blue; }
</style>
</head>
<body>
    <h2>...</h2>
    <div id="division1">...</div>
    <div id="division2">...</div>
    <p id="parrafo1">...</p>
</body>
```

Teniendo en cuenta que el lenguaje CSS, al igual que el lenguaje de etiquetas HTML, no es sensible a los saltos de línea, es posible declarar las propiedades de estilo anteriores en líneas diferentes, haciendo que el documento sea más legible:

```
<style type="text/css">
    div, h2, p {
        background-color: pink;
        border: 2px solid red;
        padding: 2cm;
        color: blue;
    }
</style>
```

b) Otra opción posible para asociar un estilo CSS definido en *<style></style>* a un elemento HTML del documento es haciendo referencia a él a través del valor de su atributo "**class**". Esto nos va a permitir diferenciar entre elementos que tengan la misma etiqueta HTML. Es decir, con la opción anterior, al asociar un estilo a un elemento, todos los elementos del documento con la misma etiqueta HTML adoptaban ese estilo sin posibilidad de diferenciar unos de otros. Ene este caso, para hacer referencia a un elemento en concreto, haremos uso del valor asignado a su atritubo "class" precedido de un punto, o del nombre de su etiqueta seguido de un punto.

```
<head>
<style type="text/css">
    .division1, h2.titulo1 { background-color: pink; border: 2px solid red; padding: 2cm; color:
    blue; }
    div.division2, .parrafo1 { background-color: blue; border: 2px dotted brown; padding: 2cm;
    color: white; }
</style>
</head>
<body>
    <h2 class="titulo1">...</h2>
    <div class="division1">...</div>
    <div class="division2">...</div>
    <p class="parrafo1">...</p>
</body>
```

c) Una última opción existente a la hora de asociar un estilo CSS definido dentro de la sección *<style></style>* a un elemento HTML del documento haciendo referencia al valor asignado a su atributo "**id**". Para ello será necesario preceder al valor de "**id**" de un carácter almohadilla "#". Aunque se trata de una opción similar a la presentada anteriormente, mediante el uso del valor atributo "**class**", esta opción se caracteriza por ser muy útil en la personlización de estilo mediante lenguaje JavaScript como veremos en el siguiente capítulo.

```
<head>
<style type="text/css">
    #division1, h2#titulo1 { background-color: pink; border: 2px solid red; padding: 2cm; color:
    blue; }
    div#division2, #parrafo1 { background-color: blue; border: 2px dotted brown; padding:
    2cm; color: white; }
```

```
</style>
</head>
<body>
    <h2 id="titulo1">...</h2>
    <div id="division1">...</div>
    <div id="division2">...</div>
    <p id="parrafo1">...</p>
</body>
```

Además, teniendo en cuenta que en HTML hay etiquetas dobles que pueden encerrar a otras, podemos hacer referencia a estas últimas, sin necesidad de declarar atributos "id" o "class", a través de la lista de elementos HTML entre los que se encuentran encerradas, separándolos mediante espacios en blanco:

```
<head>
<style type="text/css">
    div#division1 p.parrafo1 h2 { background-color: pink; border: 2px solid red; padding: 2cm;
    color: blue; }
    div.division2 img { background-color: blue; border: 2px dotted brown; padding: 4px; }
</style>
</head>
<body>
    <div id="division1"><p id="parrafo1"><h2>...</h2>...</p>...</div>
    <div class="division2"><img src="imagenes/img1.jpg" />...</div>
</body>
```

Todas estas opciones, englobadas en esta segunda forma recomendada por el W3C para declarar estilos CSS en documentos HTML/XHTML presenta el problema de que estos estilos tan sólo pueden ser utilizados por el propio documento. Es decir, si un sitio Web esta compuesto por múltiples documentos HTML/XHTML, que hacen uso de estilos CSS similares en sus etiquetas HTML, habrá que definirlos de manera independiente en la sección *<style></style>* de cada uno de los documentos por separado, sin permitir su reutilización. Para que los estilos definidos puedan ser utilizados por diferentes documentos simultáneamente, haremos uso de la tercera forma recomendada.

3. En última instancia, el estándar CSS nos permite declarar los estilos en un fichero independiente, pudiendo ser reutilizado por varios documentos simultáneamente. El contenido de este fichero de estilos es el que incluiríamos en la etiqueta doble HTML *<style></style>* en el caso de definirlos en el propio documento. Para poder reconocer el contenido de estos ficheros se hace uso de una extensión específica, "*.**css**". Una vez creado el fichero externo de estilos CSS será necesario importarlo desde aquellos documentos HTML/XHTML que quieran hacer uso de los estilos allí definidos. Para esta importación existen dos opciones posibles:

   a) La etiqueta simple HTML "**<link *atributos* />**", entre otras cosas, nos permite informar al documento HTML cual es la hoja de estilos externa que será utilizada en la presentación de los contenidos del documento por pantalla e impresión. Esta etiqueta deberá colocarse en la cabecera del documento que lo importa, *<head></head>*.Entre sus atributos destacaríamos los siguientes: "**rel**" define la relación existente entre el documento HTML y el que se importa, "**type**" informa

del tipo de contenido, "**media**" indica el medio para el cual el documento importado va destinado, y "**href**" especifica la ruta o ubicación del documento a importar. Esta etiqueta deberá utilizarse tantas veces como hojas de estilo CSS externas queramos importar.

```
<link rel="stylesheet" type="text/css" media="screen|print" href="URI/URL fichero externo *.css"/>
```

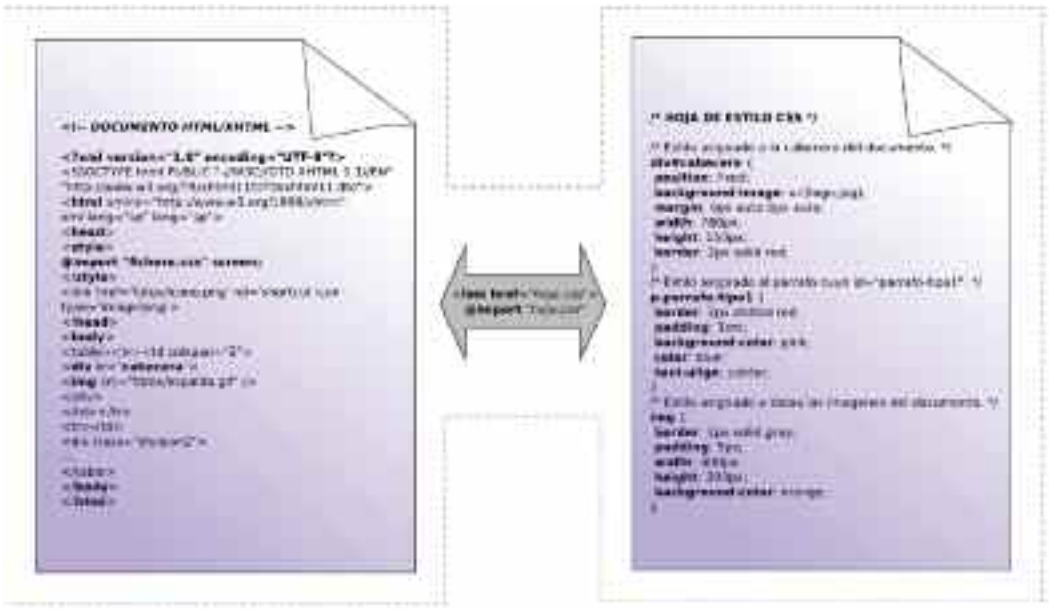


Figura 2.2. La importación de hojas de estilo externas nos permite la reutilización en diferentes documentos HTML/XHTML de los estilos CSS definidos.)

b) La directiva "**@import**" nos permite importar hojas de estilos externas desde un documento HTML/XHML. Esta directiva deberá ubicarse dentro de la etiqueta doble <style></style> seguida de la ruta donde se ubica el fichero a importar utilizando una de las siguientes sintaxis:

```
<style type="text/css">
    @import url("URL fichero externo *.css") screen|print;
    @import "URL fichero externo *.css" screen|print;
</style>
```

Al igual que la anterior etiqueta "**<link />**", la directiva "**@import**" se incluirá tantas veces como hojas de estilo externas sea necesario importar, e incluso pueden combinarse ambas, junto con las declaraciones de los estilos necesarios:

```
<head>
<link rel="stylesheet" type="text/css" media="screen" href="hoja1.css"/>
<link rel="stylesheet" type="text/css" media="screen" href="hoja2.css"/>
<style type="text/css">
```

```
    @import url("hoja3.css") screen;
    @import "hoja4.css" screen;
    .division1, h2.titulo1 { background-color: pink; border: 2px solid red; padding: 2cm; color:
    blue; }
    div.division2, .parrafo1 { background-color: blue; border: 2px dotted brown; padding: 2cm;
    color: white; }
</style>
</head>
```

Por último, indicar que en el caso de que haya estilos asignados a un mismo elemento HTML superpuestos, el navegador adoptará las propiedades de estilo de manera secuencial a como son declaradas, siempre teniendo prioridad las últimas que sean leídas por el navegador. Por ejemplo, si un documento HTML presenta la cabecera anterior, y en alguno de los ficheros importados, "hoja1.css", "hoja2.css", "hoja3.css" y "hoja4.css", se definen propiedades asociadas a un elemento del documento con atributo "class='division1'", las propiedades de color de fondo, borde, relleno y color de fuente establecidas por "**.division1** { background-color: pink; border: 2px solid red; padding: 2cm; color: blue; }", serán prioritarias respecto a cualquier otras. Por tanto, la prioridad de los estilos tan sólo depende del orden en que son definidos. Si cambiáramos el orden de las anteriores declaraciones, ocurriría justamente lo contrario, pasando a ser las más prioritarias las declaraciones de estilo de la hoja de estilos "hoja2.css", le seguirían en prioridad "hoja1.css", "hoja4.css", etc.:

```
<head>
<link rel="stylesheet" type="text/css" media="screen" href="hoja1.css"/>
<link rel="stylesheet" type="text/css" media="screen" href="hoja2.css"/>
<style type="text/css">
    .division1, h2.titulo1 { background-color: pink; border: 2px solid red; padding: 2cm; color:
    blue; }
    div.division2, .parrafo1 { background-color: blue; border: 2px dotted brown; padding: 2cm;
    color: white; }
    @import url("hoja3.css") screen;
    @import "hoja4.css" screen;
</style>
<link rel="stylesheet" type="text/css" media="screen" href="hoja1.css"/>
<link rel="stylesheet" type="text/css" media="screen" href="hoja2.css"/>
</head>
```

En el caso de que queramos que el valor asignado a una determinada propiedad sea prioritaria respecto al resto de posibles valores que pueda asignarsele, se le acompañará de la directiva "**important!**".

```
.division1, h2.titulo1 { background-color: pink important!; border: 2px solid red; padding: 2cm;
color: blue; }
```

Con la finalidad de poder documentar las hojas de estilo CSS y facilitar su posterior mantenimiento, el estándar nos permite introducir comentarios encerrando estos entre los caracteres "***/**" y "***/***": "***/* comentarios CSS */***".

En conclusión, CSS se caracteriza por asegurar una independencia entre los contenidos y su presentación, facilitando el diseño del sitio Web y su posterior mantenimiento.

3. PROPIEDADES DE ESTILO CSS

A continuación veremos las propiedades de estilo más significativas, compatibles con las recomendaciones CSS del W3C (CSS1 o CSS2), y son por las últimas versiones de los navegadores o clientes Web más conocidos, Internet Explorer, Mozilla Firefox, Opera o Netscape.

Estas propiedades CSS pueden ser clasificadas en función de la caracterítica del elemento HTML al que le afectan:

— Propiedades del borde del elemento HTML. Por defecto este borde es transparente. CSS nos permite personalizar de manera independiente el color, grosor, y forma, de cada uno de los bordes del elemento: *border-top*, *border-bottom*, *border-left* y *border-right*.

— Propiedades del margen del elemento HTML en relación a sus límites dentro del documento. Podemos establecer de manera independiente margenes en las cuatro direcciones: *margin-top*, *margin-bottom*, *margin-left*, y *margin-right*.

— Propiedades de relleno, entendiendo este como el área comprendida entre el borde del elemento HTML y su contenido. Estas propiedades CSS nos permiten establecer el grosor del relleno que rodea al área asociada al contenido en los cuatro sentidos: *padding-top*, *padding-bottom*, *padding-left* y *padding-right*.

— Propiedades del fondo del elemento HTML. Nos permiten personalizar tanto su color como la imagen de fondo: *background-color* o *background-image*, entre otras.

— Propiedades de posicionamiento del elemento en el documento. Nos permiten determinar la posición absoluta o relativa de un elemento dentro del área asociada a los contenidos en el navegador: *position*, *top*, *left*, *right*, o *bottom*.

— Propiedades de dimensionamiento del elemento HTML. Nos permiten ajustar su anchura y altura: *width*, *height*.

— Propiedades del texto. Nos permiten ajustar multitud de aspectos del contenido de tipo texto de un elemento HTML como su color, tamaño o separación de caracteres entre otros muchos.

— Propiedades asociadas a listas de elementos. Nos permiten personalizar tanto listas ordenadas como desordenadas.

— Propiedades de visualización de un elemento HTML Deciden si un elemento será visible en la presentación del documento, y de que manera.

— Pseudo-clases y pseudo-elementos. Propiedades especiales que nos permiten añadir efectos especiales a determinados elementos HTML.

A través de dos ejemplos prácticos, se mostrará la forma en que son utilizadas estas propiedades en hojas de estilo CSS, aunque al igual que los contenidos expuestos en el capítulo (lenguaje HTML), CSS es un elemento básico en el diseño de sitios Web, por lo que seguiremos viendo multitud de ejemplos prácticos de su utilización en los capítulos siguientes.

3.1. Propiedades de estilo asociadas a márgenes, rellenos y bordes del elemento HTML

Las propiedades de estilo asociadas a margenes (*margin*), rellenos (*padding*) y bordes (*border*), nos permiten adecentar la presentación del contenido de los elementos HTML del documento. Son aplicables a todo aquel elemento HTML que albergue un contenido, pudiendo ir desde el propio cuerpo del documento (BODY), a divisiones de la página (DIV), tablas (TABLE), párrafos (P), etc. La lista de estas propiedades que nos permiten su control son las siguientes:

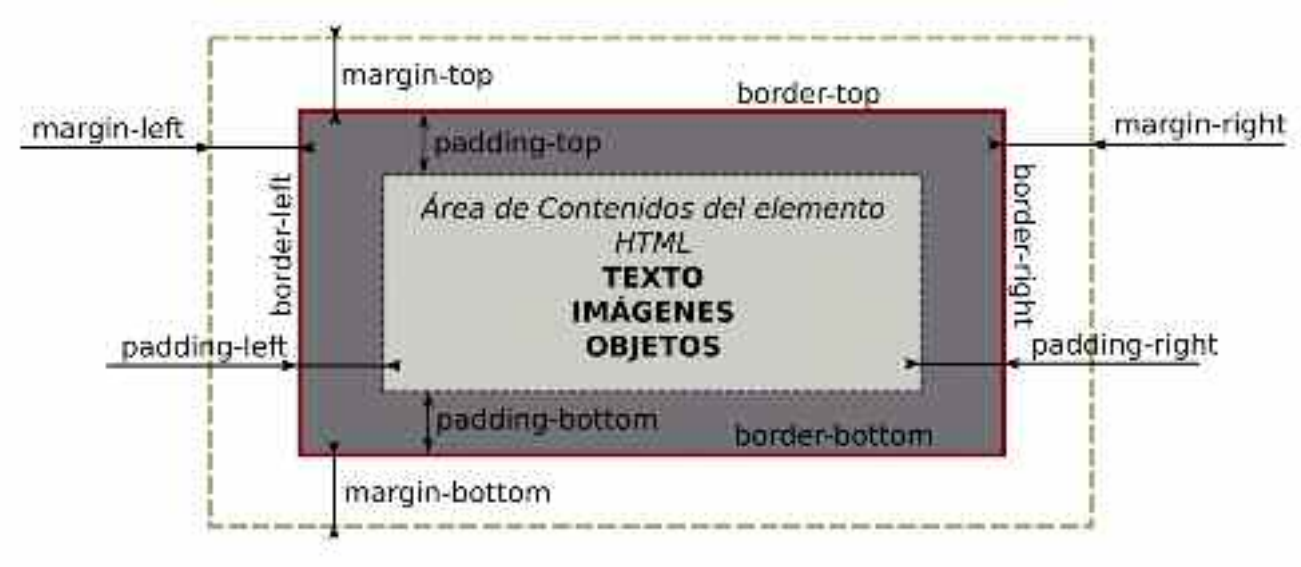


Figura 2.3. Propiedades de estilo asociadas al borde, márgenes y relleno

PROPIEDAD	DESCRIPCIÓN	VALUES
border-width	Determina el grosor del borde	thin | medium | thick *length* (*n*pt|*n*in|*n*cm|*n*px)
border-style	Aplica un estilo para el borde entre un conjunto de estilos definidos	none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset
border-color	Aplica un color al borde. Este color puede especificarse en formato RGB o mediante un nombre normalizado de color. Estos valores se muestran al final de este apartado	*color*
border border-top border-bottom border-left border-right	Propiedad que engloba en una las tres propiedades anteriores	*border-width* *border-style* *border-color*
border-top-width border-bottom-width border-left-width border-right-width	Equivalente a "border-width" pero afectando únicamente a uno de los cuatro bordes: superior, inferior, izquierdo o derecho	thin medium thick *length* (*n*pt|*n*in|*n*cm|*n*px)
border-top-style border-bottom-style border-left-style border-right-style	Equivalente a "border-style" pero afectando únicamente a uno de los cuatro bordes: superior, inferior, izquierdo o derecho	none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset
border-top-color border-bottom-color border-left-color border-right-color	Equivalente a "border-color" pero afectando únicamente a uno de los cuatro bordes: superior, inferior, izquierdo o derecho	*border-color*

PROPIEDAD	DESCRIPCION	VALUES
padding-right padding-left padding-top padding-bottom	Establece la distancia entre el contenido y el borde correspondiente: derecho, izquierdo, superior e inferior	*length* (*n*pt|*n*in|*n*cm|*n*px) *%*
padding	Propiedad que engloba las cuatro anteriores	*padding-top* *padding-right* *padding-bottom* *padding-left*
margin-top margin-right margin-bottom margin-left	Permiten establecer de manera independiente el margen superior, derecho, inferior e izquierdo, entre el elemento HTML afectado y el resto de elementos HTML de la página	auto *length* (*n*pt|*n*in|*n*cm|*n*px) *%*
margin	Propiedad que engloba las cuatro anteriores, permitiéndonos establecer todos los margenes, superior, laterales e inferior mediante una sola declaración	*margin-top* *margin-right* *margin-bottom* *margin-left*

Desde "*http://www.w3schools.com/CSS/tryit.asp?filename=trycss_border*" pueden realizarse pruebas de diseño con las propiedades anteriores o cualquier otras.

Aquellas propiedades de estilo que tengan asociado un valor de medida, es posible utilizar cualquiera de las unidades que se muestran a continuación:

UNIDADES	DESCRIPCIÓN
mm	Medida en milímetros
cm	Medida en centímetros
in	Medida en pulgadas (inches)
pt	Medida en puntos (points). 1pt = 1/72 in
pc	Medida en picas. 1pc = 12pt
px	Medida en pixels
em ex	Medidas que toman como referencia el tamaño de la fuente 1em = tamaño fuente 1ex = 1/2 tamaño fuente
%	Medida en porcentaje

Unidades de medida utilizadas en CSS

De igual forma, aquellas propiedades de estilo CSS cuyo valor asociado sea un color podrá especificarse en tres formatos diferentes:

UNIDADES	DESCRIPCIÓN
nombre_color	Nombre del color normalizado. La lista de todos los nombres de color, y su correspondiente valor en hexadecimal pueden consultarse en "http://www.w3schools.com/CSS/css_colornames.asp".

UNIDADES	DESCRIPCIÓN
rgb(rojo,verde,azul) rgb(rojo%,verde%,azul%)	La codificación RGB nos permite hacer referencia a cualquier color como mezcla de los colores rojo, verde y azul. La cantidad de cada color puede especificarse con una cantidad comprendida entre 0 y 255, o en tanto por cien.
#rrggbb	Especificación del color mediante las cantidades de rojo, verde y azul en hexadecimal, con valores entre 00 (0) y FF (255) para cada uno de los colores.

Formatos utilizados en CSS para especificar un color

3.2. Propiedades de estilo asociadas al fondo del elemento HTML

Las propiedades "background" permiten personalizar el fondo de un elemento HTML, mediante un color o imagen. Esta propiedad es aplicable a cualquier elemento que presente un fondo como son el cuerpo del documento HTML/XHTML (BODY), las divisiones HTML realizadas en una página Web (DIV), una tabla (TABLE), un párrafo (P) o un título (H1|2|3|4|...), entre otros.

PROPIEDAD	DESCRIPCION	VALUES
background-color	Asigna un color de fondo. Este color puede especificarse, al igual que en "**border-color**", en formato RGB o con un de los nombres de color normalizados, tal como se vio en el apartado anterior	*color-rgb* *color-hex* *color-nombre* transparent
background-image	Inserta una imagen de fondo. A través de la directiva "url(URL/ruta image)" se especifica su ubicación	url(*URL/ruta imagen*) none
background-repeat	Indica el modo de repetición de la imagen de fondo	repeat repeat-x repeat-y no-repeat
background-position	Posiciona la imagen de fondo indicada en "**background-image**" en la posición deseada. Se puede hacer uso de valores ya establecidos como "top center", "top right", o directamente indicar las coordenadas X e Y asociadas a la posición deseada mediante las unidades de medida mostradas en el apartado anterior, o mediante porcentajes	top left top center top right center left center center center right bottom left bottom center bottom right *x% y%* *xpos ypos*
background-attachment	Indica si la imagen quedará fija en la posición indicada en "**background-position**" o se desplazará con la barra de desplazamiento o *scrollbar* que muestra el navegador cuando los contenidos Web no caben en el área disponible	scroll fixed
background	Permite combinar las cinco propiedades anteriores en una	*background-image* *background-repeat* *background-position* *background-attachment* *background-position*

Desde "*http://www.w3schools.com/CSS/tryit.asp?filename=trycss_border*" pueden realizarse pruebas de diseño con las propiedades anteriores o cualquier otras.

3.3. Propiedades de estilo asociadas a la posición y dimensión de los elementos HTML en el documento

Propiedades de estilo asociadas a la visibilidad de los elementos en el documento.

PROPIEDAD	DESCRIPCION	VALUES
position	Establece la forma en que se posicionará un elemento HTML dentro del documento. • La opción "**static**" coloca los elementos en el orden en que son declarados en en el documento HTML, sin poder ser controlada su posición exacta dentro del documento, siendo omitidas las propiedades de posicionamiento "**top**", "**bottom**", "**left**" y "**right**". Su posición final en el documento dependerá únicamente de las dimensiones que vayan ocupando cada uno de ellos, y si su colocación lleva implícito un salto de línea o no. Es la opción por defecto. • La opción "**absolute**" nos permite decidir la posición absoluta de un elemento dentro del documento, tomando como origen de coordenadas (0,0) cualquiera de las cuatro esquinas del área de contenidos de nuestro navegador. Por ejemplo, en el caso de tomar como origen de coordenadas la esquina superior izquierda, se posicionarán los elementos con las propiedades "**top**" y "**left**", mientras si es la esquina superior derecha, "**top**" y "**right**". • La opción "**relative**" establece la posición relativa de un elemento en el documento respecto al último elemento HTML posicionado en el documento de manera "**static**" o "**relative**". En el caso de que el elemento anterior del documento haya sido posicionado de manera "**absolute**" o "**fixed**", no será tenido en cuenta para el establecimiento de la posición relativa. • La opción "**fixed**" es equivalente a la opción "**absolute**" con la diferencia de que en aquellos casos donde los contenidos HTML ocupen un área mayor que la zona visible del navegador reservada para ello, su posición será independiente del movimiento de la barra de desplazamiento o *scrollbar* que nos muestra el navegador, permaneciendo su posición siempre fija. Para su mejor comprensión observar las figuras siguientes.	**static** **absolute** **relative** **fixed**
top **bottom** **left** **right**	Establece la posición del elemento HTML en el documento. Son omitidas en la opción por defecto, "**position: static;**", siendo efectivas únicamente en los otros tres modos de posicionamiento, "**relative**", "**absolute**" y "**fixed**"	auto *%* *length*
width **height**	Indican la anchura y altura de un elemento HTML. En el caso de indicar su valor en "%", advertir que estas dimensiones son en relación a las medidas del elemento HTML que contiene al elemento afectado	auto *length* *%*
float	Desplaza un texto o imagen a la derecha o izquierda del área de contenidos, haciendo que su ubicación sea desapercibida por el resto de elementos del	left right none

PROPIEDAD	DESCRIPCION	VALUES
	documento. Esto implica que los elementos que le siguen que sean posicionados de manera ″**static**″ o ″**relative**″ no tendrán en cuenta para la determinación de su posición relativa, al elemento que haga uso de esta propiedad, dando la sensación de ser ″flotante″ dentro del documento. Mediante las propiedades ″**top**″, ″**bottom**″, ″**left**″ y ″**right**″ puede ajustarse la posición final del elemento ″flotante″. Por defecto vale ″**none**″	
clear	Evita que pueda posicionarse un elemento flotante a su lado derecho, izquierdo o ambos	left right both none
vertical-align	Determina el alineamiento vertical de los elementos. Puede ser utilizada cuando queramos alinear una imagen con el texto, mediante ″**text-top**″, ″**middle**″ o ″**text-bottom**″	baseline sub super top text-top middle bottom text-bottom *length* *%*

Desde *“http://www.w3schools.com/CSS/tryit.asp?filename=trycss_position_absolute”* pueden realizarse pruebas de diseño con las propiedades anteriores o cualquier otras.

Para poder comprender mejor las propiedades anteriores de posicionamiento anteriores, a continuación se muestran unas figuras explicativas. En primer lugar, se muestra como quedarían posicionados los elementos dentro del documento, en el caso de que tomar como opción “**absolute**” para el parámetro “**position**”. Las propiedades encargadas de determinar esta posición exacta dentro del área de contenidos del documento HTML/XHTM “**top**”, “**left**”, “**bottom**” y “**right**”, dependen de cual de las cuatro esquinas de este área se cojan como origen de coordenadas:

— Esquina superior izquierda: “**top**” y “**left**”.

— Esquina superior derecha: “**top**” y “**right**”.

— Esquina inferior derecha: “**bottom**” y “**right**”.

— Esquina inferior izquierda: “**bottom**” y “**left**”.

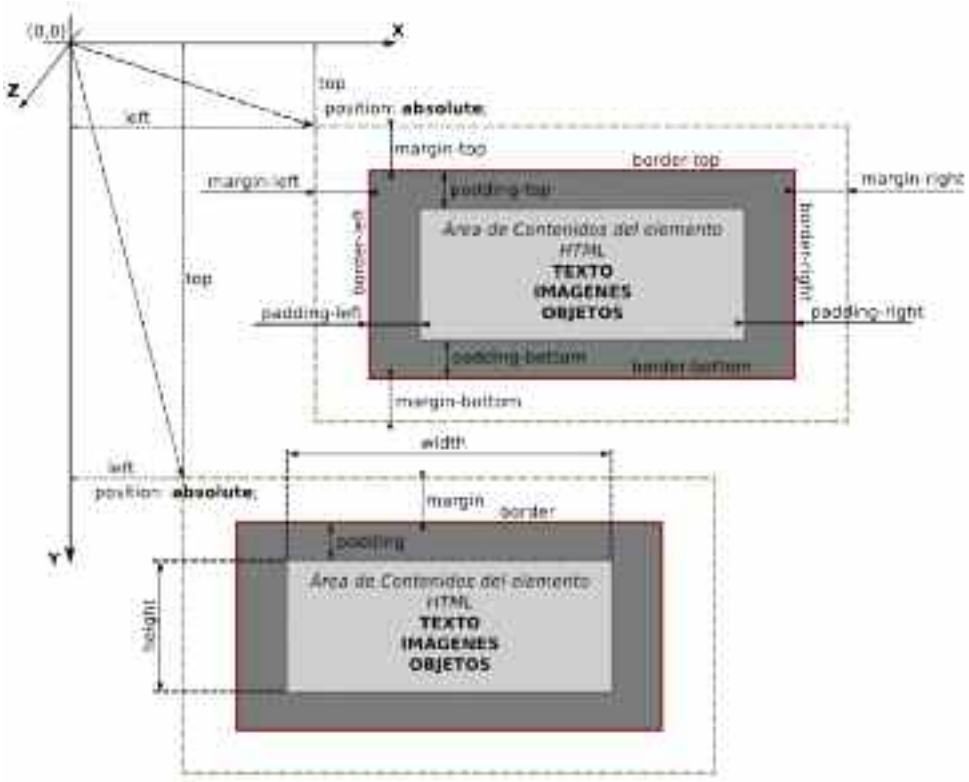


*Figura 2.4. La propiedad "**position: absolute;**" nos permite decidir la posición de un elemento HTML dentro del documento tomando como origen de coordenadas alguna de las cuatro esquinas del área de contenidos del navegador*

En relación a la opción "**relative**", el posicionamiento de un elemento HTML en el documento se establece en relación al último elemento ubicado en el área de contenidos del navegador de manera "**static**" o "**relative**", controlando esta mediante las propiedades "**top**" y "**left**".

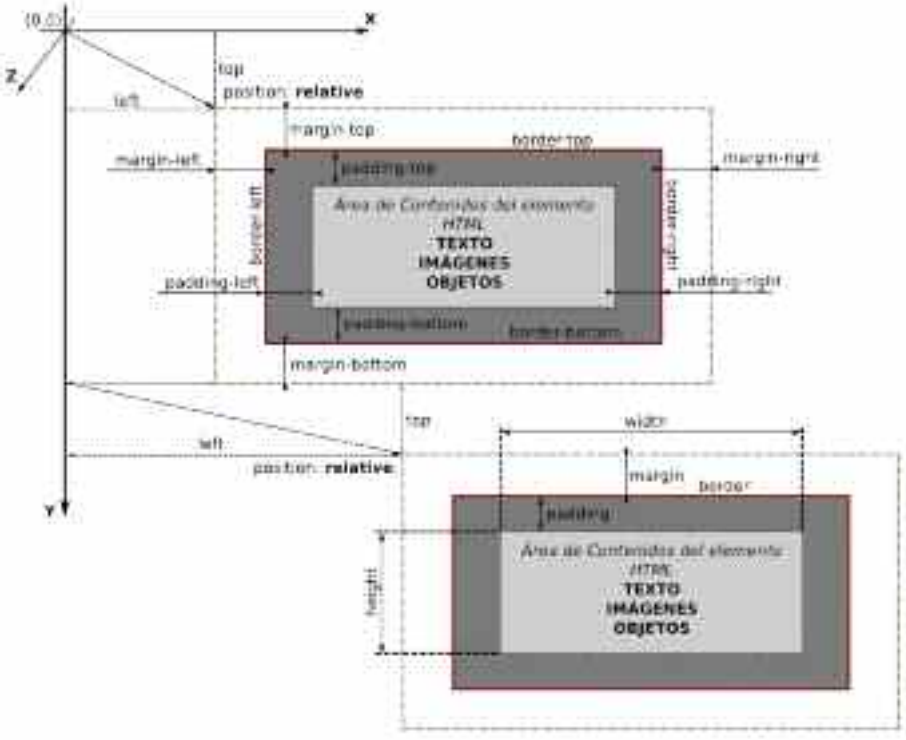


*Figura 2.5. La propiedad "**position: relative;**" nos permite decidir la posición de un elemento HTML dentro del documento en relación a la posición del último elemento posicionado de manera "static" o "relative"*

Ejercicio práctico n.º 1

Con la finalidad de poder practicar las propiedades de estilo anteriores, vamos a diseñar una plantilla Web para que los contenidos presenten el siguiente aspecto:

— En la parte superior, ocupando un área total del 80% de ancho y 20% de alto del área de contenidos del navegador, se situará la cabecera o "header" del documento, con el "logo" de la sitio Web como imagen de fondo.

— Debajo del "header" encontraremos dos zonas especialmente pensadas para dos banners de publicidad de la Web, ocupando un área igual al 15% de alto y 44% de ancho cada uno.

— Bajo los banners de publicidad, en la parte izquierda encontraremos el área asociada al menú principal del sitio Web, ocupando un ancho del 20%, y a la derecha el área asociada a los contenidos del sitio Web, con un ancho del 73%.

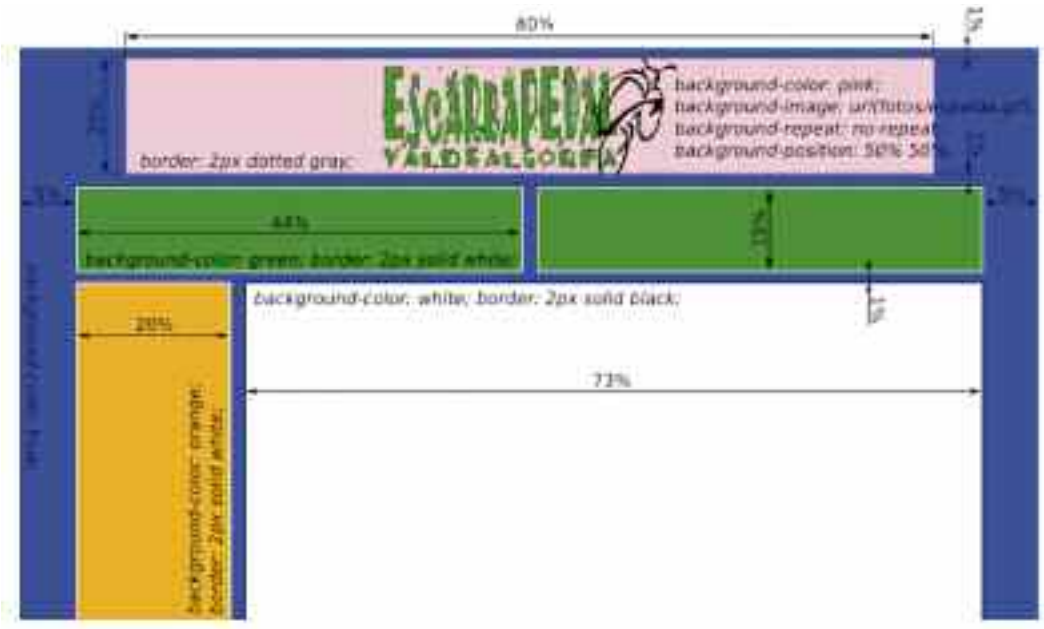


Figura 2.6. Aspecto de la plantilla diseñada para un sitio Web en el ejercicio práctico n.º1

Solución ejercicio n.º 1

Para el diseño de la plantilla Web solicitada haremos tantas divisiones HTML como zonas a delimitar queramos hacer. En concreto, en la solución que se presenta, se establecen siete divisiones: una para la cabecera, otra que albergará las dos divisiones asociadas a los banners, más otra división que contendrá las divisiones asociadas al menú principal y el área de contenidos. Para que las dos divisiones asociadas a los banners queden a la misma altura, en paralelo, mediante la propiedad "float" hará que sean flotantes y que podamos ajustar su ubicación facilmente. De igual forma se hace con las dos últimas divisiones asociadas al menú principal y contenidos.

Para asociar a cada uno de los elementos HTML de la plantilla las propiedades de estilo CSS haremos uso de los atributos "id" y "class" definidos en cada uno de ellos. Estas propiedades se han declarado en la cabeza del documento a través de la etiqueta doble <style></style>, aunque sería más correcto editar un fichero CSS externo para que estas propiedades pudieran ser reutilizadas por todos las páginas Web que formarán parte del sitio Web que hagan uso de la misma plantilla.

```
<?xml version=″1.0″ encoding=″UTF-8″?>
<html>
<head>
<style type=″text/css″>
body { background-color: blue; }
/* Definimos las propiedades CSS de la cabecera o ″header″ del documento. Con ″margin: 1% auto auto auto;″ establecemos un margen superior de un 1%, dejando que el resto de margenes se ajusten automaticamente en función de la anchura y altura definidas, ″width: 80%; height: 20%;″, garantizando de esta forma que quede centrada la división realizada. Como fondo colocamos una imagen posicionándola en el centro ″background-repeat: no-repeat; background-position: 50% 50%;″. */
div#logo { margin: 1% auto auto auto; width: 80%; height: 20%;
    background-color: pink; background-image: url(fotos/espalda.gif);
    background-repeat: no-repeat; background-position: 50% 50%;
    border: 2px dotted gray; }
/* Definimos una división que albergará las dos diviones de los banners. Se le asociada un color transparente para que no sea visible este contenedor, ″background-color: transparent;″. */
div#banner { margin-top: 1%; width: 100%; height: 15%; background-color: transparent;}
/* Las propiedades CSS que tienen en común los dos banners se las asignamos a través del atributo ″class='banner'″ que tienen en común. La altura ″height: 100%;″ es en relación a la altura del elemento que las contiene, ″div#banner″, correspondiete al ″height: 15%;″ de la altura del area de contenidos del navegador. */
div.banner { width: 44%; height: 100%; background-color: green; border: 2px solid white; }
/* Convirtiendo las divisiones en ″flotantes″ las posicionamos a la misma altura donde nos interesa. */
div#banner1 { float: left; margin-left: 5%; }
div#banner2 { float: right; margin-right: 5%; }
/* De manera similar a lo realizado con lo ″banners″ hacemos con las divisiones menú principal y área de contenidos, albergándolas en una división contenedora. */
div.contenedor { margin-top: 1%; }
div#menu { margin-left: 5%; float: left; width: 15%; height: 100%; background-color: orange; border: 2px solid white;}
div#contenidos { margin-right: 5%; float: right; width: 73%; height: 100%; background-color: white; border: 2px solid black; }
</style>
</head>
<body>
<!— Definimos la lista de los elementos que componen la plantilla Web: 7 divisiones. —>
<div id=″logo″></div>
<div id=″banner″>
    <div class= ″banner″ id=″banner1″></div><div class=″banner″ id=″banner2″></div>
</div>
<div class=″contenedor″>
    <div id=″menu″></div><div id=″contenidos″></div>
</div>
</body>
</html>
```

3.4. Propiedades de estilo asociadas al contenido tipo texto del elemento HTML

Propiedades de estilo del texto. Estas propiedades permiten personalizar la apariencia del texto en nuestra página Web dentro del área de contenido de un elemento HTML. Estas propiedades pueden asignarse a cualquier etiqueta HTML doble que encierre texto, desde el propio cuerpo del documento (BODY), úti para asignar un estilo por defecto a todo el texto del documento, a divisiones dentro una página (DIV), párrafos (P) o tablas (TABLE), entre otras.

PROPIEDAD	DESCRIPCION	VALUES
color	Asigna un color al texto. Ver tabla X	*color-rgb* | *color-hex* |*color-nombre*
font-family	Indica la lista de tipografías prioritarias que serán utilizadas para formatear el texto en el navegador Web. Las distintas familias se separarán por comas, y en el caso de que contenga espacios en blanco, se encerrarán entre comillas. En el caso de que el navegador no disponga de las fuentes especificadas, utilizará la tipografía que tenga asignada por defecto, dependiente esta del tipo de navegador	*family-name* *generic-family*
font-size	Determina el tamaño de la fuente. Puede hacerse uso de las medidas predefinidas, o indicar su tamaño haciendo uso de las unidades indicadas en la tabla X. También es posible asignarle un valor en tanto por cien, tomando como referencia el tamaño de la fuente por defecto asignada por el navegador	xx-small | x-small | small medium large | x-large | xx-large smaller | larger *medida* (*n*pt|*n*in|*n*cm|*n*px) *%*
font-style	Permite poner el texto en cursiva. Los valores ″italic″ y ″oblique″ son equivalentes	normal italic oblique
font-weight	Establece el grosor de la fuente	normal bold bolder lighter 100 | 200 | 300 | ... | 900
font-variant	Permite poner el texto en formato mayúsculas, distinguiendo mediante su tamaño a su vez entre los caracteres mayúsculas y minúsculas introducidos	normal small-caps
font	Propiedad que engloba en una sola las propiedades anteriores relacionadas con la fuente del texto	*font-style* *font-variant* *font-weight* *font-size/line-height* *font-family* caption icon menu message-box small-caption status-bar

PROPIEDAD	DESCRIPCION	VALUES
direction	Indica la dirección en que se imprimirá el texto por pantalla, de izquierda a derecha, o a la inversa	ltr *(left to right)* rtl *(left to right)*
line-height	Establece el valor de la distancia entre líneas. Por defecto su valor es "normal", pero podemos asignarle una distancia personalizada haciendo uso de las unidades de medida habituales (ver tabla X). También podemos tomar como unidad de medida el tamaño de fuente establecido, y mediante un número de estas unidades o porcentaje (%), asignar un valor a esta propiedad	normal *medida* (*n*pt|*n*in|*n*cm|*n*px) *número* | *%*
letter-spacing	Determina la separación entre caracteres	normal *medida* (*n*pt|*n*in|*n*cm|*n*px)
text-align	Establece la justificación del contenido de una etiqueta doble HTML, afectando tanto al texto como a la imágenes	left right center justify
text-decoration	Permite remarcar el texto mediante distintos efectos: subrayado, sobrerayado, tachado o parpadeo	underline overline line-through blink
text-indent	Sangra el texto, tabulando su primera línea. La distancia de este sangrado podemos indicarla haciendo uso de las unidades de medida habituales	*medida* (*n*pt|*n*in|*n*cm|*n*px) *%*
text-transform	Transforma el texto, permitiéndonos poner en mayúscula la primera letra de cada palabra, todas las letras, o todas en minúscula	none capitalize uppercase lowercase
white-space	Indica al intérprete HTML si los saltos de línea, espacios en blanco y tabulaciones utilizadas en la redacción del texto serán omitidos o tenidos en cuenta en la presentación del contenido del elemento HTML. Asignándole el valor "**pre**" muestra el texto con la misma apariencia con que fue escrito. Con el valor "**nowrap**" evitamos todo salto de línea inherente	normal pre nowrap
word-spacing	Incrementa o decrementa el espaciado entre palabras	normal *medida* (*n*pt|*n*in|*n*cm|*n*px)

Desde "*http://www.w3schools.com/CSS/tryit.asp?filename=trycss_text-decoration*" pueden realizarse pruebas de diseño con las propiedades anteriores o cualquier otras.

3.5. Propiedades de estilo asociadas a listas

Propiedades de estilo asociadas a listas de elementos HTML que nos permiten personalizar su apariencia.

PROPIEDAD	DESCRIPCION	VALUES
list-style-type	Selecciona un marcador para los elementos de la lista. Los hay para listas no numeradas (disc|circle|square), y numeradas (decimal, decimal-leading-zero, lower-roman, upper-roman, etc.)	none disc | circle | square decimal decimal-leading-zero lower-roman | upper-roman lower-alpha upper-alpha | lower-greek | lower-latin | upper-latin |hebrew | armenian | georgian | cjk-ideographic | hiragana | katakana | hiragana-iroha | katakana-iroha
list-style-position	Establece si la marca asociada a los elemento de la lista aparecerá dentro o fuera del área cuadrada delimitada por la lista	inside outside
list-style-image	Indica la imagen que será utilizada como personalizar el marcador utilizado por los distintos elementos de la lista	none *url*
list-style	Propiedad que engloba las tres anteriores	*list-style-type* *list-style-position* *list-style-image*

3.6. Propiedades de estilo asociadas a la visibilidad de los elementos en el documento

Propiedades de estilo asociadas a la visibilidad de los elementos en el documento.

PROPIEDAD	DESCRIPCION	VALUES
display	Permite decidir la forma en que se visualizará un elemento HTML en el documento. Entre las distintas opciones destacar: — **none**: hace invisible un elemento HTML. — **block**: Visualiza el elemento en el documento introduciendo un salto de linea antes y después de él. — **inline**: Visualiza el elemento eliminando los saltos de línea. — **list-item**: Visualiza los elementos de una lista. — **run-in** | **compact**: Visualiza el elemento en modo "**block**" o "**inline**" dependiendo del contexto. — **table**: Visualiza los elementos de una tabla en el formato correspondiente	none block | inline | list-item | run-in | compact table | inline-table | table-row-group | table-header-group | table-footer-group | table-row | table-column-group | table-column | table-cell | table-caption

PROPIEDAD	DESCRIPCION	VALUES
visibility	Determina si un elemento HTML es visualizado o no. La diferencia entre su valor ″hidden″ respecto a ″display: none;″, es que ″visibility″ lo oculta dejando un espacio vacío en la página Web, mientras que ″display″ no reserva ningún hueco para el elemento. Su valor ″collapse″ sería equivalente a ″display: none;″	visible hidden collapse
opacity	Establece el porcentaje de opacidad de un elemento HTML en el documento. Esta propiedad nos permite establecer un efecto de transparencia, que aunque no soportado por todos los navegadores Web comerciales, si es interpretada por los más extendidos, Internet Explorer o Mozilla Firefox. En el caso de Internet Explorer, para hacer referencia a esta propiedad utilizaremos ″filter: alpha(opacity=0|100)″ donde con un valor comprendido entre 0 y 100 decidimos el porcentaje de transparencia. Mozilla Firefox reconoce directamente la propiedad ″opacity:0|1″, donde con un valor entre 0 y 1 se obtiene el mismo efecto. Según esto, para que nuestra página sea compatible con el mayor número de clientes Web, se hará uso de las dos propiedades anteriores, desechando cada navegador las propiedades que no le son compatibles	filter: alpha(opacity=*0|100*) opacity:*0|1*
z-index	Establece la posición de un elemento HTML sobre el eje Z, perpendicular al plano. Esta propiedad es muy interesante porque permite organizar los elementos HTML en diferentes capas (ver figura X)	auto *number* (-1|0|1)
overflow	Determina el aspecto de la barra de desplazamiento o *scrollbar* que muestra la ventana del navegador Web cuando el contenido de un elemento HTML no cabe dentro de las dimensiones disponibles	visible hidden scroll auto
cursor	Establece la forma que adoptará el puntero del ratón al colocarse sobre el elemento HTML donde se declare esta propiedad. Los hay interesantes como ″**pointer**″, utilizado al pasar el ratón sobre enlaces del documento, ″**text**″, útil al pasar por párrafos de texto, ″**wait**″, cuando esta a la espera de una acción, etc. En caso de que los punteros predefinidos no nos satisfagan, podemos crear nuestros propios cursores (formato ″*.cur″) e importarlo a través de la directiva ″**url**″	auto default pointer | help | wait | text | move | crosshair e-resize | ne-resize | nw-resize | n-resize | se-resize | sw-resize | s-resize | w-resize *url*
clip	Establece las dimensiones del área delimitadora que será utilizada para visualizar el contenido del elemento HTML. La superficie se correponde con un rectángulo, cuyas coordenadas se establecen en relación a la posición relativa del elemento en el documento	auto rect (*top, right, bottom, left*)

Entre las propiedades anteriores cabría remarcar “**z-index**”. Esta nos permite tratar a los elementos HTML del documento como elementos ubicados en diferentes capas, visualizándose únicamente el elemento situado en la capa con mayor valor asociado a esta propiedad. Para poder superponer dos elementos sobre la misma posición del documento será necesario hacer uso de posicionamiento absoluto, “**position: absolute;**”. Controlando la opacidad de la elemento situado en la capa más alta, mediante “**opacity**”, podemos permitir que

se vea parte de los elementos situados en capas inferiores. Aunque en el siguiente capítulo dedicado a JavaScript veremos a través de ejercicios prácticos como hacer uso de la propiedad "**z-index**", con la finalidad de comprender mejor los aspectos tratados se muestra la siguiente figura:

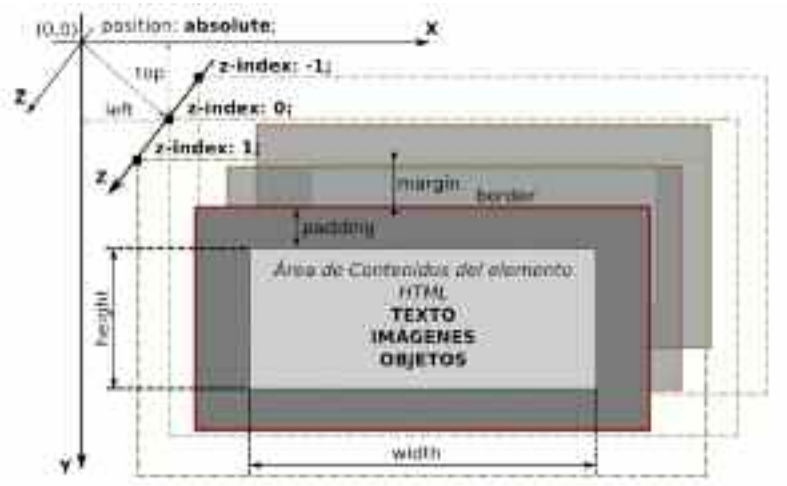


Figura 2.7. La propiedad z-index nos permite organizar los elementos en diferentes capas, pudiendo controlar la visibilidad de los elementos involucrados

3.7. Pseudoclases y pseudoelementos CSS

Además de las propiedades de estilo anteriores, el estándar CSS dispone de unas propiedades especiales llamadas "pseudo-clases" y "pseudo-elementos" que nos permiten asignar efectos especiales a los elementos de nuestro documento HTML/XHTML. Aunque la lista de estas propiedades especiales es más grande, aquí sólo se van a presentar aquellas que son reconocidas entre los navegadores Web más extendidos, "Internet Explorer", "Mozilla Firefox" o "Netscape".

PSEUDO-CLASE	DESCRIPCIÓN
:link {*estilo*}	Asigna un estilo personalizado a un enlace sin visitar de nuestra página HTML
:visited {*estilo*}	Asigna un estilo personalizado a un enlace que ha sido visitado
:active {*estilo*}	Asigna un estilo personalizado a un enlace seleccionado
:hover {*estilo*}	Asigna un estilo personalizado a un elemento HTML cuando el ratón pasa sobre él

PSEUDO-ELEMENTO	DESCRIPCIÓN
:first-letter {*estilo*}	Aplica un estilo personal a la primera letra del texto contenido en el elemento HTML
:first-line {*estilo*}	Aplica un estilo personal a la primera línea del texto contenido en elemento HTML.

La utilización de estas propiedades especiales y del resto se mostrarán en el siguiente ejercicio práctico.

Ejercicio práctico n.º 2

Diseñar la plantilla Web construida en el capítulo anterior mediante el uso de una tabla HTML haciendo uso de las propiedades de estilo vistas en este capítulo. Sus características más importantes son las siguientes:

Figura 2.8. Plantilla del sitio Web formada por tres divisiones diferentes

— La página estará dividida en tres partes: cabecerá (o header), menú principal y área de contenidos.

— La cabecera del documento ocupará un 100% de anchura y 150 pixels de altura, presentado un fondo naranja y un logo en su centro.

— El menú principal ocupará un 15% de la anchura del área disponible en el navegador para los contenidos y 800 pixels de altura, presentando fondo gris. Los items del menú se corresponderán con los elementos de una tabla HTML, donde las filas pares e impares presentarán un aspecto diferente.

— Al pasar el ratón por encima de los items de menú cambiarán su aspecto añadiendo un pequeño efecto dinámico. Para ello haremos uso de las pseudo-clases CSS.

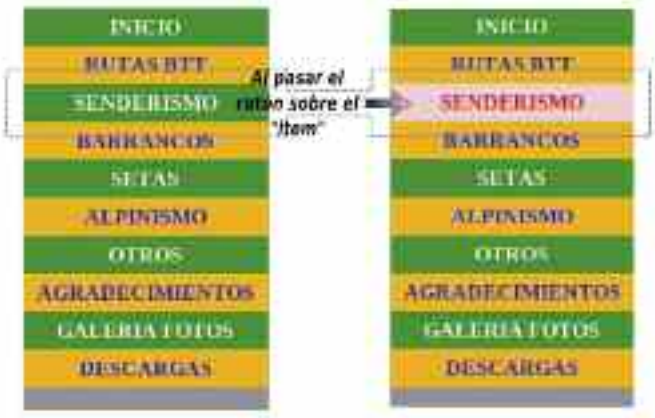


Figura 2.9. Al pasar el ratón por los items del menú cambiará su afecto aportando cierto dinamismo a la Web

Con la finalidad de poder reutilizar los estilos CSS que definamos para la plantilla en las páginas Web que componen el sitio Web , editaremos un fichero externo, "**plantilla.css**" que los contenga.

La primera letra del párrafo de texto de la división asociada a los contenidos, presentará un estilo particular: un tamaño cuatro veces más grande al resto del texto, color azul, y en cursiva.

Solución ejercicio n.º 2

Según las especificaciones de diseño del enunciado anterior, los elementos HTML que componen el documento sería el siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<html>
<head>
<!— Importamos la hoja de estilos externa "plantilla.css" ubicada en la subcarpeta "css" —>
<style>  @import "css/plantilla.css" screen; </style>
<link href='fotos/icono.png' rel='shortcut icon' type='image/png'>
</head>
<body>
<!— Creamos una división asociada a la cabecera o "header" del documento.  En lugar de colocar el logo de la Web como imagen de fondo, se ha optado por insertar un elemento imagen correspondiente al logo —>
<div class="cabecera"> <img src="fotos/logo.gif" widht="400px" height="150px"> </div>
<!— La división "contenedor" contendrá a su vez a las divisiones "menu" y "contenidos". —>
<div class="contenedor">
    <div class="menu">
<!— El menú principal se corresponde con una tabla HTML, donde cada una de sus filas es un item del menú.  Para que la tabla ocupe todo el ancho reserado por la división "menú", se establece su ancho en "100%" respecto. —>
    <table border="0" cellpadding="0" cellspacing="0" width="100%">
<!— Los elementos de la tabla y enlaces tienen asignados valores en sus atributos "class" e "id" para poder ser referenciados desde la hoja de estilos asociada al documento. —>
    <tr><td class="enlace" id="tipo1"><a href="index.html" id="tipo1">INICIO</a></td></tr>
    <tr><td class="enlace" id="tipo2"><a href="rutas.html" id="tipo2">RUTAS BTT</a></td></tr>
    <tr><td class="enlace" id="tipo1"><a href="senderismo.html" id="tipo1">SENDERISMO</a></td></tr>
    <tr><td class="enlace" id="tipo2"><a href="barrancos.html" id="tipo2">BARRANCOS</a></td></tr>
    <tr><td class="enlace" id="tipo1"><a href="setas.html" id="tipo1">SETAS</a></td></tr>
    <tr><td class="enlace" id="tipo2"><a href="alpinismo.html" id="tipo2"> ALPINISMO</a></td></tr>
```

```
<tr><td class="enlace" id="tipo1"><a href="otros.html" id="tipo1">OTROS
</a></td></tr>
<tr><td class="enlace" id="tipo2"><a href="agradecimientos.html" id="tipo2">
AGRADECIMIENTOS</a></td></tr>
<tr><td class="enlace" id="tipo1"><a href="galeria.html" id="tipo1">GALERIA
FOTOS</a></td></tr>
<tr><td class="enlace" id="tipo2"><a href="descargas.html" id="tipo2">DESCAR-
GAS</a></td></tr>
</table>
</div>
<!— La división "contenidos" contendrá dos parrafos, uno con un texto y otro con
una imagen. —>
<div class="contenidos" width="100%">
<p id="parrafo1">
El club <b>ESCARBAPEDAL</b> está compuesto por un grupo de aficionados a las
actividades de montaña. Entre estas actividades cabría señalar las rutas en bicicleta BTT,
aunque también se desarrollan otras actividades como pueden ser senderismo, barran-
quismo, etc.
</p>
<p id="parrafo2"> <img src="fotos/miembros.jpg"> </p>
</div>
</div>
</body> </html>
```

En relación a la hoja de estilos CSS importada desde el documento anterior, "**plantilla.css**", su contenido sería el siguiente:

```
/* Hoja de estilos CSS "plantilla.css". */
/* Div "cabecera": Establece sus dimensiones, formato del borde, color de fondo.
Para que la imagen insertada aparezca centrada definimos "text-align: center; ", y
para no mostrar el "scrollbar" en esta división "overflow: hidden;". */
div.cabecera { width: 100%; height: 150px; border: 1px solid red; background-color: oran-
ge; overflow: hidden; text-align: center; }
/* Div "contenedor": Albergará las divisiones asociadas al menú principal y área de
contenidos. Establece sus dimensiones y personaliza todos sus bordes menos el
top, al ser compartido con la división "cabecera". */
div.contenedor { background-color: transparent; width: 100%; height: 800px; border-bot-
tom: 1px solid red; border-left: 1px solid red; border-right: 1px solid red; }
/* Div "menu": Establece sus dimensiones, y se coloca de manera flotante en el lado
izquierdo. */
div.menu { float: left; width: 15%; height: 800px; background: gray; }
/* Personalizamos las celdas de la tabla que albergarán los items del menú: centra-
mos el texto, lo ponemos en negrita, y definimos un relleno de 5px para dar holgu-
ra al contenido de la celda */
td.enlace { text-align: center; font-weight: bold; padding-top: 5px; padding-bottom: 5px; }
/* Enlaces del menú: para evitar los enlaces <a></a> aparezcan subrayados y bajo el
color establecido por defecto en cada navegador Web, eliminamos esta decoración. */
div.menu a { text-decoration: none; }
/* Mediante la ayuda de Pseudo-Clases CSS añadimos dinamismo a los enlaces del
menú.*/
```

```
/* Se definen dos estilos para los enlaces del menú: uno para los enlaces no visitados, "link", y visitados, "visited", y otro para los enlaces que detectan pasar el ratón sobre ellos, "hover".*/
div.menu a#tipo1:link,div.menu a#tipo1:visited { color: white; background-color: transparent; }
div.menu a#tipo1:hover { color: red; background-color: transparent; }
div.menu a#tipo2:link, div.menu a#tipo2:visited { color: blue; background-color: transparent; }
div.menu a#tipo2:hover { color: red; background-color: transparent; }
/* Se definen dos estilos para las celdas de la tabla, dependiendo de si se detecta pasar el ratón por encima o no. */
div.menu td#tipo1 { background-color: green; }
div.menu td#tipo1:hover { color: red; background-color: pink; }
div.menu td#tipo2 { background-color: orange; }
div.menu td#tipo2:hover { color: red; background-color: pink; }
/* Div "contenidos": Establece sus dimensiones, y se coloca de manera flotante en el lado derecho. */
div.contenidos { float: right; width: 85%; height: 800px; background: yellow; }
/* El "parrafo1" contendrá el texto de presentación de la página de inicio del sitio Web. Para favorecer su presentación se establecen margenes superior y laterales, se justifica el texto y se sangra 1cm. */
p#parrafo1{ margin-left: 30px; margin-top: 30px; margin-right: 30px; text-align: justify; text-indent: 1cm; }
/* Mediante Pseudo-Elementos personalizamos la primera letra del parrafo para destacar el texto. */
p#parrafo1:first-letter { font-size: 400%; color: blue; font-style: italic; }
/* El "parrafo2" contendrá una imagen.  Para que esta aparezca centrada, se declará  "text-align: center;" */
p#parrafo2 { text-align: center; }
/* Para mejorar el aspecto de la imagen se establece un borde, un relleno de 5px de grosor, y a modo de ejemplo, se modifica la forma del puntero al pasar el ratón sobre ella. */
p#parrafo2 img { border: 1px solid blue; background-color: #B1F1F1; padding: 5px; cursor: help; width: 400px; height: 300px; }
```

Capítulo 3

JAVASCRIPT

En los capítulos anteriores hemos aprendido a maquetar paginas Web haciendo uso de etiquetas HTML y hojas de estilo CSS. Este tipo de documentos Web se caracterizan por ser estáticos, es decir, que los contenidos que presentan y la forma en que lo hacen no cambia, y siempre es igual independientemente de quien sea quien acceda a ella, el momento en que se haga y la manera en que se haga el acceso.

Pero en ocasiones nos puede interesar introducir elementos dinámicos en nuestros documentos con la finalidad de atraer la atención de los usuarios, entendiendo por dinámico cualquier efecto cambiante en forma, tiempo o dependencia de su acceso. Por ejemplo, en la actualidad todas las personas que tenemos acceso a Internet solemos acceder a la Web para visualizar algún vídeo que nos pueda interesar o informarnos de noticias en algún periódico digital; si los contenidos que se nos mostraran fueran siempre los mismos, accederíamos una vez, tal vez dos, pero es fácil que no accediéramos más al no encontrar nada nuevo. Algunos ejemplos de dinamismo que podemos encontrar habitualmente en la Web podrían ser los siguientes:

a) Modificar el formato de un texto (color, tamaño, contenido, ...), o una imagen al pasar con el ratón sobre estos elementos. Esto permite centrar la atención del usuario que accede desde un cliente al sitio Web, con la finalidad de transmitirle un mensaje.

b) Dependiendo de la fecha y hora en que se solicita una página Web, podemos hacer que el contenido sea uno u otro. En sitios Web informativos, podemos diferenciar, por ejemplo, actividades de mañana y tarde.

c) Mediante el uso de formularios HTML, podemos solicitar “login” y “password” al usuario, y en función de su valor, hacer que los contenidos visualizados sean distintos. Este aspecto dinámico es utilizado por infinidad de aplicaciones Web: correo Webmail, servicios de almacenamiento de información en la Web (por ejemplo, youtube), etc.

d) Los formularios HTML también son utilizados en la realización de consultas a bases de datos. Los gestores de bases de datos actuales (MySQL, ORACLE, etc.) disponen de entornos Web para interactuar con las bases de datos que contienen, pudiendo realizar sobre estas todas las operaciones habituales (selección, inserción, actualización, borrado, etc.). Este aspecto aporta un gran dinamismo a la Web, siendo utilizado por multitud de sitios Web, como “Google”, tiendas virtuales de Internet o elaboradores de rutas de carretera e itinerarios. Es decir, “Google” no es más que una gigantesca base de datos que contiene información que es consultada en base a un patrón de busqueda. De manera similar ocurre con la compra de productos a través de Internet, en cuyas tiendas virtuales, se puede consultar su catálogo y hacer todo tipo de compras.

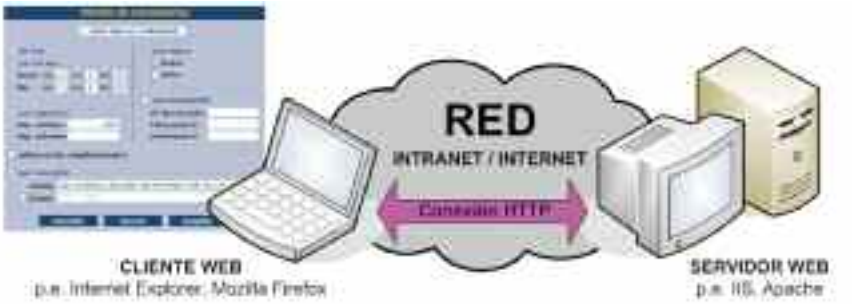


Figura 3.1. La utilización de formularios HTML es el método clásico utilizado para intercambiar datos entre un cliente y un servidor Web, permitiéndonos dinamizar los contenidos Web

Como puede advertirse a través de los ejemplos anteriores, el elemento más importante que es utilizado para dinamizar un sitio Web el el formulario HTML, indispensable para interactuar con el servidor Web y personalizar los contenidos Web que devuelve al cliente.

A la hora de hablar de contenidos dinámicos, también es importante distinguir entre dinámismo en el lado del cliente y en el lado del servidor. Esta diferencia radica en quien es el encargado de dinamizar los contenidos Web en la comunicación entre nuestro cliente Web y el servidor al que se accede. Para conseguir el dinamismo en el lado del cliente, es necesario introducir en los documentos Web algo más que etiquetas HTML y estilos CSS, normalmente líneas de código de un lenguaje de programación estandarizado que irá camuflado o embebido entre el resto del documento. Por ello es necesario que el cliente Web disponga de un software intérprete que comprenda el lenguaje añadido y se encargue de dinamizar los contenidos. A estos intérpretes se les suele llamar comunmente “plugins”, o parches software, que son agregados a nuestro navegadores, y que les confieren la función comentada. En la actualidad, los navegadores Web más extendidos, Internet Explorer, Mozilla Firefox, Netscape u Opera, suelen traer integrados estos “plugins” despreocupando al usuario de su instalación. Entre los “plugins” más afamados, están los intérpretes de JavaScript, de Java y de Flash.

Por otro lado podemos implementar dinamismo involucrando al servidor. Para conseguir este dinamismo en el lado del servidor es necesario que tras la conexión con él, exista una comunicación permanente entre el cliente y servidor durante la sesión que abrimos, con la finalidad de que el servidor vaya refrescando nuestros contenidos, o nos vaya mostrando la información deseada, introduciendo de esta forma el efecto de dinamismo. La manera más común de mantener esta comunicación con el servidor es a través de formularios HTML. Mediante la introduciendo de cajas de texto, u otros elementos, se nos permite hacer consultas al servidor en busca de la información deseada. Otra opción fue extendida es AJAX (JavasScript Asincrono y XML), donde de manera transparente para el usuario, en un segundo plano, y de manera asincrona se mantiene una comunicación entre el cliente Web y el servidor Web, permitiéndonos refrescar los contenidos de las páginas Web consultadas.

En el presente capítulo vamos a aprender una posible forma de dinamizar los contenidos Web. Este tan sólo va a ser el primero de los capítulos que se van encargar de mostrar como introducir efectos dinámicos en nuestros documentos, con la pretensión de que al final del libro hayamos aprendido a implementar una Web tan atractiva como las que podemos encontrar en Internet. En concreto, en este capítulo nos vamos a centrar en JavaScript, un lenguaje que embebido entre las etiquetas HTML, permite introducir efectos dinámicos muy atractivos que nos permitirá centrar la atención del usuario que acceda al sitio Web. Hoy por hoy, JavaScript es la solución más extendida en el mundo Web a la hora de dinamizar los contenidos Web en lado del cliente.

1. CONTENIDOS DEL CAPÍTULO

A lo largo del capítulo se van a ir introduciendo los aspectos más importantes de JavaScript, reforzado con ejemplos prácticos, y sus posibles soluciones.

— Empieza presentando el lenguaje de programación JavaScript destacando sus características más importantes.

— Una vez entendido que JavaScript es un lenguaje orientado a eventos, se presentan los distintos tipos de eventos que existen.

— A continuación se presentan los objetos más importantes que existen en JavaScript, resaltando sus principales propiedades y métodos asociados. De manera especial se informa del objeto "style", el cual nos permite desde JavaScript modificar las propiedades de estilo CSS asociadas a los elementos de un documento HTML.

— El capítulo continúa explicando la importancia de las variables y funciones declaradas por el usuario en JavaScript.

— Con la finalidad de poder controlar el flujo de ejecución de las instrucciones JavaScript, se presentan las distintas estructuras de control de flujo y los operadores.

— Termina el capítulo presentando las funciones o métodos predefinidos en JavaScript que pueden ser utilizadas en nuestros programas, facilitándonos el trabajo, y aumentando la potencia del resultado.

Como ya se ha indicado previamente, para poder comprender todas las características presentadas, mediante la ayuda de ejercicios prácticos propuestos facilitaran su comprensión. En concreto, continuaremos con la construcción del sitio Web empezado en los capítulos anteriores, que iremos mejorando poco a poco, incorporándole nuevas funcionalidades.

Para crear programas en lenguaje JavaScript, podremos seguir utilizando el editor de texto/HTML/JavaScript que ya hemos utilizado en capítulos anteriores. La interpretación y ejecución del código JavaScript y visualización de sus resultados, seguirá encomendandose a nuestro cliente o navegador Web favorito. Tan sólo será necesario invocar como hasta ahora, desde el cliente Web al documento HTML que incluya el código JavaScript colocando en la barra de navegación del cliente la URL o ruta dentro del equipo donde localiza el documento.

Para la comprobación de posibles errores producidos en la creación de los programas JavaScript existen multitud de herramientas software. En el caso de utilizar como intérprete al navegador Web "Mozilla Firefox", es posible instalar un plugin o extensión de Mozilla que nos informará de todo ello: "**Firebug**". Para su instalación iremos al menú "Herramientas/Complementos", y allí deberemos encontrar un enlace al portal de Mozilla para obtener nuevas extensiones, estableciéndose una conexión automática, a través de Internet con su sitio Web: "https://addons.mozilla.org/es-ES/firefox/". Desde allí podremos buscar el complemento deseado e instalarlo.

Figura 3.2. Firebug es un complemento de Mozilla Firefox que nos permitira depurar los programas escritos en JavaScript interpretados por este cliente Web

Una vez instalado, en caso de que Firebug detecte errores al interpretar el código JavaScript, nos informará mediante una advertencia visible en la esquina inferior derecha de nuestro navegador. Al pinchar sobre este mensaje de advertencia la información sobre los errores se ampliará.

Figura 3.3. Firebug nos informará de la detección de errores con un mensaje en la esquina inferior derecha del navegador Mozilla Firefox.
Al pinchar sobre él, se despliega una información más detallada

2. CARACTERÍSTICAS DE JAVASCRIPT

A diferencia de otros lenguajes de programación como pueden ser "C", "Pascal", "Java" o "Perl", "JavaScript" es un lenguaje pensado para el desarrollo Web. Es decir, mientras otros lenguajes han sido creados para el desarrollo de software de sistemas (GNU/Linux esta implementado en lenguaje C y Microsoft Windows en C#) o desarrollo de aplicaciones, en 1995 la empresa "Netscape" presenta el lenguaje JavaScript con la finalidad de poder incluir contenidos dinámicos en las páginas HTML que eran visualizadas desde su cliente Web, "Netscape Navigator". Las características más destacadas del lenguaje JavaScript son las siguientes:

a) Mediante el uso de la etiqueta doble HTML <SCRIPT> </SCRIPT>, JavaScript va embebido en HTML como si se tratase de una extensión más del documento, lo que garantiza que todo lo visto en HTML nos va a seguir siendo valido en este capítulo.

<script *type src*> *Instrucciones Lenguaje JavaScript* **</script>**

En cuanto a los atributos de la etiqueta HTML anterior:

— "**type**": indica el tipo de lenguaje que va a ser utilizado en el script. En scripts JavaScript su valor es "**text/javascript**". Antiguamente, para proporcionar esta información se hacía uso del atributo "language", que actualmente es desaconsejado.

— "**src**": informa de la ruta relativa al documento HTML donde se localiza un fichero externo con las instrucciones JavaScript que formarán parte del script. Este fichero externo para indicar que su contenido es código JavaScript, se hará explicitamente mediante su extensión, "***.js***". En el caso de que este atributo no aparezca, deberán incluirse las instrucciones JavaScript entre las etiquetas HTML "**<script> </script>**".

b) A diferencia de otros lenguajes de programación, como "C" o "Java" no requiere de una compilación previa para ser ejecutado, sino que es interpretado directamente por el propio cliente Web y ejecutado, mediante la ayuda del correspondiente software intérprete de JavaScript con el cuentan actualmente los navegadores Web de manera preinstalada.

c) Sin lugar a dudas, la característica más importante de JavaScript, es que se trata de un lenguaje de programación orientado a eventos, al igual que el lenguaje de desarrollo de aplicaciones "Visual Basic". Mientras otros lenguajes de programación como el lenguaje "C" sigue una secuencia establecida de ejecución, normalmente el orden en que son redactadas las instrucciones de ejecución, en el lenguaje JavaScript se va ejecutando el código siguiendo el orden de eventos que se van sucediendo, provocados estos por el usuario que hace uso del cliente o navegador Web. Eventos que pueden desencadenar la ejecución de código JavaScript, si así se le ha indicado, puede ser el simple hecho de pulsar una tecla del teclado del cliente, pasar el ratón sobre una determinada zona de la página o el simple hecho de pinchar sobre un elemento en concreto.

Al igual que en otros lenguajes de programación es posible comentar los programas JavaScript, importante este aspecto para poder documentar y facilitar el posterior mantenimiento del código. Los comentarios utilizados en HTML, ***<!—Comentario HTML—>*** no serán validos en JavaScript. Para poder incluir comentarios dentro de JavaScript existen dos opciones:

1. En el caso de querer comentar una sola línea, esta se precederá del juego de caracteres "***//***" (doble barra inclinada): ***//Comentario JavaScript de una línea***.
2. En el caso de que el comentario ocupe más de una línea, puede precederle a cada una de las líneas el juego de caracteres anterior, "***//***", o encerrando el párrafo entre el juego de caracteres "***/****" y "*****/***": ***/* Comentarios JavaScript – línea o parrafo */***.

3. EVENTOS JAVASCRIPT

Teniendo en cuenta que para poder programar en JavaScript hay que conocer los distintos tipos de eventos que pueden desencadenar la ejecución del código que introduzcamos en el documento HTML, lo primero que haremos será mostrar la lista de los diferentes eventos que pueden programarse:

CLASE DE EVENTOS	EVENTO	DESCRIPCIÓN DEL EVENTO
EVENTOS ASOCIADOS AL RATÓN	OnClick OnDblClick OnMouseOver OnMouseOut OnMouseDown OnMouseUp OnMouseMove	Eventos producidos por la acción del movimiento del ratón, tras hacer un click (On Click), un doble click (Dbl Click), al pasar sobre un elemento del documento HTML (Mouse Over), tras salir del área asignada a un elemento (Mouse Out), tras pulsar un botón del ratón (Key Down) o despulsarlo (Key Up), o simplemente por mover el ratón (Mouse Move)
EVENTOS ASOCIADOS AL TECLADO	OnKeyDown OnKeyUp OnKeyPress	Eventos producidos por la acción de pulsar una tecla (Key Down), de despulsar una tecla (Key Up), o la combinación de ambos, tras pulsar y despulsar una tecla (Key Press)
EVENTOS ASOCIADOS AL CUERPO DEL DOCUMENTO Y SUS MARCOS	OnLoad OnUnLoad	Eventos asociados a la carga y descarga del cuerpo del documento HTML o alguno de los marcos (FRAMES) que pueden formarla
EVENTOS ASOCIADOS A FORMULARIOS	OnFocus OnBlur OnSelect OnChange OnSubmit OnReset	Eventos producidos sobre los elementos que componen un formulario HTML: botones, áreas de texto, check box, etc. Los eventos que pueden desencadenar la ejecución de JavaScript pueden ir desde la posición donde se situe el foco de edición dentro del formulario, al simple hecho de que se produzca un cambio

Para desencadenar un efecto dinámico ante cualquiera de los eventos anteriores, tan sólo será necesario declarar dicho evento como si se tratase de un atributo más dentro de la etiqueta HTML del elemento en cuestión, y asociarle el efecto deseado en lenguaje JavaScript. Este efecto a asignar suele corresponderse con un cambio en el valor de alguno de sus atributos HTML o propiedades de estilo del elemento HTML dentro de nuestro documento que deseamos modificar. Su sintaxis, junto con un ejemplo, sería la siguiente:

```
<elemento_HTML atributos evento="efecto_dinámico">
<body style="background-color:pink" OnLoad="efecto_dinámico" OnClick="efecto_dinámico">
<p class="parrafo1" OnMouseOver="efecto_dinámico" OnMouseOut="efecto_dinámico">
```

El elemento HTML dentro de nuestro documento (document) que sufre en efecto dinámico, puede ser el mismo donde es declarado el evento u otro elemento del documento:

a) En el caso de que el elemento que va a ver modificado su aspecto ante la detección del evento producido sea el mismo donde fue definido, tan sólo habrá indicar el atributo a modificar y su nuevo valor. En el caso de que el atributo al que se le vaya a modificar el valor sea "style", será necesario indicar explícitamente la propiedad de estilo a cambiar, y asignarle su nuevo valor (por ejemplo, "*style.color='red';*"). Algunos ejemplos podrían ser los siguientes:

```
<img id="mifoto" src="foto1.jpg" OnMouseOver="src='foto2.jpg';" OnMouseOut="src='foto1.jpg';">
<p class="parrafo1" OnMouseOver="style.color='red';" OnMouseOut="style.color='blue';">
```

b) En el caso de que el elemento que va a ver modificado su aspecto ante la detección del evento sea otro elemento del documento, haremos referencia a él a través de su identificador mediante la función JavaScript "**getElementById**('*Identificador del elemento HTML*');" del objeto JavaScript "**document**". Para identificar a los distintos elementos del documento HTML es necesario asignarle un nuevo atributo a la etiqueta HTML, "id='*identificador del elemento HTML*'", . Su sintaxis junto con un ejemplo sería la siguiente:

```
document.getElementById("id").atributo="valor"
document.getElementById("id").style.propiedad="valor"
```

```
<div id="divisionA"> Etiquetas HTML / Texto / Imágenes / Objetos </div>
<p OnClick="document.getElementById('divisionA').style.backgroundColor='orange'">
```

Como puede advertirse a partir del ejemplo anterior, para JavaScript el documento HTML es un objeto llamado "**document**" que tiene asociado un conjunto de propiedades y métodos que permiten hacer hacer referencia a los distintos elementos que intervienen en el propio documento. Aunque de estas propiedades y métodos o funciones asociadas a los distintos objetos JavaScript se informará más adelante, en este punto del capítulo sería importante conocer las diferentes formas que hay para hacer referencia a los elementos HTML que forman parte del documento:

1.ª forma: A través del valor de su atributo "id": **document.getElementById("id")**.

2.ª forma: A través del valor de su atributo “name”, podemos referenciar al conjunto de elementos que compartan dicho valor: **document.getElementsByName(“name”)**.

3.ª forma: A través del nombre de la propia etiqueta HTML, referenciando a todos los elementos del mismo tipo: **document.getElementsByTagName(“nombre”)**.

Antes de seguir profundizando en el lenguaje JavaScript, haremos a continuación un ejemplo práctico que muestre lo visto.

Ejercicio práctico n.º 1

Diseña una página Web dentro del sitio Web que estamos implementando llamada “**agradecimientos.html**” correspondiente al enlace del menú principal “AGRADECIMIENTOS”. El texto de agradecimiento contendrá un conjunto de palabras claves que cambiarán de formato al pasar el ratón por encima. La plantilla de la página seguirá siendo la misma que la diseñada en el capítulo anterior, resultando un aspecto similar al siguiente:

Figura 3.4. Pantalla de la página Web asociada al item del menu “AGRADECIMIENTOS”. Al pasar el ratón sobre las palabras clave, en color rojo y negrita, cambiará su aspecto, tal como se observa con “MATARRAÑA”

Solución ejercicio n.º 1

Para ello seguiremos los siguiente pasos:

1) La división de la página asociada a los contenidos Web (*Contenido Web* </DIV>) contendrá párrafos de texto, “ *texto/imágenes* ”, donde la primera letra de cada uno de ellos, “p:first-letter”, para resaltarlo se le dará un tamaño grande, “font-size: xx-large;”, color rojo, “color: red”, y en cursiva, “font-style: italic;”.

En relación a las palabras clave dentro del párrafo que se desean dinamizar, para resaltarlas se pondrán en negrita, “”, color rojo, “color: red;” y con la primera letra de cada una de ellas en mayúsculas, “text-transform: capitalize;”. Para ello se definirán dentro de la etiqueta doble HTML <style></style> los estilos anteriores. Para asociar estos estilos a los párrafos y a las palabras clave, se hara uso del nombre etiqueta en el primer caso, “p”, y del nombre del identificador, id=”clave1”, en las segundas.

```
<style type=text/css>
div.division3 { width: auto; height: 800px;  background: yellow; position: relative;  m a r -
gin-left: 200px; margin-top: 0px; margin-right: 0px; text-align: justify;  border-right: 2px solid
red; text-indent: 1cm;  padding: 1cm; padding-top: 0.5cm; }
p:first-letter { font-size: xx-large; color: red; font-style: italic; }
#clave1 { font-size: 16px; color: red; text-transform: capitalize; }
</style>

<div class=˝division3˝>
    <p>Agradecimientos a todos aquellas personas que de manera desinteresada han
hecho posible el <b id=˝clave1˝>club Escarbapedal</b>, amante de la naturaleza y todo lo
que en ella se encuentra. Gracias por saber valorar mejor nuestro entorno, enseñar a valo-
rarlo y cuidarlo.</p>
    <p>Gracias especialmente a todos esos pueblos que forman las comarcas del <b
id=˝clave1˝>Bajo Aragón</b>, y bellisimas <b id=˝clave1˝>Matarraña</b> y <b
id=˝clave1˝>Maestrazgo:</b></p>
    <ul style=˝list-style-position:inside; color:brown; font-size:18px; font-weight:800;˝>
    <li>Alcañiz</li> <li>Valdealgorfa</li> <li>Torrecilla de Alcañiz</li> <li>La
Codoñera</li> <li>Beceite</li> <li>Valderrobres</li> <li>Ejulve</li> <li>Castellote</li>
<li>...</li>
    </ul>
</div>
```

2) A continuación asociaremos los eventos a detectar a los elementos HTML deseados. En concreto, queremos que al pasar el ratón, “**OnMouseOver**”, sobre las palabras clave, cambie las propiedades de su estilo, volviendo estas propiedades a su valor inicial al sacar el ratón del área de influencia de las palabras claves, “**OnMouseOut**”. Para declarar estos eventos JavaScript incluiremos ambos como si se tratasen de dos atributos más dentro de las etiquetas que encierran a las palabras clave. Según esto, el párrafo anterior quedaría de la siguiente forma:

```
<p>Agradecimientos a todos aquellas personas que de manera desinteresada han hecho
posible el <b id=˝clave1˝ OnMouseOver OnMouseOut>club Escarbapedal</b>, amante
de la naturaleza y todo lo que en ella se encuentra. Gracias por saber valorar mejor nues-
tro entorno, enseñar a valorarlo y cuidarlo.</p>
<p>Gracias especialmente a todos esos pueblos que forman las comarcas del <b
id=˝clave1˝ OnMouseOver OnMouseOut>Bajo Aragón</b>, y bellisimas <b id=˝clave1˝
OnMouseOver OnMouseOut>Matarraña</b> y <b id=˝clave1˝ OnMouseOver
OnMouseOut>Maestrazgo:</b></p>
```

3) Por último asociaremos el efecto dinámico a los eventos anteriores, correspondiente a un cambio en las propiedades de estilo. Para conseguir redifinir el estilo, en este ejemplo práctico se ha optado por definir en nuevo estilo asociado a un nuevo identificador, “id=’clave2’”, de tal forma que al provocarse el evento en cuestion el elemento afectado cambien de identificador adoptando el nuevo estilo:

```
#clave1 { font-size: 16px; color: brown; text-transform: capitalize; }
#clave2 { font-size: 18px; color: blue; text-transform: uppercase; }

<p>Agradecimientos a todos aquellas personas que de manera desinteresada han hecho posible el <b id=″clave1″ OnMouseOver=″id='clave2';″ OnMouseOut=″id='clave1'″>club Escarbapedal</b>, amante de la naturaleza y todo lo que en ella se encuentra. Gracias por saber valorar mejor nuestro entorno, enseñar a valorarlo y cuidarlo.</p>
<p>Gracias especialmente a todos esos pueblos que forman las comarcas del <b id=″clave1″ OnMouseOver=″id='clave2';″ OnMouseOut=″id='clave1'″>Bajo Aragón </b>, y bellisimas <b id=″clave1″ OnMouseOver=″id='clave2';″ OnMouseOut=″id='clave1'″> Matarraña</b> y <b id=″clave1″ OnMouseOver=″id='clave2';″ OnMouseOut= ″id='clave1'″>Maestrazgo:</b></p>
```

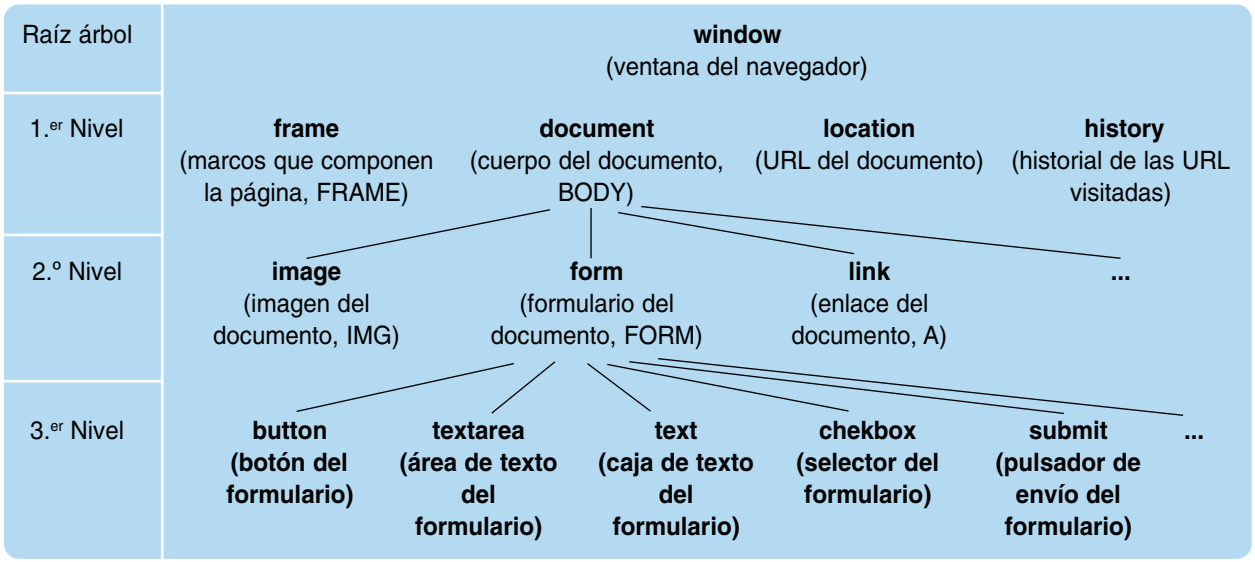
Según todo lo anterior, el documento HTML correspondiente a la página web “agradecimientos.html” quedaría de la siguiente forma:

```
<!— Contenido del documento HTML ″agradecimientos.html″ —>
<?xml version=″1.0″ encoding=″UTF-8″?>
<html><head>
<style type=text/css>
/* Importamos el fichero CSS asociado a las propiedades de estilo del resto del documento HTML definido ya en el capítulo anterior: ″css/plantilla.css″. */
        @import ″css/plantilla.css″ screen;
        div.division3 { width: auto; height: 800px;    background: yellow; position: relative; margin-left: 200px; margin-top: 0px; margin-right: 0px; text-align: justify; border-right:  2px solid red;     text-indent: 1cm;      padding: 1cm; padding-top: 0.5cm; }
        p:first-letter { font-size: xx-large; color: red; font-style: italic; }
        #clave1 { font-size: 16px; color: brown; text-transform: capitalize; }
        #clave2 { font-size: 18px; color: blue; text-transform: uppercase; }
</style>
<link href=″fotos/icono.png″ rel=″shortcut icon″ type=″image/png″>
</head>
<body>
/* La división del documento HTML ″division1″ se corresponde con la cabecera del documento, creada ya en el capítulo anterior. */
<div class=″division1″></div>
/* La división del documento HTML ″division2″ se corresponde con el menú principal, creada ya en el capítulo anterior. */
<div class=″division2″>
        <table border=″0″ cellpadding=″0″ cellspacing=″0″ width=″100%″>
        <tr><td  class=″enlace″  id=″tipo1″><a  href=″index.html″  id=″tipo1″>INICIO </a></td></tr>
        <tr><td  class=″enlace″  id=″tipo2″><a  href=″rutas.html″  id=″tipo2″>RUTAS BTT</a></td></tr>
        <tr><td class=″enlace″ id=″tipo1″><a href=″senderismo.html″ id=″tipo1″>SENDERISMO</a></td></tr>
        <tr><td  class=″enlace″  id=″tipo2″><a  href=″barrancos.html″  id=″tipo2″>BARRANCOS</a></td></tr>
        <tr><td  class=″enlace″  id=″tipo1″><a  href=″setas.html″  id=″tipo1″>  SETAS</a> </td></tr>
        <tr><td  class=″enlace″  id=″tipo2″><a  href=″alpinismo.html″  id=″tipo2″>ALPINISMO</a></td></tr>
```

```
        <tr><td class=˝enlace˝ id=˝tipo1˝><a href=˝otros.html˝ id=˝tipo1˝>OTROS
</a></td></tr>
        <tr><td class=˝enlace˝ id=˝tipo2˝><a href=˝agradecimientos.html˝ id=˝tipo2˝ >AGRA-
DECIMIENTOS</a></td></tr>
        <tr><td class=˝enlace˝ id=˝tipo1˝><a href=˝galeria.html˝ id=˝tipo1˝>GALERIA
FOTOS</a></td></tr>
        </table>
</div>
<div class=˝division3˝>
        <p>Agradecimientos a todos aquellas personas que de manera desinteresada han
hecho posible el <b id=˝clave1˝ OnMouseOver=˝id='clave2';˝ OnMouseOut=˝id='cla-
ve1'˝˝>club Escarbapedal</b>, amante de la naturaleza y todo lo que en ella se encuentra.
Gracias por saber valorar mejor nuestro entorno, enseñar a valorarlo y cuidarlo.</p>
        <p>Gracias especialmente a todos esos pueblos que forman las comarcas del <b
id=˝clave1˝ OnMouseOver=˝id='clave2';˝ OnMouseOut=˝id='clave1'˝˝>Bajo Aragón
</b>, y bellisimas <b id=˝clave1˝ OnMouseOver=˝id='clave2';˝ OnMouseOut= ˝id='cla-
ve1'˝˝>Matarraña</b> y <b id=˝clave1˝ OnMouseOver=˝id='clave2';˝
OnMouseOut=˝id='clave1'˝˝>Maestrazgo:</b></p>
        <ul style=˝list-style-position:inside; color:brown; font-size:18px; font-weight:800;˝>
                <li>Alcañiz</li>
                <li>Valdealgorfa</li>
                <li>Torrecilla de Alcañiz</li>
                <li>La Codoñera</li>
                <li>Beceite</li>
                <li>Valderrobres</li>
                <li>Ejulve</li>
                <li>Castellote</li>
                <li>...</li>
        </ul>
</div>
</body>
```

4. OBJETOS JAVASCRIPT

Como se ha comentado en el apartado anterior, en JavaScript el documento HTML es un objeto, "**document**", con sus propiedades y sus métodos. En realidad, para JavaScript todos los elementos que forman parte del cliente Web o navegador son objetos, desde la propia ventana que abre nuestro navegador, "**window**", hasta una simple imagen o enlace de nuestro documento, estructurándose estos de una manera jerárquica en un árbol invertido donde pueden diferenciarse en sus ramas diferentes niveles, donde su raíz es la ventana del navegador:



Para referenciar un elemento dentro del árbol, habrá que ir descendiendo de nivel mediante la notación, ".", y así poder hacer referencia a una de sus propiedades o métodos asociados. Es decir su sintaxis sería la siguiente:

window.document.form.textarea.*propiedad*

window.document.form.textarea.*metodo*

Un objeto de gran interés que es utilizado en JavaScript que no aparece en la lista anterior es "**event**" el cual nos puede dar información sobre el evento que se desencadena la acción. A continuación se indicarán las propiedades y métodos o funciones más importantes asociadas a los objetos más útiles:

OBJETO	Propiedades	Métodos
window	**name**: informa del nombre de la ventana. **location**: URL de la página cargada en la ventana del navegador. **innerHeight** | **innerWidth**: resolución vertical y horizontal en pixels del área de la ventana del navegador asociada a los contenidos Web, descontando las barras de navegación tareas y herramientas.	**alert(*"mensaje"*)**: abre una ventana de dialogo de alerta con el texto indicado. **confirm(*"texto"*)**: abre una ventana de dialogo de confirmación con el texto indicado. **prompt(*"pregunta"*,*"respuesta_defecto"*)**: abre una ventana de dialogo para preguntar al usuario, pudiendo este dar una respuesta. **open("URL","nombre","opciones")**: abre una nueva ventana del navegador bajo la URL indicada, permitiéndonos identificarla con un nombre, y controlar su apariencia con una lista de opciones separadas por comas. Entre las distintas opciones destacar: "fullscreen=yes|no, height=*pixels*, width=pixels, top=*pixels*, left=*pixels*, resizable=yes|no, scrollbars=yes|no, titlebar=yes|no, menubar=yes|no, toolbar=yes|no, location=yes|no, status=yes|no".
document	**URL**: URL del documento. title: titulo del documento. **domain**: nombre del dominio del documento. **cookie**: cookies asociadas al documento.	**getElementById(*"id"*)**: devuelve una referencia al primer elemento con el identificador indicado. **getElementsByName(*"name"*)**: devuelve la lista de elementos del documento que comparten el nombre indicado.

	anchors[i]: devuelve una referencia a todos los enlaces del documento. **images[i]**: devuelve una referencia a todos los enlaces del documento. **forms[i]**: devuelve una referencia a todos los formularios del documento. **links[i]**: devuelve una referencia a todos los enlaces y áreas del documento.	**getElementsByTagName(*"name"*)**: devuelve la lista de elementos con la etiqueta HTML indicada. **write(*"texto/etiquetas HTML/JavaScript"*)**: desde JavaScript nos permite escribir en el documento, texto, pudiendo incluir etiquetas HTML y funciones JavaScript.
history	**length**: informa del número de elementos almacenados en el historial del navegador.	**back**(): Carga en el navegador la URL anterior que aparece en la lista del historial. **forward**(): Carga en el navegador la siguiente URL que aparece en la lista del historial. **go**("numero|URL"): Carga en el navegador la URL correspondiente a la URL situada en la posición relativa del historial indicada. El número puede ser negativo o positivo, permitiendo hacer back o forward.
location	**href**: URL del documento visualizado. **pathname**: ruta de la URL del documento que se esta visualizando. **host**: nombre del equipo y puerto donde se visualiza el documento.	**assign**("URL"): carga el nuevo documento indicado por URL. **replace**("nuevaURL"): reemplaza el documento que se visualiza en el navegador por la URL indicada. **reload**(): recarga el documento.
event	**type**: informa del tipo de evento que desencadena la acción. Su valor puede ser: "**mouseover**", "**mouseout**", "**click**", etc. **currentTarget**: informa del elemento HTML u objeto donde se ha producido el evento. Nos permite hacer referencia a sus propiedades. Esta propiedad en Internet Explorer no funciona.	
screen	**height** | **width**: resolución vertical y horizontal de la pantalla en pixels. **availHeight** | **availWidth**: resolución vertical y horizontal de la pantalla en pixels descontando la barra de tareas del navegador. **colorDepth**: profundidad de color.	

4.1. Objeto JavaScript Style. Propiedades

Desde JavaScript podemos tanto obtener información sobre el valor de las propiedades de estilo CSS asignadas a un elemento HTML del documento, como asignarles el que deseemos. Para hacer referencia a estas propiedades de estilo, haremos uso del objeto "style" de JavaScript, donde el nombre de sus propiedades, son similares a las utilizadas en CSS (los caracteres guión, "-", se eliminan). Estas a su vez, al igual que se mostró en el capítulo de hojas de estilo CSS, se pueden dividir en distintas categorías en función a la parte del documento a la que hacen referencia. Para saber más sobre estas propiedades, ir al capítulo asociado a las hojas de estilo CSS, y buscar su propiedad correspondiente:

CATEGORIAS DE STYLE	PROPIEDADES DE ESTILO ASOCIADAS A CADA CATEGORIA
Propiedades **Background**	background backgroundAttachment backgroundColor backgroundImage backgroundPosition backgroundPositionX backgroundPositionY backgroundRepeat
Propiedades **Border** y **Margin**	border borderColor borderStyle borderWidth borderBottom borderBottomColor borderBottomStyle borderBottomWidth borderLeft borderLeftColor borderLeftStyle borderLeftWidth borderRight borderRightColor borderRightStyle borderRightWidth borderTop borderTopColor borderTopStyle borderTopWidth margin marginBottom marginLeft marginRight marginTop padding paddingBottom paddingLeft paddingRight paddingTop
Propiedades Texto y estilo de Fuente	font fontFamily fontSize fontStyle fontVariant fontWeight color letterSpacing lineHeight textAlign textDecoration textIndent textTransform whiteSpace wordSpacing
Propiedades Posicionamiento y Visuales	position top bottom left right zIndex display visibility overflow cursor direction width height
Propiedades asociadas a listas **List**	listStyle listStyleImage listStylePosition listStyleType

Ejercicio práctico n.º 2

En este segundo caso práctico dinamizaremos dos aspectos de nuestro sitio Web: 1) Personalizaremos el menú principal del sitio Web, haciendo que al pasar el ratón por encima de cada uno de los "items" del menú cambien de aspecto, modificándose tanto su fondo como el formato del texto. 2) Hacer una página Web asociada al "item" del menú descargas, formada por un conjunto de enlaces de descarga de documentos, de tal forma que al situar el ratón sobre el correspondiente enlace, aparezca sobre una área de texto una explicación informativa sobre el documento a descargar.

Figura 3.5. Pagina Web de descargas. Al pasar el ratón sobre cualquiera de los enlaces a descargar, se mostrará un texto informativo sobre la descarga

Solución ejercicio n.º 2

Para ello seguiremos los siguiente pasos:

1) Teniendo en cuenta que el menú principal esta formado por una tabla, <table> </table>, y que cada una de sus celdas, <td></td>, se corresponde con un "item" del menú, asociaremos a estas etiquetas HTML tres eventos JavaScript, de tal forma que al pasar el ratón por encima, "**OnMouseOver**", cambien sus propiedades de estilo asociadas, volviendo a sus propiedades iniciales al salir el ratón de su área de influencia, "**OnMouseOut**", y que al pinchar con el ratón sobre su área de influencia, "**OnClick**", cargue la página Web asociada al enlace.

```
<td class="enlace" id="item1" OnMouseOver OnMouseOut Onclick> enlace </td>
```

2) Al igual que en el ejemplo práctico anterior, el cambio de propiedades de estilo se producira tras detectarse el evento programado, modificando este el atributo "id", de "item1" e "item2", a "item3" e "item4", donde cada uno de los identificadores tendrá asociado un conjunto de propiedades de estilo en la sección de estilos, <style></style>, o incorporándolos en la hoja de estilo asociada a la página Web.

```
<style type=text/css>
td.enlace {
        font-weight: bold;
        padding-top: 5px;
        padding-bottom: 5px;
        text-align: center;
        cursor: pointer;
}
#item1 {
        background-color: green;
        color: white;
        font-weight: 400;
}
#item2 {
        background-color: orange;
        color: blue;
        font-weight: 400;
}
#item3 {
        background-color: Chartreuse;
        color: Crimson;
        font-weight: 900;
}
#item4 {
        background-color: gold;
        color: deeppink;
        font-weight: 900;
}
</style>
```

```
<td class="enlace" id="item1" OnMouseOver="id='item3'" OnMouseOut="id='item1'"
Onclick> enlace </td>
```

3) Teniendo en cuenta que la etiqueta HTML <td> no dispone como atributo "href" para poder asociarle una página enlazada al pinchar sobre él, haremos uso del objeto "**location**", y su propiedad "localtion.**href**". El mismo efecto se conseguiría haciendo uso de su método "**location.assign("enlace.html")**", o mediante el objeto "window" y su propiedad "window.**location**".

```
<td class="enlace" id="item1" OnMouseOver="id='item3'" OnMouseOut="id='item1'"
Onclick="location.href='enlace.html'"> enlace </td>
```

De esta forma, el menú principal del sitio Web, correspondiente a la división , quedaría de la siguiente forma:

```
<div class="division2">
    <table border="0" cellpadding="0" cellspacing="0" width="100%">
    <tr><td class="enlace" id="item1" OnMouseOver="id='item3';" OnMouseOut=
"id='item1';" Onclick="location.href='index.html'">INICIO</td></tr>
    <tr><td class="enlace" id="item2" OnMouseOver="id='item4';" OnMouseOut=
"id='item2';" Onclick="location.href='rutas.html'">RUTAS BTT</td></tr>
    <tr><td class="enlace" id="item1" OnMouseOver="id='item3';" OnMouseOut=
"id='item1';" Onclick="location.href='senderismo.html'">SENDERISMO</td></tr>
    <tr><td class="enlace" id="item2" OnMouseOver="id='item4';" OnMouseOut=
"id='item2';" Onclick="location.href='barrancos.html'">BARRANCOS</td></tr>
    <tr><td class="enlace" id="item1" OnMouseOver="id='item3';" OnMouseOut=
"id='item1';" Onclick="location.href='setas.html'">SETAS</td></tr>
    <tr><td class="enlace" id="item2" OnMouseOver="id='item4';" OnMouseOut=
"id='item2';" Onclick="location.href='alpinismo.html'">ALPINISMO</td></tr>
    <tr><td class="enlace" id="item1" OnMouseOver="id='item3';" OnMouseOut=
"id='item1';" Onclick="location.href='otros.html'">OTROS</td></tr>
    <tr><td class="enlace" id="item2" OnMouseOver="id='item4';" OnMouseOut=
"id='item2';" Onclick="location.href='agradecimientos.html'" >AGRADECIMIEN-
TOS</td></tr>
    <tr><td class="enlace" id="item1" OnMouseOver="id='item3';" OnMouseOut=
"id='item1';" Onclick="location.href='galeria.html'">GALERIA FOTOS</td></tr>
    <tr><td class="enlace" id="item2" OnMouseOver="id='item4';" OnMouseOut=
"id='item2';" Onclick="location.href='descargas.html'">DESCARGAS</td></tr>
    </table>
</div>
```

4) A continuación crearemos la lista de documentos a descargar, , y les asociaremos un el evento "**OnMouseOver**" de tal forma que al pasar el ratón sobre el correspondiente enlace, aparezca una caja de texto informando sobre el correspondiente enlace. Para ello, crearemos tanto párrafos de texto informativos, *Información sobre la descarga* , como documentos a descargar, identificándolos a través de su correspondiente atributo "**id**", donde mediante la propiedad de estilo "**display**" podemos controlar si la explicación correspondiente sea visible o no. También se muestra como modificar el color del texto del enlace al colocar el ratón sobre él, pero en lugar de hacer uso del evento "OnMouseOver", de la pseudo-clase "**:hover**" vista en el capítulo de hojas de

estilo CSS. A modo de ejemplo, para uno de los enlaces, quedaría de la siguiente forma, donde por defecto en la hoja de estilos se habrá declarado al párrafo informativo como no visible, "**display: none**":

```
/* Estilos añadidos en la hoja de estilo asociada a la página HTML*/
a.descarga:hover {
        color: blue;
        }
p.explicacion {
        display: none;
        border: 2px dotted brown;
        font-weight: 600;
        color: blue;
        padding: 0.2cm;
        text-align: justify;
        }

<ol style=˝list-style-position:inside; color:brown; font-size:16px; font-weight:800;˝>
<li><a class=˝descarga˝ href=˝pdf/cid.pdf˝ OnMouseOver=˝document.getElementById
('explicacion1').style.display='block';˝ OnMouseOut=˝document.getElementById('explica-
cion1').style.display='none';˝>Ruta del CID campeador.</a></li>
</ol>

<p class=˝explicacion˝ id=˝explicacion1˝>El CID campeador fue un guerrero mítico que ha
pasado a la historia por su conocido destierro, luchar por la reconquista de España, y ganar
su última batalla con una flecha clavada en su corazón ...</p>
```

Según todo lo anterior, la página Web asociada al link descargas del menú principal sería la siguiente:

```
<?xml version=˝1.0˝ encoding=˝UTF-8˝?>
<html>
<head>
        <script type=˝text/javascript˝ language=˝javascript˝ src=˝menu.js˝></script>
        <style type=˝text/css˝>
            @import ˝css/plantilla.css˝ screen;
            @import ˝css/plantilla.css˝ print;
        h3.titulo {
        text-align: center;
        text-decoration: blink;
        color: red;
        font-weight: 800;
        }
        a.descarga {
            text-decoration: none;
            color: brown;
            font-size: 16px;
            font-weight: 900;
        }
        a.descarga:hover {
            color: blue;
        }
        p.explicacion {
            display: none;
            border: 2px dotted brown;
            font-weight: 600;
```

```
            color: blue;
            padding: 0.2cm;
            text-align: justify;
        }
        </style>
<link href='fotos/icono5.png' rel='shortcut icon' type='image/png'>
</head>
<body>
<div class="division1"></div>
<div class="division2">
        <table border="0" cellpadding="0" cellspacing="0" width="100%">
        <tr><td class="enlace" id="item1" OnMouseOver="id='item3';" OnMouseOut=
"id='item1';" Onclick="location.href='index.html'">INICIO</td></tr>
        <tr><td class="enlace" id="item2" OnMouseOver="id='item4';" OnMouseOut="id=
'item2';" Onclick="location.href='rutas.html'">RUTAS BTT</td></tr>
        <tr><td class="enlace" id="item1" OnMouseOver="id='item3';" OnMouseOut=
"id='item1';" Onclick="location.href='senderismo.html'">SENDERISMO</td></tr>
        <tr><td class="enlace" id="item2" OnMouseOver="id='item4';" OnMouseOut=
"id='item2';"
<!—Demás Items del menú principal —>
        </table>
</div>
<div class="division3">
        <hr><h3 class="titulo">ZONA DE DESCARGAS</h3><hr>
        <p>Desde aquí podrás descargarte los documentos más importantes relacionados
con el mundo de la bicicleta:
        <ol style="list-style-position:inside; color:brown; font-size:16px; font-weight:800;">
            <li><a class="descarga" href="pdf/cid.pdf"
            OnMouseOver="document.getElementById('explicacion1').style.display='block';"

OnMouseOut="document.getElementById('explicacion1').style.display='none';">Ruta del
CID campeador.</a></li>
            <li><a class="descarga" href="pdf/viaplata.pdf"
            OnMouseOver="document.getElementById('explicacion2').style.display='block';"

OnMouseOut="document.getElementById('explicacion2').style.display='none';">Ruta de la
VIA de la PLATA.</a></li>
            <li><a class="descarga" href="pdf/camsantiago_frances.pdf"
            OnMouseOver="document.getElementById('explicacion3').style.display='block';"

OnMouseOut="document.getElementById('explicacion3').style.display='none';">Camino
Santiago Frances.</a></li>
<!—Demás Documentos a descargar de la lista —>
        </ol>
        </p>
        <p class="explicacion" id="explicacion1">El CID campeador fue un guerrero mítico
que ha pasado a la historia por su conocido destierro, luchar por la reconquista de España,
y ganar su última batalla con una flecha clavada en su corazón ...</p>
        <p class="explicacion" id="explicacion2">La Via de la Plata es una de las rutas más
fascinantes de España, caracterizada por las increibles dehesas andaluzas y extremeñas
...</p>
        <p class="explicacion" id="explicacion3">El Camino de Santiago Frances es sin
lugar a dudas la ruta BTT más entrañable que todo ciclista debe hacer ...</p>
<!—Demás Parrafos informativos asociados al resto de documentos a descargar de
la lista —>
</div>
</body>
```

5. VARIABLES Y FUNCIONES EN JAVASCRIPT

Al igual que en otros lenguajes de programación, JavaScript permite definirnos variables y funciones. Las primeras son útiles para poder almacenar temporalmente datos que podrán ser utilizados por las instrucciones JavaScript. En cuanto a las funciones, se corresponden con un bloque o conjunto de instrucciones JavaScript, identificadas por un nombre, que al ser invocadas desde cualquier parte del documento HTML son ejecutadas.

Las variables JavaScript, al igual que en otros lenguajes de programación reservan espacio en memoria con la finalidad de poder almacenar el tipo de dato que le indiquemos. Estos datos pueden ser de distintos tipos, destacando los de tipo "**number**" (números enteros o reales), "**boolean**" (verdadero o falso), "**string**" (cádena de caracteres alfanuméricos), "**object**" (objetos JavaScript de tipo fecha, formulario, etc.), o simplemente de tipo "**NULL**". Aunque lo cierto es que toda variable en JavaScript es implicitamente un objeto, a diferencia de los otros tipos de variables, para declarar una de tipo "**object**" será necesario hacer uso explicito del operador "**new**" seguido del tipo de objeto a crear: **Date**(), **Image**(), **Array**(), etc. De las variables de tipo "**object**" remarcar los "**arrays**" o listas de valores, las cuales nos permiten almacenar una lista de valores, predefinidos en el momento de la declaración, o a posteriori, haciendo referencia a la posición de la lista donde se quiere almacenar dicho valor, "***lista[posicion] = valor;***". Antes de utilizar una variable, debe haberse declarado previamente mediante la palabra reservada "**var**" seguida del nombre identificador de la variable: "**var** *nombre-variable = valor*;".

En JavaScript también existen constantes que pueden ser aprovechadas sin tener que definirlas. Entre todas ellas destacar las constantes matemáticas predefinidas como propiedades del objeto "**Math**": **Math.PI** (=3.1415...), **Math.E** (e=2.718), **Math.LN2** (ln(2)), **Math.SQRT2** (raíz de 2), etc.

Teniendo en cuenta que para JavaScript, todo elemento HTML es un objeto más que puede ser referenciado, pudiendo consultar el valor de sus propiedades e incluso modificarlas, es posible hacer uso del objeto "**this**", el cual nos permite hacer referencia sobre uno mismo.

TIPO DE DATO	EJEMPLO
number: integer o float	**var** contador = 0; **var** contador = **new Number** (0); **var** dospi = 2 * **Math.PI**;
boolean	**var** respuesta = true; **var** respuesta = **new Boolean** (true);
string	**var** correo = ″amartinr@educa.aragon.es″; var correo = **new String** (″amartinr@educa.aragon.es″);

object	**var** fecha = **new Date()**;
	var imagen = **new Image()**;
	var comunidades = new array (″Andalucia″, ″Aragón″, ..., ″La Rioja″);
	comunidades**[0]**=″Otra comunidad″;
	var valores = new array(2);
	valores**[0]** = 1025;
	valores**[1]** = -8.23;
	this.backgroundColor=″red″;

En JavaScript disponemos de la función "**typeOf**()" que nos informa del tipo de el objeto que se le suministra como parámetro. Según lo anterior, el valor devuelto por dicha función puede ser: *number*, *string*, *boolean*, *object*, *function* o *undefined*.

```
typeof(document) ? devuelve "object"
typeof(document.getElementById) ? devuelve "function"
typeof(typeof(document.getElementById)) ? devuelve "string"
```

En ocasiones nos interesará realizar operaciones aritméticas con los valores almacenados en las variables declaradas en JavaScript. En el caso de que el valor asignado a las variables involucradas en dicha operación aritmética, no haya sido a través en la misma declaración, sino que haya sido un valor obtenido a través de un formulario o envío desde una función, por defecto el tipo de dato almacenado será un string o cadena de caracteres, de hay que sea necesaria la utilización de funciones predefinidas en JavaScript encargadas de convertir dicha cadena en un número entero, "**parseInt(***cadena***)**", o a un número real, "**parseFloat(***cadena***)**". En el caso de que esta conversión no pueda llevarse a cabo, la función nos devolverá el valor "**NaN**" (Not a Number). En el caso de dudar si el tipo de dato almacenado en una variable es numérico o no, existe la función "**isNaN(***valor*|*variable***)**" *(is Not a Number)*, la cual devuelve un valor booleano: "true" en el caso de que no se trate de un dato numérico, y "false" en el caso de que si lo sea.

En cuanto al ámbito de utilización de la variable, abarca el área donde esta fue declarada, no siendo reconocida fuera de ella.

En relación a las funciones en JavaScript, las hay predefinidas, que nos permiten llevar a cabo acciones básicas sobre los diversos objetos JavaScript, a las cuales se les suele llamar **métodos**, y las hay definidas por el usuario, donde mediante la utilización de variables y funciones básicas predefinidas, podemos realizar acciones más complejas. Para implementar una función propia deberemos seguir la siguiente sintaxis:

```
function nombre-funcion (variable1, variable2, ...) {
        instrucciones JavaScript;
}
```

Donde *nombre-función* es el identificador de la función a través del cual la invocaremos desde cualquier parte del documento para que se ejecuten las instrucciones JavaScript que incluye. A esta función, de manera opcional, podemos pasarle un conjunto de valores que serán almacenados en las variables *variable1*, *variable2*, ... que podrán ser utilizadas por las instrucciones de la función.

Opcionalmente estas funciones pueden devolver algo a quien la invoco. Para ello se hace uso de la función "**return**(resultado)". Esta opción suele utilizarse cuando desde una función definida por el usuario se llama a otra de función también definida por el usuario, y se espera que la segunda le devuelva algo a la primera (ver ejemplo práctico n.º 4).

```
function nombre-funcion (variable1, variable2, ...) {
        instrucciones JavaScript;
        return(resultado);
}
```

La declaración de estas variables y funciones de usuario, dentro del documento HTML pueden incluirse de varias formas:

a) Dentro del propio documento HTML, haciendo uso de las etiqueta HTML doble "**<script> </script>**". El uso de estas etiquetas suele hacerse dentro de la sección

"**<head> </head>**" del documento del documento, aunque pueden utilizarse en cualquier otra parte del documento desde la cual se quiera realizar una invocación explicita a una función JavaScript:

```
<head>
<script type=´text/javascript´>
        // Declaración de variables globales a las funciones
        var variable-global1, variable-global2, ...;
        // Declaración de las funciones y sus variables locales
        function funcion1 (variable1, variable2, ...) {
        var variable-local1, variable-local2, ...;
         instrucciones Javascript;
        }
        function funcion2 (variable1, variable2, ...) {
        var variable-local1, variable-local2, ...;
         instrucciones Javascript;
        }
        ...
</script>
</head>
```

```
<body>
...
<script type=´text/javascript´> funcion1(valor1, valor2, ...); </script>
...
</body>
```

b) Fuera del documento HTML, mediante la creación de un fichero JavaScript con extensión "*.js", que posteriormente será importado desde el propio documento haciendo uso de la etiqueta doble ?**<script src="*.js"> </script>**", donde su atributo "src" se encargará de indicar la ruta relativa al documento HTML donde se localiza el fichero JavaScript a importar:

```
/* Contenido del fichero *.js */
//Declaración de las variables globales a las funciones declaradas en *.js
var variable-global1, variable-global2, ...;
// Declaración de las funciones y sus variables locales
function funcion1 (variable1, variable2, ...) {
        var variable-local1, variable-local2, ...;
        instrucciones Javascript;
}
function funcion2 (variable1, variable2, ...) {
        var variable-local1, variable-local2, ...;
        instrucciones Javascript;
}
...

<head>
...
<script type=´text/javascript´ src="*.js"> </script>
...
</head>
```

6. OPERADORES Y ESTRUCTURAS DE CONTROL DE FLUJO

Para poder encauzar el flujo de ejecución de las instrucciones dentro de una función es posible hacer uso de alguna de las clásicas estructuras de control de flujo:

if if...else if...else if ... else	**for**
if (condición) { *instrucciones JavaScript;* } else if (condición) { *instrucciones JavaScript;* } else { *instrucciones JavaScript;* }	for (variable-contador = valor; condición; incremento) { *instrucciones JavaScript;* }
while	**do ... while**
while (condición) { *instrucciones JavaScript;* }	do { *instrucciones JavaScript;* } while (condición)
for ... in	**switch**

for (var variable in lista)
{ *instrucciones JavaScript;* }
	switch (expresión)
	{
	case *caso1*: *instrucciones JavaScript;*
	break;
	case *caso2*: *instrucciones JavaScript;*
	break;
	...
	default: *instrucciones JavaScript;*
	}

Para el establecimiento de las condiciones utilizadas en el control de flujo, suelen utilizarse alguno de los operadores siguientes:

Tipo de Operadores	**Operadores**		
Aritméticos	*+ suma* *- resta y cambio de signo* ** multiplicación*	*/ división* *\ división entera* *% resto de la división*	*++ incremento* *— decremento*
Relacionales	*== igual a* *!= distinto a*	*> mayor que* *>= mayor o igual que*	*< menor que* *<= menor o igual que*
Lógicos	*&& and*	*|| or*	*! negación*
Cadena de caracteres	*+ concatena valores y strings*		

Ejercicio práctico n.º 3

Teniendo en cuenta que el menú principal de nuestro sitio Web debe estar presente en todas las páginas Web que lo componen, tenemos dos opciones: a) Incluir el código HTML asociado a la tabla que hace de menú principal en cada una de las páginas, o b) hacer un "script" externo en lenguaje JavaScript encargado de imprimir un tabla equivalente a la del menú principal, e importarlo desde cada una de las páginas HTML donde se quiere utilizar, garantizando una reutilización del código. Teniendo en cuenta que la reutilización del código es deseable en programación con la finalidad de minimizar líneas de código, en este ejemplo práctico veremos como hacerlo.

Solución ejercicio n.º 3

1) Creamos mediante un editor de texto un documento llamado "menu.js". Este fichero se corresponderá con un script en lenguaje JavaScript correspondiente a la generación del menú principal del sitio Web. El script comienza definiendo tantas variables como enlaces hay en el menú a las distintas páginas Web que forman parte del sitio Web.

```
var enlace1=˝index.html˝, enlace2=˝rutas.html˝, enlace3=˝senderismo.html˝,
enlace4=˝barrancos.html˝, enlace5=˝setas.html˝, enlace6=˝otros.html˝, enlace7=˝agrade-
cimientos.html˝, enlace8=˝galeria.html˝, enlace9=˝descargas.html˝;
```

2) Creamos la función "**enlazar(enlace)**". Será invocada al pinchar sobre cualquiera de los items del menú, "OnClick", a la cual se le pasará como parámetro la variable correspondiente al enlace que tiene que cargarse en el navegador. Para ello se hace uso del objeto JavaScript "location", donde su propiedad "href", nos permite indicar la URL que se carga en el navegador.

```
function enlazar(enlace) {
        location.href=enlace;
}
```

3) Crearemos la función "**cambiar(objeto,evento)**". Será invocada tanto al pasar el ratón sobre cualquiera de los items de menú, "OnMouseOver", como al salir de su área de influencia, "OnMouseOut", pasándole como parámetros en elemento HTML u objeto donde se produce el evento, y el tipo de evento que la ha producido, con la finalidad de distinguir entre los distintos items del menú, y el evento que la invoca. En concreto, esta función se va a encargar de realizar el cambio de estilo y formato del item de menú al pasar el ratón sobre él, tal como se ha hecho hasta ahora en los ejemplos prácticos anteriores. Para ello, la función consulta el valor del atributo "id" del item que la invoca, para saber cual es su estilo asociado en la hoja de estilos, y saber de esta forma cual le corresponde. Para conocer cual de los dos eventos invocó a la función, se hace uso de la propiedad "**type**" del objeto "**event**", cuyos posibles valores coincide con el nombre de los eventos sin el prefijo "On".

```
function cambiar(objeto,evento) {
  if (evento.type == ″mouseover″ && objeto.id == ″item1″) {
  objeto.id=″item3″; }
  if (evento.type == ″mouseover″ && objeto.id == ″item2″) {
  objeto.id=″item4″; }
  if (evento.type == ″mouseout″ && objeto.id == ″item3″) {
  objeto.id=″item1″; }
  if (evento.type == ″mouseout″ && objeto.id == ″item4″) {
  objeto.id=″item2″; }
}
```

4) Creamos la función principal "**menu_principal()**" encargada de crear el menú. Es invocada desde el documento HTML, desde la posición donde debería mostrarse el menú. Desde esta función se llamará a las funciones anteriores. Para ello se hace uso de la función o método "**write**" asociado al objeto "**document**", la cual nos permite desde JavaScript escribir tanto etiquetas HTML como JavaScript en el documento. Según esto el fichero "menu.js" quedaría de la siguiente forma:

```
/* Contenido del SCRIPT JavaScript ″menu.js″ */
var enlace1=″index.html″, enlace2=″rutas.html″, enlace3=″senderismo.html″, enlace4=″barrancos.html″, enlace5=″setas.html″, enlace6=″otros.html″, enlace7=″agradecimientos.html″, enlace8=″galeria.html″, enlace9=″descargas.html″;
function cambiar(objeto,evento) {
        if (evento.type == ″mouseover″ && objeto.id == ″item1″) {
        objeto.id=″item3″; }
        if (evento.type == ″mouseover″ && objeto.id == ″item2″) {
        objeto.id=″item4″; }
        if (evento.type == ″mouseout″ && objeto.id == ″item3″) {
        objeto.id=″item1″; }
        if (evento.type == ″mouseout″ && objeto.id == ″item4″) {
        objeto.id=″item2″; }
}
function enlazar(enlace) { location.href=enlace; }
function menu_principal() {
document.write('<table border=″0″ cellpadding=″0″ cellspacing=″0″ width=″100%″>');
document.write('<tr><td class=″enlace″ id=″item1″ OnMouseOver=″cambiar (this,event);″ OnMouseOut=″cambiar(this,event);″ OnClick=″enlazar(enlace1);″>INICIO</td></tr>');
document.write('<tr><td class=″enlace″ id=″item2″ OnMouseOver=″cambiar (this,event);″ OnMouseOut=″cambiar(this,event);″OnClick=″enlazar(enlace2);″>RUTAS BTT</td></tr>');
document.write('<tr><td class=″enlace″ id=″item1″ OnMouseOver=″cambiar (this,event);″ OnMouseOut=″cambiar(this,event);″OnClick=″enlazar(enlace3);″>SENDERISMO</td></tr>');
document.write('<tr><td class=″enlace″ id=″item2″ OnMouseOver=″cambiar (this,event);″ OnMouseOut=″cambiar(this,event);″OnClick=″enlazar(enlace4);″>BARRANCOS</td></tr>');
document.write('<tr><td class=″enlace″ id=″item1″ OnMouseOver=″cambiar (this,event);″ OnMouseOut=″cambiar(this,event);″ OnClick=″enlazar(enlace5);″>SETAS </td></tr>');
```

```
document.write('<tr><td class="enlace" id="item2" OnMouseOver="cambiar (this,event);"
OnMouseOut="cambiar(this,event);" OnClick="enlazar(enlace6);">OTROS </td></tr>');
document.write('<tr><td class="enlace" id="item1" OnMouseOver="cambiar(this,event);"
OnMouseOut="cambiar(this,event);"OnClick="enlazar(enlace7);">AGRADECIMIENTOS
</td></tr>');
document.write('<tr><td class="enlace" id="item2" OnMouseOver="cambiar(this,event);"
OnMouseOut="cambiar(this,event);"OnClick="enlazar(enlace8);">GALERIA FOTOS
</td></tr>');
document.write('<tr><td class="enlace" id="item1" OnMouseOver="cambiar(this,event);"
OnMouseOut="cambiar(this,event);"OnClick="enlazar(enlace9);">DESCARGAS </td></tr>');
document.write('</table>');
}
```

5) Por último, sería necesario importar el fichero "menu.js" desde el documento HTML que quiere hacer uso de las funciones creadas, e invocar a la función "menu_principal()" encargada de generar el menú desde la posición del documento correspondiente al menú:

```
<?xml version="1.0" encoding="UTF-8"?>
<html>
<head>
    <!—Importamos el script JavaScript "menu.js" indicando su ubicación en
"src" —>
    <script type="text/javascript" src="menu.js"></script>
    <!—Importamos las hojas de estilos. En ellas deben estar definidos los esti-
los para "item1", "item2", "item3", y "item4", utilizados en el script "menu.js" en
la generación del menu principal —>
    <style type="text/css">
     @import "css/plantilla.css" screen;
     @import "css/plantilla.css" print;
    </style>
<link href='fotos/icono.png' rel='shortcut icon' type='image/png'>
</head>
<body>
<div class="division1"></div>
 <div class="division2">
    <script type="text/javascript" language="javascript">
     menu_principal();
    </script>
</div>
<div class="division3">
    <!—Contenido de la division3 —>
</div>
</body>
```

7. FUNCIONES PREDEFINIDAS DE JAVASCRIPT

JavaScript dispone de multitud de funciones y métodos ya predefinidos que podemos utilizar en nuestros programas escritos en JavaScript, ahorrándonos trabajo. Estas las podemos clasificar en función de los objetos JavaScript a las que están asociadas, destacándose las siguientes:

Funciones JavaScript asociadas al objeto **String**

length: propiedad de un objeto string que informa del número de caracteres que forman la cadena de caracteres.

small() | **big**(): asigna un tamaño pequeño o grande a la cadena de caracteres.

bold() | **italic**() | **blink**() | **strike**(): decoran el texto permitiendonos ponerlo en negrita, cursiva, parpadeando o tachado, respectivamente.

fontsize(tamaño) | **fontcolor**(″color″): formatean el texto permitiendo asignar un tamaño y color al string.

toLowerCase() | **toUpperCase**(): transforma la cadena de caracteres poniendo todos los caracteres todos en minúscula o mayúsculas respectivamente.

sub() | **sup**(): transforma la cadena de caracteres en un subíndice o superíndice respectivamente.

link(″url″): convierte el string en un enlace HTML, <a href>, a la URL indicada.

search(/patron/[i]): busca el patrón especificado en la cadena de caracteres devolviendo la posición donde se encontro. La opción ″i″ indica que no distinga entre mayúsculas y minúsculas. Devuelve un ″-1″ en caso de no encontrarse.

IndexOf(″patron″,[desde]) | **lastIndexOf**(″patron″,[desde]): informa de la posición dentro de la cadena donde se encuentra el patrón indicado. Distingue entre mayúsculas y minúsculas. Devuelve un ″-1″ en caso de no encontrarse.

substr(posini,[cantidad]): extrae una subcadena del string a partir de la posción indicada en ″posini″. Opcionalmente puede indicarse cuantos caracteres extraer.

substring(posini,[posfin]): extrae una subcadena del string a partir de la posción indicada en ″posini″. Opcionalmente puede indicarse la posición final.

replace(/cadena1/[i],cadena2): busca la cadena1 en el string y la substituye por la cadena2. La opción ″i″ indica que no distinga entre mayúsculas y minúsculas.

split(″delimitador″,[ntrozos]): rompe la cadena de caracteres en trozos utilizando como delimitador el carácter especificado. Opcionalmente puede indicarse el número de trozos a recuperar.

concat(cadena1, ..., cadenaN): concatena las cadenas de caracteres indicadas al string. Ofrece el mismo resultado que el operador de concatenación ″**+**″.

Ejemplos:

```
<?xml version=″1.0″ encoding=″UTF-8″?>
<html><head> ... </head>
<body>
<script type=″text/javascript″>
var nombre = new String (″Arturo Martín Romero″);
var correo = new String (″amartinr@educa.aragon.es″);
var dni = new String (″73123765A″);
var direccion = new String (″C/ Don Pelayo:19 – 6ºB:Alcañiz:Teruel″);
var trozos_direccion = new Array ();
var mienlace = new String (″Sitio Web IngeMartin″);
document.write(″<hr>″ + nombre.bold().italics().fontsize(20).blink().fontcolor(″Blue″) + ″<hr>″);
document.write(″Para cualquier duda, ponerse en contacto con: ″ + correo.bold().fontcolor
(″Orange″));
```

```
/* A partir de "amartinr@educa.aragon.es" generamos otra dirección de correo,
"otrocorreo@educa.aragon.es" reemplazando "amartinr" por "otrocorreo". */
document.write("<br>También puedes ponerte en contacto con: " + correo.replace(correo.substring(0,correo.indexOf("@")),"otrocorreo"));
/* Troceamos la cadena de caracteres asociada a la variable dirección utilizando como parámetro
delimitador ":". Cada uno de los trozos de "dirección" será un componente del Array trozos_direccion. */
trozos_direccion = direccion.split(":");
document.write("<br>La dirección de " + correo + " es " + trozos_direccion[0] + ", Nº" + trozos_direccion[1] + ". " + trozos_direccion[2] + " - " + trozos_direccion[3]);
/* Con la propiedad "length" calculamos la longitud de la cadena "dni", y "link" crea un enlace. */
document.write("<br>La longitud del dni " + dni + " es " + dni.length + ".");
document.write("También puedes visitar la Web" + mienlace.link("http://www.ingemartin.es") + ".")
</script>
</body></html>
```

Funciones JavaScript asociadas al objeto **Math**

Math.round(x) | **Math.floor**(x) | **Math.ceil**(x): redondea un número al entero más cercano, al entero menor más cercano o al entero mayor más cercano respectivamente.
Math.random(): genera un número aleatorio entre 0 y 1.
Math.abs(x): devuelve el valor absoluto del número indicado.
Math.min(x,y) | **Math.max**(x,y):
Math.sin(x) | **Math.cos**(x) | **Math.tan**(x) | **Math.asin**(x) | **Math.acos**(x) | **Math.atan**(x): devuelven el seno, coseno, tangente, arcoseno, arcocoseno, y arcotangente del número indicado.
Math.sqrt(x): devuelve la raíz cuadrada del número indicado.
Math.pow(x,y): devuelve el valor de "x" elevado a "y".
Math.exp(x): devuelve el resultado de elevar el número de Euler "e" a "x".
Math.log(x): devuelve el logaritmo natural del número indicado.

Ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<html><head> ... </head>
<body>
<script type="text/javascript">
/* Calculamos el área y el perimétro del circulo: */
document.write ("El valor de una onda senoidal en 0, 90, 180 y 360 grados es: " + Math.round( Math.sin(0) ) + " - " + Math.round (Math.sin (Math.PI/2)) + " - " + Math.round (Math.sin (Math.PI)) + " - " + Math.round (Math.sin (2*Math.PI)) + ".");
</script>
</body></html>
```

Funciones JavaScript asociadas al objeto **Date**

Date(): haciendo uso del operador "new" permite crear un objeto de tipo fecha, con la fecha y hora actuales. En caso de querer asignar valores diferentes a los actuales se seguirá el formato: "*Date(año,mes,día,hora,minutos,segundos)*".
getDay(): devuelve el día de la semana (0-6) de un objeto de tipo Date().
getDate() | **getMonth**() | **getFullYear**(): devuelven el día del mes (1-31), el mes (0-11) y el año con cuatro dígitos, de un objeto de tipo Date().

getHours() | **getMinutes**() | **getSeconds**() | **getMiliseconds**(): devuelven la información de la hora (0-23), los minutos (0-59), los segundos (0-59) y los milisegundos (0-999), de un objeto de tipo Date().

getUTCDay() | **getUTCDate**() | **getUTCMonth**() | **getUTCFullYear**() | **getUTCHours**() | **getUTCMinutes**() | **getUTCSeconds**() | **getUTCMiliseconds**(): igual que las funciones anteriores pero de acuerdo al tiempo universal coordinado (UTC).

getTime(): devuelve el número de milisegundos transcurridos desde el 1 de enero de 1970 hasta la fecha y hora indicadas en un objeto de tipo Date().

setDay() | **setDate**() | **setMonth**() | **setFullYear**() | **setHours**() | **setMinutes**() | **setSeconds**() | **setMiliseconds**() | **setUTCDay**() | **setUTCDate**() | **setUTCMonth**() | **setUTCFullYear**() | **setUTCHours**() | **setUTCMinutes**() | **setUTCSeconds**() | **setUTCMiliseconds**(): asigna un día de la semana (0-6), un día del mes (0-31), un mes (0-11), un año con cuatro dígitos, una hora (0-23), unos minutos (0-59), unos segundos (0-59), o milisegundos (0-999), a un objeto de tipo Date().

toString(): convierte un objeto de tipo Date() en un string.

Ejemplos:

```
<?xml version=˝1.0˝ encoding=˝UTF-8˝?>
<html><head> ... </head>
<body>
<script type=”text/javascript”>
var nombre = new String (”Arturo Martín Romero”);
/* Obtenemos la fecha y hora acutuales y se la asignamos a la variable ”fecha” */
var fecha = new Date ();
/* Asignamos la fecha de nacimiento de la persona.  Se puede hacer de dos formas: */
var fechanacimiento = new Date (1977,9,13);
/* La Asignación de fecha anterior es equivalente a: */
fechanacimiento.setDate(13);
fechanacimiento.setMonth(9);
fechanacimiento.setFullYear(1977);
/* Calculamos la edad de la persona.  Para ello hacemos uso de la función ”getTime()”, que nos proporciona el tiempo transcurrido en milisegundos en relación al 1 de enero de 1970. */
document.write (˝<br>Tu edad de ” + nombre + ” es: ˝ + Math.floor((fecha.getTime() - fechanacimiento.getTime()) / (1000 * 60 * 60 * 24 * 365)));
</script>
</body></html>
```

Funciones JavaScript asociadas al objeto **Array**

length: propiedad de un Array que nos informa del número de elementos que lo forman.

pop(): devuelve el último elemento de un Array, y lo elimina del Array.

shift(): devuelve el primer elemento de un Array, y lo elimina del Array.

push(nuevoele1, nuevoele2, ...): Añade uno o más elementos al final del Array, y devuelve su nueva longitud.

unshift(nuevoele1, nuevoele2, ...): Añade uno o más elementos al inicio del Array, y devuelve su nueva longitud.

reverse(): cambia el orden de los elemento de un Array.

sort(): devuelve los componentes de un Array de manera ordenada.

join(”delimitador”): devuelve un string con todos los elementos del Array unidos a traves del carácter delimitador indicado.

concat(Array1, Array2, ...): concatena Arrays en uno.

toString(): convierte un array en una cadena de caracteres.

Ejemplos:

```
<?xml version=″1.0″ encoding=″UTF-8″?>
<html><head> ... </head>
<body>
<script type=″text/javascript″>
var grupos = new Array (″1ºA ESO″, ″1ºB ESO″, ..., ″2º Bachillerato Letras″);
document.write (″<br>El número de grupos es: ″ + grupos.length);
/* La función ″push()″ añade elementos a la lista y nos devuelve su nueva longitud. */
document.write (″<br>Añadiendo dos grupos a la lista pasan a ser: ″ + grupos.push(″1º ESI GM″,″2º ESI GM″));
document.write (″<br>La lista ordenada de grupos es: ″ + grupos.sort());
/* La función ″pop()″ extrae el último elemento de la lista, y lo elimina de la lista. */
document.write (″<br>El último de la lista es: ″ + grupos.pop());
document.write (″<br>Ahora son: ″ + grupos.length);
</script>
</body></html>
```

Otras Propiedades y Funciones JavaScript

innerHTML: propiedad que permite tanto informar como asignar un contenido en formato HTML a un elemento HTML del documento.
ParseInt(): devuelve el resultado de convertir en un número entero el valor del objeto afectado.
ParseFloat(): devuelve el resultado de convertir en un número real el valor del objeto afectado.
Number(): convierte el valor de un objeto en número.
String(): convierte en una cadena de caracteres el objeto indicado.
isFinite(): comprueba si una cantidad es finita.
typeOf(): devuelve un string informando del tipo de objeto. Este puede ser *number*, *string*, *boolean*, *object*, *function* o *undefined*.

Ejemplos:

```
<?xml version=″1.0″ encoding=″UTF-8″?>
<html><head> ... </head>
<body>
<hr><p id=″saludos″>Aqui va el saludo JavaScript</p><hr>
<script type=″text/javascript″>
var usuario = new String();
/* La función o método ″prompt″ del objeto ″window″ muestra una ventana permitiendo preguntar al usuario por algo, pudiendo este contestar insertando texto en una caja de texto. El valor insetado es devuelto por esta función. */
usuario=window.prompt(″Indica cual es tu nombre:″);
/* Personalizamos el contenido del mensaje de bienvenida de la página Web, modificando el contenido del párrafo HTML. */
document.getElementById(″saludos″).innerHTML=″Bienvenido a la Web D. ″ + usuario.bold(). fontcolor(″blue″);
</script>
</body></html>
```

Ejercicio práctico n.º 4

Diseñar una página Web referente a una galería de imágenes. Esta estará asociada al enlace del menú principal "Galería Fotos". Tal como hacen las modernas galerías de imágenes, la página mostrará todas las fotos incluidas en la galería en tamaño pequeño, comúnmente llamadas "thumbnail". Al pinchar sobre cualquiera de las imágenes de muestra, aparecerá un velo que oscurecerá el fondo de la página Web, dejando la imagen seleccionada en el medio de la pantalla con una resolución aceptable, a modo de visor de fotos. Debajo de la imagen se mostrará un texto explicativo personalizado para cada una de las imágenes. Al pinchar sobre ella, automáticamente nos mostrará la siguiente imagen de la galería. Para salir del visor de imágenes, y volver a la página Web, deberá pincharse con el ratón fuera del área ocupada por la imagen.

Figura 3.6. Aspecto de la Galería de imágenes. Al pinchar sobre uno de los thumbnails la foto se agranda, permitiéndonos ver el resto de imágenes de la galería, pinchando sobre la foto

Solución ejercicio n.º 4

¡¡Sugerencia!! Con la finalidad de que no le cueste cargar mucho a la galería de fotos, es conveniente que las imagenes sean de baja resolución. Para poder redimensionar el tamaño de una imagen a un valor deseado o generar "thumbnails" puede hacerse uso de alguna herramienta software de tratamiento de imágenes como pueden ser PhotoShop o GIMP, pero también es posible utilizar otro tipo de herramientas que nos agilizarán el trabajo en operaciones tan sencillas como esta. Este es caso del comando "**convert**" del paquete software "**ImageMagick**", disponible tanto en entornos Microsoft Windows como GNU/Linux (*http://www.imagemagick.org*). Mediante "**convert**" se puede ajustar el tamaño en unos pocos segundos, invirtiendo menos tiempo del que cuesta por ejemplo arrancar a "PhotoShop". El comando en cuestión sería:

```
[usuario@linux] convert -resize anchoxalto imagen-original imagen-ajustada
[usuario@linux] convert -resize 400x150 logo.gif logo-fondo.gif
[usuario@linux] convert -resize 150x110 foto.jpg foto-thumbnail.gif
```

1) La estructura de la página Web asociada a la galería de fotos estará dividida como hasta ahora: una primera división donde se pone el "logo" del sitio Web, una segunda división correspondiente al menú principal del sitio Web, y una tercera división donde se ubican los contenidos de la página, formada en este caso por un título y las imágenes thumbnails de las fotos de la galería. El contenido HTML de esta tercera división del documento, , quedaría así (*en el ejemplo, la galería esta formada por 8 fotos ubicadas dentro de una subcarpeta "fotos"*):

```
<div class="division3">
<hr><h3 class="titulo">GALERIA DE FOTOS</h3><hr>
<p>Galeria de imagenes del club Escarbapedal.</p>
<p class="fotos" style="text-align: center; text-indent: 0cm;">
<img class="thumbnail" src="fotos/thumbnail-foto1.jpg"
OnMouseOver="style.cursor='pointer'" Onclick="ampliar('1');">
<img class="thumbnail" src="fotos/thumbnail-foto2.jpg"
OnMouseOver="style.cursor='pointer'" Onclick="ampliar('2');">
<img class="thumbnail" src="fotos/thumbnail-foto3.jpg"
OnMouseOver="style.cursor='pointer'" Onclick="ampliar('3');">
<img class="thumbnail" src="fotos/thumbnail-foto4.jpg"
OnMouseOver="style.cursor='pointer'" Onclick="ampliar('4');">
<img class="thumbnail" src="fotos/thumbnail-foto5.jpg"
OnMouseOver="style.cursor='pointer'" Onclick="ampliar('5');">
<img class="thumbnail" src="fotos/thumbnail-foto6.jpg"
OnMouseOver="style.cursor='pointer'" Onclick="ampliar('6');">
<img class="thumbnail" src="fotos/thumbnail-foto7.jpg"
OnMouseOver="style.cursor='pointer'" Onclick="ampliar('7');">
<img class="thumbnail" src="fotos/thumbnail-foto8.jpg"
OnMouseOver="style.cursor='pointer'" Onclick="ampliar('8');">
</p>
</div>
```

Donde las propiedades de estilo definidas en la hoja de estilos CSS asociadas a las imágenes thumbnails son:

```
/*Propiedades de estilo declaradas dentro de "galeria.css" asociadas a los thumbnail.*/

img.thumbnail { border: 1px solid black; background-color: white; padding: 4px; width: 150px; height: 110px; }
```

Para que las fotos de la galería aparezcan centradas, se han introducido dentro de un párrafo HTML, , y se ha hecho uso de la propiedad de estilo "**text-align: center**", que por defecto centra todo su contenido, y se ha eliminado la tabulación por defecto definida en la hoja de estilos con "**text-indent: 0cm;**". En relación a los eventos declarados sobre el objeto <img>, "OnMouseOver" cambia el estilo del puntero del ratón con la finalidad de que el usuario al pasar el ratón sobre ella este cambio le incite a pinchar sobre la imagen, y "OnClick", de tal forma que al pinchar sobre una de las imagenes, se invocará a la función JavaScript "ampliar('num_imagen');", pasándole como parámetro el número de la imagen a ver ampliada.

2) Al igual que en el ejercicio práctico anterior, el menú principal de la página Web se carga a través de una llamada a una función JavaScript ubicada en un fichero JavaScript externo, "menu.js". De manera similar haremos con las funciones JavaScript usadas en la galería de imágenes encargadas de aportar el dinamismo ubicadas en otro fichero externo que llamaremos "galeria.js". Ambos ficheros, ubicados en una subcarpeta llamada "js/" deberán ser importados.

En cuanto a los estilos asignados a los distintos elementos que componen la página, los definiremos en ficheros CSS externos que deberán ser importados: "galeria.css" define los estilos utilizados en la galería de imágenes, y "plantilla.css" el resto de estilos utilizados en el resto del documento, ya utilizados en el resto de páginas que componen el sitio Web. La importación de todos estos ficheros externos, "*.js" y "*.css", será declarada en la cabecera del documento Web "galeria.html" quedando de la siguiente forma:

```
<!—Cabecera del documento HTML "galeria.html" —>
<head>
    <!—Importamos el fichero JavaScript "menu.js" que crea el menú principal —>
    <script type="text/javascript" src="js/menu.js"></script>
    <!—Importamos el fichero JavaScript "galeria.js" que dinamiza la galeria —>
    <script type="text/javascript" src="js/galeria.js"></script>
    <style type="text/css">
    <!—Importamos los ficheros CSS donde se definen los estilos del documento —>
                    @import "css/plantilla.css" screen;
                    @import "css/galeria.css" screen;
    </style>
<link href='fotos/icono.png' rel='shortcut icon' type='image/png'>
</head>
```

3) Al pinchar sobre cualquiera de las imágenes thumbnail de la galería, aparecerá una nueva capa superpuesta a los contenidos de la página Web que ocultará su contenido, garantizando que centremos la vista en la imagen que se mostrará en el centro de la pantalla, en tamaño grande y con una resolución aceptable.

Figura 3.7. Tras pinchar sobre un thumbnail, se visualizará la imagen centrada en tamaño grande, sobre una capa semitransparente que ocultará el fondo

La capa que se encargará de hacer de "velo" semitransparente ocultando el fondo y resaltando la imagen central, se corresponderá con una etiqueta HTML que ocupará toda la pantalla.

```
/*Propiedades de estilo declaradas dentro de "galeria.css" asociadas a la capa que hace de velo.*/
#velo { z-index: -1; background-color: gray; position: fixed; width: 100%; height: 100%; top: 0px; left: 0px; opacity: 0.8; filter: alpha(opacity=80); }
```

```
<!—División o capa creada en "galeria.html" que hará de "velo"—>
<div id=˝velo˝ onclick=˝salir();˝></div>
```

Como puede observarse en las propiedades de estilo asociadas a la división identificada como "velo":

— **z-index: -1**": Por defecto, la capa queda ocultada por debajo del resto de divisiones de la página Web. Al pinchar sobre cualquiera de las fotos de la galería, desde JavaScript, modificaremos el valor de este parámetro a "z-index: 1", pasando de quedar por debajo, oculta, a quedar superpuesta por encima, ocultando lo que queda detrás.

¡¡Recuerda!! La propiedad de estilo "z-index" permite colocar los elementos que componen el documento en diferentes capas. Es decir, "z-index" asume que existe un tercer eje de coordenadas, eje Z, perpendicular al plano formado por los ejes X e Y, donde su valor asignado a un elemento del documento indica la posición ocupada sobre este eje provocando que determinados elementos queden por encima o por debajo.

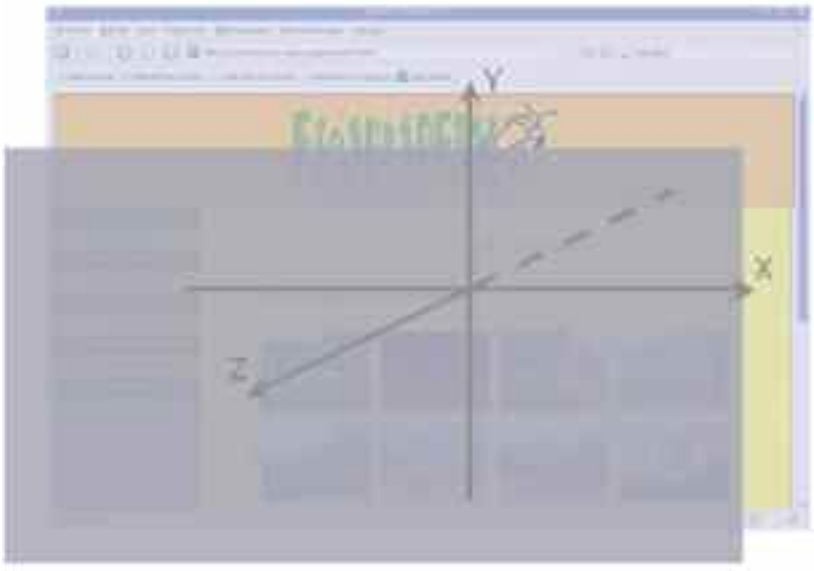


Figura 3.8. z-index permite posicionar los elementos HTML en diferentes capas sobre un eje Z

— **position: fixed; width: 100%; height: 100%; top: 0px; left: 0px;**": Para que el "velo" oculte lo que hay por debajo, posicionaremos la capa de manera fija, ?**position: fixed;**", sobre las coordenadas (x,y)=(0,0), "**top: 0px; left: 0px;**", ocupando toda la pantalla, "**width: 100%; height: 100%;**".

— **opacity: 0.8; filter: alpha(opacity=80);**": Para que el velo no oculte totalmente lo que queda por debajo, y haga un efecto de semitransparencia. Esta propiedad de estilo sólo funcionará bajo los navegadores Web Mozilla Firefox, "**opacity: 0.8;**" e Internet Explorer, "**filter: alpha(opacity=80);**", no siendo reconocidas estas propiedades por otros clientes Web.

4) Una última división o capa, , se encargará de mostrar la imagen seleccionada en tamaño grande y de mayor resolución, junto con un texto informativo asociado a la foto. Al igual que la capa "velo", ésta por defecto también quedará oculta, "**z-index: -1;**" visualizándose únicamente al pinchar sobre uno de los thumbnails. En relación a los comentarios, serán gestionados desde JavaScript mediante una función, mostrándose en cada momento el que corresponda.

```
<!—División o capa creada en "galeria.html" encargada de mostrar la foto seleccionada—>
<div id=˝capaimagen˝>
<img id=˝mifoto˝ onclick=˝siguiente(this);˝><br>
<tt id=˝comentario˝></tt>
</div>
```

```
/*Propiedades de estilo declaradas dentro de "galeria.css" asociadas a la capa que hace de velo.*/
#capaimagen { z-index: -1; position: fixed; top: 15%; height: 70%; text-align: center; background-color: transparent; }
#mifoto { padding: 4px; background-color: white; border: 1px solid black; width: 100%; height: 100%; }
#comentario { display: inline; background-color: black; color: white; border: 1px dotted white; font-size: 12px; line-height: 25px; padding: 4px; }
```

En relación con las propiedades de estilo asignadas a la nueva capa destacar las siguientes:

— **position: fixed; top: 15%; height: 70%;**": Junto con "**left**" y "**width**" son las propiedades encargadas centrar la nueva capa que albergará la foto seleccionada, independientemente de la posición de la barra de desplazamiento (scrollbar), "**position: fixed;**", y la resolución de la pantalla, al estar asignado su valor en tanto por cien. Tan solo se han definido las propiedades "*top*" y "*height*", ya que si tenemos en cuenta que nos podemos encontrar diferentes resoluciones de pantalla en el cliente Web, para poder guardar una proporción 4x3 en las fotos y así estas se vean sin deformar, será necesario desde JavaScript determinar la resolución de pantalla, u dimensiones en pixels, del área dentro de la ventana del navegador donde se encuentran los contenidos Web (sin tener en cuenta las barras de herramientas del navegador). Una vez que tengamos la resolución vertical y horizontal en pixels, "**window.innerHeight**" y "**window.innerWidth**", podremos conocer con cuantos pixels se corresponde el "**height: 70%**" definido, y así determinar la anchura teniendo en cuenta la relación 4x3. Los cálculos, que se mostrarán más adelante mediante el código JavaScript, serían los siguientes:

```
alto (pixels) =  window.innerHeight * 0.7;
ancho (pixels) = alto * (4 / 3);
width (%) = ancho * 100 / window.innerWidth;
left (%) = (100 – width) / 2;
```

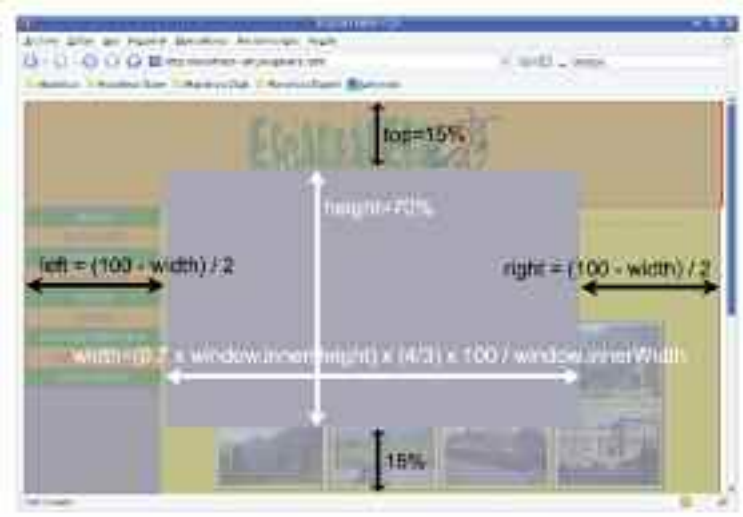


Figura 3.9. Controlando el valor de las propiedades, top, left, width y height, haciendo uso de porcentajes puede centrarse una imagen

— **background-color: transparent;**": Provoca que la capa encargada de albergar al foto ampliada tenga fondo transparente. En realidad su valor es indiferente ya que su superficie será cubierta en su totalidad por la foto seleccionada <img id="mifoto()"></img>, al haber definido como propiedades de la imagen "**width: 100%; height: 100%;**".

— **font-size: 12px; line-height: 25px; padding: 4px; border: 1px dotted white;**": Ajusta el tamaño del texto del comentario que se mostrará debajo de la foto, la distancia entre líneas para que no se solapen y el grosor del relleno entre el texto y el borde definido para el área de texto, garantizando de esta forma que se lea con claridad.

5) Según todo lo anterior, el contenido del documento HTML quedaría de la siguiente forma:

```
<?xml version="1.0" encoding="UTF-8"?>
<html>
<head>
        <script type="text/javascript" src="js/menu.js"></script>
        <script type="text/javascript" src="js/galeria.js"></script>
        <style type="text/css">
                @import "css/plantilla.css" screen;
                @import "css/galeria.css" screen;
        </style>
<link href='fotos/icono.png' rel='shortcut icon' type='image/png'>
</head>
<body>
<div class="division1"></div>
<div class="division2">
<script type="text/javascript" language="javascript"> menu_principal(); </script>
</div>
<div class="division3">
        <hr><h3 class="titulo">GALERIA DE FOTOS</h3><hr>
```

```
        <p> Galeria de imagenes del club Escarbapedal.</p>
        <p class="fotos" style="text-align: center; text-indent: 0cm;">
        <img class="thumbnail" src="fotos/foto1.jpg" OnMouseOver="style.cursor='pointer'" Onclick="ampliar('1');">
        <img class="thumbnail" src="fotos/foto2.jpg" OnMouseOver="style.cursor='pointer'" Onclick="ampliar('2');">
        <!—Resto de imágenes de la galería—>
        </p>
</div>
<div id="velo" onclick="salir();"></div>
<div id="capaimagen" >
        <!—Imagen y comentario—>
        <img id="mifoto" onclick="siguiente(this);"><br>
        <tt id="comentario"></tt>
</div>
</body></hml>
```

6) La hoja de estilos "galeria.css" quedaría de la siguiente forma:

```
h3.titulo { text-align: center; text-decoration: blink; color: red; font-weight: 800; }
img.thumbnail { border: 1px solid black; background-color: white; padding: 4px; width: 150px; height: 110px; }
#velo { z-index: -1; background-color: gray; position: fixed; width: 100%; height: 100%; top: 0px; left: 0px; opacity: 0.8; filter: alpha(opacity=80); }
#capaimagen { z-index: -1; position: fixed; top: 15%; height: 70%; text-align: center; background-color: transparent; }
#mifoto { padding: 4px; background-color: white; border: 1px solid black; width: 100%; height: 100%; }
tt.comentarios { display: inline; background-color: black; color: white; border: 1px dotted white; font-size: 12px; line-height: 25px; padding: 4px; }
```

7) En cuanto al fichero externo "**galeria.js**", esta compuesto por cinco funciones que son invocadas desde la galería de imágenes del documento HTML:

   1. *function* **ampliar(nfoto)**: función encargada de ampliar el thumbnail seleccionado. Para que la función sepa cual es la foto a mostrar, al invocarla desde el documento HTML se le pasa como parámetro el número de foto dentro de la galeria, **OnClick="ampliar('1');"**. Teniendo en cuenta que el nombre que se les ha dado a las imágenes de la galería es la palabra "foto" seguido del número de imagen, "foto1.jpg", "foto2.jpg", ... será tan sencillo asignar un valor al atributo "**src**" del elemento HTML **<img *src*>** como concatenar a la palabra foto el valor del parámetro enviado a la función. Pero para que sea visible la foto, antes, deberemos superponer las capas asociadas al velo semitransparente que oculta el fondo y a la imagen al resto de capas, modificando la propiedad de estilo "**z-index**". En relación a la capa asociada a la imagen, tal como se ha explicado antes, se centrará haciendo uso de las propiedades "**window.innerWidth**? y "**window.innerHeight**?.

```
function ampliar(nfoto) {
/* Calculamos el ancho que tiene que tener la imagen en % para guardar la proporción 4x3 respecto al alto definido en la hoja de estilos, "height: 70%;", y el margen izquierdo para que quede centrada. */
```

```
var ancho=parseInt((window.innerHeight * 0.7) * (4 / 3) * 100 / window.innerWidth);
var izquierda=parseInt((100 - ancho) / 2);
/* Damos prioridad a la capa identificada como "velo" con "zIndex" superponiéndose al documento. */
document.getElementById("velo").style.zIndex="1";
/* Asignamos los valores calculados a la capa asociada a la imagen y la superponemos al resto. */
document.getElementById("capaimagen").style.left=izquierda+"%";
document.getElementById("capaimagen").style.width=ancho+"%";
document.getElementById("capaimagen").style.zIndex="1";
/* Asignamos al atributo "src" del elemento "mifoto" la ruta de la foto seleccionada de la galería. */
document.getElementById("mifoto").src="fotos/foto"+nfoto+".jpg";
/* Llamada a la función JavaScript encargada de asignar un comentario a la foto. */
comentario_foto(nfoto);
}
```

2. *function* **siguiente(foto)**: función encargada de mostrar la siguiente foto de la galería. Esta será invocada al pinchar sobre la imagen ya ampliada, **<img id=»mifoto» onclick=»siguiente(this);»>**, pasándole como parámetro la imagen que en ese momento se esta viendo, con la finalidad de poder conocer de esta forma cual es la siguiente que le corresponde. Si tras pinchar varias veces se llega a la última foto de la galería, automáticamente pasara a mostrarse la primera de todas ellas, haciendo de esta forma un bucle.

```
function siguiente (foto) {
/* Declaramos tres variables encargadas de almacenar el numero de foto que estaba viendose, el numero de foto correspondiente a la siguiente en la galeria que corresponde verse, y el número de fotos total. */
var nfoto, nfoto_siguiente, numero_fotos=8;
/* Mediante las funciones JavaScript "indexOf" y substring, rescatamos el número de foto que se esta viendo, el cual utilizaremos para saber cual es la siguiente imagen que corresponde verse.  Teniendo en cuenta que el nombre de las fotos es "fotoNUM_FOTO.jpg", para obtener el número de la foto, tan sólo deberemos coger el carácter que hay antes de la extensión ".jpg". */
nfoto=foto.src.substring(foto.src.indexOf(".jpg") - 1, foto.src.indexOf(".jpg"));
/* Para saber si hemos llegado a la última foto de la galéria, comparamos el numero de foto obtenido con el número de fotos total de la galeria.  En casa de que sea la ultima se volverá a mostrar la primera. */
if (nfoto < numero_fotos) {
    /* Una vez calculado el número de la siguiente foto, modificamos el atributo "src" de la imagen que se esta viendo en ese momento,  y mostramos el comentario de la foto que corresponde a la siguiente. */
    nfoto_siguiente = parseInt(nfoto) + 1;
    document.getElementById("mifoto").src="fotos/foto"+nfoto_siguiente+".jpg";
    /* Convertimos a carácter el numero de foto, y se le envia como parámetro a la función encargada de colocar el comentario adecuado. */
    comentario_foto(nfoto_siguiente.toString());
    }
    else { document.getElementById("mifoto").src="fotos/foto1.jpg";
    comentario_foto('1');
    }
}
```

3. *function* **comentario_foto(nfoto)**: función encargada de mostrar un comentario bajo cada una de las fotos visualizadas en tamaño grande en la galería. En la llamada a la función se le pasa como parámetro el numero de la foto a la que hay que asociarle el comentario. Para ello se definen tantas variables como comentarios haya para cada una de las fotos, y mediante la ayuda del método o función JavaScript "**getElementById("id").innerHTML**", personalizamos el contenido HTML del elemento ***<tt></tt>*** del documento. Junto al comentario pondremos la fecha en que la foto fue colocada en la página Web, y según esta el número de días que lleva alojada.

Figura 3.10. El comentario bajo la foto esta compueto por dos partes. Información de la imagen, y su fecha, junto con el número total de días alojada en la Web

```
function comentario_foto (nfoto) {
/* Definimos dos variables comentario y fecha por cada una de las fotos de la galería*/
var comentario1="Foto 1: Imagen de los maravillosos puertos de beceite.";
var fechafoto1 = new Date(2008,11,10);
var comentario2="Foto 2: Via Verde Valdealgorfa - Delta del Ebro. Estación del Prat del Compte.";
var fechafoto2 = new Date(2008,10,1);
var comentario3="Foto 3: Virgen de la Balma. El santuario se encuentra en Castellón, en el límite con la provincia de Teruel, en Zorita, cercano a Morella.";
var fechafoto3 = new Date(2008,10,8);
var comentario4="Foto 4: Espectacular subida al tosal de en grillo, cerca de Horta de San Juan, en plenos puertos de Beceite.";
var fechafoto4 = new Date(2008,9,24);
var comentario5="Foto 5: Vistas desde Fredes.";
var fechafoto5 = new Date(2008,9,6);
var comentario6="Foto 6: Via de la Plata.  En el límite de provincia entre Extremadura y Castilla.  Sierra de Bejar.";
var fechafoto6 = new Date(2008,9,5);
var comentario7="Foto 7: Acueducto de Merida, Extremadura. Via de la Plata.";
var fechafoto7 = new Date(2008,9,5);
var comentario8="Foto 8: Anfiteatro romano de Merida. Via de la Plata.";
var fechafoto8 = new Date(2008,8,30);
/* Mediante un condicional "switch" decidimos que comentario poner a la foto. Para indicar la fecha de la foto, y los días que lleva alojada se llama a la función JavaScript "dias_alojadas(fechafoto)". */
```

```
switch (nfoto) {
case "1":
document.getElementById("comentario").innerHTML=comentario1+dias_alojadas(fe
chafoto1); break;
case "2":
document.getElementById("comentario").innerHTML=comentario2+dias_alojadas(fe
chafoto2); break;
case "3":
document.getElementById("comentario").innerHTML=comentario3+dias_alojadas(fe
chafoto3); break;
case "4":
document.getElementById("comentario").innerHTML=comentario4+dias_alojadas(fe
chafoto4); break;
case "5":
document.getElementById("comentario").innerHTML=comentario5+dias_alojadas(fe
chafoto5); break;
case "6":
document.getElementById("comentario").innerHTML=comentario6+dias_alojadas(fe
chafoto6); break;
case "7":
document.getElementById("comentario").innerHTML=comentario7+dias_alojadas(fe
chafoto7); break;
case "8":
document.getElementById("comentario").innerHTML=comentario8+dias_alojadas(fe
chafoto8); break;
        }
}
```

4. *function* **dias_alojadas(fechafoto)**: función encargada de poner la fecha y días totales que lleva la foto insertada en la página Web. En la llamada a la función se le pasa como parámetro la fecha de la foto sobre la que se hace el comentario. Mediante esta fecha suministrada, restándosela a la fecha actual, podremos conocer el número de días que lleva en la Web.

Figura 3.11. Formato de Fecha y dias de alojamiento

```
function dias_alojadas(fechafoto) {
/* Obtenemos la fecha actual mediante "new Date();" */
var fechaactual = new Date();
/* La función devuelve, "return", la fecha de la foto en formato "dia/mes/año" seguido del número de días que lleva alojada la foto. Mediante "getDate()", "getMonth()" y "getFullYear()" recuperamos el día, mes y año de la fecha pasada como parámetro a la función. Teniendo en cuenta que "getMonth()" nos devuelve el mes en formato (0-11), es necesario sumarle 1. "getTime" nos devuelve el tiempo en milisegundos transcurrido desde el 1 de enero de 1970, permitiendonos determinar con una simple resta el número de días transcurridos entre la fecha actual y la fecha de la foto. */
return("<br>Fecha: " + fechafoto.getDate() + "/" + (fechafoto.getMonth() + 1) + "/" +
fechafoto.getFullYear() + ". Dias alojada: " + Math.round((fechaactual.getTime() -
fechafoto.getTime())/ (1000 * 60 * 60 * 24)) + ".");
}
```

5. *function* **salir()**: función encargada de mandar las capas velo e imagen por debajo de las divisiones del documento modificando la propiedad de estilo "z-index" (para JavaScript "**zIndex**"). Para invocar la función será necesario pinchar fuera del área utilizada por la imagen seleccionada, es decir, sobre el área asociada a la capa "velo", <div id=»velo» **onclick=»salir();»**> .

```
function salir() {
/* Asignando "zIndex='-1'" estamos enviando la capa correspondiete al fondo. */
        document.getElementById("velo").style.zIndex="-1";
        document.getElementById("capaimagen").style.zIndex="-1";
}
```

Capítulo 4

ANIMACIONES FLASH

Tras habernos familiarizado en el capítulo anterior con JavaScript, el método más comúnmente utilizado para conseguir dinamizar los contenidos de nuestro sitio Web en el lado del cliente, en el presente capítulo introduciremos una posible alternativa. Aunque surgió inicialmente para el diseño de presentaciones interactivas[7], tras más de diez años de veteranía, Flash se ha convertido en la actualidad en un referente a hora de diseñar las páginas que componen un sitio Web, gracias a sus destacables características, y el agradable aspecto que aporta a la Web. Al igual que JavaScript, Flash nos permite controlar el comportamiento de nuestros documentos HTML/XHTML aportándoles un dinamismo que provoca que el usuario centre la atención en determinadas zonas de la página que pueden ser estratégicas (por ejemplo, publicidad). Aunque existen otras posibles alternativas para garantizar ese dinamismo en el lado del cliente como son Visual Basic Script o los applets de Java, estos han quedado relegados por los problemas que presentan de compatibilidad[8] en el primer caso, y riesgos en la seguridad del equipo cliente, en el segundo, al poderse programar en Java cualquier aplicación de propósito general[9], además de una lentitud de ejecución destacable en relación a otras tecnologías existentes como Flash .

Posterior al afianzamiento de Flash, en el año 2001, el consorcio internacional W3C convirtió el lenguaje SVG (*Scalable Vector Graphics*) en una recomendación a la hora de implementar gráficos vectoriales bidimensionales, tanto de manera estática como de manera dinámica, dando la posibilidad, al igual que JavaScript y Flash de interactuar con el usuario mediante un conjunto de manejadores de eventos similares a los vistos en JasvaScript. De esta forma SVG esta compitiendo actualmente con el ya asentado Flash en la creación de animaciones Web interactivas, con la ventaja de que SVG es un estándar abierto, al contrario de Flash que es propietario de la empresa de software Adobe. Pensando que este será el formato del futuro, se dedicará el siguiente capítulo a esta recomendación.

Más recientemente, en el año 2007, Microsoft ha sacado a la luz SilverLight, como una alternativa a Flash y SVG. Pensado igualmente para el diseño de animaciones Web, permite trabajar gráficos vectoriales, y todo tipo de archivos multimedia (música y videos), ofreciendo igualmente unos resultados sorprendentes. Como inconveniente cabría destacar que al igual que Flash, se trata de código no abierto, ni recomendado por el W3C, que presenta la temeridad de que si lle-

---

7 En 1996 Macromedia se hace con los derechos de Flash tras la buena aceptación por parte de compañias como Microsoft, Disney o Fox en el diseño de sus sitios Web. En 2005 Adobe adquiere Macromedia, y con ella los derechos de Flash.

8 Visual Basic Script (VBScript) es un lenguaje propietario de Microsoft Windows que tan solo es interpretado por su propio navegador, Internet Explorer, y no por otros tan extendidos como Mozilla Firefox u Opera.

9 Aunque los applets de Java tuvieron una época de gran esplendor a finales de los años 90, la aparición de multitud de estos con carácter malicioso, hizo que fueran considerados de poca confianza, y requirieran de un certificado digital que indicase lo contrario. Esto se ha traducido que en la actualidad apenas se implementen.

gase a extender su utilización, Microsoft controlase el mundo Web haciendo que solo fuese accesible desde plataformas Windows. En entornos GNU/Linux existe una alternativa libre llamada MoonLight.

1. CONTENIDOS DEL CAPÍTULO

El presente capítulo sigue la siguiente estructura:

— Comienza informando del software necesario para la realización de los ejercicios prácticos que se plantean, señalando los sitios Web desde donde puede realizarse su descarga.

— A continuación se indican las características más importantes de Flash, sus ventajas e inconvenientes.

— Para aprender como se desarrollan animaciones con **Adobe Flash**, se presentan de manera detallada dos ejercicios prácticos.

— Termina presentando a **SWFTools** como una alternativa a **Adobe Flash** para la generación de archivos ***swf***.

2. SOFTWARE DE DESARROLLO

Para poder probar los ejercicios prácticos que se muestran en el presente capítulo, una posibilidad es descargar e instalar la última versión de Adobe Flash (http://www.adobe.com/es/downloads/), pero también es posible utilizar cualquier otra versión anterior de Adobe Flash, o incluso Macromedia Flash, al haber respetado Adobe el entorno de desarrollo que Flash presentaba ya con Macromedia. Habrá que tener presente que la versión descargable desde el sitio Web de Adobe es de prueba válida durante 30 días, siendo necesario pagar una licencia para conseguir un número de serie para un uso ilimitado.

Figura 4.1. Desde www.adobe.com/es/downloads/ *podemos descargar Adobe Flash Professional para probarlo durante 30 días*

El problema que vamos encontrar es que esta herramienta software, como ya se ha comentado en sus características, solo es instalable bajo sistemas operativos Microsoft Windows y Macintosh[10]. Esto implica que en el caso de que trabajemos bajo GNU/Linux, será necesario hacer uso de máquina virtual (por ejemplo, VirtualBox[11]) e instalar sobre ella Microsoft Windows, o crear una nueva partición en nuestro disco duro, e instalar Windows de manera independiente.

Figura 4.2. Mediante VirtualBox es posible instarlar Microsoft Windows bajo GNU/Linux, permitiéndonos trabajar con Adobe Flash como si se tratase de una aplicación más

El resultado obtenido por Flash (*objeto SWF*) será incrustado en un documento HTML/XHTML reproducible desde nuestro navegador o cliente Web preferido (por ejemplo, *Mozilla Firefox* o *Internet Explorer*), tal como se mostró en el apartado X del primer capítulo del libro.

Como alternativa a Adobe Flash cabría destacar el software de desarrollo **SWiSH**. Con un entorno de trabajo muy similar, es famoso por generar potentes animaciones para títulos, con multitud de efectos dinámicos predefinidos, sin apenas trabajo y de manera muy intuitiva. Presenta el inconveniente de ser un software privado, siendo necesaria la adquisición de una licencia para su plena utilización.

En el caso de que la presentación Flash que se desea generar, sea muy sencilla, también será posible utilizar el software libre **SWFTools** como alternativa, disponible tanto para entornos Microsoft Windows como GNU/Linux (*http://www.swftools.org/download.html*), tal como se mostrará al final del presente capítulo.

3. CARACTERÍSTICAS DE FLASH

Entre las características más reseñables que han hecho de Flash un estándar en el diseño de animaciones en el mundo Web serían las siguientes:

— *Resultados con reducido tamaño*. Al trabajar Adobe Flash con gráficos vectoriales posibilita aumentar las dimensiones del gráfico resultante sin perder calidad, y sin incrementar

10 Puede instalarse en GNU/Linux bajo el emulador de Windows wine, pero se desaconseja por problemas futuros que puede presentar.

11 VirtualBox es un software libre disponible tanto para Microsoft Windows como GNU/Linux que nos permite trabajar simultáneamente con distintos sistemas operativos.

el tamaño del archivo resultante. Esta característica, junto a los vertiginosos anchos de banda que actualmente suministran los proveedores de Internet (por ejemplo, Telefónica), garantizan que los archivos Flash pueden descargarse rápidamente.

— *Eficiencia en la reproducción*. Si a la característica anterior le añadimos que Flash permite el "streaming", una técnica en virtud de la cual el usuario puede comenzar a ver una animación sin que haya sido descagada por completo, esta garantizado que las animaciones resultantes puedan reproducirse en el navegador o cliente Web sin apenas retardos.

Además, una vez que el archivo Flash ha sido descargado completamente, queda almacenado en la caché del navegador Web, de tal forma que no será necesario volver a repetir la operación de descarga en el caso de que sea solicitada por parte del usuario una nueva reproducción.

— *Compatibilidad entre navegadores*. Para que las animaciones generadas en Adobe Flash sean reproducibles en nuestro navegador es inprescindible que este tenga instalado el complemento o plugin correspondiente. Actualmente Flash se encuentra tan extendido y es tan ampliamente usado por los diseñadores de sitios Web, que ha provocado que los desarrolladores de los navegadores Web más populares ya incorporen por defecto este complemento sin que sea necesaria su instalación por parte del usuario.

— *Manejo de elementos multimedia*. Las animaciones Flash ofrecen la oportunidad de introducir todo tipo de archivos multimedia, audio y video. La combinación de todo ello asegura un éxito en vistosidad.

— *Entorno de desarrollo intuitivo*. Macromedia y Adobe Flash se caracterizan por ofrecer unas herramientas de desarrollo gráficas muy elaboradas y depuradas, que permiten poder empezar a desarrollar animaciones simplemente haciendo uso de la intuición. Esto le diferencia radicalmente de otras alternativas como SVG, el cual no dispone actualmente de una herramienta gráfica potente, y es necesario redactar el código manualmente.

— *Fogramas y capas*. Las animaciones generadas por Flash podemos considerarlas como la reproducción de un conjunto de fotogramas a una velocidad constante, que permiten dar la sensación de movimiento, donde los distintos elementos que intervienen simultáneamente se localizan en capas que solapan sobre el eje Z de tal forma que no se interfieren entre sí. De todos los fotogramas, el usuario tan sólo tiene que diseñar los llamados fotogramas clave, ya que los fotogramas intermedios los creará Adobe Flash como resultado de interpolación entre dos fotogramas clave.

— *ActionScript*. Adobe Flash dispone de este lenguaje de programación que nos permite añadir dinamismo a la Web en el lado del cliente al igual que JavaScript. Este nos permite controlar el comportamiento del sitio Web e interactuar con el usuario.

— *Archivos "*.fla" y "*.swf"*. La extensión de los documentos de Flash sobre los cuales se realiza el diseño de lo que será la película o animación es "*.fla". Una vez terminada, Adobe Flash combina todos los elementos que intervienen en ella, generando un archivo resultado de su empaquetamiento con extensión "*.swf". Este último será el objeto SWF que se podrá invocar desde un documento HTML mediante la etiqueta *<object></object>* para su reproducción con la ayuda del adecuado complemento o plugin de Flash[12].

Como resumen de todo lo anterior, podríamos decir que Adobe Flash nos permite crear animaciones Web con un alto nivel de dinamismo, apariencia muy agradable, posibilidad de añadir todo tipo de archivos multimedia, todo con muy poco esfuerzo, y con un tamaño de archivo resultante reducido. No obstante, no todo son ventajas en Flash, cambien podríamos destacar algunos inconvenientes:

---

12 Heredado de Macromedia, Adobe llama a la acción de generación del objeto SWF a partir del documento "*.fla" *publicar*.

— El resultado de la animación diseña desde Adobe Flash es un objeto SWF compacto no editable ni por la propia herramienta que lo ha generado. La única forma de introducir una modificación en el texto, imágenes, animaciones o archivos multimedia que incluya ese objeto, es editar el fichero "*.fla" a partir del cual se genero, dando lugar a un nuevo objeto SWF. Esta pequeña complicación a la hora de llevar a cabo actualizaciones en sus contenidos, hace que a veces se detecte en los sitios Web diseñados enteramente en Flash un cierto carácter estático, en el sentido que siempre se ve lo mismo. Este aspecto también complica su posterior mantenimiento, sobre todo cuando la persona encargada de ello no fue quien lo generó, y no dispone del archivo "*.fla".

— Sin el fichero "*.fla" asociado al objeto SWF (*.swf) no es posible reutilizarlo por parte de los desarrolladores y diseñadores de aplicaciones software en Flash.

— Al igual que ocurre con los documentos HTML que hacen uso de marcos o frames, los navegadores Web pueden encontrar problemas de navegación entre las páginas de sitio Web cuando hasta los enlaces y botones de control de navegación se implementan enteramente en Flash.

— De manera similar al inconveniente anterior, los buscadores de información de Internet encuentran problemas al escanear los contenidos de un sitio Web realizado enteramente en Flash, al encontrarse todo compacto en un objeto SWF no editable. Por estos motivos, es muy complicado que un buscador como Google nos pueda enlazar con una de las páginas que componen un sitio Web en Flash más allá de su página de inicio. Los buscadores están pensados para rastrear documentos de tipo texto etiquetados mediante marcas HTML.

— Las herramientas de desarrollo Flash solo esta disponible en entornos Microsoft Windows y Apple Macintosh. Esto obliga a diseñadores Web que hagan uso de otro tipo de sistemas operativos como GNU/Linux, sea necesario trabajar sobre una máquina virtual (por ejemplo, VirtualBox).

Por todos los motivos anteriores, en este capítulo no se va a explicar como diseñar un sitio Web entero en Flash navegable. Simplemente se va a explicar como diseñar pequeñas animaciones Flash útiles para la colocación de anuncios o publicidad en nuestra Web[13].

4. ENTORNO DE DESARROLLO ADOBE FLASH

Tras la descarga e instalación de Adobe Flash, ejecutaremos esta herramienta software de desarrollo desde el menú de programas. Al abrirse nos encontraremos el siguiente entorno, el cual es personalizable por el usuario desde el menú superior *Ventana*:

Figura 4.3. Aspecto del entorno de desarrollo de Adobe Flash

13 A los anuncios que se colocan en las páginas que componen un sitio Web se les denomina banners, caracterizándose por llamar mucho la atención al visitante.

— **Escenario**: Muestra el contenido del fotograma actualmente seleccionado en la línea de tiempo.

— **Barra de herramientas**: Conjunto de utilidades ofrecidas por Adobe Flash para el diseño de los contenidos mostrados en el escenario. Las hay para representar formas geométricas, pintar a mano alzada o rellenar superficies entre otras. Además de estas herramientas intrínsecas, Flash nos permite reutilizar elementos generados por otras aplicaciones mediante su importación[14].

— **Capas**: Organizan los elementos mostrados en el escenario sobre un eje Z imaginario. Los elementos situados sobre capas superiores ocultarán a los que se encuentren en capas inferiores. Esta división en capas de los elementos es fundamental para posibilitar la animación de cada uno de ellos por separado, sin que interfiera sobre el resto.

— **Línea de tiempo**. Organiza los contenidos mostrados en el escenario en el tiempo haciendo uso de distintos fotogramas. El fotograma que se mostrará en cada instante de tiempo dependerá de la velocidad de reproducción de la película de Flash medida en fotogramas por segundo, la cual se puede personalizar desde el menú *Modificar*, opción *Documento*. Entre los distintos fotogramas que forman parte de la animación cabría distinguir a los fotogramas clave sobre el resto, ya que son estos los que el desarrollador debe personalizar, siendo utilizados por Flash el resto de fotogramas intermedios para crear una interpolación entre ambos, ofreciendo de esta forma una sensación de movimiento.

— **Propiedades**. Ventana que nos permite editar las propiedades del objeto que tengamos seleccionado, que puede ser el propio escenario, un elemento que este contenga o un fotograma de la línea de tiempo.

— **Biblioteca**. Almacena y organiza los distintos elementos que forman parte de la película o animación Flash que estamos diseñando. Estos elementos pueden ser reutilizados mediante instancias, optimizando de esta forma el tamaño de la película resultante. Flash también nos da la posibilidad de reutilizar esta biblioteca en otras películas Flash mediante su importación.

— **Menús**. Esta barra de menús clásica en entornos de ventanas personalizado para Adobe Flash, nos permite controlar y gestionar todos los aspectos de la película que se diseña.

Para conocer más de cerca esta herramienta software, a continuación se proponen dos ejercicios prácticos. Que mejor que aprender practicando.

14 Desde el ménu *Archivo*, la opción *Importar* nos permite que imágenes en diversos formatos jpg/png/gif/bmp/wmf, archivos de música mp3/wav o videos avi/mpg/mov/flv puedan ser importados al escenario de Flash.

Ejercicio práctico n.º 1

En el presente apartado se mostrará como crear una animación Flash de duración 20 segundos a partir de un conjunto de tres imágenes y dos textos, a modo de ejemplo. Las imágenes se secuenciarán, apareciendo y desapareciendo poco a poco mediante un efecto de transparencia (propiedad *alfa*). De manera similar, los textos, superpuestos a las imágenes irán apareciendo en el transcurso de la reproducción de la película poco a poco. El resultado será colocar en el header o cabecera de un documento HTML/XHML como publicidad o anuncio sobre el tema que decidas.

Figura 4.4. Animación Flash utilizada como encabezado del documento HTML/XHTML

Los pasos que seguiremos serán los siguientes:

1. A modo de ejemplo, el anuncio presentará unas dimensiones de 635 pixels de ancho y 150 pixels de alto. En consecuencia será necesario ajustar el tamaño de las imágenes que formarán parte de la animación a las medidas indicadas. Para ello, utilizaremos alguna herramienta software de retoque de imagen digital, destacándose a PhotoShop, en entornos Microsoft Windows, y Gimp, software libre disponible tanto en entornos Windows como GNU/Linux, con una potencia similar.

¡¡Sugerencia!! Utilizar Photoshop o Gimp para una simple operación de redimensionamiento de una imagen es como matar moscas a cañonazos. Para operaciones tan concretas, existe un software libre disponible tanto para Windows como GNU/Linux muy potente llamado ***ImageMagick*** (*http://www.imagemagick.org*). Su característica más notable es que es un editor de imágenes sin necesidad de una interfaz gráfica, al realizarse todas las operaciones de edición desde la línea de comandos de Windows o consola de GNU/Linux. Entre sus comandos podríamos destacar "convert", utilizado para convertir imágenes, "**identify**" para obtener información de una imágen, o "display/imdisplay"[15] para visualizar una imágen entre otros muchos. En concreto el comando "convert" nos permite redimensionar una imagen con la opción "**-resize**", o recortarla mediante la opción "**-crop**", siguiendo la siguiente sintaxis:

```
[prompt@Windows|Linux] convert [-verbose][16] -crop AnchoxAlto+coordX+coordY imagen-origen imagen-destino

[prompt@Windows|Linux] convert [-verbose] -resize AnchoxAlto imagen-origen imagen-destino
```

15 GNU/Linux hace uso del comando *display* para visualizar la imagen indicada, mientras que en Windows se utiliza *imdisplay*.

16 La opción *-verbose* proporciona información por pantalla sobre el resultado de la operación solicitada.

Tras su descarga e instalación, habiendo seleccionado las imágenes que queremos utilizar en la animación Flash, ejecutaremos los siguientes comandos:

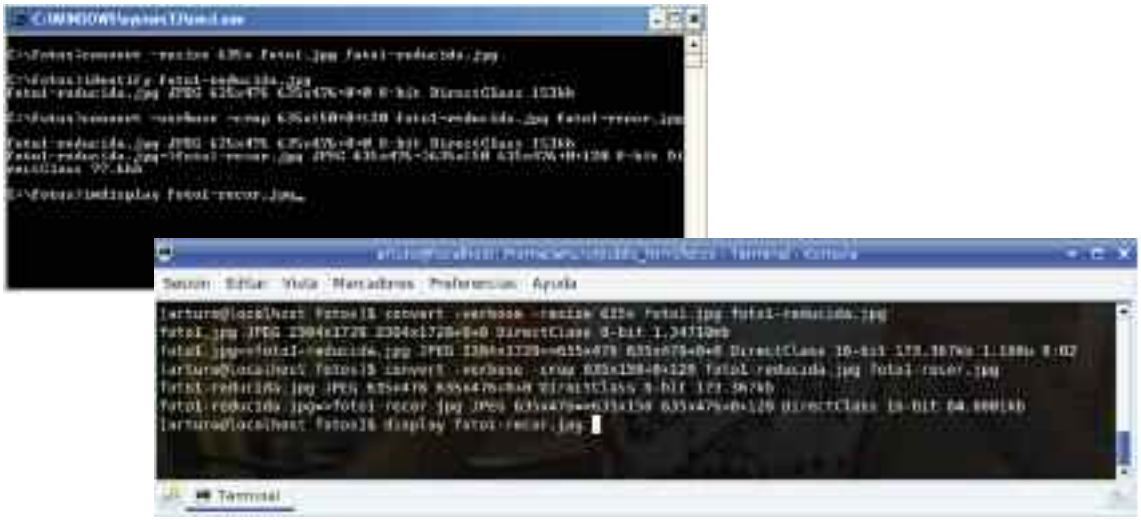


Figura 4.5. Capturas de pantalla en Windows y GNU/Linux con las operaciones de edición de imagenes

1. Redimensionamos la foto original ajustándola al ancho de la imagen que usaremos en la animación, 635 pixels. Al no indicar valor a la altura, el comando "convert" le asignará la que le corresponda evitando deformar la imagen.

convert [-verbose] **-resize** 635x imagen-origen imagen-origen-reducida

2. Recortamos la imagen anterior con la anchura y altura deseadas, 635 pixels x150 pixels, indicando la esquina superior izquierda del cuadro de recorte mediante sus coordenadas X e Y:

convert [-verbose] **-crop** 635x150+0+coorY imagen-origen-reducida imagen-origen-recortada

Figura 4.6. Ejemplo del resultado de edición de imagen mediante la utilidad de ImageMagick convert

El número de imágenes que formarán parte de la animación es arbitrario. El ejemplo que se muestra hará uso de tres.

2. Ajustaremos el tamaño del documento Flash a las dimensiones del objeto que tratamos de generar (*635x150 pixels*). Para ello, iremos al menú ***Modificar***, la opción ***Documento***.

Figura 4.7. Desde Modificar/Documento *podemos personalizar las propiedades del documento Flash*

Otra propiedad del documento a destacar es la *velocidad de fotogramas*. Contra mayor sea la cantidad de fotogramas por segundo que se especifique, mayor será la sensación de movimiento que adquirirá la animación, a costa también de un mayor tamaño del fichero reproducible (**.swf*) resultante. En el caso de que la película Flash contenga elementos de vídeo, o animaciones es recomendable el valor de 24 fps[17], en caso contrario, se detectarán discontinuidades. En este ejemplo, en el que se va a mostrar una transición entre las imágenes, se va a asignar un valor de 12 fps, siendo aconsejable rediseñar la pélicula Flash con 24 fps para detectar las diferencias.

¡¡Importante!! Teniendo en cuenta que la animación queremos que dure 20 segundos, y que la velocidad de reproducción de los fotogramas seleccionada anteriormente en las *propiedades del documento* es de 12 fps, la película de Flash constará de *20sg x 12fps = 240 fotogramas*. De estos 240 fotogramas, tan sólo es necesario editar los fotogramas clave. El resto de fotogramas situados entre estos fotogramas clave serán generados por Flash como resultado de su interpolación.

3. Creamos tantas capas como elementos dinámicos vayan a formar parte de la animación: 2 capas para los textos, y 3 capas para las tres imágenes. Es conveniente que cada uno figure en una capa diferente para que su animación asignada no interfiera con la del resto. Además teniendo en cuenta que las capas superiores se superponen visualmente sobre las capas inferiores, las dos capas asociadas a los textos se colocarán en primer lugar.

Figura 4.8. Desde la ventana de capas podemos crear/borrar/ocultar/bloquear/organizar *estas*

17 Científicamente esta comprobado que con 25 fps el ojo humano no es capaz de detectar transiciones de movimiento. Esta es la cadencia a la que se refresca las imágenes en televisión.

Haciendo un doble click sobre el nombre de la capa, Adobe Flash nos permitirá editarlo, pudiendo identificarla con un nombre más acorde con lo que contenga. En el caso de que el número de capas sea elevado, es posible hacer uso de carpetas para organizarlas (ver imagen anterior).

4. Empezaremos editando el primero de los textos. Seleccionando el primer fotograma clave en la línea de tiempo de la capa asociada al primer texto, y la *herramienta de Texto* en la barra de herramientas, introduciremos el primer título de la animación en el escenario del documento Flash, seleccionando previamente en la ventana de propiedades el tipo de fuente, su tamaño, color y estilo, pudiendo asignar a cada uno de los caracteres que lo forman propiedades diferentes.

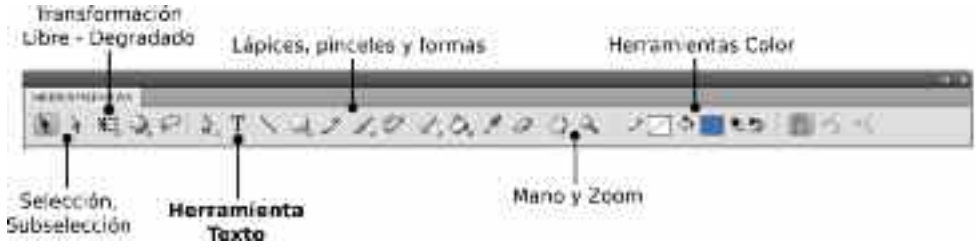


Figura 4.9. La herramienta Texto nos permite introducir texto en la animación Flash

Los fotogramas clave se distinguen del resto, al tener impreso un circulo en su parte inferior. Mientras el fotograma clave esta vacío de contenidos este círculo permanecerá en color blanco, y al introducir en él un texto, una imagen o cualquier otro elemento, adoptará un color negro.

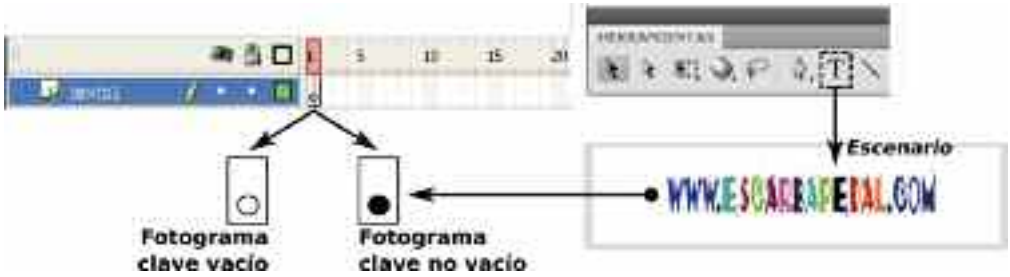


Figura 4.10. Un círculo en la parte inferior del fotograma nos informará de que se trata de un fotograma clave. Su color nos indicará si está vacío o no

5. Tras introducir el texto anterior, lo seleccionaremos y lo convertiremos en símbolo. Esta conversión facilita su animación en la película Flash, favorece su reutilización y garantiza que el tamaño del archivo *swf* resultante sea menor.

Figura 4.11. Desde Modificar/Convertir en símbolo *podemos convertir un texto, una forma o un mapa de bits entre otros, en un símbolo Flash*

Para ello, iremos al menú ***Modificar*** y seleccionaremos la opción ***Convertir en símbolo***. De los tipos de símbolos utilizados por Flash, *Gráfico*, *Botón* y *Clip de película*, elegiremos la primera opción.

6. Crearemos otros tres fotogramas clave equidistantes espaciados en la línea de tiempo 80 fotogramas en la capa asociada al primer texto. Estos fotogramas se crearán con una copia o instancia del mismo símbolo que se encuentra el primer fotograma clave, editables de manera independiente. Tal como veremos a continuación, estos fotogramas clave serán los encargados de determinar el transcurso de la animación.

Figura 4.12. La inserción de fotogramas clave determina el curso de la película Flash

7. Para la animación del texto anterior, jugaremos con su tamaño y la propiedad de color del símbolo alfa, editables ambas desde la ventana de propiedades. Otra forma de ajustar fácilmente el tamaño del símbolo sobre el escenario es haciendo uso de la ***herramienta de transformación libre*** que encontramos en la barra de herramientas (ver figura X).

Figura 4.13. Desde la ventana de **propiedades** *podemos ajustar las propiedades de los elementos que forman parte del escenario de Flash*

En concreto, su animación consistirá en aumentar paulatinamente el tamaño del texto, partiendo en el fotograma clave nº1 de un tamaño muy reducido, con un transparencia casi total (*alfa=20%*), agrandándose a medida que avanza el tiempo, adquiriendo un tamaño acorde al del escenario del documento (*635x150 pixels*), con una opacidad total (*alfa=100%*), al cabo de un tercio de la duración de la animación (*fotograma clave nº80*), manteniéndose durante otra tercera parte (*hasta el fotograma clave nº160*), para terminar aumentado de tamaño y produciéndose su disolución (*alfa=15%*), en el fotograma clave nº240, tal como se puede observar en la siguiente figura.

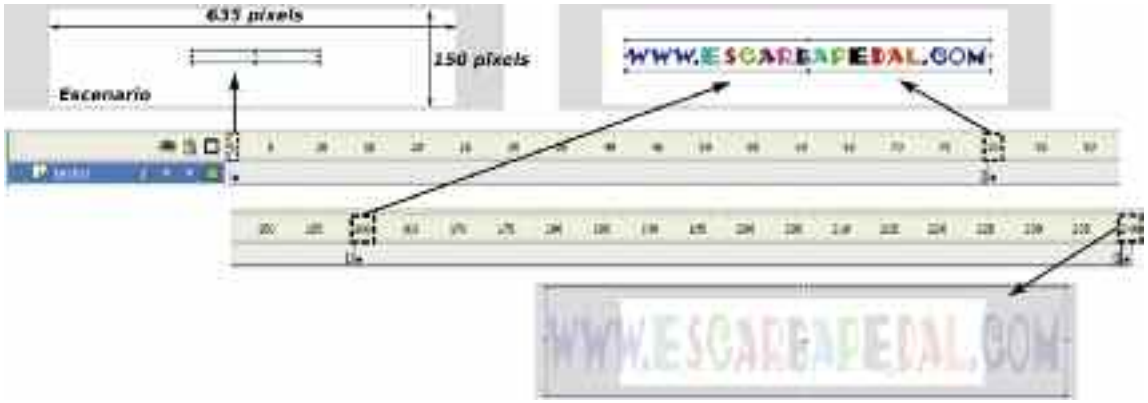


Figura 4.14. Contenido de los fotogramas clave 1, 80, 160 y 240

8. Los fotogramas intermedios entre los cuatro fotogramas clave anteriores (1, 80, 160, y 240), serán interpolados por Adobe Flash garantizando una sensación de movimiento. Para ello, será necesario ***crear interpolación de movimiento*** entre cada dos fotogramas clave, pinchando con el botón derecho del ratón sobre estos fotogramas (1, 80 y 160).

Figura 4.15. La interpolación de movimiento entre dos fotogramas clave es utilizada para proporcionar sensación de movimiento en las animaciones Flash

Para comprobar el efecto de la interpolación podemos reproducir la película Flash desde el menú ***Control***, seleccionando ***Probar película***.

9. A continuación haremos algo parecido con el segundo título en una segunda capa. Su animación comenzará a mitad de película, en el fotograma nº120 con efecto de transparencia prácticamente total (*alfa=15%*).

Figura 4.16. Pasos para la edición del segundo título de la animación

Para ello, como puede observarse en la anterior figura, seleccionaremos el fotograma n.º 120 asociado a la segunda capa (*texto2*) e insertaremos un fotograma clave vacío, que rellenaremos mediante la herramienta ***Texto*** introduciendo un segundo título, que convertiremos en símbolo gráfico (*Modificar / Convertir en símbolo*), al que asignaremos desde la ventana de propiedades un alfa del 15%. A continuación, realizaremos las siguientes acciones para completar su animación:

— Fotograma n.º 180 de la capa *texto2*: insertamos un fotograma clave, copía del anterior, asignando un alfa del 100%, sin modificar su tamaño.

— Fotograma n.º 240 de la capa *texto2*: insertamos un fotograma clave, copía del anterior, asignando un alfa del 15%, sin modificar su tamaño.

— Creamos interpolación de movimiento entre los fotogramas clave 120-180, y 180-240 proporcionando sensación de continuidad en la aparición y desaparición de este segundo título.

— Desde el menú ***Control***, seleccionando ***Probar película*** comprobaremos el resultado de la animación.

10. Tras animar los textos anteriores, importaremos a la biblioteca del documento Flash las imágenes que formarán parte de la animación, seleccionadas al comienzo del ejercicio práctico. Para ello, desde el menú ***Archivo***, en la opción ***Importar***, podremos ***Importar a biblioteca*** los archivos necesarios. Tras ello, la lista de archivos aparecerá en la ventana asociada a la biblioteca del documento.

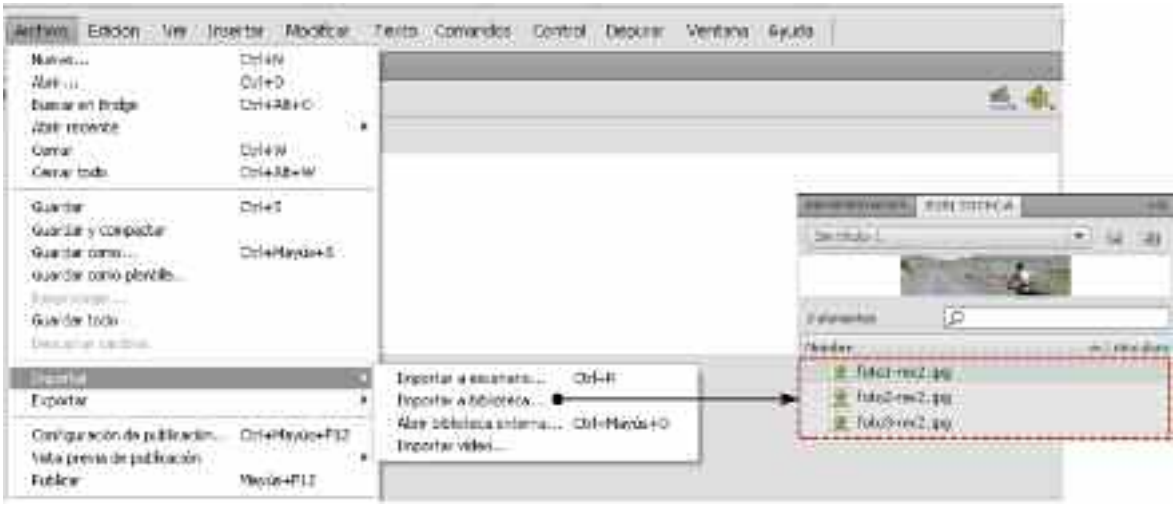


Figura 4.17. Adobe Flash *nos permite importar archivos multimedia (imagen / audio / video) para poderlos utilizar en nuestra animación*

Los pasos que seguiremos para animar las imágenes son:

1. Fotograma n.º 1 – capa n.º 3 (*foto1*): Arrastramos la primera imagen desde la biblioteca al escenario. Convertimos la imagen en un símbolo gráfico (*Modificar / Convertir en símbolo*). Mediante la herramienta de ***Transformación libre*** ajustamos su tamaño, a un valor próximo al doble del asignado al escenario. Desde la ventana de propiedades, asignamos un valor de *alfa* del 15%.
2. Fotograma n.º 40 – capa n.º 3 (foto1): Insertamos un fotograma clave. Seleccionando la imagen, ajustamos su tamaño al del escenario {nota pie pagina: Desde la ventana de propiedades, podemos ajustar fácilmente la anchura, altura y posición de la imagen.}, y le asignamos un valor de *alfa* del 100%.

3. Fotograma n.º 80 – capa n.º 3 (foto1): Insertamos un fotograma clave, y le asignamos a la imagen un valor de *alfa* del 15%.
4. Creamos interpolación de movimiento entre los fotogramas clave 1-40 y 40-80 de la capa n.º 3.

Figura 4.18. Resultado de la edición de la imagen n.º 1

5. Fotograma nº60 – capa nº4 (foto2): Insertamos un fotograma clave vacío. Arrastramos la segunda imagen desde la biblioteca al escenario. Mediante la herramienta de ***Transformación libre*** ajustamos su tamaño, a un valor próximo al doble del asignado al escenario, y le asignamos un valor de *alfa* del 15%.
6. Fotograma nº120 – capa nº4 (foto2): Insertamos un fotograma clave. Ajustamos el tamaño de la imagen al tamaño del escenario, y le asignamos un valor de *alfa* del 100%.
7. Fotograma nº160 – capa nº4 (foto2): Insertamos un fotograma clave. Asignamos un valor de *alfa* del 15% a la segunda imagen.
8. Creamos interpolación de movimiento entre los fotogramas clave 60-120 y 120-160 de la capa nº4.
9. Fotograma nº140 – capa nº5 (foto3): Insertamos un fotograma clave vacío. Arrastramos la tercera imagen desde la biblioteca al escenario. Ajustamos su tamaño, a un valor próximo al doble del asignado al escenario, y asignamos un valor de *alfa* del 15%.
10. Fotograma nº200 – capa nº5 (foto3): Insertamos un fotograma clave. Ajustamos el tamaño de la imagen al tamaño del escenario, y le asignamos un valor de *alfa* del 100%.
11. Fotograma nº240 – capa nº5 (foto3): Insertamos un fotograma clave. Asignamos un valor de *alfa* del 15% a la tercera imagen.
12. Creamos interpolación de movimiento entre los fotogramas clave 140-200 y 200-240 de la capa nº5.
13. Desde el menú ***Control***, seleccionando ***Probar película*** comprobaremos el resultado de la animación. Posteriormente, modifica la velocidad de reproducción de fotogramas (*Modificar*/*Documento*) a 24 fps, y comprueba que los saltos de discontinuidad se ven reducidos.
14. Desde el menú ***Archivo***, seleccionando ***configuración de publicación***, generaremos el archivo *swf* que insertaremos como objeto, <*object*></*object*>, en nuestro documento HTML/XHML. Desde esta ventana Adobe Flash también nos ofrece generar un documento HTML con el el objeto SWF incluido, pudiéndonos aprovechar de su código para completar la página Web que estemos diseñando.

Ejercicio práctico n.º 2

En este segundo ejercicio práctico se mostrará cómo generar una animación, donde la trayectoria de movimiento que siguen los elementos que la forman es un trazado establecido previamente por el diseñador: ***Interpolación de movimiento utilizando capa guía***.

El ejercicio que se propone, se corresponde con la animación de duración 10 sg del logotipo de una organización imaginaria cualquiera. Compuesta por dos textos y dos imágenes, irán apareciendo los elementos sobre el escenario progresivamente, comenzando desde una posición inicial externa al escenario, definida en un fotograma clave inicial, hasta un posición final en su interior, definida por un fotograma clave final, donde jugando con la propiedad del color alfa, se proporcionará la sensación de movimiento mediante su interpolación, tal como se muestra en la siguiente figura.

Figura 4.19. Animación del ejercicio práctico n.º 2, representando la posición inicial, final e intermedias de los cuatro elementos que formarán parte de ella

A simple vista, podría parecer un ejemplo más similar al ejercicio práctico anterior, sin embargo, en esta animación el contenido de los fotogramas generados por Flash en la interpolación de movimiento entre los dos fotogramas clave, inicial y final, asociados al segundo de los textos, será establecido por el trazado indicado a través de una capa especial vinculada a la capa que alberga dicho texto que hará de guía. Resumiendo los pasos serían los siguientes:

1. Configuraremos las propiedades del documento (menú *Modificar/Documento*). Dimensiones, 700 pixels de anchura por 100 pixels de altura, y velocidad de reproducción de los fotogramas 24 fps (*10 sg -> 240 fotogramas*).
2. Creamos cuatro capas, dos para los textos (en el ejemplo, “cossio” y “.net”) y otras dos para las imágenes del logotipo.
3. Seleccionando el primer fotograma clave asociado a las dos primeras capas, y haciendo uso de la herramienta Texto insertaremos sobre el escenario los textos que formarán parte del logotipo en las posiciones iniciales de la animación.
4. Importaremos las imágenes del logotipo a la biblioteca del documento Flash (menú *Archivo/Importar/Importar a biblioteca*). Posteriormente las arrastraremos desde la biblioteca hacia el escenario, ubicándolas en la posición en que deseemos que se encuentren al comienzo de la animación (primer fotograma clave asociado a cada una de sus capas).

Figura 4.20. Posición inicial de los elementos que forman parte de la animación

5. Seleccionándolos uno a uno, convertiremos en símbolos gráficos los textos e imágenes anteriores (menú *Mofificar/Convertir en símbolo*).
6. Modificaremos la propiedad de color alfa de los símbolos (*alfa=10%*) en su posición inicial para que su aparición sobre el escenario sea suave.
7. ***Insertaremos un fotograma clave*** en la capa asociada al segundo texto (*texto2*) a mitad de animación (*fotograma n.º 120*). Con este fotograma seleccionado, arrastraremos el texto hasta su posición final en el escenario, con un alfa 100%, para a continuación pinchando con el botón derecho del ratón sobre el primero, ***crear interpolación de movimiento*** entre los fotogramas claves 1-120. Para terminar con este elemento, ***insertaremos otro fotograma clave*** en la capa *texto2* al final de la animación (*fotograma n.º 240*), de tal forma que entre los fotogramas 120-240 no se perciba movimiento por parte de este elemento.

Figura 4.21. Contenido de los fotogramas de la capa texto2

8. En las capas asociadas a las imágenes del logotipo (*imagen1* e *imagen2*), ***insertaremos un fotograma*** clave al final de la animación (*fotograma n.º 240*), copia del inicial. Con este fotograma seleccionado, arrastraremos las imágenes sobre el escenario hasta su posición final, y les asignaremos un *alfa* 100%. Finalmente ***crearemos interpolación de movimiento*** entre los fotogramas 1-240 de ambas capas.
9. Para terminar la animación, en la capa asociada al primer texto, insertaremos un fotograma clave a mitad de animación (*fotograma n.º 120*). Con este fotograma seleccionado, y mediante la ayuda de las ***herramientas selección*** y ***transformación libre*** arrastraremos y colocaremos el símbolo texto2 en su posición final dentro del escenario, asignándole un *alfa* 100%.
10. Siguiendo en la capa anterior, en lugar de crear interpolación de movimiento entre sus fotogramas clave 1-120, le asignaremos una capa que le hará de ***guía de movimiento***. Para ello, pincharemos con el botón derecho del ratón sobre la capa texto1, y seleccionaremos la opción ***Añadir guía de movimiento***.
11. Selecciona el fotograma n.º 1 de la nueva capa guía, y mediante la ***herramienta Lápiz*** dibuja la trayectoria[18] que deseas que siga el texto desde su posición inicial hasta su posición final. Tras establecer la guía, asegurate que el centro del símbolo *texto1* en su fotograma n.º 1 esta al comienzo de la guía, y de forma similar en el fotograma n.º 120 respecto a su final. Para ello, ayudate de la ***herramienta Selección***, con su opción ***Ajustar a Objetos*** activada.

18 Una vez seleccionada la herramienta Lápiz, desde la ventana de Propiedades podremos personalizarlo. Entre sus propiedades destacar el Suavizado, encargada de suavizar los tramos curvas.

Figura 4.22. La herramienta Selección nos permite ajustar el simbolo texto1 a la guía

12. A continuación, situándote en el primer fotograma de la capa *texto1*, ***crea interpolación de movimiento*** entre este fotograma clave y el n.º 120, Adobe Flash se encargará de generar los fotogramas intermedios que proporcionarán la sensación de movimiento siguiendo el trazado marcado en la capa guía.

Figura 4.23. Pasos para añadir guía de movimiento a una capa

13. El resultado de la animación resultante podemos comprobarla desde el menú ***Control***, seleccionando ***Probar película***.
14. Desde el menú ***Archivo***, seleccionando ***configuración de publicación***, generaremos el archivo *swf* que insertaremos como objeto, <*object*></*object*>, en nuestro documento HTML/XHML.

5. SWFTOOLS

Uno de los grandes inconvenientes que presenta Adobe Flash es que se trata de un software cerrado y privativo, motivo por el cual, no existen otras herramientas software comparables que nos permitan generar archivos *swf* con prestaciones similares. No obstante, en muchas ocasiones prácticas Adobe Flash resulta ser una herramienta fastuosa para lo que se quiere hacer. Para la realización de animaciones Flash que no requiran de muchas florituras **SWFTools** es puede ser una alernativa. Esta es una herramienta software libre de código abierto disponible tanto en entornos Microsoft Windows como GNU/Linux que puede descargarse desde "*http://www.swftools.org/download.html*".

Figura 4.24. SWFTools es una alernativa libre a Adobe Flash útil para realizar animaciones sencillas

Entre las distintas utilidades disponibles en el paquete software SWFTools podríamos destacar las siguientes:

1. **jpeg2swf|png2swf|gif2swf**: Generan una animación Flash a partir de un conjunto de imágenes en formato *jpeg*, *png* o *gif* respectivamente. La sintaxis de utilización de estos comandos es la siguiente:

gif2swf img1.gif [img2.gif ...] [-o objeto.swf -X anchura-pixels -Y altura-pixels -r frames/segundo]

jpeg2swf img1.jpg [img2.jpg ...] [-o objeto.swf -X anchura-pixels -Y altura-pixels -r frames/segundo -j calidad]

png2swf img1.png [img2.png ...] [-o objeto.swf -X anchura-pixels -Y altura-pixels -r frames/segundo -q calidad]

Ejemplo de su utilización en Microsoft Windows[19] o GNU/Linux podría ser el siguiente:

c:/sfwtools> jpeg2swf.exe|gif2swfexe|png2swf.exe img1.jpg|gif|png img2.jpg|gif|png ... -o animacion.swf -X 500 -Y 400 -r 1

[usuario@equipo]$ jpeg2swf|gif2swf|png2swf img1.jpg|gif|png img2.jpg|gif|png ... -o animacion.swf -X 500 -Y 400 -r 1

2. **wav2swf**: Genera una animación Flash a partir de una canción en formato *wav*. La sintaxis de utilización de este comando es la siguiente:

wav2swf cancion.wav [-o objeto.swf -s muestras/segundo -r frames/segundo]

Ejemplo de su utilización en Microsoft Windows o GNU/Linux podría ser el siguiente:

c:/sfwtools> wav2swf.exe cancion.wav -o cancion.swf -s 22050

[usuario@equipo]$ wav2swf cancion.wav -o cancion.swf -s 22050

19 Tras la instalación de SWFTools en Microsoft Windows se crea una carpeta en "c:/swftools" donde se localizan las utilidades ejecutables. En los ejemplos se supone que los archivos que se manejan se encuentran en su interior.

¡¡Sugerencia!! En el caso de dispongamos de música en formato *mp3*, y queramos convertirla en formato *swf* para insertarla en un documento HTML/XHTML como objeto Flash, será necesaria una conversión intermedia de *mp3* a *wav*. Para ello, es posible hacer uso del software libre "**ffmpeg**" disponible tanto para Microsoft Windows como para GNU/Linux (*http://ffmpeg.org/download.html*). Su sintaxis sería la siguiente:

```
ffmpeg -i cancion.mp3 cancion.wav
```

ffmpeg puede resultar igualmente útil para convertir videos en formato mpeg, vob, flv, ... a formato avi (*ffmpeg -i pelicula.mpg|mov|flv|vob|mp4|... pelicula.avi*) para posteriormente obtener una película Flash mediante la utilidad de SWFTools **avi2swf**.

3. **pdf2swf**: Genera una animación Flash a partir de una presentación en formato pdf (Portable Document Format). La sintaxis de utilización de este comando es la siguiente:

```
pdf2swf fichero.pdf [-o objeto.swf -p rango-paginas -j calidad]
```

Ejemplo de su utilización en Microsoft Windows o GNU/Linux podría ser el siguiente:

```
c:/sfwtools> pdf2swf.exe presentacion.pdf -o presentacion.swf -j 90
[usuario@equipo]$ pdf2swf.exe presentacion.pdf -o presentacion.swf -j 90
```

Esta utilidad de SWFTools puede resultar muy útil en el caso de que dispongamos de una presentación en formato *pps*, *ppt* (presentación Microsoft PowerPoint), *odp* (presentación OpenOffice) u obtenida a partir de cualquier otro programa, que al imprimirla en formato *pdf*, puede ser convertida en un archivo u objeto *swf*.

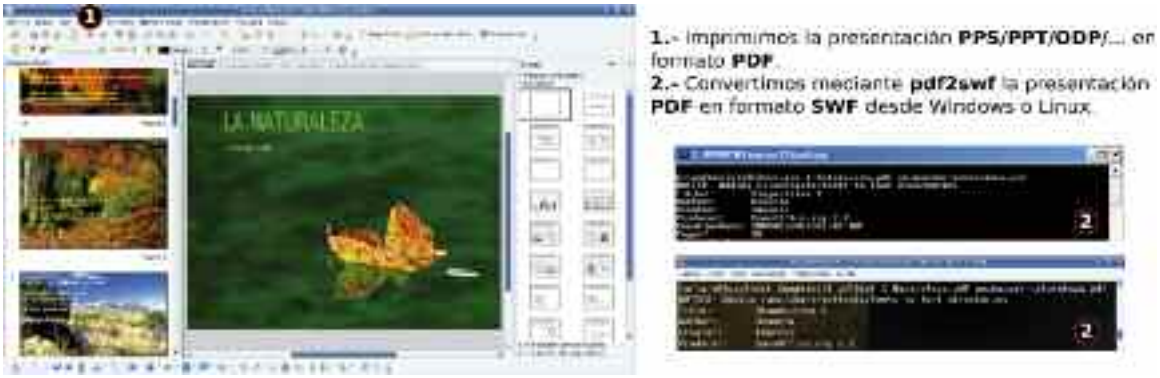


Figura 4.25. Ejemplo de conversión de una presentación en formato PDF a formato Flash SWF

4. **avi2swf**: Genera una animación Flash a partir de un video en formato avi. La sintaxis de utilización de este comando es la siguiente:

```
avi2swf pelicula.avi [-o objeto.swf -n numero-frames-covertir -m mp3-bitrate -r muestras-mp3/segundo -q calidad -s escala]
```

Ejemplo de su utilización en Microsoft Windows o GNU/Linux podría ser el siguiente:

```
c:/sfwtools> avi2swf.exe pelicula.avi -o pelicula.swf -r 44100 -q 50 -s 80
[usuario@equipo]$ avi2swf pelicula.avi -o pelicula.swf -r 44100 -q 50 -s 80
```

5. **swfc**: Genera una animación Flash a partir de un script. La sintaxis de utilización de este comando es la siguiente:

```
swfc archivo-script.sc [-o objeto.swf]
```

Para saber más sobre la utilidad **swfc** se recomienda consultar la documentación ofrecida por SWFTools en su Web: “http://www.swftools.org/swfc/swfc.html”.

Capítulo 5

GRAMÁTICAS XML PARA GRÁFICOS: SVG

Bajo la recomendación de la W3C, el mundo de las comunicaciones en Internet tiende hacia una programación basada en estándares auspiciados por gramáticas (o vocabularios) XML. Estos estándares en la mayoría de los casos no son capaces de manejar gráficos con una calidad media-alta. En muchos casos únicamente son capaces de leer datos numéricos sin formato[20] para hacer una sencilla representación, normalmente con una potencia menor que la que tiene el elemento `img' existente en HTML. SVG trata de rellenar ese espacio vacío ofreciendo una descripción vectorial y estructurada de gráficos, a la vez que permite el uso de imágenes basadas en píxeles. A su vez puede ser usado como un lenguaje autónomo (stand-alone) o como un espacio de nombres XML de otras gramáticas, por ejemplo se podrían utilizar conjuntamente XHML, SVG y MathML para crear documentos científicos de gran calidad gráfica.

Al tener una gramática XML, SVG tiene unos sólidos cimientos sobre los que asentarse y aprovechar toda la implementación existente entorno a esta gramática para manejar sonidos, animaciones y respuestas a eventos convirtiéndose en un estándar abierto que compite directamente con el Flash de Adobe. La unión de SVG con SMIL, DOM y CSS ofrece, desde un principio, mayores posibilidades de futuro que este último.

En la creación de portales y páginas Web, SVG ofrece varias opciones para utilizar toda su potencia, ajustándose a las necesidades del programador:

— Funcionamiento stand-alone: En este caso el navegador web carga directamente el archivo ofreciéndolo como página web. Este será el modelo utilizado en este capítulo para mostrar los ejemplos creados.

— Embebido por referencia: Una página web invoca a un documento SVG indicando que debería ser embebido como un componente de esa página web. Esta referenciación podría hacerse básicamente de tres formas:

1. A través del elemento **img**, en el que el ancho y alto del gráfico pueden darse como atributos. En este caso se requiere del atributo `alt', usado para ofrecer una cadena de texto alternativa, para aquellos clientes web que tienen configurado el navegador para que no carge las imágenes.

2. A través del elemento **object** que puede contener otros elementos anidados en su interior. Esto permite ofrecer diferentes formatos anidados, de tal forma que se mostrará por pantalla el más externo de estos elementos anidados que a su vez sea legible para el navegador.

---

20 Cuando hablamos de datos sin formato hacemos referencia a formas de almacenamiento de bajo nivel como cvs (datos separados por ;) o similar.

3. El elemento **applet** que puede invocar a un applet de Java para visionar contenido SVG dentro de una página web. El uso habitual de estos applets es mostrar imágenes en formatos inusuales o aquellos que, por cualquier razón, necesitan estar bajo el control de un programa.

— Embebido en el propio documento: Esto sucede cuando el propio código SVG está contenido en el archivo .html mostrado por el navegador. Esto sucede por ejemplo en páginas web XHTML.

— Link externo a través del elemento **a**: Esto se utiliza para referenciar a un SVG que puede ser mostrado por cualquier visor para SVG's stand-alone.

— Referenciado desde una propiedad CSS2 o XSL.

1. CONTENIDOS DEL CAPÍTULO

El presente capítulo sigue la siguiente estructura:

— Comienza dando una introducción muy simple al formato y estructura de un documento SVG.

— A continuación se informa del software necesario para la visualización de los ejercicios prácticos que se plantean, señalando los sitios Web desde donde puede realizarse su descarga.

— Seguidamente se muestran dos sencillos ejemplos para mostrar las posibilidades de SVG.

— En la siguiente sección se hace un recorrido más detallado por los elementos de dibujo.

— Posteriormente se muestran diferentes usos de SVG para crear y animar gráficos, mostrando la posibilidad de interactuar con el usuario final o cliente web.

2. FORMATO DE UN DOCUMENTO SVG

Un documento SVG es un documento XML por lo que la estructura será fácil de entender y relacionar con otros tipos de documentos XML que el lector conozca. Como documento XML, necesita de un namespace (espacio de nombres), que es http://www.w3.org/2000/svg. También es recomendable proporcionar un public identifier (identificador público): PUBLIC "-//W3C//DTD SVG 1.1//EN"; al igual que un system identifier (identificador de sistema), que viene dado por http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd. En un archivo .dtd se especifican los elementos que pueden formar parte del documento XML y de esta forma comprobar que se contiene una estructura válida. Una declaración de documento SVG podría ser:

```
1  <?xml version="1.0" standalone="no"?>
2  <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
3        "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
4  <svg width="21cm" height="29cm" version="1.1"
5     xmlns="http://www.w3.org/2000/svg">
6      ...
7  </svg>
```

Listado 1. Cabecera de un archivo SVG

La primera línea indica la versión de XML y si el gráfico está pensado para visualizarse solo (**standalone="yes"**) y no depende de validaciones externas o si el procesador SVG debe acceder

a ellas para comprobar la validez del documento (**standalone="no"**). En este último caso la validación se realiza a través de un fichero público denominado **svg11.dtd**. En cualquier caso, normalmente los parser conocen suficiente acerca del formato de los documentos SVG, que no requieren de ese acceso externo para la validación lo que ha provocado que el uso previsto inicialmente para **standalone** no haya sido completamente implementado. Por esta razón la cabecera del documento SVG puede ser reducida eliminando la parte de identificadores.

A continuación comienza la definición del cuerpo SVG con la definición de la anchura (**width**) y altura (**height**) del dibujo, así como la versión y el namespace (definido a través de **xmlns**) con el que se evitan conflictos entre los nombres que se dan a los elementos del documento XML, que podría estar compuesto por otros sub-documentos XML.

Los elementos gráficos que contiene un documento SVG se dividen en: formas básicas (**rect**, **circle**, **ellipse**, **line**, **polyline** y **polygon**), elementos de referenciación (**image** y **use**), rutas (**path**) y textos (**text**). La forma de definir un color es por medio de una secuencia RGB, aunque hay un gran número de nombres de colores predefinidos (ver apéndice A).

Se pueden definir contenedores para los elementos gráficos o para contener otros contenedores a través de **svg**, **g**, **defs**, **symbol**, **clipPath**, **mask**, **pattern**, **marker**, **a** y **switch**. Y también contenedores para texto: **text**, **tspan**, **tref**, **textPath** y **altGlyph**.

La animación y respuesta ante eventos generados por el usuario se hace a través de las interfaces DOM (document object model). El DOM se puede entender como una interfaz de programación de aplicaciones (API) que permite programar scripts (con un estilo orientado a objetos) que interactuarán con documentos XML. Se pueden utilizar diferentes lenguajes de programación para interactuar con el DOM, pero inicialmente está pensado para ECMAScript (inspirado en JavaScript).

3. SOFTWARE PARA VISUALIZAR EL FORMATO SVG

Puesto que se trata de un estándar la mayor parte de los navegadores están haciendo grandes esfuerzos por soportar completamente este formato. En la dirección **http://www.codedread.com/svg-support.php** se puede encontrar una relación de navegadores calificados en función de su soporte al formato SVG. Esta calificación está basada en los ficheros de prueba proporcionados en la página del W3C: **http://www.w3.org/Graphics/SVG/WG/wiki/Test_Suite_Overview**. Al entrar con un navegador es posible ir comprobando las habilidades del mismo para reproducir apropiadamente este formato. En el momento de escribir estas líneas, únicamente el navegador Opera ofrece es capaz de trabajar con casi el 100% de las posibilidades de SVG. En la página http://www.opera.com/docs/specs/svg/index.dml, se muestra el conjunto de los elementos SVG soportados por este navegador. Prácticamente, el resto de los navegadores se limitan a soportar características de comportamiento estático de SVG, aunque en muy poco tiempo, navegadores como por ejemplo Firefox, lo ofrecerán en sus nuevas versiones.

Para realizar los ejercicios de este capítulo es recomendable instalar Opera, disponible para muchos sistemas operativos y descargable desde **http://www.opera.com/**. En este caso bastará con visualizar los ejemplos directamente a través del navegador.

Si se desea utilizar Firefox/Mozilla, y la versión con la que trabajas no soporta las características SVG exigidas en este libro, existe otra posibilidad aunque con mayores limitaciones en cuanto a las opciones soportadas: hacer uso del proyecto FakeSmile, descargable desde **http://leunen.d.free.fr/fakesmile/index.html**. FakeSmile consiste de un programa en java que emula el comportamiento dinámico de SVG. Se mostrará su uso en la siguiente sección junto con el primer ejemplo de SVG animado. Existen otras posibilidades para FireFox que puedes encontrar como **adds-on** y que están listados en **http://wiki.svg.org/Viewer_Matrix**.

Otra posible opción es usar Batik, un proyecto dentro de Apache (**http://xmlgraphics.apache.org/batik/**). Es una herramienta basada en Java que permite desarrollar aplicaciones o

applets que usan SVG. En **http://xmlgraphics.apache.org/batik/status.html** se muestra las características SVG soportadas por Batik. Para hacerlo funcionar basta con descargarse los archivos **.jar** que componen el proyecto y ejecutarlos:

```
[user@portatil]$ java -jar fichero.jar
```

4. PRIMER EJEMPLO: RÁPIDO, SENCILLO Y VISTOSO

Muchas empresas muestran en su página web una composición de fotografías y texto en movimiento para hacerla más atractiva. Como se ha visto en el capítulo anterior, una posibilidad es crear dichas animaciones por medio de Flash. Actualmente, con SVG existe la posibilidad de hacer este tipo de creaciones con un formato completamente libre. En esta sección se va a mostrar como crear una animación similar a la mostrada en el capítulo dedicado a Adobe Flash donde se publicita nuestro ya bien conocido Escarbapedal.

Para visualizar adecuadamente este ejemplo es necesario utilizar el navegador Opera o en su defecto Mozilla Firefox con el complemento FakeSmile. El primer paso será crear una carpeta donde serán almacenados los archivos: **fotos_transicion.svg**, **foto1.jpg** y **smil.user.js**[21].

Tomemos inicialmente la fotografía de tamaño de 635x150 utilizada en el capítulo anterior (ver figura 5.1). La renombramos como **foto1.jpg** y comprobamos el tamaño:

```
[usuario@portatil]$ identify foto1.jpg
foto1-rec2.jpg JPEG 635x150 635x150+0+0 DirectClass 8-bit 77.3496kb
```

Figura 5.1. Imagen que utilizaremos para las primeras animaciones

Para mostrar como trabajar con diferentes tamaños, supondremos que la imagen debe ocupar una anchura en el sitio web de únicamente 550 píxeles, esto significa que debería ser escalada con un factor de: 550/635 = 0.8661. Así, el tamaño del dibujo SVG debería ser[22] de 550x130. Un archivo SVG necesita de una cabecera, cuyas partes serán explicadas en la próxima sección detalladamente. Para este ejemplo basta con saber que en ella se indica el tamaño del gráfico a crear, así las primeras líneas del archivo **fotos_transicion1.svg**, el cual vamos a crear, serían:

---

21 El archivo **smil.user.js** puedes descargarlo de **http://leunen.d.free.fr/fakesmile/faq.html**, en el enlace **download link** que se encuentra en la pregunta **How do I use it in my browser?**, pero recuerda que únicamente es necesario si vas a utilizar Firefox en lugar de Opera. El fichero **fotos_transicion.svg** obviamente estará inicialmente vacio. Por último **foto1.jpg** puede ser cualquier imagen.

22 Observa que 635*0.8661=550 y 150*0.8661=130.

```
1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2  <svg xmlns="http://www.w3.org/2000/svg"
3      xmlns:xlink="http://www.w3.org/1999/xlink"
4      width="550" height="130">
5     <script type="text/ecmascript" xlink:href="smil.user.js"/>
```

Listado 2. Primeras líneas del archivo SVG

Observa que la línea **<script type="text/ecmascript" xlink:href="smil.user.js"/>** no es necesaria si utilizas Opera, pero es aconsejable incluirla en este fichero para que un usuario de Mozilla/Firefox pueda visualizar tu animación (recuerda que el script **smil.user.js** debe encontrarse en la misma carpeta que el archivo **fotos_transicion1.svg**).

A través del elemento **<image > </image>** se incrusta una imagen **.png** o **.jpg** dentro de un gráfico SVG. Los atributos mínimos de este elemento son:

1. Un enlace a la ubicación de la imagen, en este caso se supone que en el mismo directorio.
2. La posición de la esquina superior izquierda de la imagen, en este caso se supondrá el punto (0,0).
3. El tamaño al cual se debe escalar la imagen. En este caso se elige el mismo que el tamaño del gráfico SVG, es decir 550x130.

El código podría ser:

```
<image xlink:href="./foto1.jpg" x="0" y="0" width="550" height="130">
  </image>
```

O sencillamente:

```
<image xlink:href="./foto1.jpg" x="0" y="0" width="550" height="130"/>
```

debido a que el elemento no tiene ningún contenido. El código completo sería:

```
1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2  <svg xmlns="http://www.w3.org/2000/svg"
3      xmlns:xlink="http://www.w3.org/1999/xlink"
4      width="550" height="130">
5     <image xlink:href="./foto1.jpg" x="0" y="0" width="550" height="130"/>
6  </svg>
```

Listado 3. Incrustar imágenes en SVG

Para dar un aspecto de movilidad podría realizarse un escaldo de la imagen desde un tamaño, digamos un 30% mayor (es decir escalar la imagen por 1.3). Para ello bastaría con modificar el elemento **<image > </image>** añadiendo un atributo **AnimateTransform** del tipo **scale**, tal como se muestra a continuación:

```
<image xlink:href="./foto1.jpg" x="0" y="0" width="550" height="130">
    <animateTransform attributeName="transform" type="scale" dur="6s" from="1.3" to="1"/>
  </image>
```

Se está indicando que muestre por pantalla la imagen transformándose desde un tamaño[23] 715x169 a un tamaño de 550x130. El proceso de transformación durará 6 segundos.

Una vez que transcurran esos 6 segundos la imagen quedará estática. Si deseamos que el proceso se repita continuamente bastaría con añadir la propiedad **repeatCount="indefinite"** al atributo **AnimateTransform**:

```
<animateTransform attributeName="transform" type="scale" dur="6s" from="1.3" to="1"
  repeatCount="indefinite"/>
```

Cuando la imagen alcanza la escala normal es escalada de nuevo en un factor de 1.3 indefinidamente. Esto produce un choque visual que podría calificarse de no agradable por lo que una posible solución podría ser el hacer la imagen apareciera suavemente, modificando la propiedad **opacity** a través del atributo **animate**. Se debería añadir al elemento **<image > </image>** la siguiente línea:

```
<animate attributeName="opacity" dur="6s" from="0" to="1" repeatCount="indefinite"/>
```

En cualquier caso el efecto continua sin ser de nuestro agrado. Sería deseable comenzar con una opacidad 0 (completamente transparente), que esta se fuera incrementando y que finalmente volviera a decrecer hasta 0. Puesto que esto no podría hacerse utilizando las propiedades **from** y **to** fueron inventadas las propiedades **values** y **keyTimes** que contienen una secuencia de valores que determinan como evoluciona la animación a lo largo del tiempo. Por ejemplo:

```
<animate attributeName="opacity" dur="6s" values="0; 1; 0" keyTimes="0; 0.5; 1"
  repeatCount="indefinite"/>
```

keyTimes contiene una secuencia de valores comprendida entre 0 y 1 haciendo una referencia en tanto por 1 a la duración expresada a través de la propiedad **dur**, en este caso 6 segundos. En este ejemplo se establece que la propiedad **opacity** tomará los valores 0, 1 y 0 en los instantes de tiempo inicial (0*6 s=0 s), mitad (0.5*6 s=3 s) y final (1*6 s=6 s), consiguiéndose de esta forma, el efecto deseado. El contenido del fichero .svg sería:

```
1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2  <svg xmlns="http://www.w3.org/2000/svg"
3       xmlns:xlink="http://www.w3.org/1999/xlink"
4       width="550" height="130">
5
6     <image xlink:href="./foto1.jpg" x="0" y="0" width="550" height="130">
7        <animate attributeName="opacity" dur="6s" values="0; 1; 0" keyTimes="0; 0.5; 1"
             repeatCount="indefinite"/>
8        <animateTransform attributeName="transform" type="scale" dur="6s" from="1.3" to="1"
             repeatCount="indefinite"/>
9     </image>
10 </svg>
```

Listado 4. Incrustar imágenes en SVG

23 Observa que 550*1.3=715 y que 130*1.3=169.

5. UNA SECUENCIA DE IMÁGENES

Supongamos ahora que deseamos crear una secuencia de tres imágenes repartidas a lo largo de 20 segundos, sobre las cuales se superpondrá un texto publicitario. Cada imagen comenzará con una opacidad nula (transparente), se hará completamente opaca y volverá a ser transparente. Además inicialmente estará escalada en un factor de x1.3 que se irá reduciendo paulatinamente hasta un factor de escala x1. La secuencia de cada una de ellas podría ser la mostrada en la figura 5.2.

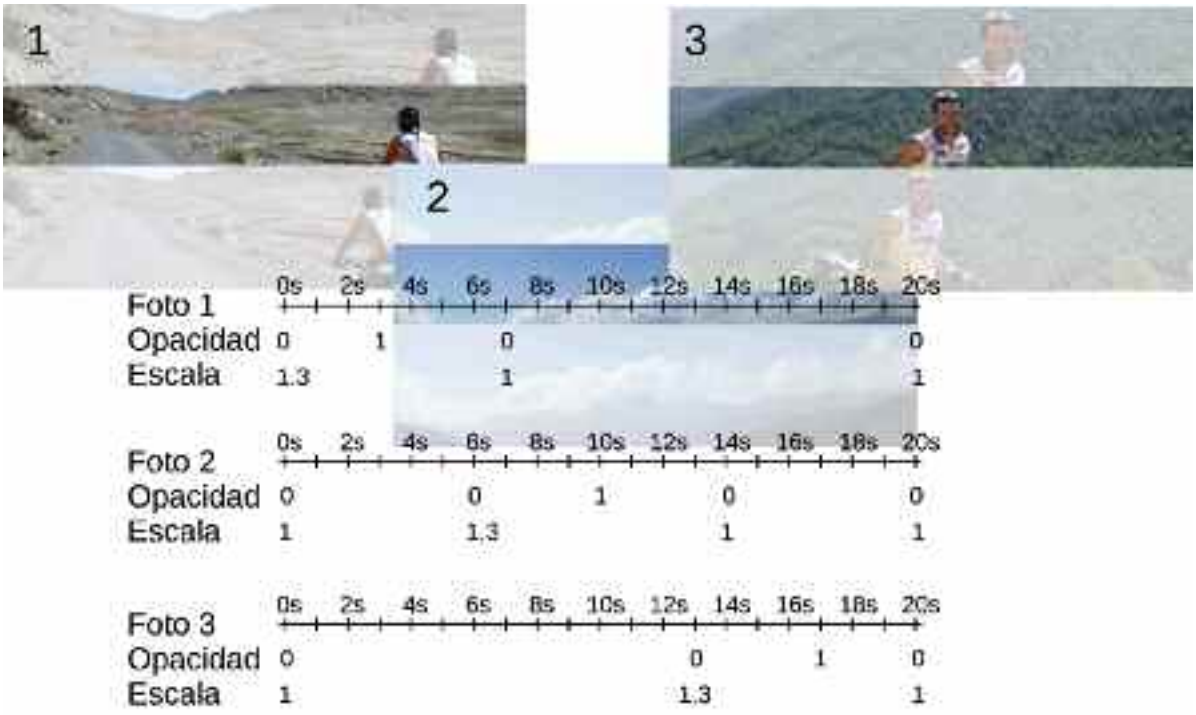


Figura 5.2. Tiempos de cambio en la secuencia de fotos

En la figura 5.2 se muestra los instantes de tiempos en los que los atributos de opacidad (opacity) y de escala (scale) tomarán los diferentes valores que producirán el efecto deseado. Por ejemplo la imagen catalogada como Foto1 (foto1.jpg) sufre cambios importantes en los instantes 0s, 3s, 7s y 20s, lo que ocurra entre estos tiempos será interpolado por el visor SVG utilizado. Estos instantes de tiempo deben estar indicados en la propiedad **keyTimes** en tanto por uno. Esto significa que:

Tiempo:	0s	3s	7s	20s
Tanto por uno:	0/20 = **0**	3/20 = **0.15**	7/20 = **0.35**	**1**

En estos instantes de tiempo tanto la opacidad, como la escala toman los valores indicados en la figura 5.2, lo que da lugar al siguiente código SVG:

```
<animate attributeName="opacity" dur="20s" values="0; 1; 0; 0" keyTimes="0; 0.15; 0.35; 1"
  repeatCount="indefinite"/>
<animateTransform attributeName="transform" type="scale" dur="20s" values="1.3; 1; 1"
  keyTimes="0; 0.35; 1" repeatCount="indefinite"/>
```

De igual forma se pueden obtener los valores **keyTimes** para las imágenes **foto2.jpg** y **foto3.jpg** obteniendo el código mostrado en el siguiente listado al que se han añadido dos nuevas imágenes que componen los textos publicitarios tal y como se muestra en la figura 5.3.

```
1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2  <svg xmlns="http://www.w3.org/2000/svg"
3      xmlns:xlink="http://www.w3.org/1999/xlink"
4      width="550" height="130">
5
6      <image xlink:href="./foto1.jpg" x="0" y="0" width="550" height="130">
7          <animate attributeName="opacity" dur="20s" values="0; 1; 0; 0" keyTimes="0; 0.15;
               0.35; 1" repeatCount="indefinite"/>
8          <animateTransform attributeName="transform" type="scale" dur="20s" values="1.3;
               1; 1" keyTimes="0; 0.35; 1" repeatCount="indefinite"/>
9      </image>
10
11     <image xlink:href="./foto2.jpg" x="0" y="0" width="550" height="130">
12         <animate attributeName="opacity" dur="20s" values="0; 0; 1; 0; 0" keyTimes="0;
               0.3; 0.5; 0.7; 1" repeatCount="indefinite"/>
13         <animateTransform attributeName="transform" type="scale" dur="20s" values="1;
               1.3; 1; 1" keyTimes="0; 0.3; 0.7; 1" repeatCount="indefinite"/>
14     </image>
15
16     <image xlink:href="./foto3.jpg" x="0" y="0" width="550" height="130">
17         <animate attributeName="opacity" dur="20s" values="0; 0; 1; 0" keyTimes="0; 0.65;
               0.85; 1" repeatCount="indefinite"/>
18         <animateTransform attributeName="transform" type="scale" dur="20s" values="1;
               1.3; 1" keyTimes="0; 0.65; 1" repeatCount="indefinite"/>
19     </image>
20
21     <image xlink:href="./titulos-texto1.png" x="60" y="20" width="475" height="94" id="texto"
           opacity="0">
22         <animate attributeName="opacity" dur="20s" values="0; 1; 0" keyTimes="0; 0.5; 1"
               fill="remove" repeatCount="indefinite"/>
23     </image>
24
25     <image xlink:href="./titulos-texto2.png" x="60" y="20" width="475" height="94" id="texto"
           opacity="0">
26         <animate attributeName="opacity" dur="20s" values="0; 0; 1; 0" keyTimes="0; 0.4;
               0.7; 1" fill="remove" repeatCount="indefinite"/>
27     </image>
28 </svg>
```

Listado 5. Fichero para la transición de imágenes

Figura 5.3. Instantánea de la animación SVG creada

Suponiendo que el fichero es guardado finalmente con el nombre **fotos_transicion.svg**, la incrustación en un documento HTML sería:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<object type="image/svg+xml" data="fotos_transicion.svg" width="550px" height="130px"/>
</html>
```

6. CONOCIENDO LAS BASES DE SVG

Como casi siempre, la mejor forma de aprender a utilizar un nuevo lenguaje es a través de ejemplos que ilustren las diferentes posibilidades. En primer lugar se van a mostrar algunas de las posibilidades SVG cuando se desean crear contenidos estáticos.

```
1  <?xml version="1.0" standalone="no"?>
2  <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
3        "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
4  <svg width="21cm" height="29cm" version="1.1"
5     xmlns="http://www.w3.org/2000/svg">
6     <desc> Diferentes elementos de dibujo  </desc>
7
8     <rect x="0.5cm" y="0.5cm" width="2cm" height="1cm" rx="0" ry="0" fill="blue"/>
9     <circle cx="11cm" cy="4cm" r="3.2cm" fill="none" stroke="red"/>
10
11    <linearGradient id="gradiente" gradientUnits="objectBoundingBox">
12          <stop offset="0" style="stop-color:blue"/>
13          <stop offset="0.5" style="stop-color:orange"/>
14          <stop offset="1" style="stop-color:green"/>
15    </linearGradient>
16
17    <ellipse cx="4cm" cy="7cm" rx="3.2cm" ry="1.50cm"  style="fill:url(#gradiente)"/>
18    <g fill="none" stroke="coral" stroke-width=".1cm">
19          <line x1="8cm" y1="10cm" x2="12cm" y 2="2.5cm" opacity="0.7" />
20          <polyline points="50,570 75,470 100,570 125,470 150,570 175,470" />
21          <polygon points=" 250,450 297,484 279,540" />
22    </g>
23    <text x="3.5cm" y="3cm">
24          Esto es un texto
25          <tspan x="1cm" dy="1.5cm" font-size="36" style="fill:orange;stroke:green">Esto es
            otro texto</tspan>
26    </text>
27 <text x="0.5cm" y="9.5cm" font-size="46" stroke-width=".05cm" style="fill:yellow;stroke:brown"
            rotate="40 0 70">
28       B M
29 </text>
30 <!— Este es un comentario que podría explicar el código —>
31
32    <path id="curva" fill="none" stroke="none" d="M 100,483 C 100,483 145,379 210,375 C
            274,372 281,473 356,470 C 431,466 446,373 508,360 C 570,347 602,418 602,
            418" />
33
34 <text style="font-size:10">
35       <textPath xlink:href="#curva">
36          Este es un texto que va siguiendo la curva desde el principio hasta el final.
            Subiendo y bajando.
37    </textPath>
38 </text>
```

```
39
40  <g style="fill-opacity:0.5">
41      <rect x="14cm" y="1cm" rx="0.52cm" ry="0.52cm" width="3.5cm" height="6cm" fill="green"/>
42      <circle cx="11cm" cy="12cm" r="2.2cm" fill="blue" stroke="none"/>
43      <rect x="9cm" y="12.5cm" rx="0.2cm" ry="0.2cm" width="7cm" height="3cm" fill="violet"/>
44  </g>
45
46  <rect x=".01cm" y=".01cm" width="18cm" height="16cm"  fill="none" stroke="blue" stroke-
      width=".2cm" />
47  </svg>
```

Listado 6. Ejemplo con elementos SVG básicos

Este fichero **.svg** al ser visualizado con un navegador que soporte adecuadamente este formato[24] mostraría algo similar a lo indicado en la figura 5.4.

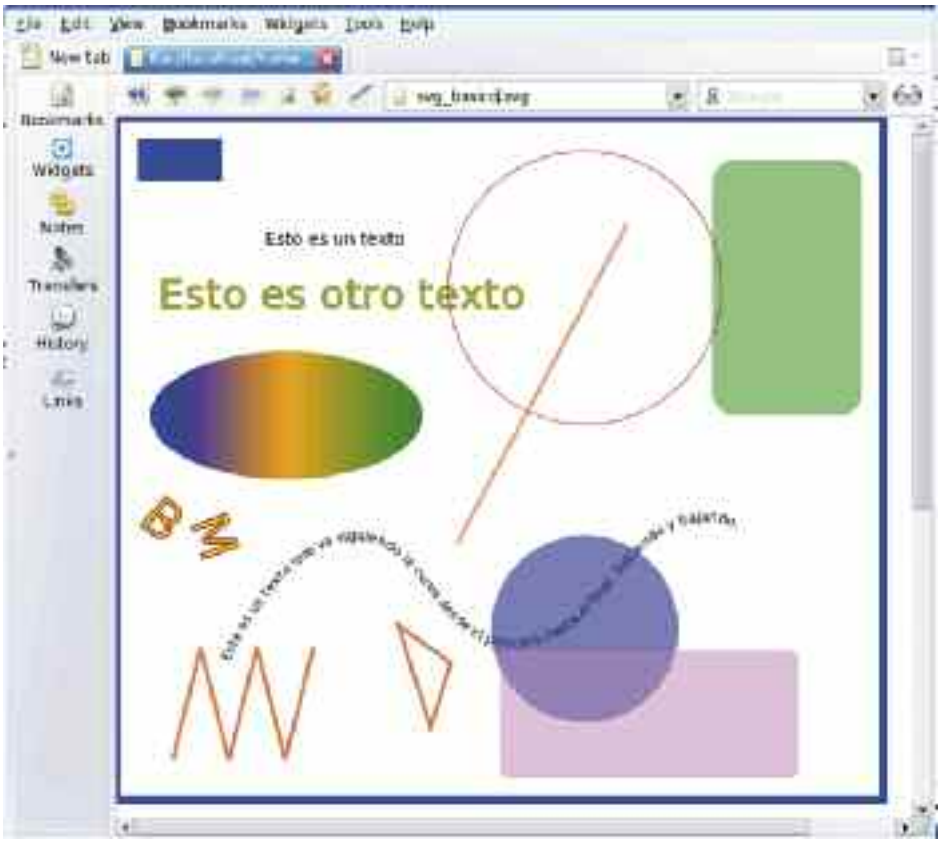


Figura 5.4. Resultado proporcionado por el navegador Opera al procesar el fichero

Tras las líneas de inicialización del documento XML y de la subparte SVG con las dimensiones máximas del dibujo y el namespace este ejemplo contiene las etiquetas **<desc> ... </desc>**. Son utilizadas para realizar una descripción de un elemento gráfico. En realidad cada elemento contenedor o elemento gráfico puede tener asociado un **desc** o un **title** que no repercutirá (no serán renderizados) en el dibujo como parte de los gráficos, pero que pueden ser utilizados por los visores SVG, por ejemplo para ser mostrados como tips al pasar el ratón por encima de ellos.

24 En el momento de escribir estas líneas únicamente el navegador Opera soporta en un porcentaje cercano al 100% el formato SVG.

El primer elemento gráfico es un rectángulo:

```
<rect x="0.5cm" y="0.5cm" width="2cm" height="1cm" rx="0" ry="0" fill="blue"/>
```

Cada elemento gráfico tiene unos parámetros propios y otros parámetros comunes a otros elementos gráficos. En este caso **x** e **y** están indicando las coordenadas **índice:coordenadas** de la esquina superior izquierda del rectángulo a dibujar. Los parámetros **width** y **height** indican la anchura y altura respectivamente. A continuación, **rx** y **ry** indican los radios de redondeo elípticos para las esquinas del rectángulo; como por defecto toman el valor cero, estos dos parámetros podrían haberse obviado. Por último, **fill** permite determinar el color[25] de relleno para este elemento de dibujo.

El siguiente elemento es 'circle' que renderiza un círculo en la pantalla:

```
<circle cx="11cm" cy="4cm" r="3.2cm" fill="none" stroke="red"/>
```

Los parámetros **cx**, **cy** y **r** son las coordenadas del centro y el radio respectivamente. En este caso el círculo no tiene relleno, pero el borde del mismo será de color rojo (**stroke="red"**).

```
<linearGradient id="gradiente" gradientUnits="objectBoundingBox">
   <stop offset="0" style="stop-color:blue"/>
   <stop offset="0.5" style="stop-color:orange"/>
   <stop offset="1" style="stop-color:green"/>
</linearGradient>
```

Estas líneas definen un gradiente lineal para un determinado relleno que variará desde el color azul (blue) al color verde (green) pasando por el naranja (orange). El parámetro **id** se utiliza para definir un identificador del elemento, en este caso **gradiente**; este identificador es nombre elegido por el usuario. En SVG se hace uso de referencias URI (uniform resource indicator)[26] para hacer una referencia al objeto identificado en otra parte el gráfico. Como se verá a continuación la referenciación se hace utilizando la estructura **url(#nombre_identificador)**.

El siguiente parámetro es **gradientUnits** al cual se le ha asignado **"objectBoundingBox"**. Esto indica que el vector gradiente que define el cambio de colores es definido como porcentajes o fracciones de la bounding box[27] del objeto a rellenar con el gradiente.

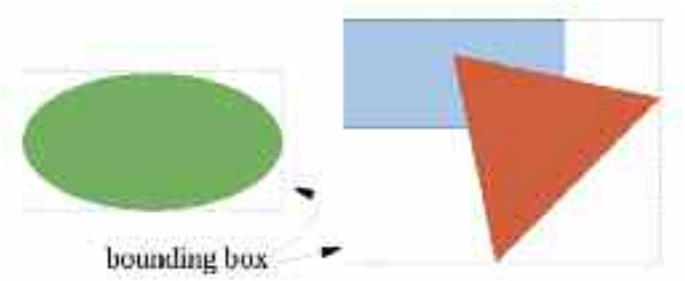


Figura 5.5. Indicación del bounding box de un gráfico o conjunto de gráficos

25 Ver apéndice Colores para conocer posibles valores para asignar a **fill**. En caso de no desear un relleno puede ser asignado el valor **none**.

26 URI es una forma de encapsular un nombre en un espacio de nombres registrados, y etiquetarlo con el espacio de nombres, produciendo un miembro del conjunto universal. Por ejemplo, una URL es un tipo de URI utilizado para identificar un objeto (una WEB) en Internet.

27 El bounding box es el rectángulo imaginario que encierra de forma ajustada a un objeto gráfico. Ver figura 5.5.

Los **stop offset** indican el punto en el que el color indicado por **stop-color** estará en estado puro. Por tanto en la posición 0 (la parte izquierda de la bounding box) el color de relleno será únicamente azul, en la posición 0.5 (la posición central de la bounding box) el color será naranja, y por último en la posición 1 (la parte derecha de la bounding box) será verde, tal y como se muestra en la figura 5.4.

```
<ellipse cx=”4cm” cy=”7cm” rx=”3.2cm” ry=”1.50cm”  style=”fill:url(#gradiente)”/>
```

La definición de una elipse es similar a la del círculo con la salvedad de que son necesarios dos radios. En este caso se hace uso del atributo **style** para asignar un relleno al elemento de dibujo. La forma de asignar ese relleno es a través del identificador del **linearGradient** definido anteriormente.

Las siguientes líneas de código representan un conjunto de elementos gráficos que conforman un grupo SVG:

```
<g fill=”none” stroke=”coral” stroke-width=”.1cm”>
   <line x1=”8cm” y1=”10cm” x2=”12cm” y2=”2.5cm” opacity=”0.7” />
   <polyline points=”50,570 75,470 100,570 125,470 150,570 175,470” />
   <polygon points=”250,450 297,484 279,540” />
</g>
```

Los elementos de este grupo son una línea (**line**), una poli-línea (**polyline**) y un polígono (**polygon**). El grupo se construye haciendo uso de la etiqueta **<g> </g>**, en la que se declaran una serie de características comunes a todos los elementos de ese grupo; en este caso no tendrán relleno (**fill=”none”**), las líneas que definen el borde de los elementos serán de color **coral** (**stroke= ”coral”**) y el grosor de dichos bordes será 0.1 centímetros (**stroke-width=”.1cm”**).

Para construir una línea se deben proporcionar las coordenadas x e y de dos puntos (**x1= ”8cm” y1=”10cm” x2=”12cm” y2=”2.5cm”**). Las distancias se miden desde la esquina superior izquierda.

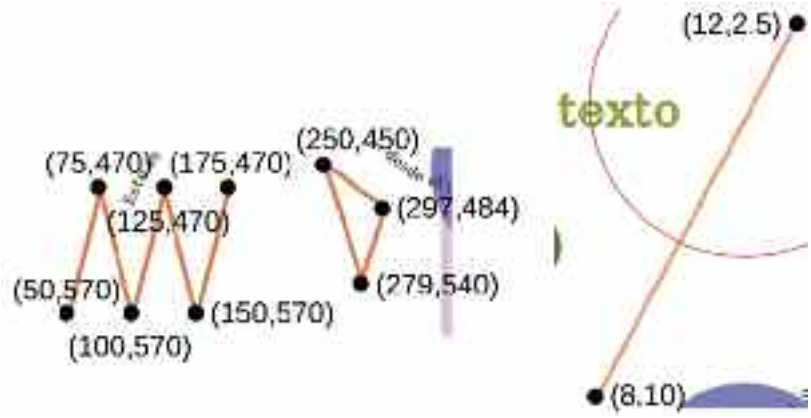


Figura 5.6. Puntos que definen líneas, poli-líneas y polígonos

El atributo **opacity** (que puede tomar un valor entre 0 y 1) hace semitransparente a la línea de tal forma que permite que el círculo rojo se vea a través de ella.

La figura 5.6 muestra las coordenadas de los puntos utilizados en la línea, así como los empleados en la construcción de la poli-línea y el polígono. La diferencia entre estos dos elementos de dibujo es que el elemento **polygon** une el primer y último punto de la secuencia proporcionada, en cambio **polyline** no hace esa unión.

Obsérvese el uso de las unidades del dibujo. Hasta el momento las medidas del dibujo se habían proporcionado en centímetros (cm), pero esta no es la única unidad de medida posible. SVG admite em (igual al tamaño de la actual fuente utilizada), ex (igual a la altura de la x en la fuente utilizada), px (pixel), pt (un punto, que es igual a 1/72 pulgadas), pc (una pica, que es igual a 12pt), cm (centímetros), mm (milímetros), in (pulgadas) y porcentajes (con respecto a la altura y anchura del dibujo).

```
<text x="3.5cm" y="3cm">
   Esto es un texto
   <tspan x="1cm" dy="1.5cm" font-size="36" style="fill:orange;stroke:green">Esto es otro
       texto</tspan>
</text>
```

Para escribir texto en un gráfico se hace uso de **<text atributos> </text>**. En este caso únicamente se están indicando las coordenadas de la esquina inferior izquierda de la caja que contiene el texto.

Si dentro de un elemento texto se desea hacer una modificación de formato, posición, ... se puede hacer uso de **<tspan atributos> </tspan>**. En el ejemplo se da una nueva coordenada **x** absoluta (**x="1cm"**) y otra relativa con respecto a la dada al elemento **text** (**dy="1.5cm"**); en este caso, esto significa que la nueva coordenada **y** del texto será 3cm+1.5cm=4.5cm. Se hace una nueva definición del tamaño de la fuente (**font-size="36"**) y por medio del atributo **style** se indica el color de relleno y del borde (**style="fill:orange;stroke:green"**) del nuevo texto a escribir.

```
<text x="0.5cm" y="9.5cm" font-size="46" stroke-width=".05cm" style="fill:yellow;stroke:brown"
  rotate="40 0 70">
  B M
</text>
```

Este es otro ejemplo de texto en un dibujo SVG. En este caso la única novedad es el atributo **rotate** que permite rotar carácter a carácter el texto a renderizar en la pantalla. En este caso hay tres caracteres: la **B**, un espacio en blanco y la **M**, a los que se ha asignado las rotaciones de 40, 0 y 70 grados respectivamente. Si se desea rotar toda la cadena de texto debes reemplazar **rotate="40 0 70"** por **transform="rotate(ángulo_a_rotar,cx,cy)"** que rotará el texto entorno al centro definido por las coordenadas **cx** y **cy** el ángulo en grados indicado. Si no se indica el centro se toma por defecto el punto 0,0.

```
<!— Este es un comentario que podría explicar el código —>
```

Como en cualquier documento XML <!— —> sirve para realizar comentarios.

```
<path id="curva" fill="none" stroke="none" d="M 100,483 C 100,483 145,379 210,375 C
  274,372 281,473 356,470 C 431,466 446,373 508,360 C 570,347 602,418 602,418" />
```

Por medio de **path** es posible crear una infinidad de figuras debido a su enorme flexibilidad. en el ejemplo se define en primer lugar un identificador del elemento a dibujar: **id="curva"**. A continuación se indica que la figura a crear no debe tener relleno y tampoco borde, lo que significa que será invisible. Por último el atributo **d** (data) almacena una secuencia de puntos y parámetros que definirán al dibujo por completo. En esta secuencia se utilizan letras (**M, L, C, Z,** ...) para indicar diferentes operaciones en coordenadas absolutas, y las mismas letras (**m, l, c, z,** ...) para indicar operaciones similares en coordenadas relativas. La siguiente tabla muestra un pequeño resumen.

Comando	Significado	Parámetros	Descripción
M, m	Mover a	x,y	Comienza una nueva subpath en la coordenada indicada. Si la secuencia comienza con **m** las coordenadas indicadas se supondrán absolutas. Detrás de **M** o **m** pueden ir varios puntos, eneste caso se supone una secuencia implícita de **línea a** (line to).
Z, z	Cerrar path	ninguno	Cierra la subpath actual uniendo el último punto indicado con el de comienzo.
L, l	Línea a	x,y	Traza una línea desde el punto actual al dado. Si se proporciona una secuencia de puntos, estos se unirán para construir una poli-línea.
H, h	Línea horizontal a	x	Traza una línea desde el punto actual (xac,yac) hasta el nuevo punto (x,yac) si se utiliza **H**, y hasta el punto (xac+x,yac) si se utiliza **h**.
V, v	Línea vertical a	y	Traza una línea desde el punto actual (xac,yac) hasta el nuevo punto (xac,y) si se utiliza **V**, y hasta el punto (xac,yac+y) si se utiliza **v**.
C, c	Curva a	x1,y1 x2,y2 x,y	Dibuja una curva de Bézier cúbica desde el punto actual al punto (x,y) utilizando como punto de control (x1,y1) al comienzo de la curva y (x2,y2) como punto de control al final de la misma.
S, s	Curva suave a	x2,y2 x,y	Dibuja una curva de Bézier cúbica desde el punto actual al punto (x,y). El primer punto de control se toma la reflexión del segundo punto de control del comando previo al actual punto (si no hay comando previo o este no fuera **C**, **c**, **S** o **s** se asume que el primer punto de control es el actual punto. (x2,y2) es el segundo punto de control.
Q, q	Bézier cuadrático a	x1,y1 x,y	Dibuja una curva de Bézier cúbica desde el punto actual al punto (x,y) usando (x1,y1) como punto de control.
T, t	Bézier cuadrático suave a	x,y	Dibuja una curva de Bézier cúbica desde el punto actual al punto (x,y). Se toma como punto de control la reflexión del punto de control del anterior comando. usando (x1,y1) como punto de control. Si el comando anterior no es **Q**, **q**, **T** o **t** se asume que el punto de control es el actual punto.
A, a	Arco elíptico	rx.ry k p j x,y	Dibuja un arco elíptico del punto actual al punto (x, y). Para ayuda acudir al punto 8.3.8 de la recomendación W3C para SVG.

La figura 5.7 muestra algunos ejemplos de utilización de los anteriores comandos.

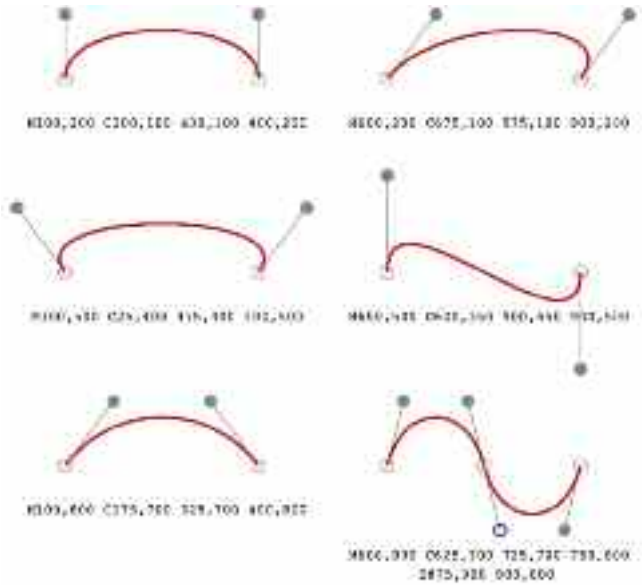


Figura 5.7. Ejemplos de uso de algunos comandos

```
<text style="font-size:10">
    <textPath xlink:href="#curva">
        Este es un texto que va siguiendo la curva desde el principio hasta el final. Subiendo
        y bajando.
    </textPath>
</text>
```

El **path** creado anteriormente tenía como finalidad marcar un camino o ruta para una cadena de texto. Este conjunto de líneas define un texto al que se ha asignado como único atributo el tamaño de la fuente a través de **style**. A continuación <textPath identificador> </textPath> permite relacionar un texto con un **path** indicado a través de un identificador. El identificador es un link o enlace definido a través de uno de los atributos **xlinl**, en este caso **href** que posibilita referenciar elementos identificados en otra parte del documento a través de **id**. La sintaxis es **xlink: href="#nombre_objeto"**. De esta forma se pueden conseguir efectos como el mostrado en el ejemplo.

```
<g style="fill-opacity:0.5">
    <rect x="14cm" y="1cm" rx="0.52cm" ry="0.52cm" width="3.5cm" height="6cm"
        fill="green"/>
    <circle cx="11cm" cy="12cm" r="2.2cm" fill="blue" stroke="none"/>
    <rect x="9cm" y="12.5cm" rx="0.2cm" ry="0.2cm" width="7cm" height="3cm" fill="violet"/>
</g>
```

Este ejemplo es para mostrar de nuevo el uso de grupos <g atributos> </g> utilizando otros atributos como por ejemplo en este caso **fill-opacity**, que es asignado a todos y cada uno de los elementos incluidos en el grupo.

```
<rect x=".01cm" y=".01cm" width="18cm" height="16cm"  fill="none" stroke="blue" stroke-
        width=".2cm" />
</svg>
```

Estas dos últimas líneas se encargan de dibujar un marco para todo el dibujo por medio de un rectángulo sin rellenar, con un borde azul de grosor 0.2 centímetros y finalizar el dibujo (**</svg>**).

7. UTILIZACIÓN DE INKSCAPE COMO AYUDA DE DIBUJO

Una de las ventajas de SVG es que puede describirse un programa (un dibujo) a través de un simple editor de texto, pero entonces aparece uno de los grandes problemas de los paradigmas de programación no basados en WYSIWYG (what you see is what you get, lo que ves es lo que tienes): No tienes una realimentación en tiempo real a través de la pantalla de lo que estás programando. En trabajos técnicos las aplicaciones no WYSIWYG tienen cabida y hasta superan con creces a las que lo son, como por ejemplo ocurre con Tex (o LaTex) al comparlo con un procesador como OpenOffice Writer o Microsoft Word. Sin embargo cuando se trata de trabajos artísticos es muy conveniente tener una realimentación directa de lo que se está creando.

Lamentablemente no existe, al menos de momento, una aplicación capaz de construir dibujos SVG's y que soporte todas sus características y posibilidades. A pesar de esto, existen aplicaciones que permiten la creación de SVG's como por ejemplo InkScape, disponible para Windows y GNU/Linux, que ofrece una gran cantidad de posibilidades. Una vez que el fichero creado con estas aplicaciones está guardado, es posible abrirlo con un editor de textos y modificarlo a nuestro antojo.

En esta sección se va a mostrar la edición y modificación de un dibujo SVG creado con InkScape. Es un ejemplo sencillo que nos permitirá ir descubriendo las características de este formato.

El uso de InkScape es bastante intuitivo y no es objeto de este libro explicar su funcionamiento. Por esta razón se opta por la creación de un sencillo logotipo como el mostrado en la figura logo.eps, compuesto por dos cadenas de texto: **"Cossio"** y **".net"**, así como por dos conjuntos de curvas que lo delimitan. Crearemos una de estas familias de curvas y la otra bastará con copiarla y aplicarle una transformación en espejo.

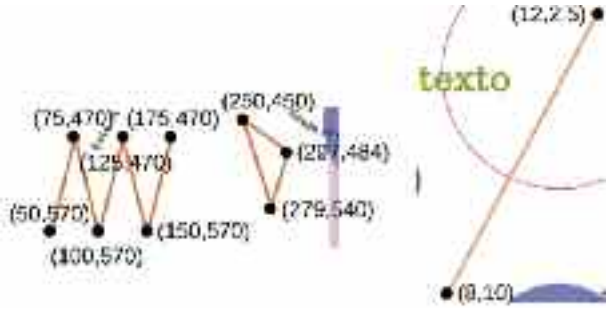


Figura 5.8. Logotipo sobre el que se va a aplicar animación

Tras la creación del dibujo deberá ser guardado con el formato **svg plano** tal y como se muestra en la figura 5.9.

Figura 5.9. Tras la creación del logotipo con InkScape se guarda como SVG plano

Tras esta operación ya es posible abrirlo (logo_ini.svg) a través de un editor de textos, obteniéndose algo similar a:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!— Created with Inkscape (http://www.inkscape.org/) —>
<svg
   xmlns:svg="http://www.w3.org/2000/svg"
   xmlns="http://www.w3.org/2000/svg"
   version="1.0"
   width="786.20135"
   height="93.85067"
   id="svg2232">
  <defs
     id="defs2249" />
  <text
     x="122.32668"
     y="91.934593"
     style="font-size:109.40000153px;line-height:125%;fill:#3e3e3d;font-family:Segoe Print"
     id="text2234">
    <tspan
       id="tspan2236"><tspan
   style="fill:#aa0000"
   id="tspan2238">cossio</tspan>
.net</tspan>
  </text>
  <g
     transform="matrix(0.967309,0,0,0.967309,-14.853496,-614.13157)"
     style="stroke:#008000;stroke-width:6.96453476;stroke-miterlimit:4"
     id="g3356">
  <path
        d="M 62.306783,726.85414 C 62.306783,726.85414 62.758076,718.5503
           60.840461,710.87982 C 58.922845,703.20936 54.636321,696.17228
           54.636321,696.17228 L 93.041063,675.2678 C 89.265026,711.51322
           139.01146,703.84276 139.01146,703.84276"
        style="fill:none;fill-opacity:0.75;fill-rule:evenodd;stroke:#008000;stroke-
           width:6.96453476;stroke-linecap:butt;stroke-linejoin:miter;stroke-miterli
           mit:4;stroke-dasharray:none;stroke-opacity:1"
      id="path3358" />
  <path
        d="M 39.488314,699.87938 C 19.033748,693.31858 112.34476,656.55453
           112.34476,656.55453"
        style="fill:none;fill-opacity:0.75;fill-rule:evenodd;stroke:#008000;stroke-
           width:6.96453476;stroke-linecap:butt;stroke-linejoin:miter;stroke-miterli
           mit:4;stroke-dasharray:none;stroke-opacity:1"
        id="path3361" />
  <path
        d="M 25.412807,703.84276 C -11.506994,683.38817 117.45842,639.92219
           117.45842,639.92219"
        style="fill:none;fill-opacity:0.75;fill-rule:evenodd;stroke:#008000;stroke-
           width:6.96453476;stroke-linecap:butt;stroke-linejoin:miter;stroke-miterli
           mit:4;stroke-dasharray:none;stroke-opacity:1"
        id="path3366" />
  </g>
  <g
  transform="matrix(-0.967309,0,0,0.967309,800.91965,-612.30952)"
        style="fill:none;stroke:#008000;stroke-width:6.96453476;stroke-miterlimit:4;stroke-
           dasharray:none;stroke-opacity:1;display:inline"
```

```
43              id="g3368">
44      <path
45              d="M 62.306783,726.85414 C 62.306783,726.85414 62.758076,718.5503
                        60.840461,710.87982 C 58.922845,703.20936 54.636321,696.17228
                        54.636321,696.17228 L 93.041063,675.2678 C 89.265026,711.51322
                        139.01146,703.84276 139.01146,703.84276"
46              style="fill:none;fill-opacity:0.75;fill-rule:evenodd;stroke:#008000;stroke-
                        width:6.96453476;stroke-linecap:butt;stroke-linejoin:miter;stroke-miterli
                        mit:4;stroke-dasharray:none;stroke-opacity:1"
47              id="path3370" />
48      <path
49              d="M 39.488314,699.87938 C 19.033748,693.31858 112.34476,656.55453
                        112.34476,656.55453"
50              style="fill:none;fill-opacity:0.75;fill-rule:evenodd;stroke:#008000;stroke-
                        width:6.96453476;stroke-linecap:butt;stroke-linejoin:miter;stroke-miterli
                        mit:4;stroke-dasharray:none;stroke-opacity:1"
51              id="path3372" />
52      <path
53              d="M 25.412807,703.84276 C -11.506994,683.38817 117.45842,639.92219
                        117.45842,639.92219"
54              style="fill:none;fill-opacity:0.75;fill-rule:evenodd;stroke:#008000;stroke-
                        width:6.96453476;stroke-linecap:butt;stroke-linejoin:miter;stroke-miterli
                        mit:4;stroke-dasharray:none;stroke-opacity:1"
55              id="path3374" />
56      </g>
57    </svg>
```

Listado 7. Contenido SVG del logotipo creado

De la observación del fichero se desprende la existencia en el mismo de mucha información innecesaria y otra que es redundante. Observad que se definen dos espacios de nombres[28] (namespaces). Sin embargo el espacio de nombres utilizando el prefijo **svg** (**xmlns:svg="http://www.w3.org/2000/svg"**) no se utiliza en ningún momento, por tanto puede ser eliminado.

Cada elemento tiene asociado un identificador: **id="nombre"**. Estos identificadores sirven para referenciar los elementos a lo largo del documento, pero si no se utilizan no son necesarios. En este documento no se hace ninguna referencia, luego pueden ser eliminados.

Los elementos de dibujo definidos dentro del elemento **<def> </def>** no son representados cuando el parser SVG lee el contenido del fichero. Se utiliza para identificar elementos que posteriormente pueden ser referenciados para una posterior utilización. En este caso no tiene ninguna utilidad y por tanto puede ser eliminado.

Como se desea que los textos **"Cossio"** y **".net"** se comporten de forma diferente será conveniente separarlos en dos elementos de texto diferentes.

Las curvas se han definido por medio del elemento **path**, al que se han asociado muchos atributos que toman los valores por defecto, así pues, pueden ser eliminados.

Por último observar que las unidades de los elementos están dadas en pixels. InkScape ha dado mucha precisión al dibujo al proporcionar tantos decimales a las diferentes expresiones. En la mayor parte de los casos los decimales podrían eliminarse produciendo una modificación del dibujo despreciable. Tras realizar todas estas modificaciones el programa, que será llamado **logo_ini2.svg**, quedaría de la siguiente forma:

---

28 Recordad que un espacio de nombres se define a través de la palabra reservada **xmlns**.

```
1   <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2   <svg
3     xmlns="http://www.w3.org/2000/svg"
4     width="786"
5     height="94">
6
7     <text x="122" y="91" style="font-size:109px;line-height:125%;fill:#aa0000;font-family:Segoe
            Print">
8       cossio
9     </text>
10    <text x="460" y="91" style="font-size:109px;line-height:125%;fill:#3e3e3d;font-family:Segoe
            Print">
11      .net
12    </text>
13    <g
14      transform="matrix(0.967309,0,0,0.967309,-14.853496,-614.13157)"
15      style="fill:none;stroke:#008000;stroke-width:7;stroke-miterlimit:4">
16      <path d="M 62,726 C 62,726 62,718 60,710 C 58,703 54,696 54,696 L 93,675 C 89,711
            139,703 139,703"/>
17      <path d="M 39,699 C 19,693 112,656 112,656"/>
18      <path d="M 25,703 C -11,683 117,639 117,639"/>
19    </g>
20    <g
21      transform="matrix(-0.967309,0,0,0.967309,800.91965,-612.30952)"
22      style="fill:none;stroke:#008000;stroke-width:7;stroke-miterlimit:4">
23      <path d="M 62,726 C 62,726 62,718 60,710 C 58,703 54,696 54,696 L 93,675 C 89,711
            139,703 139,703"/>
24      <path d="M 39,699 C 19,693 112,656 112,656"/>
25      <path d="M 25,703 C -11,683 117,639 117,639"/>
26    </g>
27  </svg>
```

Listado 8. Archivo tras las modificaciones propuestas

Observa que el fichero se ha hecho mucho más legible al eliminar el texto redundante. Puedes comprobar que al abrirlo con InkScape el resultado es el mismo. Se ha tenido que hacer un cambio utilizando el método de prueba y error: Al separar los textos **Cossio** y **.net** ha sido necesario cambiar la coordenada **x** para situar el texto **.net** a un valor 460.

La creación de las curvas que delimitan el texto del logotipo no se hizo dos veces. Se crearon las de la izquierda y después mediante la herramienta de simetría se construyó la de la derecha. Por esta razón las definiciones de los elementos **path** son idénticas. Lo que cambia es la matriz de transformación del grupo **<g></g>** que hace que aparezcan en posiciones diferentes. Por esta razón hubiera sido posible definir en **<defs></defs>** el conjunto de curvas y después reutilizarlas. El programa, que se llamará **logo_ini3.svg**, quedaría:

```
1   <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2   <svg
3     xmlns="http://www.w3.org/2000/svg"
4     xmlns:xlink="http://www.w3.org/1999/xlink"
5     width="786"
6     height="94">
7
8     <defs>
```

```
9       <g id="curvas"
10        style="fill:none;stroke:#008000;stroke-width:7;stroke-miterlimit:4">
11        <path d="M 62,726 C 62,726 62,718 60,710 C 58,703 54,696 54,696 L 93,675 C 89,711
              139,703 139,703"/>
12        <path d="M 39,699 C 19,693 112,656 112,656"/>
13        <path d="M 25,703 C -11,683 117,639 117,639"/>
14      </g>
15    </defs>
16
17    <text x="122" y="91" style="font-size:109px;line-height:125%;fill:#aa0000;font-family:Segoe
              Print">
18      cossio
19    </text>
20    <text x="460" y="91" style="font-size:109px;line-height:125%;fill:#3e3e3d;font-family:Segoe
              Print">
21      .net
22    </text>
23
24    <use transform="matrix(-0.967309,0,0,0.967309,800.91965,-612.30952)" xlink:href=
              "#curvas"/>
25    <use transform="matrix(0.967309,0,0,0.967309,-14.853496,-614.13157)" xlink:href=
              "#curvas"/>
26  </svg>
```

Listado 9. Reutilización de código y uso de predefiniciones

En **<defs></defs>** se ha definido un grupo de curvas que no son representadas hasta que no son llamadas desde otra parte del fichero; por esta razón se les ha asignado el identificador **id="curvas"** que permitirá su referenciación. Por otro lado, para referenciar elementos dentro de un archivo SVG es necesario utilizar el atributo **href** dentro de un espacio de nombres XLink, tal como se estipula en **http://www.w3.org/TR/xlink/**. Para ello hay que definir el nombre de espacio: **xmlns:xlink="http://www.w3.org/1999/xlink"**. En el resto del documento la referenciación de un elemento mediante su identificador se consigue utilizando la estructura: **xlink:href= "#identificador"** y el elemento **<use></use>**. En el ejemplo el uso de este elemento:

```
<use transform="matrix(-0.967309,0,0,0.967309,800.91965,-612.30952)" xlink:href="#cur-
vas"/>
```

Utiliza el elemento identificado por **id="curvas"** aplicando una transformación definida a través de la matriz **"matrix(-0.967309,0,0,0.967309,800.91965,-612.30952)"**.

Al aplicar una matriz de transformación diferente a cada curva se obtienen representaciones diferentes. Una matriz de transformación cambia un sistema de coordenadas en otro a través de la ecuación:

$$\begin{pmatrix} x_{previa} \\ y_{previa} \\ 1 \end{pmatrix} = \begin{pmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_{nueva} \\ y_{nueva} \\ 1 \end{pmatrix}$$

Como la matriz de transformación sólo necesita seis valores se puede expresar como un vector (a, b, c, d, e, f), así en el archivo tomaría la forma **"matrix(a,b,c,d,e,f)"** escrita por ejemplo como: **"matrix(-0.967309,0,0,0.967309,800.91965,-612.30952)"**.

8. ANIMACIÓN DE UN DIBUJO SVG

Una vez que se conocen ligeramente algunos de los elementos SVG ya se pueden introducir otras posibilidades del formato, como por ejemplo la animación. En esta sección se va a mostrar un sencillo ejemplo que muestra la creación de un logotipo animado basada en el que fue creado en la sección anterior.

La animación consistirá en que las curvas que delimitan el texto se moverán desde una posición a otra siguiendo una línea recta, cambiando además de color durante el recorrido. El texto **Cossio** se desplazará de izquierda a derecha y el texto **.net** se moverá recorriendo un sector de circunferencia. La figura 5.10 muestra algunas secuencias del movimiento.

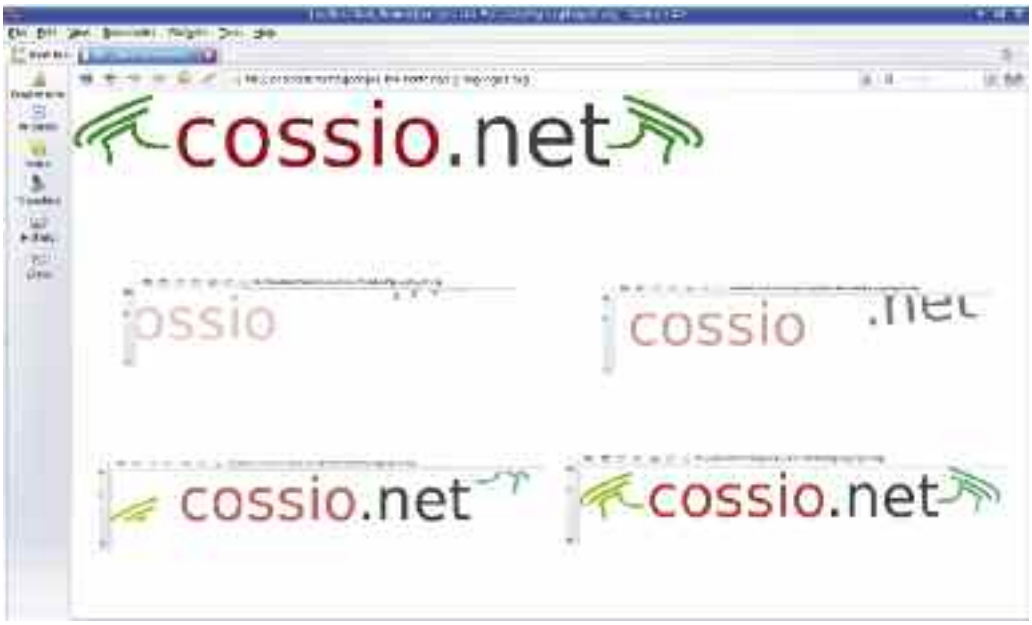


Figura 5.10. Visión final en el navegador y la evolución del movimiento

La animación del texto **Cossio** se realizaría añadiendo las siguientes líneas a su elemento texto asociado:

```
<text x="122" y="91" style="font-size:109px;line-height:125%;fill:#aa0000;font-family:Segoe
        Print;opacity:0.5">
  cossio
  <animate attributeName="opacity" from="0" to="1" begin="0s" dur="9s" fill="freeze" />
        <animateTransform attributeName="transform" type="translate" from="-350" to="0"
        begin="0s" dur="5s"/>
</text>
```

Se han añadido dos nuevos: un elemento **<animate *atributos* />** y otro **<animate Transform *atributos* />**. Cada uno de estos elementos necesita en primer lugar designar el atributo sobre el que se aplica la animación por ejemplo:

```
<animate attributeName="opacity" from="0" to="1" begin="0s" dur="9s" fill="freeze" />
```

En este caso la animación se aplica sobre la transparencia u opacidad (**opacity**) del texto que variará desde un valor 0 (completamente transparente) hasta un valor 1 (completamente opaco). El proceso de animación comenzará en el instante de cargar la imagen (a los 0 segundos) y el pro-

ceso de cambio durará 9 segundos. Finalmente el atributo **fill** define como quedará el dibujo al finalizar la animación y puede tomar dos valores **“freeze”** y **“remove”**. Si toma el valor **“freeze”** el atributo, objeto de animación, se quedará con el último valor que tomó durante el proceso de animación, en este caso **opacity=”1”**. Si toma el valor **“remove”** tras la animación el atributo objeto de la misma tomará el valor definido en el elemento correspondiente. En este ejemplo el elemento **<texto></texto>** asigna al atributo **opacity** el valor 0.5, por tanto si **fill** toma el valor **“remove”**, al finalizar la animación el texto adquirirá una opacidad de ese valor.

El valor por defecto que toma el atributo **opacity** es 1, por tanto si no utilizamos este atributo dentro del elemento **<texto></texto>** es posible prescindir de usar el atributo **fill**. Esto es así debido a que el valor del atributo al final de la animación y el valor del mismo en el elemento **<texto></texto>** coinciden. Por tanto el siguiente código hace la misma labor:

```
<text x=”122” y=”91” style=”font-size:109px;line-height:125%;fill:#aa0000;font-family:Segoe
        Print”>
  cossio
  <animate attributeName=”opacity” from=”0” to=”1” begin=”0s” dur=”9s”/>
  <animateTransform attributeName=”transform” type=”translate” from=”-350” to=”0”
        begin=”0s” dur=”5s”/>
</text>
```

El elemento **<animateTransform *atributos* />** se utiliza para construir animaciones de traslación (**type=”translate”**), escalado (**type=”scale”**), rotación (**type=”rotate”**) y de deformación oblicuas (**type=”skewX | skewY”**). En este caso se muestra la de traslación. El texto se desplazará desde la posición **x=-350** hasta la posición **x=0**, coordenadas relativas a las dadas en el elemento **<texto></texto>**. A **from** (posición desde la que empieza a moverse el texto) se le pueden asignar coordenadas **x** e **y** separadas por un espacio: **from=”-350 0”**. Lo mismo ocurre con **to** (posición final del texto). Si no se especifica la coordenada **y** se sobreentiende que vale 0. Obsérvese que no es necesario utilizar el atributo **fill** con el valor **freeze** porque el texto acaba en la posición relativa (0,0). Si acabara en otra posición sería necesario su uso. Prueba a cambiar el valor de **to** a “40 50” para entender este efecto.

Los atributos **begin** y **dur** indican los instante en los que comienza y termina la animación.

La animación del texto **.net** se podría realizar añadiendo las siguientes líneas a su correspondiente elemento **<texto></texto>**:

```
<text x=”460” y=”91” style=”font-size:109px;line-height:125%;fill:#3e3e3d;font-family:Segoe
        Print”>
  .net
  <animate attributeName=”opacity”  from=”0” to=”1” begin=”0s” dur=”5s”/>
  <animateTransform attributeName=”transform” type=”rotate” from=”-15” to=”0” begin=”0s”
        dur=”5s”/>
</text>
```

Estas líneas son similares a las utilizadas con el texto **Cossio**. En este caso se utiliza el tipo **“rotate”** para el elemento **<animateTransform *atributos* />**, que girará el texto en un arco de circunferencia con centro el origen del texto (esquina inferior izquierda) desde -15 grados hasta 0 grados. La animación comienza en el instante de cargar el dibujo y dura 5 segundos. Obsérvese, que de nuevo no es necesario utilizar el valor **freeze** para el atributo **fill**.

La animación que se dará a las curvas que delimitan el texto se muestra a continuación. En este caso es necesario utilizar un grupo para que las animaciones únicamente afecten a los elementos de ese grupo:

```
<g>
  <use transform="matrix(-0.967309,0,0,0.967309,800.91965,-612.30952)" xlink:href=
        "#curvas" />
  <animateTransform attributeName="transform"  type="translate" from="75 -250" to="0 0"
        begin="0s" dur="7s"/>
 <animateColor attributeName="stroke"  from="rgb(0,255,255)" to="rgb(0,128,0)" begin="3s"
        dur="8s"/>
</g>
```

La línea correspondiente al elemento **<animateTransform *atributos* />** no merece comentario. A continuación se define otro elemento de animación: **<animateColor *atributos* />** utilizado para modificar colores. El cambio de color es aplicado al atributo **stroke** (las propias líneas) y evoluciona de un color **rgb(0,255,255)**, cían, hasta el verde final **rgb(0,128,0)**.

En el otro grupo de curvas, las líneas de código son similares. El código del archivo completo, llamado logo_ini4.svg, tendría la siguiente forma:

```
1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2  <svg
3    xmlns="http://www.w3.org/2000/svg"
4    xmlns:xlink="http://www.w3.org/1999/xlink"
5    width="786"
6    height="94">
7
8    <defs>
9      <g id="curvas"
10       style="fill:none;stroke:#008000;stroke-width:7;stroke-miterlimit:4">
11       <path d="M 62,726 C 62,726 62,718 60,710 C 58,703 54,696 54,696 L 93,675 C 89,711
             139,703 139,703"/>
12       <path d="M 39,699 C 19,693 112,656 112,656"/>
13       <path d="M 25,703 C -11,683 117,639 117,639"/>
14     </g>
15   </defs>
16
17   <text x="122" y="91" style="font-size:109px;line-height:125%;fill:#aa0000;font-family:Segoe
             Print">
18     cossio
19     <animate attributeName="opacity" from="0" to="1" begin="0s" dur="9s"/>
20     <animateTransform attributeName="transform" type="translate" from="-350" to="0"
             begin="0s" dur="5s"/>
21   </text>
22
23   <text x="460" y="91" style="font-size:109px;line-height:125%;fill:#3e3e3d;font-family:Segoe
             Print">
24     .net
25     <animate attributeName="opacity"  from="0" to="1" begin="0s" dur="5s"/>
26     <animateTransform attributeName="transform" type="rotate" from="-15" to="0" begin="0s"
             dur="5s"/>
27   </text>
28
29   <g>
30     <use transform="matrix(-0.967309,0,0,0.967309,800.91965,-612.30952)" xlink:href=
             "#curvas" />
31     <animateTransform attributeName="transform"  type="translate" from="75 -250" to="0 0"
             begin="0s" dur="7s"/>
```

```
32      <animateColor attributeName="stroke"  from="rgb(0,255,255)" to="rgb(0,128,0)" begin="3s"
              dur="8s"/>
33    </g>
34
35    <g>
36      <use transform="matrix(0.967309,0,0,0.967309,-14.853496,-614.13157)" xlink:href=
              "#curvas" />
37      <animateTransform attributeName="transform" type="translate" from="-75 250" to="0"
              begin="0s" dur="7s"/>
38      <animateColor attributeName="stroke" from="rgb(255,255,0)" to="rgb(0,128,0)" begin="3s"
              dur="8s"/>
39    </g>
40  </svg>
```

Listado 10. Archivo final tras todas las modificaciones propuestas

Por último te puedes plantear como ejercicio analizar las diferentes aplicaciones de dibujo que conozcas, para hacer una clasificación de aquellas que permiten guardar en formato SVG. También puedes buscar programas que permiten la conversión entre formatos, de entre ellos los autores consideran interesante el **pdf2svg' (http://www.cityinthesky.co.uk/pdf2svg.html)** que convierte los .pdf en .svg de una forma muy eficiente.

9. RESPONDIENDO A EVENTOS

El formato SVG está pensado para responder a eventos desde el momento en que SVG toma a SMIL como referencia para crear animaciones.

La respuesta a eventos necesita de la programación a través del lenguaje EMACScript o similar y que se describe completamente en el capítulo 18 de la especificación SVG dada por la W3C (**http://www.w3.org/TR/SVG11/script.html**). Para no hacer excesivo el contenido de este capítulo en este apartado se muestra un sencillo ejemplo en el listado 11.

```
1   <?xml version="1.0" standalone="no"?>
2   <svg width="6cm" height="5cm"
3       xmlns="http://www.w3.org/2000/svg" version="1.1">
4       <!— ECMAScript para cambiar radio del círculo con cada click —>
5       <script type="text/ecmascript">
6           <![CDATA[
7               function circle_click(evt) {
8                   var circle = evt.target;
9                   var radioactual = circle.getAttribute("r");
10                  if (radioactual == 40)
11                      circle.setAttribute("r", radioactual*2);
12                  else
13                      circle.setAttribute("r", radioactual*0.5);
14              }
15          ]]>
16      </script>
17
18      <circle onclick="circle_click(evt)" cx="100" cy="100" r="40" fill="red"/>
19  </svg>
```

Listado 11. Respuesta a los click del ratón

El ejemplo en sí mismo es suficientemente explicativo para comprender como funciona el proceso de respuesta a eventos utilizando el atributo `onclick' que puede ser asociado a una gran cantidad de elementos de dibujo SVG. En **http://www.w3.org/TR/SVG11/interact.html#SVGEvents** se encuentra la lista de todos los posibles eventos soportados en SVG.

El enlazado a otras páginas web a través de clicks de ratón no necesita de scripts sino que está soportado por SVG a través del elemento **a** (similar a como se define en HTML). El siguiente ejemplo muestra un enlace a páginas web haciendo click sobre elementos de dibujo.

```
1   <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2   <svg width="250" height="450" version="1.1"
3       xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
4
5     <a xlink:href="http://www.cossio.net">
6       <rect x="20" y="30" width="180" height="50" rx="10" ry="10" style="fill:pink;stroke:gray" />
7       <text x="40" y="65" font-size="24" style="fill:white;stroke:green">IES Cossío</text>
8     </a>
9     <a xlink:href="http://www.iesrioarba.es">
10      <rect x="20" y="130" width="180" height="50" rx="10" ry="10" style="fill:pink;stroke:gray"
            />
11      <text x="40" y="165" font-size="24" style="fill:white;stroke:green">IES Río Arba</text>
12    </a>
13  </svg>
```

Listado 12. Enlaces entre contenidos

El código es muy sencillo y legible y apenas requiere explicación. Observa la utilización[29] de **encoding="UTF-8"** para habilitar el uso de las tildes en el texto escrito en los rectángulos redondeados.

El último ejemplo muestra el uso del elemento <set></set>. Este elemento permite cambiar atributos de un objeto durante una cierta cantidad de tiempo o cuando sucede algún evento.

```
1   <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2   <svg xmlns="http://www.w3.org/2000/svg"
3       xmlns:xlink="http://www.w3.org/1999/xlink"
4         width="200" height="300" >
5
6   <defs>
7     <path id="rutatexto" d="M19,197 C 86,187 157,201 157,201 z" />
8     <path id="fondotexto" d="M 151,163 C 151,163 78,161 13,183 L 19,197 C 86,187 157,201
          157,201 z" />
9   </defs>
10
11    <rect x="0" y="0" width="100%" height="100%" fill="green" opacity="0.3" />
12    <text x="30" y="50" font-size="24" fill="brown">
13      <tspan dx="0" dy="0">Pulsa sobre </tspan>
14      <tspan dx="-155" dy="30">alguno de los </tspan>
15      <tspan dx="-150" dy="30">siguientes </tspan>
16      <tspan dx="-115" dy="30">enlaces </tspan>
```

29 El valor de **encoding="UTF-8"** es el que tomaría por defecto, luego puede no indicarse. En cualquier caso, observa la codificación utilizada por tu editor para indicarla en el archivo si usas tildes.

```
17      </text>
18
19      <use y="25" xlink:href="#fondotexto" fill="lavender" fill-opacity="0.7">
20         <set begin="cossio.mouseover" attributeName="fill" to="darkblue" />
21         <set begin="cossio.mouseout" attributeName="fill" to="lavender" />
22      </use>
23      <use y="75" xlink:href="#fondotexto" fill="lavender" fill-opacity="0.7">
24         <set begin="rioarba.mouseover" attributeName="fill" to="darkblue" />
25         <set begin="rioarba.mouseout" attributeName="fill" to="lavender" />
26      </use>
27
28      <a  xlink:href="http://www.cossio.net">
29        <text id="cossio" transform="translate(0,23.5)" font-size="15" fill="blue">
30           <set begin="mouseover" attributeName="fill" to="lavender" />
31           <set begin="mouseout" attributeName="fill" to="blue" />
32           <textPath startOffset="5" xlink:href="#rutatexto">
33              I.E.S. M.B. Cossío
34           </textPath>
35        </text>
36      </a>
37
38      <a  xlink:href="http://www.iesrioarba.es">
39         <text id="rioarba" transform="translate(0,72.5)" font-size="15" fill="blue">
40            <set begin="mouseover" attributeName="fill" to="lavender" />
41            <set begin="mouseout" attributeName="fill" to="blue" />
42            <textPath startOffset="10" xlink:href="#rutatexto" >
43               I.E.S. Río Arba
44            </textPath>
45         </text>
46      </a>
47
48   </svg>
```

Listado 13. Enlaces y respuesta a eventos

Observa que el código es sencillo y además se repite dos veces: una para crear el dibujo de enlace a **http://www.cossio.net** y otra para el enlace a **http://www.iesrioarba.es**.

En el elemento **<defs></defs>** se definen dos rutas. La llamada **rutatexto** se utilizará para escribir el texto sobre ella de forma que aparezca curvado y la que es llamada **fondotexto** se utiliza para crear un fondo del texto que remarca los enlaces a las páginas web.

A continuación se define un rectángulo y un texto de la forma habitual y ya conocida. Posteriormente se definen elementos **<use></use>** para hacer uso de la ruta **fondotexto**. Centrémonos en el primero de ellos, ya que el segundo es similar: por medio del elemento **<set></set>** se indica que la propiedad **fill** del elemento referenciado dentro de **<use></use>** cambie al color **darkblue** cuando el ratón esté encima del elemento **cossio** (un texto definido posteriormente). De la misma forma por medio de otro elemento **<set></set>** se indica que cuando el ratón salga del elemento **cossio** el color cambie a **lavender**. Obsérvese que al no estar dentro del elemento **cossio** es necesario hacer referencia a él utilizando **cossio.mouseover** y **cossio.mouseout**.

Por último se hace uso de un elemento **<a>**.para crear un enlaces a páginas web. Dentro de él se define un texto que a su vez anida de nuevo elementos **<set></set>**. En este caso **mouseover** y **mouseout** no son precedidos por **cossio.** debido a que están dentro de dicho elemento y por defecto hacen referencia a él. Por supuesto podrían haberse utilizado **cossio.mouseover** y **cossio.mouseout** pero no es necesario.

Capítulo 6

SERVIDOR WEB APACHE

Tras habernos centrado en los primeros capítulos en el diseño de documentos Web directamente interpretables por un cliente Web, en esta segunda parte del libro fijaremos la atención en las funciones e importancia de un servidor Web, sin el cual no podría concebirse la actual situación de acceso a todo tipo de información a través de Internet.

Figura 6.1. Pasos necesarios para comunicar un cliente con un servidor Web

Una de las pegas más importantes que podríamos señalar que presenta el trabajo realizado hasta ahora, es que los sitios Web desarrollados solo pueden ser visualizados desde el propio equipo donde se encuentran almacenados, sin que puedan ser accesibles por otro equipo de la red en la que nos encontremos, a no ser que lo compartamos mediante algún protocolo de compartición de archivos (NetBios, Samba/NetBios, FTP, NFS, etc.).

Para comprender la necesidad de introducir un servidor Web es necesario tener presente los siguientes aspectos:

— El protocolo HTTP (*Hypertext Transfer Protocol, Protocolo de Transferencia de Hipertexto*) surge de la necesidad de compartir información compuesta por texto, imágenes y sonido. Anterior a él, otros protocolos como FTP (*Protocolo de Transferencia de Ficheros*) permitían compartir archivos entre un servidor y equipos clientes, siendo necesario abrir explícitamente el archivo transferido para conocer la información que contenía. En HTTP, una vez transferida la información, es interpretada por el cliente y visualizada de manera automática.

— La función más importante del servidor Web o HTTP es garantizar que las páginas que componen un sitio Web que haya sido previamente alojado en él, u otro dispositivo de almacenamiento accesible por este, sean alcanzables y visualizadas por el resto de equipos que se encuentran conectados en red haciendo uso del protocolo HTTP.

Figura 6.2. El protocolo HTTP permite compartir documentos Web entre equipos

— La actual Web 2.0 requiere que el servidor HTTP cuente con un sistema gestor de bases de datos, donde son almacenados y gestionados la gran cantidad elementos que componen los sitios Web a los que da servicio. Originalmente las páginas Web estaban compuestas por un número reducido de elementos que podrían organizarse en simples carpetas.

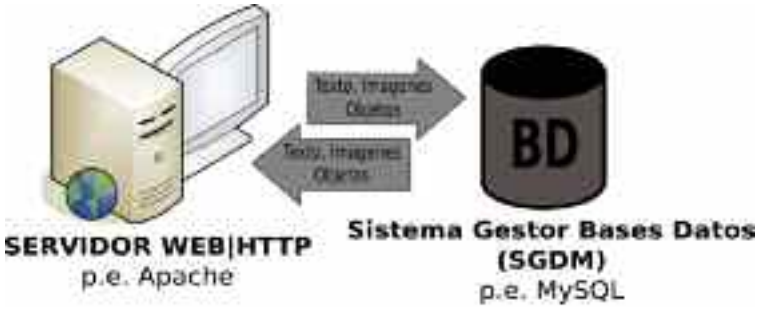


Figura 6.3. Los actuales servidores Web se apoyan en una base de datos para gestionar sus sitios Web

— El dinamismo en los contenidos que presentan las actuales páginas Web requieren de un servidor que este consultando constantemente bases de datos de las que extrae la información requerida por el usuario. Para ello, los servidores hacen uso de aplicaciones Web desarrolladas mediante la utilización de lenguajes como PHP o Java, siendo totalmente transparente al usuario final. Ejemplo de ello, son las tiendas virtuales, compra-venta de billetes de tren o avión online, buscadores de información como Google o el servicio de correo Webmail.

A lo largo del presente, y posteriores capítulos comprenderemos la importancia de un servidor Web y aprenderemos a configurar el software servidor más destacado en la actualidad, **Apache**.

1. CONTENIDOS DEL CAPÍTULO

El presente capítulo sigue la siguiente estructura:

— Presentación del software servidor que se utilizará a lo largo de los siguientes capítulos, la forma de obtenerlo de Internet e instalarlo.

— Características más destacables de Apache.

— Listado de las directivas más importantes del servidor Apache.

— Implementación de Hosts Virtuales basados en IP y nombre de dominio para dar servicio a múltiples sitios Web de manera simultánea.

— Estrategias para publicación de contenidos por parte de los usuarios vía FTP o desde el Home de los usuarios del sistema.

— Implementación de Sitios Web no anónimos con autenticación de tipo *basic* y *digest*.

2. SOFTWARE DE DESARROLLO

Para realizar los ejercicios prácticos que se presentan en el capítulo será necesario instalar el software **Apache** en nuestro equipo convirtiéndolo en un servidor Web. Para ello se muestran dos posibles alternativas:

1. Descargar el software servidor **Apache** desde su Web "*http://httpd.apache.org*" e instalarlo. En el caso de que el equipo de prácticas funcione bajo el sistema operativo GNU/Linux, no será necesaria su descarga, ya que este software esta disponible para su instalación en prácticamente todas las distribuciones GNU/Linux, o en alguno de los repositorios que tenga configurados[30].

   El gran inconveniente que presenta esta primera alternativa, en relación a la que se presenta a continuación, es que **Apache** tan sólo será el primero de los programas de una gran lista que serán necesarios instalar para poder realizar las prácticas que se proponen desde este capítulo hasta el final del libro. Un sistema gestor de bases de datos, un software servidor para la transferencia de archivos FTP, y un conjunto de módulos de **Apache** que le asignarán multitud de funcionalidades, tan sólo son un ejemplo del software que será necesario instalar posteriormente.

2. Descargar el software servidor **Xampp** desde su Web "*http://www.apachefriends.org/es/xampp.html*" e instalarlo. **Xampp** es un software libre de servidor que aglutina en un solo los servicios más importantes necesarios para poner en marcha un servidor Web 2.0:

   — **Microsoft Windows**: servidor Web Apache + PHP, servidor FTP FileZilla, gestor de bases de datos MySQL + phpMyAdmin.

   — **GNU/Linux**: servidor Web Apache + PHP, servidor FTP ProFTPd, gestor de bases de datos MySQL + phpMyAdmin.

Figura 6.4. Xampp es un software libre multiservicio que puede ser descargado desde su Web

Por su simplicidad y comodidad, para la realización práctica de todos los ejemplos de configuración del servicio HTTP que se presentan en el libro, se asumirá que se ha escogido esta segunda alternativa.

---

30 En esta práctica se supone que el usuario conoce como instalar software tanto en Microsoft Windows como en la distribución correspondiente de GNU/Linux.

Tras la descarga de Xampp, se llevará a cabo su instalación y postinstalación siguiendo los pasos indicados en su propia Web. Tal como allí se indica, para comprobar su funcionamiento, arrancaremos su servicio[31], y abriendo un cliente HTTP, realizaremos una solicitud de conexión poniendo en la barra de direcciones "*http://127.0.0.1*" o "*http://localhost*", siendo *localhost* un alias asociado a la dirección IP 127.0.0.1 de lazo interno definido en el fichero *hosts* del sistema, ubicado en "*c:\winnt\system32\drivers\etc\hosts*" en Windows NT/2000, en "*c:\windows \system32\drivers\etc\hosts*" en sistemas operativos Windows XP/2003/Vista, y en "*/etc/hosts*" en distribuciones GNU/Linux.

Figura 6.5. Tras arrancar el multiservicio Xampp podemos comprobar el correcto funcionamiento de Apache desde un cliente Web, "http://localhost"

El control de los servicios Xampp dependerá del sistema operativo utilizado:

— **Microsoft Windows**: Tras la instalación de Xampp[32], en el menú de *Inicio|Programas* hallaremos la opción *Apache Friends|Xampp|Xampp* Control Panel, desde donde podremos controlar todos los servicios que incluye Xampp.

Figura 6.6. Xampp Control Panel gestiona los servicios instalados

31 En el caso de que tengamos instalado otro software servidor que trabaje bajo los mismos puertos que Xampp, 21|FTP, 80|HTTP o 3306|MySQL, será necesario parar sus servicios para evitar conflictos. En caso contrario no funcionará.

32 Xampp ofrece dos versiones, una instalable, y una descomprimible. Aquí asumiremos ques se ha optado por la primera opción.

También es posible encontrar la lista de todos los servicios gestionados por Windows accediendo a el *Panel de Control*/*Herramientas Administrativas*/*Servicios* o pinchando con el botón derecho del ratón sobre *Mi PC*, eligiendo la opción *Administrar*, tendremos acceso igualmente a *Servicios*. Allí podemos encontrar los servicios instalados mediante Xampp: Apache (servicio HTTP), FileZilla (servicio FTP) y MySQL (servicio SGDB MySQL). Pinchando con el botón derecho sobre cualquiera de estos servicios nos permitirá iniciarlo, reiniciarlo o pararlo.

Figura 6.7. Ventana de gestión de servicios en Microsoft Windows

— **GNU/Linux**: La lista de los servicios gestionados por GNU/Linux se encuentran en el directorio "*/etc/init.d/*". Como puede observarse a través de sus permisos *"ls -l /etc/init.d/"*, se trata de un conjunto de scripts ejecutables que permiten controlar el estado del servicio:

[root@linux]# **/etc/init.d/***nombre-servicio* **status** *(informa del estado del servicio indicado)*

[root@linux]# **/etc/init.d/***nombre-servicio* **start|restart|stop** *(inicia, reinicia o para el servicio indicado)*

El modo en que se inician estos servicios al arrancar el equipo servidor lo podemos consultar ejecutando el comando "*chkconfig –list*". Este nos proporciona una lista con todos los servicios ofrecidos junto con el estado en que se encuentran al arrancar el equipo en función del modo de arranque programado[33]. Para modificar este comportamiento ejecutaremos el siguiente comando: "*chkconfig —level niveles nombre-servicio off|on*".

[root@linux]# **chkconfig —list** *nombre-servicio (información del estado del servicio indicado tras arrancar el equipo)*

[root@linux]# **chkconfig —level** *niveles nombre-servicio* **off|on** *(asigna el estado en que se encontrará el servicio indicado tras el próximo arranque del equipo en los modos de arranque indicados)*

[root@linux]# **chkconfig —level 35 network off** *(establece que el servicio de red se encuentre desabilitado en el siguiente arranque del servidor tras iniciarse en modo 3 ó 5)*

A diferencia de Microsoft Windows, podrá observarse como en GNU/Linux los servicios instalados mediante Xampp no figuran en la lista anterior, a no ser que se hayan instalado de manera independiente. Como puede observarse en la Web de Xampp, su instalación

33 El fichero */etc/inittab* establece el modo de arranque de GNU/Linux. Los modos más comunes son el 3 y el 5, ambos multiusuario con el sistema de red activo, diferenciándose en que el primero presenta una interfaz de usuario en modo comando, y el segundo en modo gráfico.

bajo GNU/Linux, se limita a una simple descompresión de un archivo en formato *tar.gz* en la ruta sugerida */opt*, "*tar xvfz xampp-linux.tar.gz -C /opt*", siendo transparente para el sistema[34]. Por ello, en el caso de seguir todos los pasos sugeridos por Xampp, el control de su servicio se establecerá invocando a un script llamado lampp ubicado en la carpeta creada en la descompresión anterior:

```
[root@linux]# /opt/lampp/lampp start|restart|stop (iniciar, reinicia o para todos los servicios Xampp)
[root@linux]# /opt/lampp/lampp startapache|stopapache|reloadapache (iniciar, para o recarga únicamente el servicio HTTP ofrecido por Apache)
```

Para conocer todas las opciones posibles de gestión de los servicios Xampp, puede ejecutarse el script anterior sin parámetros, "*/opt/lampp/lampp*".

Recordar que para que no haya conflictos entre Xampp y otros servicios que puedan estar instalados en el sistema y que hagan uso de los mismos puertos de comunicaciones, ***21|FTP***, ***80|HTTP*** o ***3306|MySQL***, será necesaria la desinstalación de este otro software, o su parada mediante la herramienta de administración de servicios de que disponga, tal como se ha mostrado anteriormente.

¡¡Sugerencia!! A la hora de decidir el sistema operativo bajo el que hacer funcionar el servidor HTTP Apache, Microsoft Windows o GNU/Linux, en la práctica real es aconsejable hacer uso del segundo, al ahorrarnos el coste de su licencia y presentar menos vulnerabilidades a los ataques informáticos, como características más destacables, entre otras muchas ventajas.

3. CARACTERÍSTICAS DEL SERVIDOR WEB APACHE

Apache es el software servidor HTTP más ampliamente utilizado en el mundo con una implementación superior al 50% del total de los servidores públicos. Tan sólo el software servidor ofrecido por Microsoft (**IIS**, *Internet Information Server*) se vislumbra como un posible competidor, aunque con muchas limitaciones funcionales respecto a **Apache**, debido a las vulnerabilidades que presenta y menores opciones de configuración[35].

Figura 6.8. Comparativa de uso de los principales software de servidor HTTP

34 La desinstalación del software Xampp en GNU/Linux no se realiza por la vía habitual, simplemente hay que borrar ese directorio, *"rm -rf /opt/lampp"*.

35 Un estudio y comparativa actualizados del uso de las distintas alternativas de software de servidor HTTP disponibles en el mercado puede encontrarse en *"http://news.netcraft.com/archives/web_server_survey.html"*.

Apache aparece públicamente en abril de 1995 (*v0.6.2*) como una alternativa al servidor del NCSA (**httpd**) cuyo desarrollo quedó definitivamente suspendido en 1998, aprovechándose prácticamente todo su código. Actualmente la ASF (*Apache Software Foundation, Fundación del Software Apache*) es la organización no lucrativa encargada de gestionar y desarrollar el proyecto apache. Entre sus características cabe destacar:

1. *Software libre*. Libertad para poder usar el software Apache para cualquier propósito, con código abierto disponible para poderlo estudiar y adaptar a las necesidades particulares de cada usuario, pudiendo hacer tantas copias uno quiera y distribuirlas sin ningún tipo de problemas de licencias.
2. *Multiplataforma*. Hay versiones disponibles para entornos Microsoft Windows, GNU/Linux, Macintosh y UNIX/Solaris.
3. *Facilidad de configuración*: Mediante un simple editor de textos, modificando el fichero de configuración **httpd.conf** podemos personalizar el comportamiento del servicio Web.
4. *Modularidad*: Dispone de multitud de módulos independientes formados por un conjunto de directivas de configuración que permiten asignar funcionalidades a Apache. Puede distinguirse entre módulos base compilados en tiempo de instalación, módulos que pueden ser cargados dinámicamente DSO (*Dynamic Shared Objects*) y módulos extra que pueden ser instalados a posteriori.
5. *Soporte para Webs dinámicas*: Dispone de módulos que le permiten ejecutar scripts CGI (*Perl, lenguaje C, ...*), e interpretar etiquetas SSI y lenguajes como PHP o Python embebidos en los documentos HTML/XHTML.
6. *Comunicaciones seguras*: A través del módulo SSL (*Secured Socket Layer, Seguridad de la Capa de Transporte*) haciendo uso de algoritmos criptográficos Apache nos permite establecer comunicaciones seguras (*confidencialidad y autentificación*) entre el cliente y el servidor HTTP.
7. *Servicio multiweb*: La implementación de Hosts Virtuales basados en dirección IP y en nombre de dominio nos ofrece la posibilidad de dar servicio a diferentes sitios Web simultáneamente.

4. CONFIGURACIÓN DE APACHE

Para configurar el servicio HTTP ofrecido por el software **Apache** tan sólo será necesario editar su fichero de configuración ***httpd.conf*** con un simple editor de textos y modificar o añadir directivas. Tras su instalación a través de **Xampp**[36] lo encontraremos en el siguiente directorio:

a) **Microsoft Windows**: Durante la instalación de la versión *installer* de Xampp su asistente gráfico nos permitirá indicar la ubicación de su directorio raíz (p.e. *"c:\xampp"*). En su interior encontraremos un directorio asociado al servicio Apache, "*apache*", el cual contiene el directorio de configuración "*conf*" donde se encuentra el fichero ***httpd.conf*** (por ejemplo: "*c:/xampp/apache/conf/httpd.conf*").

36 En la Web de Xampp *"http://www.apachefriends.org/es/xampp.html"* se nos informa de los detalles de su instalación y postinstalación tanto bajo Microsoft Windows como en GNU/Linux.

Figura 6.9. Ubicación del fichero de configuración de Apache en Windows

b) **GNU/Linux**: En el caso de haber seguido los pasos de instalación sugeridos en la Web de Xampp, "*tar -xvfz xampp-linux.tar.gz -C /opt*", podremos encontrar el fichero **httpd.conf** en "*/opt/lampp/etc/httpd.conf*".

Al abrir este fichero mediante nuestro editor de textos preferido, podremos observar que su contenido esta compuesto por comentarios de ayuda precedidos por el carácter almohadilla, "**#**", y diversas directivas de configuración, entre las cuales cabría destacar por su importancia las siguientes:

— **ServerRoot** *ruta-directorio-raíz*. Informa de la ubicación del directorio donde se localizan todos los archivos y subdirectorios asociados al servicio Apache:

a) Microsoft Windows: **ServerRoot "c:/xampp/apache"**

b) GNU/Linux: **ServerRoot "/opt/lampp"**

Las rutas relativas que se asignen al resto de directivas de configuración siempre son en relación a este directorio raíz.

— **Listen** [*direccion-IP:*]*puerto* [*protocolo*]. Establece bajo que dirección IP y puerto de comunicaciones Apache atenderá a las peticiones HTTP que lleguen al equipo servidor. En el caso de no indicar ninguna dirección IP, se entenderá que debe escuchar por el puerto indicado a través de cualquiera de las IP que tenga asignadas nuestro equipo. El protocolo soló se indicará de manera opcional, cuando el puerto indicado no sea un puerto well-know (*0:1023*).

```
Listen 80
Listen 192.168.1.1:8000 https
```

Teniendo en cuenta que un equipo puede tener asignadas múltiples direcciones IP sobre una misma interfaz de red o varias, esta directiva de configuración nos permite indicar si deseamos escuchar solicitudes de conexión bajo cualquiera de estas IP, o únicamente por alguna de ellas. Puede utilizarse tantas veces como sea necesario, pero teniendo en cuenta que no entren en conflicto entre sí. Por ejemplo, las siguientes directivas entrarían en conflicto, ya que no puede una directiva no puede restringir la escucha por un puerto a una determinada IP y otra indicar que bajo cualquiera:

```
Listen  12345
Listen 192.168.1.1:12345
```

— **ServerName** [protocolo://]*nombre-dominio*[:*puerto*]. Asigna un nombre de dominio cualificado (*FQDN, Fully Qualified Domain Name*) al servicio Web al que da servicio Apache. En el caso de que Apache se configure para dar servicio a múltiples sitios Web bajo una misma dirección IP, el criterio que utiliza para su diferenciación es hacer uso de un nombre de dominio diferente (*Hosts Virtuales basados en nombre*). Un ejemplo sería:

```
ServerName http://www.midominio.es:8080
```

Este nombre será utilizado desde el cliente Web en su barra de direcciones para hacer referencia al sitio Web que se desea visitar, "*http://www.midominio.es*". Para que este nombre de dominio pueda ser utilizado en Internet es necesario de que sea cualificado y valido, para lo cual deberá haberse registrado previamente el dominio "*midominio.es*" y haberse configurado un equipo en él con el identificador "*www*" asociandole la correspondiente dirección IP pública proporcionada por el proveedor de servicios de Internet al router que lo conecta a Internet[37].

A nivel local, para hacer pruebas del correcto funcionamiento de nuestro servidor HTTP, podremos hacer uso de nombres de dominio sin necesidad de registrar uno públicamente, ni de tener que hacer solicitudes de resolución de estos nombres a su correspondiente dirección IP a un servidor DNS (*Domain Name Server*). Esto es posible ya que todo equipo cliente, independientemente de que el sistema operativo sea Microsoft Windows o GNU/Linux, al solicitar una conexión HTTP a través de un nombre de dominio, primero intenta su resolución haciendo uso del fichero local *hosts*, y en el caso de que en él no encuentre su IP asociada, realizará una solicitud de resolución de nombres a el servidor DNS que tenga configurado. También es posible definir nombres alias del nombre de dominio especificado mediante la directiva "**ServerAlias** *nombre-equipo* [*otros-nombres-equipo*]".

Como puede observarse en la sintaxis de uso de la directiva **ServerName**, opcionalmente es posible indicar igualmente el tipo de protocolo utilizado y el puerto de escucha. Para ser congruentes, el puerto indicado deberá haberse especificado previamente en la directiva ***Listen***.

— **DocumentRoot** "*directorio-documentos*". Indica la ruta del directorio raíz que contiene todos los documentos que componen el sitio Web que deseamos servir. Xampp recomienda que todos los directorios raíz asociados a los distintos sitios Web se ubiquen en "*/opt/lampp/htdocs*" en GNU/Linux, o en "*c:/xampp/htdocs*" en Microsoft Windows, aunque puede hacerse uso de cualquier otra ruta.

```
ServerName www.web1.es
DocumentRoot /opt/lampp/htdocs/sitioweb1 #En entorno GNU/Linux
DocumentRoot "c:/xampp/htdocs/sitioweb1" #En entorno Microsoft Windows
```

Según las directivas del ejemplo anterior, si desde un cliente Web realizáramos una petición HTTP "*http://www.web1.es/galeria.html*" a Apache, estaríamos solicitando el documento Web ubicado en el servidor bajo GNU/Linux o Windows en "*/opt/lampp/htdocs/sitioweb1/galeria.html*" o "*c:/xampp/htdocs/sitioweb1/galeria.html*" respectivamente.

— **DirectoryIndex** *lista-archivos-inicio*. Indica cual será el documento que será servido por defecto ante una solicitud HTTP entre todos los que componen el sitio Web. En caso de no

37 El último capítulo del libro trata todos los aspectos necesarios para poner en marcha los servicios vistos a través de Internet.

asignársele valor a esta directiva, por defecto, el documento que será servidor por defecto será "*index.html*", "*index.pl*", "*index.cgi*", "*index.php*", etc.

```
Listen 80
ServerName www.web1.es
DocumentRoot /opt/lampp/htdocs/sitioweb1/indice.html #En entorno GNU/Linux
DocumentRoot "c:/xampp/htdocs/sitioweb1" #En entorno Microsoft Windows
DirectoryIndex indice.html
```

En el ejemplo anterior, al recibir una petición HTTP por el puerto 80 desde un cliente Web "*http://www.web1.es*", Apache le devolverá la página Web de inicio del sitio Web "*indice.html*" ubicada en "*/opt/lampp/htdocs/sitioweb1/indice.html*" o "*c:/xampp/htdocs/sitioweb1/indice.html*" dependiendo de si el sistema operativo bajo el que trabaja Xampp es GNU/Linux o Windows.

— **LogFormat** *formato nombre-formato*. **CustomLog** *ruta-fichero-log formato*. **TransferLog** *ruta-fichero-log*. **ErrorLog** *ruta-fichero-log*. Un aspecto muy importante que hay que tener en cuenta cuando se ofrece un servicio es la auditoría. Esta nos permite conocer quien, cuando, desde donde y que hizo al solicitar el servicio. Esta información de auditoría se almacena en los llamados ficheros *log*. La directiva **LogFormat** nos permite definir el formato de la información que será almacenada en los ficheros de auditoría, CustomLog especifica la ruta del fichero donde se guarda dicha información haciendo uso de alguno de los formatos creados con **LogFormat**, y **TransferLog** es similar a **CustomLog**, con la diferencia de que no se puede especificar cualquier formato de auditoría, sino que se hará uso de un formato por defecto definido en la configuración de Apache.

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access_log common
TransferLog logs/access_log
```

Para la definición del formato de auditoría Apache dispone de un conjunto de variables, entre las que podrían destacarse las siguientes:

Variable	Significado
%a %h	Dirección IP del equipo cliente que solicita la conexión HTTP.
%l %u	Nombre de usuario con que se accede al servidor.
%t	Hora en que fue recibida la solicitud de conexión.
%r	Información sobre la primera línea de la solicitud recibida.
%s	Información sobre el estado de la solicitud.
%b %O	Tamaño en bytes de la respuesta del servidor,sin incluir o incluyendo la cabecera HTTP.

También es posible auditar los errores que puedan darse en el servicio haciendo uso de la directiva **ErrorLog**:

```
ErrorLog logs/error_log
```

Advertir que para indicar las rutas de los ficheros de auditoría se ha hecho uso de rutas relativas en relación a el directorio raíz del servidor Apache indicada en la directiva **ServerRoot**, pero también puede hacerse uso de rutas absolutas.

— **<VirtualHost** *direccion-IP*[:*puerto*] [*direccion-IP*[:*puerto*]]**>** *directivas-configuración* **</VirtualHost>**. Directiva de configuración utilizada cuando queremos dar servicio a más de un sitio Web. Es necesario indicar una dirección IP y opcionalmente el puerto de comunicaciones del equipo servidor bajo la cual se realizarán escuchas de solicitudes de conexión HTTP al sitio Web que da servicio el VirtualHost.

```
Listen 8088
<VirtualHost 192.168.1.1:8088 172.30.1.1:8088>
    ServerName www.sitioweb1.es
    DocumentRoot /opt/lampp/htdocs/sitioweb1
    DirectoryIndex inicial.html
</VirtualHost>
```

En el ejemplo anterior, al recibir el servidor una petición HTTP por el puerto *8088* a través de alguna de las direcciones IP indicadas, *192.168.1.1* o *172.30.1.1*, resultado de la resolución del nombre de dominio "*www.sitioweb1.es*", se le devolverá al cliente su página de inicio "*/opt/lampp/htdocs/sitioweb1/inicial.html*".

En el caso de que se desee dar servicio a más de un sitio Web bajo la misma IP y por el mismo puerto de comunicaciones, será necesario diferenciarlos a través del nombre de dominio asignado a la directiva **ServerName**, informando de ello a Apache mediante la directiva "**NameVirtualHost** *direccion-IP*". Ha esta configuración se denomina *Hosts Virtuales basados en nombre*.

```
Listen 80
Listen 8088
NameVirtualHost 192.168.1.1
NameVirtualHost 127.0.0.1
<VirtualHost 192.168.1.1:80 127.0.0.1:8088>
    ServerName www.sitioweb1.es
    DocumentRoot /opt/lampp/htdocs/sitioweb1
</VirtualHost>
<VirtualHost 192.168.1.1:80 127.0.0.1:8088>
    ServerName www.sitioweb2.es
    DocumentRoot /opt/lampp/htdocs/sitioweb2
</VirtualHost>
```

Como se puede observar, los números de puerto asociados a las direcciones IP especificadas en los *VirtualHost* tienen que ser compatibles con los puertos de escucha de la directiva **Listen**. En caso contrario no funcionará el servicio.

— **Maxclients** *cantidad*. Estable un límite en el número máximo de conexiones o solicitudes HTTP que serán aceptadas simultáneamente por Apache. Su valor tiene relación con el asignado a la directiva "**ServerLimit** *cantidad*", siendo este último el valor máximo de procesos hijos de Apache.

```
ServerLimit 1000
MaxClients 800
```

— **LoadModule** *nombre-modulo ruta-fichero-modulo*. Carga dinámicamente el modulo indicado desde la ruta especificada. La modularidad de Apache es una de sus características más importantes al poder personalizar su comportamiento de manera ajustada agre-

gando únicamente los módulos necesarios. Cada módulo tiene asociadas un conjunto de directivas que pueden ser utilizadas en la configuración de Apache, por lo que tan sólo sería necesario cargar aquellos módulos asociados a las directivas que son utilizadas. Por ejemplo, para poder configurar sitios Web no anónimos, donde los usuarios que se autentifiquen en el acceso sean validados por un gestor de bases de datos (*DBM, Data Base Management*) será necesario tener cargado el siguiente módulo:

```
LoadModule authn_dbm_module modules/mod_authn_dbm.so
```

Todos aquellos módulos que son cargados sin que sean utilizadas sus directivas en la configuración de Apache sólo provocará un mayor consumo de recursos del sistema. En el fichero **httpd.conf** podemos encontrar una larga lista de módulos, donde aquellos que no sean útiles pueden deshabilitarse comentando la línea correspondiente precediéndola con el carácter almohadilla "**#**".

— Aunque hay directivas como *DocumentRoot* o <*VirtualHost*></*VirtualHost*> que pertenecen al núcleo de Apache, modulo *core*, y que pueden utilizarse sin tener cargado ningún módulo dinámico DSO, otras directivas, como *DirectoryIndex*, solo puede utilizarse al encontrarse cargado el módulo *mod_dir*, o *LogFormat*, *CustomLog* y *TransferLog* si lo esta el módulo *mod_log_config*.

```
LoadModule dir_module modules/mod_dir.so #Permite utilizar la directiva DirectoryIndex
LoadModule log_config_module modules/mod_log_config.so #LogFortat, CustomLog, TransferLog
```

— **Include** *ruta-fichero-configuración*. Permite incluir en la configuración del servidor Apache las directivas que se encuentren otros ficheros auxiliares. Al poder dividir en diferentes ficheros la configuración, hace que el fichero **httpd.conf** sea más legible al no cargarlo con todas las directivas.

```
Include etc/extra/mis-directivas.conf
```

Advertir que para indicar la ruta del fichero auxiliar de configuración se puede hacer uso de una ruta relativa en relación a la raíz del servidor Apache asignada a la directiva **ServerRoot**. Es decir, la ruta anterior, sería equivalente a haber puesto "*/opt/lampp/etc/extra/mis-directivas.conf*" en GNU/Linux, o "*c:/xampp/apache/conf/extra/mis-directivas.conf*".

Una información más detallada de estas y otras directivas podemos encontrarla en la documentación Web del proyecto Apache "*http://httpd.apache.org/docs/*".

Figura 6.10. Documentación Web de Apache: "http://httpd.apache.org/docs"

4.1. Configuración de Hosts Virtuales en Apache

Una característica fundamental de todo servidor Web, y por tanto de Apache, es la capacidad de dar servicio a más de un sitio Web simultáneamente. Para ello, Apache nos ofrece dos alternativas diferentes mediante la implementación de Hosts Virtuales basados en dirección IP y puerto de comunicaciones, o Hosts Virtuales basados en nombre de dominio. Ambas configuraciones serán explicadas mediante la realización de dos ejercicios prácticos.

Ejercicio práctico n.º 1

Configura Apache mediante la implementación de Hosts Virtuales basados en direcciones IP o puertos de escucha diferentes para que además de dar servicio al sitio Web de Xampp, *http://localhost:80*, de servicio a otros tres sitios Web que hayas diseñado en los capítulos anteriores.

Dirección IP / Puerto	Nombre Dominio ServerName	Directorio Raíz DocumentRoot	Página Inicio DirectoryIndex
127.0.0.1:8088	www.sitioweb1.es	/opt/lampp/htdocs/sitioweb1 c:/xampp/htdocs/sitioweb1	index.html
192.168.1.1:80 192.168.1.1:12345	www.sitioweb2.es	/opt/lampp/htdocs/sitioweb2 c:/xampp/htdocs/sitioweb2	indice.html
172.30.1.1:80 172.30.1.1:22122	www.sitioweb3.es	/opt/lampp/htdocs/sitioweb3 c:/xampp/htdocs/sitioweb3	inicio.html

Solución ejercicio n.º 1

Para resolver el ejercicio práctico propuesto, se aconseja seguir los siguiente pasos de configuración:

Paso n.º 1. Las direcciones IP que aparecen en el ejercicio práctico son un ejemplo. Cada uno debe hacer uso de las que ya tenga asignadas. La dirección 127.0.0.1 es la dirección de lazo interno que todo equipo puede utilizar para hacer referencia a si mismo sin necesidad de tener conexión de red. La dirección IP 192.168.1.1 se corresponderá con la dirección de Intranet que tenga asignada el equipo servidor con que se realizan las prácticas. Una tercera dirección IP puede agregarse al equipo en que caso de no tener otra asignada:

a) En GNU/Linux puede agregarse una nueva dirección IP a la interfaz de red (*eth0*) de nuestro equipo servidor con carácter temporal[38] haciendo uso de los comandos "ifconfig" o "ip addr":

```
[root@linux]# ifconfig eth0:1 172.30.1.1
[root@linux]# ip addr add 172.30.1.1/16 dev eth0
```

38 Al reiniciar el servicio de red o el equipo servidor, se perderán las direcciones IP agregadas con los comandos ifconfig o ip addr. Para que la configuración tenga carácter permanente será necesario modificar los scripts de red.

b) En Microsoft Windows se puede agregar una nueva dirección IP desde la ventana de *Propiedades de Conexión de área local* asociada a la interfaz de red del equipo, accediendo a las *Propiedades* del *Protocolo Internet (TCP/IP)*, seleccionando *Opciones Avanzadas*, tal como se muestra en la figura X.

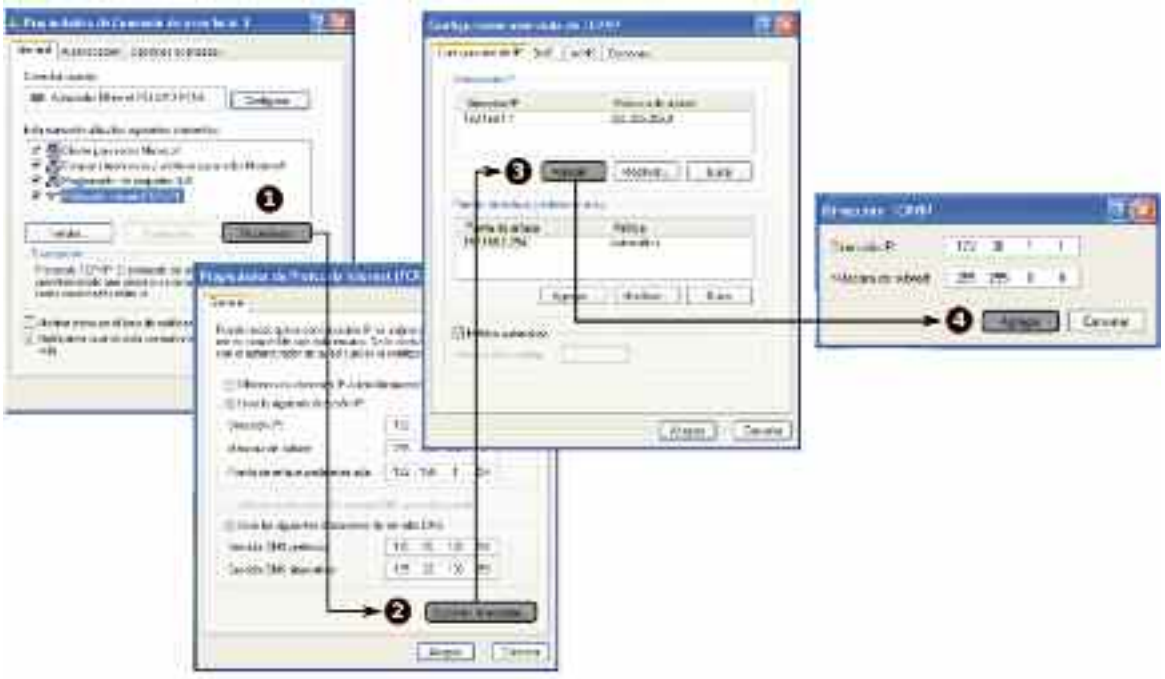


Figura 6.11. Pasos para agregar una dirección IP a un equipo bajo Microsoft Windows

Paso n.º 2. Para poder hacer uso de los nombres de dominio especificados en el enunciado del ejecicio, configuraremos localmente nuestro equipo para que lleve a cabo su resolución. Para ello, editaremos el fichero ***hosts*** localizado en /etc/hosts en GNU/Linux, en "*c:\winnt\system32\drivers\etc\hosts*" en Windows NT/2000, y en "*c:\windows\system32\drivers\etc\hosts*" en sistemas operativos Windows XP/2003/Vista, añadiendo las siguientes líneas:

```
# Contenido del fichero hosts: Direccion IP -> Nombre Equipo
127.0.0.1 localhost #Esta asociación se encuentra siempre por defecto en "hosts"
127.0.0.1 www.sitioweb1.es
192.168.1.1       www.sitioweb2.es
172.30.1.1        www.sitioweb3.es
```

¡¡Advertencia!! La resolución anterior solo tiene sentido a nivel de práctica local. Para que el acceso a los sitios Web anteriores haciendo uso de sus respectivos nombres de dominio sea posible desde un cliente Web localizado en cualquier ubicación de la Intranet/Extranet/Internet es necesario registrar públicamente los nombres de dominio, para que sean los servidores DNS los encargados de llevar a cabo la resolución. Estos aspectos serán tratados en el capítulo X.

Paso n.º 3. Crearemos los directorios raíz *web1*, *web2* y *web3* dentro del subdirectorio *htdocs* del sistema Xampp. En su interior copiaremos los documentos Web que compongan cada uno de los tres sitios Web, renombrando sus páginas de inicio por *index.html*, *indice.html* e *inicio.html* respectivamente.

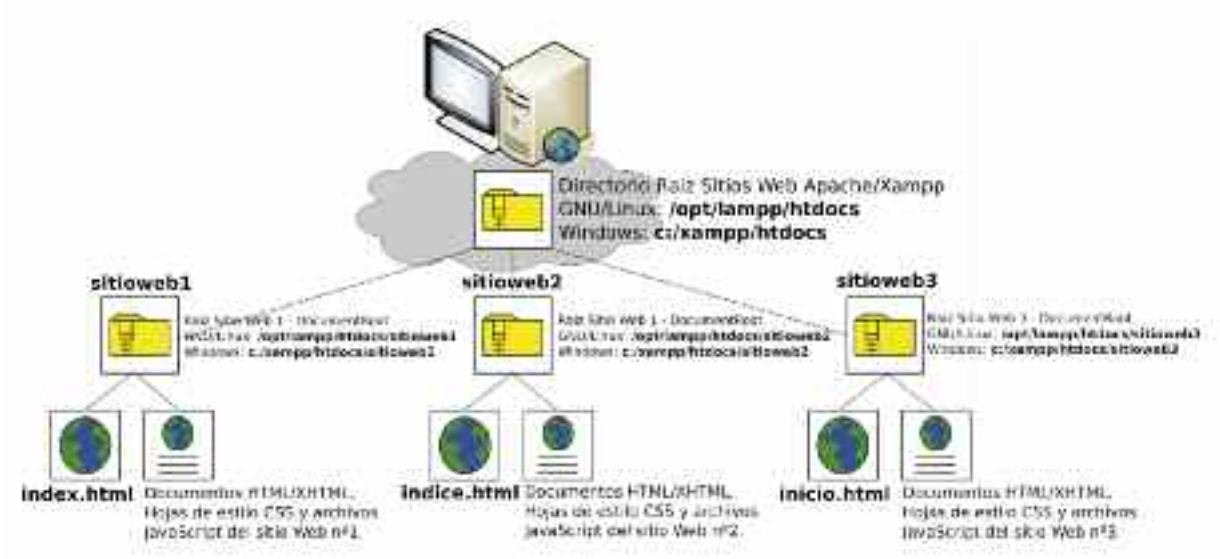


Figura 6.12. Esquema del sistema de ficheros del ejercicio práctico n.º 1

Paso n.º 4. Configuración del servidor Web Apache. Para ello abriremos el fichero **httpd.conf** con un editor de textos e incluiremos las siguientes directivas de configuración:

```
# Contenido del fichero de configuración de Apache: httpd.conf
# Las primeras tres directivas se encuentran en el fichero httpd.conf por defecto.  No deben
# modificarse para que el servicio Xampp pueda seguir funcionando vía Web: http://localhost.
Listen 80 # Puerto de escucha de Apache por defecto, a través de cualquier dirección IP
DocumentRoot /opt/lampp/htdocs # Directorio Raíz de Xampp en GNU/Linux
DocumentRoot c:/xampp/htdocs # Directorio Raíz de Xampp en Microsoft Windows
ServerName localhost # Nombre del equipo servidor Xampp
################ NUEVAS DIRECTIVAS DE CONFIGURACIÓN #################
# Directivas añadidas a httpd.conf para dar servicio a los tres sitios Web del ejercicio:
Listen 127.0.0.1:8088 # Escuchará peticiones HTTP a través de la IP 127.0.0.1 y puerto 8088
Listen 192.168.1.1:12345 # Escuchará peticiones HTTP a través de la IP 192.168.1.1 y puerto 12345
Listen 172.30.1.1:22122 # Escuchará peticiones HTTP a través de la IP 172.30.1.1 y puerto 22122
<VirtualHost 127.0.0.1:8088>
    ServerName www.sitioweb1.es # Nombre de dominio del sitio Web 1
    DocumentRoot /opt/lampp/htdocs/sitioweb1 # Directorio Raíz del sitio Web 1 en GNU/Linux
    DocumentRoot c:/xampp/htdocs/sitioweb1 # Directorio Raíz del sitio Web 1 en Windows
    DirectoryIndex index.html # No es necesaria, al ser index.html la página de inicio por defecto
</VirtualHost>
<VirtualHost 192.168.1.1>
    ServerName www.sitioweb2.es # Nombre de dominio del sitio Web 2
    DocumentRoot /opt/lampp/htdocs/sitioweb2 # Directorio Raíz del sitio Web 2 en GNU/Linux
    DocumentRoot c:/xampp/htdocs/sitioweb2 # Directorio Raíz del sitio Web 2 en Windows
    DirectoryIndex indice.html # Página de inicio a servir del sitio Web
</VirtualHost>
```

```
<VirtualHost 172.30.1.1>
    ServerName www.sitioweb3.es # Nombre de dominio del sitio Web 3
    DocumentRoot /opt/lampp/htdocs/sitioweb3 # Directorio Raíz del sitio Web 3 en GNU/Linux
    DocumentRoot c:/xampp/htdocs/sitioweb3 # Directorio Raíz del sitio Web 3 en Windows
    DirectoryIndex inicio.html # Página de inicio a servir del sitio Web
</VirtualHost>
### FINAL DE LA SECCIÓN DE DIRECTIVAS AÑADIDAS – EJERCICIO PRACTICO 1 ###
# A continuación, el resto de directivas de configuración de Apache que vienen por defecto ...
```

Otra alternativa hubiera sido crear un fichero auxiliar de configuración (por ejemplo, *ejercicio-practico1.conf*) que contuviera todas las directivas añadidas en el fichero **httpd.conf** en el ejemplo anterior, e incluirlo mediante la directiva **Include**:

```
# Contenido del fichero de configuración de Apache: httpd.conf
Listen 80 # Puerto de escucha de Apache por defecto, a través de cualquier dirección IP
DocumentRoot /opt/lampp/htdocs # Directorio Raíz de Xampp en GNU/Linux
DocumentRoot c:/xampp/htdocs # Directorio Raíz de Xampp en Microsoft Windows
ServerName localhost # Nombre del equipo servidor Xampp
# Incluimos el fichero auxiliar con todas sus directivas en la configuración de Apache
Include /opt/lampp/etc/extra/ejercicio-practico1.conf # Inclusión en GNU/Linux
Include c:/xampp/apache/conf/extra/ejercicio-practico1.conf # Inclusión en Windows
# A continuación, el resto de directivas de configuración de Apache que vienen por defecto ...
```

Paso n.º 5. Para que los cambios en la configuración de Apache surtan efecto es necesario reiniciar el servicio.

— Para reiniciar el servicio bajo GNU/Linux existen diferentes alternativas:

```
[root@linux]# /opt/lampp/lampp restart (reinicia todos los servicios Xampp)

[root@linux]# /opt/lampp/lampp stopapache (para el servicio apache)
[root@linux]# /opt/lampp/lampp startapache (inicia el servicio apache)

[root@linux]# /opt/lampp/bin/httpd -k restart (reinicia únicamente el servicio Apache)
[root@linux]# /opt/lampp/bin/apachectl -k restart (reinicia únicamente el servicio Apache)
```

— Para reiniciar el servicio bajo Microsoft Windows, puede pararse e iniciarse el servicio desde la interfaz gráfica *Xampp Control Panel*, o ejecutando uno de los siguientes comandos:

```
C:\xampp> xampp_restart.exe (reinicia todos los servicios iniciados Xampp)
C:\xampp\apache\bin> httpd.exe -k restart (reinicia únicamente el servicio Apache)
C:\xampp\apache\bin> apache.exe -k restart (reinicia únicamente el servicio Apache)
```

Paso n.º6. Por último, comprobaremos el correcto funcionamiento de la configuración anterior ejecutando nuestro cliente Web preferido (por ejemplo, *Mozilla Firefox*, *Internet Explorer*, u *Opera*), y escribiendo en la barra de direcciones el protocolo a utilizar, *http://*, seguido del nombre de dominio, *www.sitioweb1.es*, y el puerto de comunicaciones, en el caso de que el puerto de servicio no sea el puerto por defecto, *80*.

— "***http://localhost***", mostrará el sitio Web de Xampp.

— "***http://www.sitioweb1.es:8088***", mostrará la página de inicio del sitio Web n.º 1.

— "***http://www.sitioweb2.es***" o "***http://www.sitioweb2.es:12345***", mostrará la página de inicio del sitio Web n.º 2.

— "***http://www.sitioweb3.es***" o "***http://www.sitioweb3.es:22122***", mostrará la página de inicio del sitio Web n.º 3.

Ejercicio práctico n.º 2

Configura Apache mediante la implementación de Hosts Virtuales basados en nombre, al hacer uso de una misma dirección IP y puerto de comunicaciones para dar servicio a más de un sitio Web. Además del sitio Web de Xampp, *http://localhost:80*, deberá dar servicio a otros tres sitios Web que hayas diseñado en los capítulos anteriores.

Dirección IP / Puerto	Nombre Dominio ServerName	Directorio Raíz DocumentRoot	Página Inicio DirectoryIndex
127.0.0.1:80	www.sitioweb1.es	/opt/lampp/htdocs/sitioweb1 c:/xampp/htdocs/sitioweb1	index.html
192.168.1.1:80 192.168.1.1:12345	www.sitioweb2.es	/opt/lampp/htdocs/sitioweb2 c:/xampp/htdocs/sitioweb2	indice.html
172.30.1.1:80 172.30.1.1:22122	www.sitioweb3.es	/opt/lampp/htdocs/sitioweb3 c:/xampp/htdocs/sitioweb3	inicio.html

Solución ejercicio n.º 2

Para resolver el ejercicio práctico propuesto, se aconseja seguir los siguiente pasos de configuración:

Paso n.º 1. A diferencia del ejercicio práctico anterior, no será necesario agregar ninguna dirección IP al equipo servidor, al hacer uso de la dirección IP de lazo interno *127.0.0.1* y aquella que tenga asignada dentro de la Intranet (p.e. *192.168.1.1*).

Paso n.º 2. Modificaremos el contenido del fichero **hosts** realizado en el ejercicio práctico n.º 1 para la nueva resolución de nombres.

```
# Contenido del fichero hosts: Direccion IP -> Nombre Equipo
127.0.0.1     localhost # Esta asociación se encuentra siempre por defecto en "hosts"
127.0.0.1     www.sitioweb1.es
192.168.1.1   www.sitioweb2.es
192.168.1.1   www.sitioweb3.es # Modificamos la resolución del nombre www.sitioweb3.es
```

Paso n.º 3. Haremos uso de la misma estructura del sistema de ficheros, directorios raíz de los sitios Web y contenidos, que en el ejercicio práctico n.º 1.

Paso n.º 4. Al hacer uso de una misma dirección IP y puerto de comunicaciones en más de un sitio Web a servir, *127.0.0.1:80*, *192.168.1.1:80* y *192.168.1.1:12345*, es obligatoria la configuración de Apache haciendo uso de Hosts Virtuales basados en nombre. Para advertir de ello

a Apache, es necesario hacer uso de la directiva **NameVirtualHost**. Tras eliminar o comentar[39] las directivas de configuración añadidas en el fichero **httpd.conf** para la realización del ejercicio práctico anterior, estableceremos la nueva configuración. En este caso, haremos uso de un fichero auxiliar de configuración externo, **ejercicio-practico2.conf**, que contendrá la mayor parte de las directivas de la nueva configuración:

```
# Contenido del fichero de configuración de Apache: httpd.conf
Listen 80 # Puerto de escucha de Apache por defecto, a través de cualquier dirección IP
DocumentRoot /opt/lampp/htdocs # Directorio Raíz de Xampp en GNU/Linux
DocumentRoot c:/xampp/htdocs # Directorio Raíz de Xampp en Microsoft Windows
ServerName localhost # Nombre del equipo servidor Xampp
#### NUEVAS DIRECTIVAS DE CONFIGURACIÓN EJERCICIO Nº2 ######
NameVirtualHost 127.0.0.1 # Informamos a Apache de que la dirección IP 127.0.0.1 va a usarse en más de 1 sitio Web
NameVirtualHost 192.168.1.1 # Informamos a Apache de que la dirección IP 192.168.1.1 va a usarse en más de 1 sitio Web
Listen 192.168.1.1:12345 # Escuchará peticiones HTTP a través de la IP 192.168.1.1 por el puerto 12345
# Incluimos el fichero auxiliar con todas sus directivas en la configuración de Apache
#Include /opt/lampp/etc/extra/ejercicio-practico1.conf # Comentamos la configuración anterior
#Include c:/xampp/apache/conf/extra/ejercicio-practico1.conf # Comentamos la configuración anterior
Include /opt/lampp/etc/extra/ejercicio-practico2.conf # Inclusión en GNU/Linux
Include c:/xampp/apache/conf/extra/ejercicio-practico2.conf # Inclusión en Windows
# A continuación, el resto de directivas de configuración de Apache que vienen por defecto ...
```

```
# Contenido del fichero auxiliar de configuración de Apache: ejercicio-practico2.conf
<VirtualHost 127.0.0.1>
  ServerName www.sitioweb1.es # Nombre de dominio del sitio Web 1
  DocumentRoot /opt/lampp/htdocs/sitioweb1 # Directorio Raíz del sitio Web 1 en GNU/Linux
  DocumentRoot c:/xampp/htdocs/sitioweb1 # Directorio Raíz del sitio Web 1 en Windows
</VirtualHost>
<VirtualHost 192.168.1.1>
  ServerName www.sitioweb2.es # Nombre de dominio del sitio Web 2
  DocumentRoot /opt/lampp/htdocs/sitioweb2 # Directorio Raíz del sitio Web 2 en GNU/Linux
  DocumentRoot c:/xampp/htdocs/sitioweb2 # Directorio Raíz del sitio Web 2 en Windows
  DirectoryIndex indice.html # Página de inicio a servir del sitio Web
</VirtualHost>
<VirtualHost 192.168.1.1>
  ServerName www.sitioweb3.es # Nombre de dominio del sitio Web 3
  DocumentRoot /opt/lampp/htdocs/sitioweb3 # Directorio Raíz del sitio Web 3 en GNU/Linux
  DocumentRoot c:/xampp/htdocs/sitioweb3 # Directorio Raíz del sitio Web 3 en Windows
  DirectoryIndex inicio.html # Página de inicio a servir del sitio Web
</VirtualHost>
```

39 Todas aquellas líneas que estén precedidas del carácter almohadilla "#" pasarán a ser comentarios, e ignoradas por el interprete de Apache en la configuración del servicio HTTP.

Paso n.º 5. Reiniciaremos el servicio ofrecido por Apache para que los cambios en la configuración surtan efecto tal como se hizo en el ejercicio práctico anterior.

*Paso n.º 6.*Por último, comprobaremos el correcto funcionamiento de la configuración de Hosts Virtuales basados en nombre de dominio desde un cliente Web.

— "***http://localhost***", mostrará el sitio Web de Xampp.

— "***http://www.sitioweb1.es***", mostrará la página de inicio del sitio Web n.º 1.

— "***http://www.sitioweb2.es***" o "***http://www.sitioweb2.es:12345***", mostrará la página de inicio del sitio Web n.º 2.

— "***http://www.sitioweb3.es***" o "***http://www.sitioweb3.es:12345***", mostrará la página de inicio del sitio Web n.º 3.

4.2. Publicación de contenidos en Apache

En los ejemplos anteriores, se da por supuesto que es la cuenta de usuario *root* o *Administrador* quien gestiona, administra y pública los contenidos Web, al ser ésta la única que tiene privilegios para crear directorios e introducir documentos dentro del directorio ***htdocs***.

Para que sean otros los usuarios que puedan publicar contenidos, tenemos dos opciones:

1.ª opción. Que el usuario root o Administrador prepare los directorios en ***htdocs***, les de permisos de escritura a las cuentas de usuario a las que se desea permitir la publicación de contenidos, y configurar en el equipo servidor un servicio FTP para que puedan subirlos.

2.ª opción. En ocasiones nos puede interesar que determinadas cuentas de usuario dadas de alta en el equipo servidor puedan publicar contenidos Web a través de Apache desde su propio directorio HOME o perfil asignado. Para ello será necesario cargar el módulo **userdir**: *mod_userdir* en GNU/Linux y *userdir_module* en Microsoft Windows.

Estas dos opciones serán tratadas en los siguientes subapartados, junto con un ejercicio práctico para su mejor comprensión.

4.2.1. Publicación de contenidos vía FTP

Esta es la opción que habitualmente se adopta para permitir que las cuentas de usuario deseadas puedan publicar contenidos en la Web. Un ejemplo muy habitual de esta situación suele darse en Internet con el alquiler de *hosting* para que particulares u organizaciones pueden colgar sus sitios Web en Internet. Para comprender mejor este tipo de implementación se propone la realización del siguiente ejercicio práctico.

Ejercicio práctico n.º 3

Configura los servicios Xampp HTTP (*Apache*) y FTP (*FileZilla | Windows o ProFTPd | GNU / Linux*), para que una cuenta de usuario, p.e. *usuario1,* pueda subir documentos Web al servidor vía FTP dentro del subdirectorio "*htdocs / usuario1*", y puedan ser consultados vía HTTP desde un cliente Web mediante la URL "*http: / / usuario1.sitioweb.es*".

Dirección IP / Puerto	Nombre Dominio ServerName	Directorio Raíz DocumentRoot	Página Inicio DirectoryIndex
192.168.1.1:80	usuario1.sitioweb4.es	/opt/lampp/htdocs/usuario1 c:/xampp/htdocs/usuario1	index.html

Solución ejercicio n.º 3

Para resolver el ejercicio práctico propuesto, se aconseja seguir los siguiente pasos de configuración:

Paso n.º 1. El *root* o *administrador* del sistema creará el directorio raíz donde el usuario subirá los documentos que componen su sitio Web al servidor dentro del directorio Xampp **htdocs**: *c:/xampp/htdocs/usuario1* en Microsoft Windows y */opt/lampp/htdocs/usuario1* en GNU/Linux.

Paso n.º 2. El *root* o *administrador* del sistema creará la cuenta de usuario FTP con privilegios suficientes sobre el directorio anterior para poder leer, escribir y modificar sus contenidos. Esta operación de configuración dependerá de si Xampp se ha instalado bajo Microsoft Windows o GNU/Linux, al ser el software servidor FTP diferente en cada caso:

— Microsoft Windows: Tras crear el directorio destinado al usuario "*c:/xampp/htdocs/usuario1*", configuraremos la cuenta FTP desde la interfaz de administración del servidor FTP que incluye Xampp, **FileZilla**. Accediendo desde el menú programas al *Xampp Control Panel*, con el servicio FTP *FileZilla* iniciado, pinchando sobre su botón "*Admin...*" asociado, tras indicar la dirección IP del servidor, puerto de gestión del servicio y la password de administración[40], accederemos a la interfaz *FileZilla Server*. Pinchando sobre el icono *Users*, o desde menú *Edit*, opción *Users*, accederemos a la ventana de gestión de cuentas de usuario FTP. Desde allí agregaremos la cuenta *usuario1*, tal como se muestra en la figura 6.13.

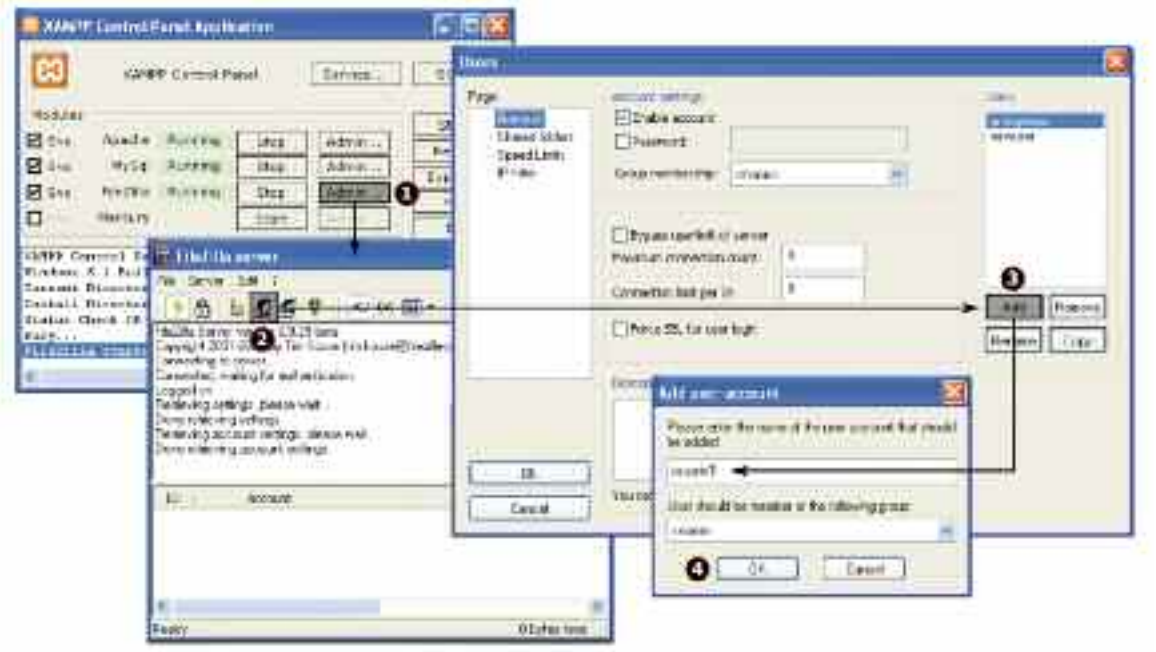


Figura 6.13. Configuración de la cuenta de usuario FTP en FileZilla Server

Tras crear la cuenta FTP, en *Shared Folders* le asignaremos el directorio "*c:/xampp/htdocs/usuario1*" como directorio *Home* (opción, *Set as home dir*)[41], con permisos de lectura, escritura y modificación para que pueda subir al servidor todos los documentos que componen su sitio Web, tal como se muestra en la figura 6.14.

40 Por defecto, el software servidor FTP FileZilla Server no tiene contraseña de administración de servicio. Por seguridad es conveniente modificarla desde el menú Edit/Settings/Admin Interface Settings.

41 El directorio Home de un usuario es aquel al que se accede por defecto tras autentificarse ante el servidor FTP.

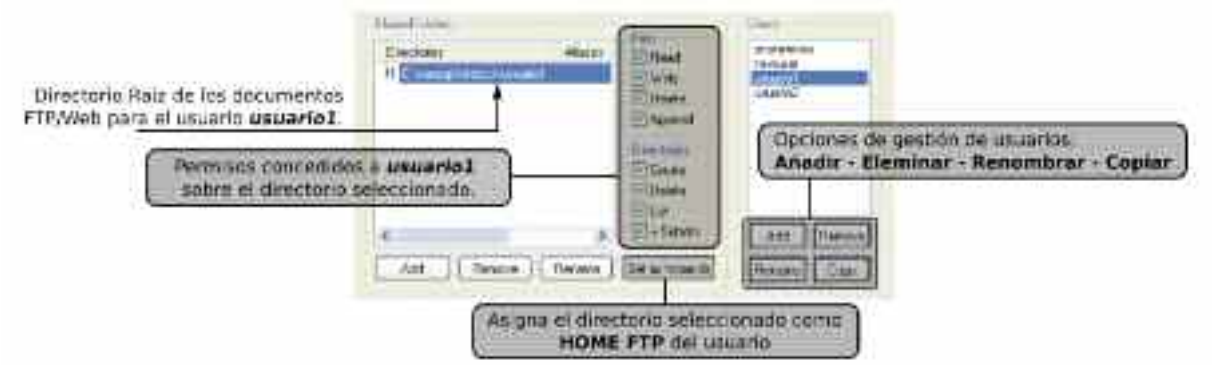


Figura 6.14. Configuración de la cuenta de usuario FTP en FileZilla

Tras la configuración anterior, tan sólo cabe comprobar que la cuenta de usuario FTP tiene acceso al servidor y puede subir contenidos. Abriendo un cliente FTP (por ejemplo, *FileZilla Client*, *Internet Explorer*, etc.), escribiremos en la barra de direcciones la URL correspondiente al sitio FTP, "*ftp://usuario1@direccion-IP-Servidor-FTP/Web*"[42].

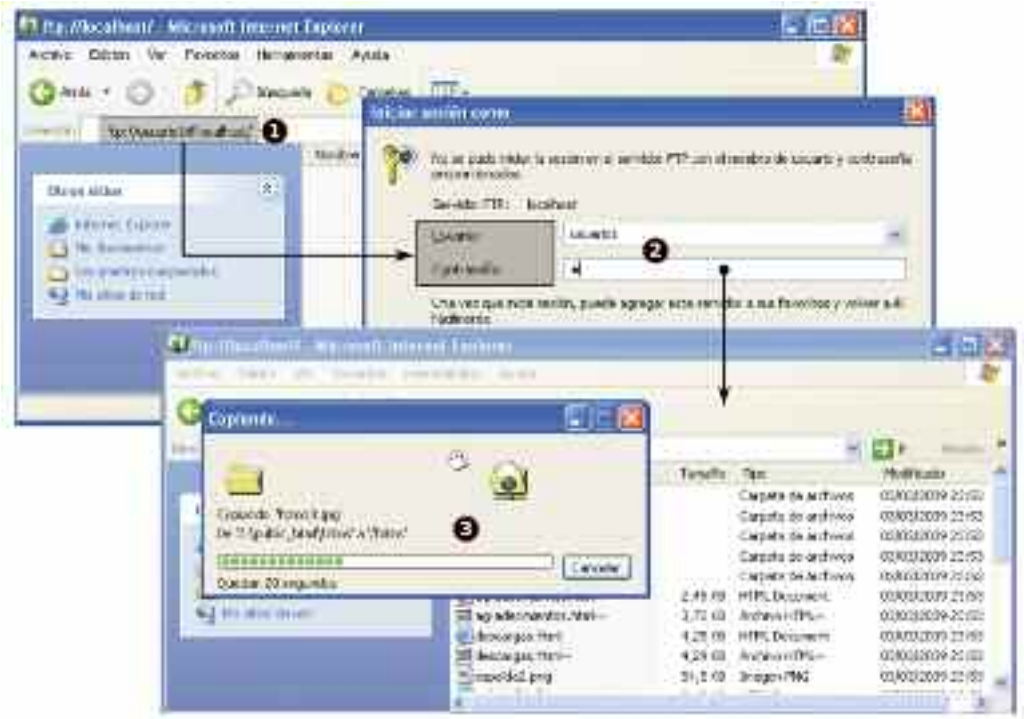


Figura 6.15. Comprobación de la subida de documentos Web al servidor FTP/HTTP desde un cliente

¡¡Observación!! Advertir que la cuenta de usuario FTP creada en FileZilla bajo Microsoft Windows, no es una cuenta valida dentro del sistema. Es decir, la cuenta *usuario1* nunca podrá iniciar sesión en Windows. Tan sólo podrá conectarse vía FTP al servidor. Este aspecto es importante es importante por cuestión de seguridad, ya que en caso contrario, el usuario podría conectarse e iniciar sesión remotamente vía TELNET o SSH, y convertirse en una amenaza para el sistema.

42 Para poder acceder mediante un cliente FTP/HTTP desde Internet al servidor FTP/HTTP ubicado en una Intranet, será necesario configurar la NAT en el router suministrado por nuestro proveedor de servicios de Internet. En el último capítulo se explican estos aspectos con más detalle.

— GNU/Linux: Tras crear el directorio destinado al usuario "/opt/lampp/*htdocs*/*usuario1*", configuraremos la cuenta FTP en **ProFTPd**. A diferencia de Windows, en GNU/Linux, esta cuenta será una cuenta del sistema:

```
[root@linux]# useradd -d directorio-Home -s shell -g grupo-usuarios nombre-usuario
[root@linux]# useradd -d /opt/lampp/htdocs/usuario1 -s /bin/false usuario1
[root@linux]# passwd usuario1
```

El comando *useradd* anterior, creará la cuenta especificada (*usuario1*), con todos los permisos sobre el directorio *Home* asignado (*-d /opt/lampp/htdocs/usuario1*), con una *Shell* o interprete de comandos falsa (*-s /bin/false*), para evitar que dicha cuenta de usuario pueda conectarse e iniciar sesión en el servidor remotamente vía TELNET o SSH. *passwd* le asignará una contraseña de acceso al sitio FTP.

Tras asegurarnos de que el servicio *ProFTPd* esta iniciado, el usuario anterior, al igual que en Microsoft Windows con *FileZilla*, ejecutando un cliente FTP (p.e. *konqueror, FileZilla, etc.*), podrá comprobar que es posible subir contenidos al servidor: "*ftp://usuario1@direccion-IP-Servidor-FTP/Web*".

```
[root@linux]# /opt/lampp/lampp stopftp (para el servicio ProFTPd)
[root@linux]# /opt/lampp/lampp startftp (inicia el servicio ProFTPd)
```

Paso n.º 3. El *root* o *administrador* del sistema editará los ficheros *hosts* y *httpd.conf* para resolver localmente el nombre de dominio *usuario1.sitioweb4.es* y configurar el servidor Apache para que de servicio al sitio Web subido por el *usuario1*.

```
# Resolución a incluir en el fichero hosts
192.168.1.1        usuario1.sitioweb4.es
```

```
# Directiva a incluir en el fichero de configuración de Apache httpd.conf
Include /opt/lampp/etc/extra/ejercicio-practico3.conf # Inclusión en GNU/Linux
Include c:/xampp/apache/conf/extra/ejercicio-practico3.conf # Inclusión en Windows
```

```
# Contenido del fichero auxiliar de configuración de Apache: ejercicio-practico3.conf
<VirtualHost 192.168.1.1>
  ServerName usuario1.sitioweb4.es # Nombre de dominio del sitio Web 2
  DocumentRoot /opt/lampp/htdocs/usuario1 # Directorio Raíz en GNU/Linux
  DocumentRoot c:/xampp/htdocs/usuario1 # Directorio Raíz en Windows
</VirtualHost>
```

Paso n.º 4. Por último, tras reiniciar el servicio HTTP ofrecido por Apache, comprobaremos mediante un cliente Web el acceso público al sitio Web anterior: "*http://usuario1.sitioweb4.es*".

```
[root@linux]# /opt/lampp/lampp restart (reinicia todos los servicios Xampp en GNU/Linux)
C:\xampp> xampp_restart.exe (reinicia todos los servicios Xampp en Windows)
```

— "***http://usuario1.sitioweb4.es***", mostrará la página de inicio *index.html* de *usuario1*.

Ejercicio práctico n.º 4

Modifica la configuración de Apache para que el acceso vía HTTP al sitio Web alojado por la cuenta de usuario FTP *usuario1* se haga a través de un directorio virtual, *usu1*, del tercer sitio Web que ya tiene configurado el servidor identificado por *www.sitioweb3.es*. La dirección URL que habrá que escribir en este caso en cliente Web será "*http://www.sitioweb3.es/usu1*".

Solución ejercicio n.º 4

Para resolver el ejercicio práctico propuesto, se aconseja seguir los siguiente pasos de configuración:

Paso n.º 1. Se denomina directorio virtual a aquel que sin ser subdirectorio de uno raíz, es visto por el visitante Web como si lo fuera. El ejercicio propone acceder al sitio Web ubicado en *htdocs/usuario1* mediante la URL "*http://www.sitioweb3.es/usu1*" como si *usu1* fuera un subdirectorio del directorio raíz o *DocumentRoot* de este tercer sitio Web, *htdocs/sitioweb3/usu1*. Para Apache, *usu1* tan sólo va a ser un alias del directorio original. Su configuración implica la utilización de la directiva "**Alias /***nombre-alias ruta-directorio*" dentro del Host Virtual asociado a este sitio Web, e inclusión de la página de inicio del usuario, *index.html*, en la lista indicada por su directiva **DirectoryIndex**. Para ello editaremos el fichero auxiliar de configuración *ejercicio-practico2.conf* creado durante la realización del ejercicio práctico 2.

```
# Directiva del fichero httpd.conf para incluir en su configuración a ejercicio-practico2.conf
Include /opt/lampp/etc/extra/ejercicio-practico2.conf # Inclusión en GNU/Linux
Include c:/xampp/apache/conf/extra/ejercicio-practico2.conf # Inclusión en Windows
```

```
# Contenido del fichero auxiliar de configuración de Apache: ejercicio-practico2.conf
# Configuración del tercer sitio Web realizada en el ejercicio práctico Nº2
<VirtualHost 192.168.1.1>
  ServerName www.sitioweb3.es # Nombre de dominio del sitio Web 3
  DocumentRoot /opt/lampp/htdocs/sitioweb3 # Directorio Raíz en GNU/Linux
  DocumentRoot c:/xampp/htdocs/sitioweb3 # Directorio Raíz en Windows
  DirectoryIndex index.html inicio.html # Páginas de inicio a servir del sitio Web
  Alias /usu1 /opt/lampp/htdocs/usuario1 # Alias en GNU/Linux
  Alias /usu1 c:/xampp/htdocs/usuario1 # Alias en Windows
</VirtualHost>
... # Resto de directivas de configuración del archivo ejercicio-practico2.conf
```

Paso n.º 2. Tras reiniciar el servicio Xampp o Apache para que los cambios en la configuración surtan efecto, comprobar el acceso a los documentos del usuario a través del alias anterior.

```
[root@linux]# /opt/lampp/lampp restart (reinicia todos los servicios Xampp en GNU/Linux)
C:\xampp> xampp_restart.exe (reinicia todos los servicios Xampp en Windows)
```

— "***http://www.sitioweb3.es/usu1***", mostrará la página de inicio index.html de *usuario1*.

4.2.2. Publicación de contenidos desde el home de los usuarios

En ocasiones nos puede interesar que determinadas cuentas de usuario dadas de alta en el sistema, GNU/Linux o Microsoft Windows, puedan publicar contenidos vía Web a través de Apache desde su propio *Home*. El *Home* de un usuario son el conjunto de directorios dentro del sistema de ficheros creados en el momento en que se crea un usuario e inicia sesión, donde este tiene todos los privilegios sobre ellos: lectura, escritura y modificación. En Microsoft Windows al *Home* del usuario también se le denomina perfil, y se crea por defecto en sus últimas versiones en "***c:\Documents and Settings****nombre-usuario*". En GNU/Linux, si al crear un usuario mediante el comando *useradd* no se indica explícitamente la opción "-d *directorio-home*", por defecto, el directorio *Home* se localizará en "***/home/****nombre-usuario*".

Las directivas de configuración de Apache que permiten esta publicación de contenidos desde el *Home* de los usuarios pertenecen al modulo DSO *userdir_module*, que deberemos tener cargado:

```
# Es necesario que el módulo userdir este cargado en el fichero de configuración httpd.conf
LoadModule userdir_module modules/mod_userdir.so
```

UserDir es la directiva encargada de especificar la ruta del directorio raíz o *DocumentRoot* dentro del *Home* de los usuarios del sistema donde dejarán los documentos Web que serán publicados por Apache. También nos permite decidir si esta publicación de contenidos es extensible a todas las cuentas de usuario del sistema, o esta restringido a unos determinados usuarios (*enabled*|*disabled*). Si no se indica lo contrario, por defecto, todo usuario podrá aprovecharse de esta opción.

```
Userdir ruta-directorio-publicacion
# Por defecto, si no se indica lo contrario, toda cuenta de usuario podrá publicar contenidos Web
Userdir enabled lista-nombres-cuentas-usuario-permitidas
Userdir disabled lista-nombres-cuentas-usuario-no-permitidas
```

Si advertimos que los directorios Home de las cuentas de usuario tienen los permisos restringidos para evitar unos puedan husmear los contenidos de otros, es necesario indicar a Apache que los documentos que se encuentran dentro del subdirectorio asignado a **UserDir** pueden ser publicados sin restricciones:

```
<Directory directorios-publicación-usuarios>
  # Permitimos el acceso desde cualquier equipo al directorio de publicación del Home
      Allow from all
</Directory>
```

Para comprender mejor estos aspectos se propone a continuación la realización de un ejercicio práctico.

Ejercicio práctico n.º 5

Configura el equipo servidor para que las cuentas de usuario del sistema *ususis1* y *ususis2* puedan publicar los documentos Web que se encuentren dentro del subdirectorio *miweb* que tengan en su *Home*, a través del siguiente sitio Web con página de inicio *index.html*:

Dirección IP / Puerto	Nombre Dominio	Directorio Raíz DocumentRoot	Directorio Usuario UserDir
192.168.1.1:80	www.sitioweb5.es	/opt/lampp/htdocs/sitioweb5 c:/xampp/htdocs/sitioweb5	/home/*/miweb c:/Documents and Settings/*/miweb

Solución ejercicio n.º 5

Para resolver el ejercicio práctico propuesto, se aconseja seguir los siguiente pasos de configuración:

Paso n.º 1. El *root* o *administrador* del sistema configurará Apache para ofrecer el servicio descrito en el enunciado del ejercicio anterior. Para ello se hará uso de un fichero auxiliar de configuración, *ejercicio-practico5.conf*, que deberá ser incluido en *httpd.conf*. También será necesario registrar localmente en el fichero *hosts* la resolución del nombre *www.sitioweb5.es*, haciéndole corresponder la dirección IP que tenga asignada el equipo bajo el cual se realizan prácticas dentro de la Intranet (por ejemplo, *192.168.1.1*).

```
# Resolución a incluir en el fichero hosts
192.168.1.1          www.sitioweb5.es
```

```
# Fichero de configuración de Apache httpd.conf: inclusión del fichero auxiliar.
Listen 80
NameVirtualHost 192.168.1.1
Include /opt/lampp/etc/extra/ejercicio-practico5.conf # Inclusión en GNU/Linux
Include c:/xampp/apache/conf/extra/ejercicio-practico5.conf # Inclusión en Windows
... # Resto de directivas de configuración
```

```
# Contenido del fichero auxiliar de configuración de Apache: ejercicio-practico5.conf
<VirtualHost 192.168.1.1>
  ServerName www.sitioweb5.es # Nombre de dominio del sitio Web 2
  DocumentRoot /opt/lampp/htdocs/sitioweb5 # Directorio Raíz en GNU/Linux
  DocumentRoot c:/xampp/htdocs/sitioweb5 # Directorio Raíz en Windows
  UserDir /home/*/miweb # Directorio de publicación en GNU/Linux
  UserDir "C:/Documents and Settings/*/miweb" # Directorio de publicación en Windows
  UserDir enabled ususis1 ususis2 # Solo permitimos públicar a esos dos usuarios
  UserDir disabled # El resto de usuarios tiene deshabilitada la publicación Web
  <Directory /home/*/miweb> # Directorio en GNU/Linux
  <Directory "C:/Documents and Settings/*/miweb"> # Directorio en Windos
          Allow from all # Permiso a todo equipo a consultar la docuemtnación Web del directorio especificado en Directory
  </Directory>
</VirtualHost>
```

Paso n.º 2. El *root* o *administrador* del sistema reiniciará el servicio Xampp o Apache para que surtan efecto los cambios realizados en la configuración.

```
[root@linux]# /opt/lampp/lampp restart (reinicia todos los servicios Xampp en GNU/Linux)
C:\xampp> xampp_restart.exe (reinicia todos los servicios Xampp en Windows)
```

Paso n.º 3. El *root* o *administrador* del sistema creará las cuentas de usuario *ususis1* y *ususis2*.

— Microsoft Windows: Las cuentas de usuario pueden ser creadas desde el *Panel de control* de Windows o desde la interfaz gráfica de *Administración de equipos*, accesible pinchando con el botón derecho del ratón sobre *Mi PC*, opción *Administrar*.

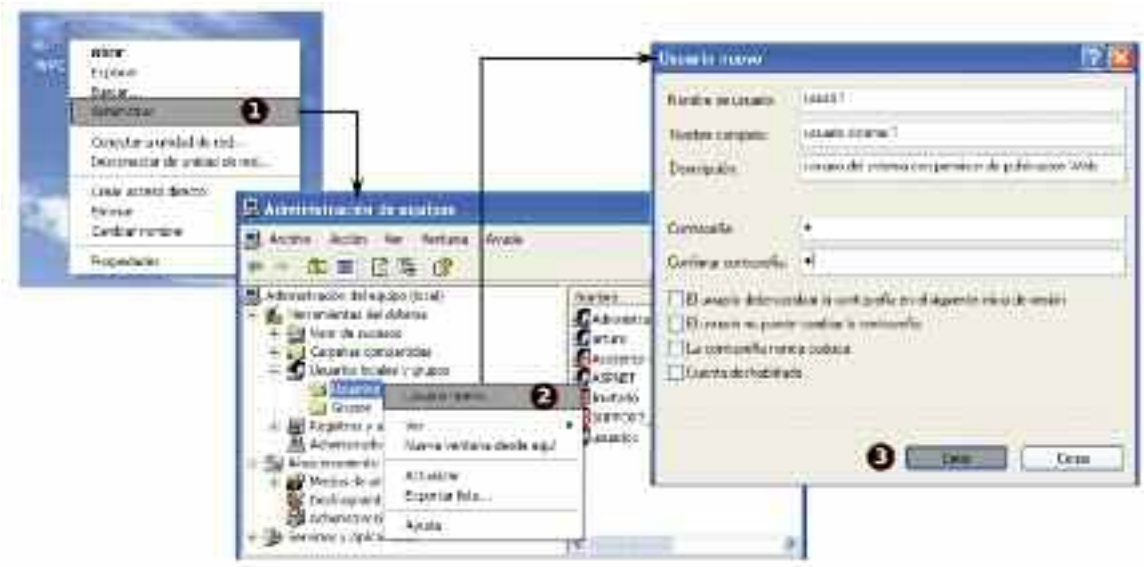


Figura 6.16. Creación de cuentas de usuario en Microsoft Windows

¡¡Aclaración!! Aunque en las capturas de pantalla relativas a la realización práctica del ejercicio puede advertirse que el sistema operativo de Microsoft elegido es Windows XP Professional por motivos de compatibilidad, al ser el más implantado, la publicación de contenidos desde el Home de los usuarios tiene más sentido en sistemas Windows multiusuario como Windows 2000/2003/2008 Server donde todos los perfiles de los usuarios del dominio son gestionados de manera centralizada en el servidor con Active Directory. Windows XP es un sistema operativo monousuario, donde solo puede haber iniciada simultáneamente una única sesión.

— GNU/Linux: Los comandos useradd y passwd nos permiten crear las cuentas de usuario del sistema y asignarles una contraseña de inicio de sesión respectivamente. Al no hacer uso de la opción "-d directorio-home", el directorio Home de los usuarios se creará en */home/ususis1* y */home/ususis2*.

```
[root@linux]# useradd ususis1
[root@linux]# passwd ususis1
[root@linux]# useradd ususis2
[root@linux]# passwd ususis2
```

Paso n.º 4. Las cuentas de usuario anteriores deberán iniciar sesión en el equipo servidor para crear su perfil en su directorio *Home*. A continuación crearán el directorio *miweb* en el directorio raíz de su perfil, *c:\Documents and Settings\nombre-usuario* en Windows y */home/nombre-usuario* en GNU/Linux, e introducirán en su interior el sitio Web que desean sea servido vía Web por Apache.

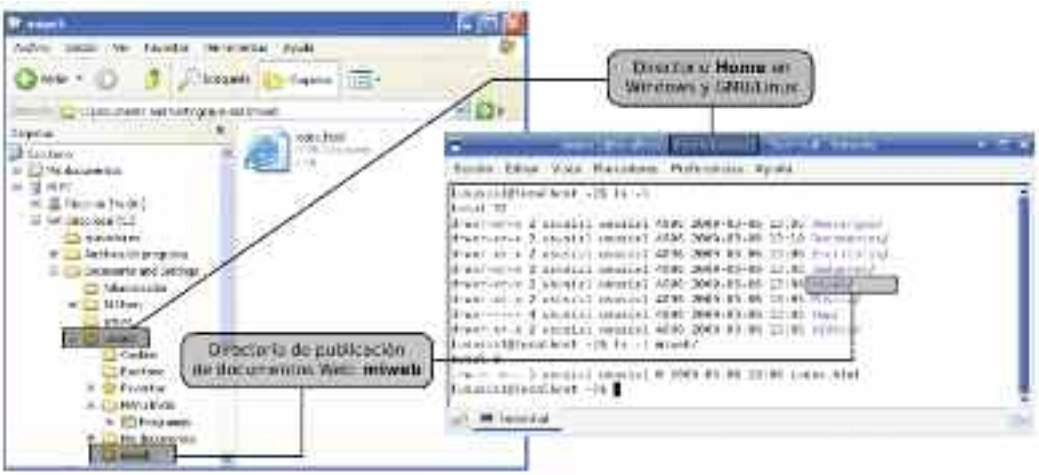


Figura 6.16. Directorios Home y de publicación de contenidos Web en Windows y GNU/Linux

¡¡Observación!! En Windows, el directorio Home, junto con todos los subdirectorios y archivos que componen el perfil del usuario son creados en el momento en que este inicia sesión. En cambio, en GNU/Linux si que se crea el directorio Home al crear el usuario, junto con algún subdirectorio básico. Esto permite que en GNU/Linux sea posible la conexión remota de los usuarios vía TELNET o SSH al equipo servidor sin necesidad de que estos hayan iniciado una sesión.

Paso n.º 5. Para comprobar el correcto funcionamiento de la publicación de contenidos desde los *Home* de los usuarios, deberemos escribir en la barra de direcciones de nuestro cliente Web preferido las siguientes URL:

— "***http://www.sitioweb5.es***", mostrará la página de inicio *index.html* del sitio Web 5. En el ejemplo se ha dado por supuesto, que el administrador o root del sistema a guardado dentro del directorio *htdocs/sitioweb5* las páginas Web que componen el sitio Web.

— "***http://www.sitioweb5.es/~ususis1***", mostrará la página de inicio *index.html* del sitio Web del usuario *ususis1*. En el ejemplo se ha dado por supuesto que la cuenta de usuario *ususis1* a guardado dentro del directorio *miweb* de su *Home* las páginas Web que componen su sitio Web.

— "***http://www.sitioweb5.es/~ususis2***", mostrará la página de inicio *index.html* del sitio Web del usuario *ususis2*. En el ejemplo se ha dado por supuesto que la cuenta de usuario *ususis2* a guardado dentro del directorio *miweb* de su *Home* las páginas Web que componen su sitio Web.

Como puede observarse en las URL, para hacer referencia al sitio Web de un usuario del sistema, es necesario hacer uso del carácter especial "~", denominado *Tilde* en el sistema Unicode (U+007E). Para imprimirlo en GNU/Linux, puede hacerse uso de la combinación de teclas *AltGr+Ñ*. En Windows puede obtenerse con *Alt+126*, o yendo a *Inicio/Accesorios/Herramientas del sistema/Mapa de caracteres*.

Figura 6.18. Mapa de caracteres enWindows

5. CONFIGURACIÓN DE SITIOS WEB NO ANÓNIMOS

En ocasiones nos puede interesar que determinados documentos Web servidos por Apache no sean accesibles públicamente y sea requerida una autenticación previa (*login y password*). Todos los sitios Web servidos hasta el momento por Apache han sido anónimos, de tal forma que cualquier usuario podía consultar la información publicada en la Web sin ningún tipo de restricción. Al diseñar un sitio Web no anónimo, puede interesarnos que su acceso se encuentre restringido totalmente, o que tan sólo se encuentre en parte, siendo esta última la situación más habitual. Es muy habitual encontrarnos al navegar por Internet secciones dentro de un sitio Web, o enlaces de descarga, que al tratar de acceder a ellas, o descargarnos el correspondiente archivo, se nos muestra una ventana de dialogo para la autenticación. En este último apartado del capítulo se indicarán las directivas más importantes que permiten configurar Apache para dar este tipo de servicio, junto con dos ejercicios prácticos que tratarán de aclararlo.

Para ello deberemos asegurarnos en primer lugar que los módulos de Apache que están relacionados con la autenticación, **mod_auth_basic**, **mod_auth_digest**, **mod_authn_file** y **mod_authn_dbm**, se cargan al iniciar el servicio, eliminando el carácter "#", en caso de que lo encontremos precediendo a las directivas **LoadModule** que se encargan de ello en el fichero de configuración *httpd.conf*.

```
# Modulos Apache relacionados con la autenticación que deben cargarse en httpd.conf
LoadModule auth_basic_module modules/mod_auth_basic.so
LoadModule auth_digest_module modules/mod_auth_digest.so
LoadModule authn_file_module modules/mod_authn_file.so
LoadModule authn_dbm_module modules/mod_authn_dbm.so
```

Las directivas más importantes pertenecientes a los módulos anteriores de las que vamos a hacer uso serán las siguientes:

Directiva	Significado
AuthType Basic|Digest Ej: **AuthType** Basic	Indica el tipo de autenticación que se va a realizar sobre las áreas restringidas a usuarios. El tipo **basic** requiere del módulo *mod_auth_basic* y **digest** de *mod_auth_digest*. Su diferencia esencial, es que en el primer caso, las contraseñas viajan entre cliente y servidor sin cifrar, y en el segundo cifradas. No todos los clientes Web soportan la autenticación **digest**.

Directiva	Significado
AuthName "*Dominio Autenticacion - Realm*" Ej: **AuthName** "Zona Restrigida N1"	Establece un dominio de autenticación o realm. Permite diferenciar entre diferentes areas de acceso restringido dentro de un mismo sitio Web. La mayoría de los clientes Web lo muestran en la ventana de autenticación con la finalidad de que el usuario reconozca dentro de que área de acceso restringido trata de entrar. Una vez validado el nombre y contraseña, el usuario ya no tendrá que volver a introducirlos al acceder a cualquier recurso que pertenezca al mismo dominio de autenticación.
AuthBasicProvider file|dbm|dbd|ldap **AuthDigestProvider** file|dbm|dbd Ej: **AuthBasicProvider** dbm	Indican cual es módulo de Apache que le va a proveer en la autenticación, para la gestión de cuentas de usuario Web y contraseñas. Puede ser **file** (*mod_authn_file*), **dbm** (*mod_authn_dbm*), **dbd** (*mod_authn_dbd*) o **ldap** (*mod_authnz_ldap*). Dependiendo del valor de AuthType, Basic|Digest, se utilizará una u otra.
AuthDBMType DB|SDBM|GDBM|NDBM Ej: **AuthDBMType** SDBM	En el caso de hacer uso de un gestor de bases de datos (*DBM, DataBase Management*) para la gestión de usuarios y contraseñas, por ejemplo, **AuthBasicProvider dbm**, esta directiva le indicará a Apache el tipo de gestor dentro de los cuatro posibles, aunque no tienen porque estar disponibles todos los tipos, al depender de como fue compilado Apache.
AuthUserFile *ruta-Fichero-usuarios* **AuthGroupFile** *ruta-Fichero-grupos* **AuthDBMUserFile** *ruta-DB-usuarios* **AuthDBMGroupFile** *ruta-DB-grupos* Ej: **AuthUserFile** usuarios.file Ej: **AuthDBMUserFile** permitidos.db	Indican la ruta donde se localiza el fichero o base de datos donde se almacenan los nombres de las cuentas de usuario Web y contraseñas que serán utilizados para validar el *login* y *password* introducidos por el usuario durante la autenticación. Se hará uso de uno u otro dependiendo de si el valor de la directiva **AuthBasicProvider** o **AuthDigestProvider** es *file* o *dbm*.
Require user *lista-usuarios* **Require** group *lista-grupos-usuarios* Ej: **Require** user usu1 usu2 Ej: **Require valid-user**	Establece que cuentas de usuario de las registradas en **AuthUserFile** o **AuthDBMUserFile** serán válidas en el acceso. En el caso de que se desee indicar que todas las cuentas registradas son válidas, se usara el argumento **valid-user**.
Order Allow Deny Ej: **Order deny,allow deny from 192.168.2.0/24 allow from all**	Restringen el acceso al servicio Web a determinados equipos mediante la especificación de sus direcciones IP o nombres de dominio. Las directivas **allow** y **deny** especifican respectivamente cuales son los equipos permitidos, y cuales no. **Order** indica el orden en que serán consultadas las directivas anteriores.

Para la creación de los ficheros o bases de datos que se encargarán de la gestión de las cuentas de usuario, Apache dispone de un conjunto de utilidades que se describen a continuación:

— **htpasswd**: Genera e inserta en un fichero plano los usuarios Web y sus contraseñas. Es utilizado cuando se realiza autenticación de tipo *Basic*, basado en fichero, *AuthBasicProvider file*. Su sintaxis más básica es la siguiente:

htpasswd [-c] ruta-fichero nombre-usuario

La opción "**-c**", *create*, solo debe utilizarse para crear el fichero en el momento de insertar el primer usuario Web. Para insertar el resto de usuarios debe quitarse. Tras ejecutar el comando, se solicitará la contraseña para el usuario[43].

43 Aunque en los ejemplos prácticos que se proponen a continuación no es utilizada esta utilidad, pueden encontrarse ejemplos de su utilización en el libro *Servicios de Internet* de los mismos autores.

— **htdigest**: Genera e inserta en un fichero plano los usuarios Web y sus contraseñas. Es utilizado cuando se realiza autenticación de tipo *Digest*, basado en fichero, *AuthDigestProvider file*. Su sintaxis más básica es la siguiente:

```
htdigest [-c] ruta-fichero-digest “dominio-autenticacion|realm/Authname” nombre-usuario
```

La opción “**-c**”, *create*, solo debe utilizarse para crear el fichero en el momento de insertar el primer usuario Web. Para insertar el resto de usuarios debe quitarse. Importante advertir que esta utilidad requiere que se le especifique el dominio de autenticación donde será valida la cuenta a crear. Este valor se corresponde con el valor asignado a la directiva *AuthName*. Tras ejecutar el comando, se solicitará la contraseña para el usuario.

Para consultar la lista de usuarios que están registrados en un fichero, podemos hacer uso del comando more, o de un simple editor de textos. La eliminación de un usuario de la lista, implica el simple borrado de la línea correspondiente.

```
more ruta-fichero-digest
```

— **htdbm**: Genera e inserta en una base de datos (*BD*) los usuarios Web y sus contraseñas, lo que agiliza la busqueda y validación de usuarios en la autenticación Web en relación a la utilización de un fichero plano. Puede utilizarse tanto en autenticaciones de tipo *Basic*, como *Digest*, basadas en una base de datos, *AuthBasicProvider dbm* o *AuthDigestProvider dbm*. Su sintaxis más básica es la siguiente:

```
htdbm [-c] -TtipoBD ruta-BD nombre-usuario
```

La opción “**-c**”, *create*, solo debe utilizarse para crear el fichero en el momento de insertar el primer usuario Web. Para insertar el resto de usuarios debe quitarse.

La opción “-T”, *type*, es utilizada para indicar cual de los cuatro tipos de gestor de bases de datos va a utilizarse: DB | SDBM | GDBM | NDBM.

Para consultar la lista de usuarios que están registrados en una base de datos, haremos uso de la opción “**-l**”, *list*, pudiendo eliminar alguno de ellos mediante la opción “-x”.

```
htdbm -l -TtipoBD ruta-BD
htdbm -x -TtipoBD ruta-BD nombre-usuario
```

Dependiendo el sistema operativo bajo el cual este instalado Xampp, GNU/Linux o Microsoft Windows, las utilidades anteriores las encontraremos en */opt/lampp/bin* o *c:/xampp/apache/bin*.

Ejercicio práctico n.º 6

Configura Apache para dar servicio a un sexto sitio Web, con directorio raíz *htdocs/sitioweb6*, bajo la dirección IP que tenga asignada el equipo servidor en la Intranet, al igual que en los sitios Web anteriores, implementando un nuevo Host Virtual basado en nombre de dominio, *www.sitioweb6.es*. El sitio Web tendrá dos áreas restringidas, un documento HTML con información privada, *privado.html*, y un directorio con archivos multimedia, *descargas*, de tal forma que solo tengan acceso un grupo de usuarios registrados en Apache: *usuweb1*, *usuweb2* y *usuweb3*. Todas las cuentas de usuario se almacenarán en una base de datos llamada *usuarios.db* de tipo SDBM, ubicada en un subdirectorio explícitamente creado denominado seguridad. Por compatibilidad con todo tipo de clientes Web, la autenticación será de tipo básico. La página de inicio será *index.html*.

Dirección IP / Puerto	**Nombre Dominio**	**Directorio Raíz DocumentRoot**	**Áreas Restringidas**
192.168.1.1:80	www.sitioweb6.es	/opt/lampp/htdocs/sitioweb6 c:/xampp/htdocs/sitioweb6	sitioweb6/**privado.html** sitioweb6/**descargas**

Área restringida	**Base de datos de usuarios (SDBM)**	**Usuarios autorizados**
sitioweb6/**privado.html**	/opt/lampp/seguridad/usuarios.db	usuweb1, usuweb2
sitioweb6/**descargas**	c:/xampp/apache/seguridad/usuarios.db	Todos los usuarios. *valid-user*

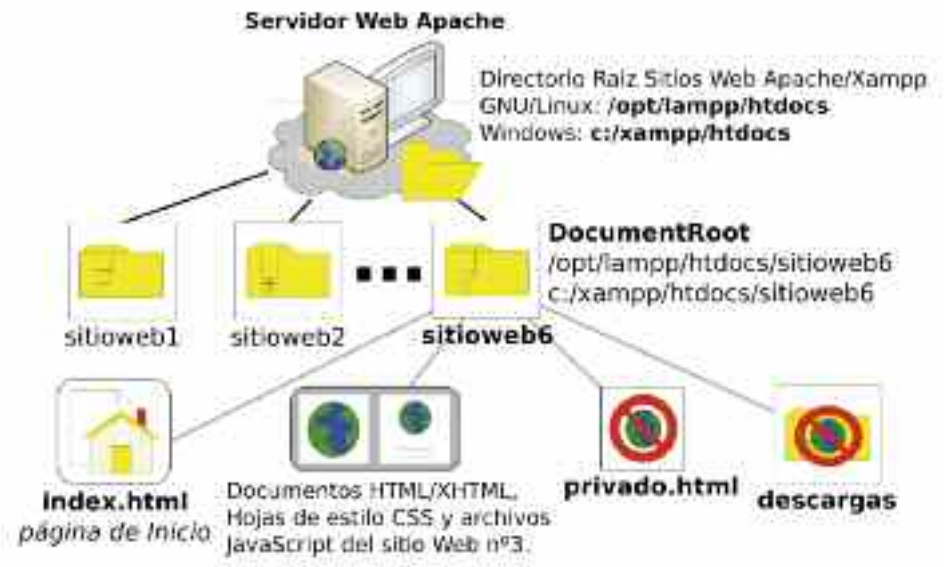


Figura 6.19. Estructura del sitio Web con áreas restringidas

Solución ejercicio n.º 6

Para resolver el ejercicio práctico propuesto, se aconseja seguir los siguiente pasos de configuración:

Paso nº1) El *root* o *administrador* del sistema creará las cuentas de usuario Web mediante la utilidad **htdbm**, tras crear el directorio *seguridad* donde guardaremos la base datos de los usuarios Web. Recordad que la opción "**-c**" sólo debe utilizarse para crear la base de datos, al insertar el primer usuario Web.

```
[root@linux]# /opt/lampp/bin/htdbm -c -TSDBM /opt/lampp/seguridad/usuarios.db usuweb1
[root@linux]# /opt/lampp/bin/htdbm -TSDBM /opt/lampp/seguridad/usuarios.db usuweb2
[root@linux]# /opt/lampp/bin/htdbm -TSDBM /opt/lampp/seguridad/usuarios.db usuweb3
C:\xampp\apache\bin> htdbm.exe -c -TSDBM c:/xampp/apache/seguridad/usuarios.db usuweb1
C:\xampp\apache\bin> htdbm.exe -TSDBM c:/xampp/apache/seguridad/usuarios.db usuweb2
C:\xampp\apache\bin> htdbm.exe -TSDBM c:/xampp/apache/seguridad/usuarios.db usuweb3
```

Paso n.º 2. El *root* o *administrador* del sistema añadirá la resolución del nombre *www.sitioweb6.es* en el fichero *hosts* e incluirá en la configuración de Apache un nuevo fichero auxiliar, *ejercicio-practico6.conf*, que contendrá el Host Virtual asociado al sitio Web con la declaración de las áreas restringidas.

```
# Resolución a incluir en el fichero hosts
192.168.1.1        www.sitioweb6.es
```

```
# Fichero de configuración de Apache httpd.conf: inclusión del fichero auxiliar.
Listen 80
NameVirtualHost 192.168.1.1
Include /opt/lampp/etc/extra/ejercicio-practico6.conf #  Inclusión en GNU/Linux
Include c:/xampp/apache/conf/extra/ejercicio-practico6.conf # Inclusión en Windows
LoadModule auth_basic_module modules/mod_auth_basic.so  # Modulo Autenticación Basic
LoadModule auth_digest_module modules/mod_auth_digest.so
LoadModule authn_dbm_module modules/mod_authn_dbm.so # Proveedor Autenticación dbm
... # Resto de directivas de configuración
```

```
# Contenido del fichero auxiliar de configuración de Apache: ejercicio-practico5.conf
<VirtualHost 192.168.1.1>
  ServerName www.sitioweb6.es # Nombre de dominio del sitio Web
  DocumentRoot /opt/lampp/htdocs/sitioweb6 # Directorio Raíz en GNU/Linux
  DocumentRoot c:/xampp/htdocs/sitioweb6 # Directorio Raíz en Windows
  <Files privado.html>
  <Files privado.html”>
        AuthName “Acceso Restringido – Zona Descargas” # Descripción
        AuthType Basic # Tipo de autenticación básica
        AuthBasicProvider dbm # Autenticación proveida por un DataBase Management
        AuthDBMUserFile /opt/lampp/seguridad/usuarios.db # Base datos de usuarios
        AuthDBMUserFile  c:/xampp/apache/seguridad/usuarios.db  # Base datos de usuarios
        Require user usuweb1 usuweb2  # Lista de usuarios autorizados
  </Files>
  <Directory /opt/lampp/htdocs/sitioweb6/descargas> # Directorio en GNU/Linux
  <Directory “c:/xampp/htdocs/sitioweb6/descargas”>  # Directorio en Windos
        AuthName “Acceso Restringido – Zona Descargas” # Descripción
        AuthType Basic # Tipo de autenticación básica
```

```
        AuthBasicProvider dbm # Autenticación proveida por un DataBase Management
        AuthDBMUserFile /opt/lampp/seguridad/usuarios.db # Base datos de usuarios
        AuthDBMUserFile  c:/xampp/apache/seguridad/usuarios.db  # Base  datos  de
usuarios
        Require valid-user  # Autorización para todos los usuarios de la base de datos
    </Directory>
</VirtualHost>
```

Paso n.º 3. El *root* o *administrador* del sistema reiniciará el servicio Apache o Xampp para que surtan efecto los cambios de configuración.

```
[root@linux]# /opt/lampp/lampp restart (reinicia todos los servicios Xampp en GNU/Linux)
C:\xampp> xampp_restart.exe (reinicia todos los servicios Xampp en Windows)
```

Paso n.º 4. Dado por hecho que se ha diseñado un sitio Web con la estructura mostrada en el enunciado del ejercicio práctico, comprobaremos el correcto funcionamiento del servicio Web, advirtiendo que para poder acceder a las áreas restringidas será necesario autenticarnos (*login y password*). Escribiendo en la barra de direcciones de nuestro cliente Web preferido las siguientes URL, debería suceder que:

— "***http://www.sitioweb6.es***", mostrará la página de inicio *index.html* del sitio Web 6.

— "***http://www.sitioweb6.es/privado.html***" (*o pinchando en un enlace de la página de inicio a privado.html*), aparecerá un ventana de dialogo solicitando un nombre de usuario y contraseña, que tras validarse mostrará la página *privado.html*.

— Al pinchar sobre cualquier enlace a un archivo ubicado en el directorio descargar , , aparecerá un ventana de dialogo solicitando un nombre de usuario y contraseña, que tras validarse, comenzará la descarga.

Figura 6.20. Diálogo de autenticación que aparece al pinchar sobre un enlace al contenido del directorio de descargas

Ejercicio práctico n.º 7

Modifica la configuración de Apache del ejercicio anterior, y utiliza autenticación de tipo Digest para garantizar que las contraseñas viajen entre el equipo cliente y el servidor de manera cifrada, con la finalidad de evitar posibles problemas de seguridad.

Solución ejercicio n.º 7

Para resolver el ejercicio práctico propuesto, se aconseja seguir los siguiente pasos de configuración:

Paso n.º 1. El *root* o *administrador* del sistema creara el fichero que contendrá las cuentas de usuarios Web mediante la utilidad **htdigest**.

```
[root@linux]# /opt/lampp/bin/htdigest -c /opt/lampp/seguridad/usuarios.digest permitidos usuweb1
[root@linux]# /opt/lampp/bin/htdigest /opt/lampp/seguridad/usuarios.digest permitidos usuweb2
[root@linux]# /opt/lampp/bin/htdigest /opt/lampp/seguridad/usuarios.digest permitidos usuweb3
C:\xampp\apache\bin> htdigest.exe -c c:/xampp/apache/seguridad/usuarios.digest permitidos usuweb1
C:\xampp\apache\bin> htdigest.exe c:/xampp/apache/seguridad/usuarios.digest permitidos usuweb2
C:\xampp\apache\bin> htdigest.exe c:/xampp/apache/seguridad/usuarios.digest permitidos usuweb3
```

Paso n.º 2. El *root* o *administrador* del sistema modificará el fichero auxiliar de configuración del ejercicio práctico anterior, *ejercicio-practico6.conf*.

```
# Contenido del fichero auxiliar de configuración de Apache: ejercicio-practico5.conf
<VirtualHost 192.168.1.1>
  ServerName www.sitioweb6.es # Nombre de dominio del sitio Web
  DocumentRoot /opt/lampp/htdocs/sitioweb6 # Directorio Raíz en GNU/Linux
  DocumentRoot c:/xampp/htdocs/sitioweb6 # Directorio Raíz en Windows
  <Files privado.html>
  <Files privado.html>
        AuthName “permitidos” # Debe coincidir con el Realm indicado en htdigest
        AuthType Digest # Tipo de autenticación básica
        AuthDigestProvider file # Autenticación validada contra un fichero
        AuthUserFile /opt/lampp/seguridad/usuarios.digest # File Usuarios
        AuthUserFile c:/xampp/apache/seguridad/usuarios.digest # File usuarios
        Require user usuweb1 usuweb2 # Lista de usuarios autorizados
  </Files>
  <Directory /opt/lampp/htdocs/sitioweb6/descargas> # Directorio en GNU/Linux
  <Directory “c:/xampp/htdocs/sitioweb6/descargas”>  # Directorio en Windos
        AuthName “permitidos” # Debe coincidir con el Realm indicado en htdigest
        AuthType Digest # Tipo de autenticación básica
        AuthDigestProvider file # Autenticación validada contra un fichero
        AuthUserFile /opt/lampp/seguridad/usuarios.digest # File Usuarios
        AuthUserFile c:/xampp/apache/seguridad/usuarios.digest # File usuarios
        Require valid-user  # Autorización para todos los usuarios de la base de datos
  </Directory>
</VirtualHost>
```

Paso n.º 3. Por último, al igual que en el resto de ejercicios prácticos, el *root* o *administrador* del sistema reiniciará el servicio Apache o Xampp para que surtan efecto los cambios de configuración, y comprobará su correcto funcionamiento desde un cliente Web.

Capítulo 7

PHP

El diseño y desarrollo de las aplicaciones Web actuales (*Web 2.0*) no sería posible sin la utilización de algún lenguaje que permitiera ejecutar tareas en el servidor: consultas sobre una base de datos, gestión de los servicios ofrecidos por el servidor (correo electrónico, transferencia de ficheros, ...), etc. Tal como se indicó en el capítulo referente a Apache, actualmente todo sitio Web dispone de formularios para el establecimiento de una comunicación entre el cliente y servidor, y tiene asociado una base de datos donde se almacenan los datos y muchos de los elementos que forman parte de los documentos que son servidos. Es decir, con las herramientas de diseño vistas hasta ahora, HTML, CSS, JavaScript y SVG, es posible implementar una Web atractiva, con aspectos dinámicos en el lado del cliente, pero imposible interactuar con el servidor. Lenguajes de programación como PHP, ASP, Python, o JSP, permiten desarrollar potentes aplicaciones Web transparentes para el usuario final, al ser ejecutado todo el código en el servidor.

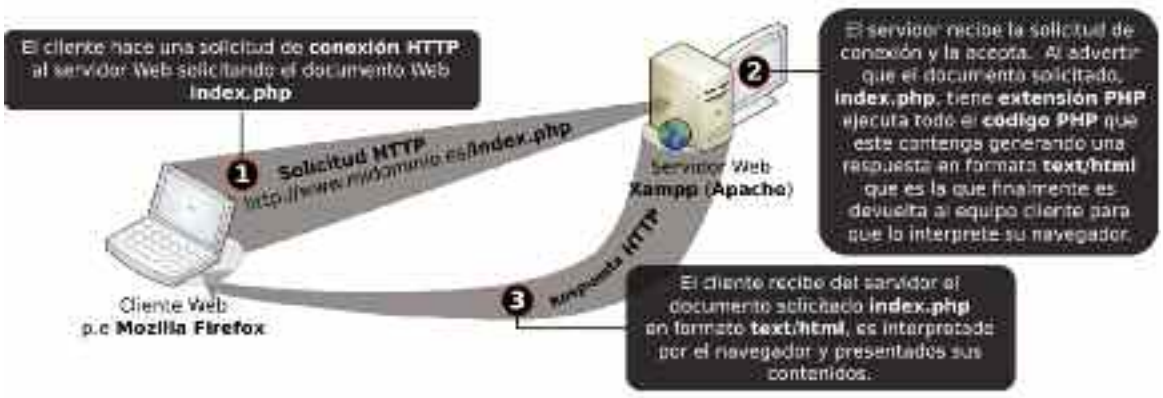


Figura 7.1. Comunicación entre un cliente y un servidor al solicitar un documento PHP

A modo de ejemplo, puede observarse en en la figura X, como al solicitar al servidor un documento PHP, este ejecuta el código PHP que este incluya[44], generando un documento en formato *text/html* que es el que finalmente es entregado al cliente Web, siendo transparente para este si el documento original hacia uso de un lengua u otro.

Antes del desarrollo de estos lenguajes de programación orientados al mundo Web, la única forma de programar tareas en el servidor que permitirán gestionar el servidor vía protocolo

44 Los lenguajes PHP, ASP, Python, ... son denominados lenguajes de servidor al no poder ser interpretados por un navegador Web. En el lado del servidor, Apache necesitará un módulo externo que le permita interpretar éstos lenguajes y generar un resultado interpretable por el cliente Web.

HTTP era haciendo uso de los clásicos lenguajes de programación pensados para el desarrollo de software de sistemas o de aplicación, tales como el lenguaje C o Perl. A estas aplicaciones desarrolladas se les denominaba scripts CGI (*Common Gateway Interface*), al hacer de interface o intermediarias entre el equipo servidor y el mundo Web[45].

En el presente capítulo se presentará a uno de los lenguajes de servidor más utilizado en el desarrollo de aplicaciones Web dinámicas, PHP, haciendo especial incapie en el uso de formularios HTML para el intercambio de información entre el cliente y el servidor.

1. CONTENIDOS DEL CAPÍTULO

El presente capítulo presenta la siguiente estructura:

— En primer lugar se indica el software necesario para la implementación de los ejercicios prácticos propuestos a lo largo del capítulo.

— Después se muestran las características más relevantes del lenguaje de programación PHP: tipos de variables, operadores, estructuras de control de flujo y bucles de iteración y funciones más importantes.

— A continuación, se presenta la herramienta de diálogo más habitual entre cliente y servidor, los formularios HTML, y la forma en que PHP gestiona los datos enviados por estos al servidor.

— Se presentan los eventos JavaScript que están relacionados con los formularios HTML: *onsubmit*, *onreset*, *onselect*, *onchange*.

— El capítulo termina mostrando la importancia del control de sesiones y cookies en PHP.

2. SOFTWARE DE DESARROLLO

En relación al software necesario para desarrollar aplicaciones en lenguaje PHP, al menos necesataríamos de los siguiente programas:

— Interprete PHP: En el caso de que hayamos instalado *Apache* bajo el paquete integrado *Xampp* en el servidor, no tendremos que preocuparnos por nada, al venir con el interprete PHP ya configurado. En caso contrario, si hemos optado por instalar Apache de manera independiente, será necesaria la instalación del interprete PHP en el equipo servidor de forma explicita: (a) En GNU/Linux será necesario instalar el paquete *apache-mod_php* que venga con la propia distribución y (b) en Windows descargar el interprete PHP de sitios Web como softonic (http://php.softonic.com).

— Editor PHP: Aunque cualquier editor de textos nos serviría a la hora de redactar código PHP, con la finalidad de facilitar la edición de los documentos PHP es aconsejable hacer uso de algún editor que disponga de algún reconocedor sintáctico que nos informe de alguna forma (*p.e. mediante colores*) de si lo que estamos escribiendo es PHP. En el caso de estar trabajando bajo GNU/Linux, existen multitud de herramientas de software libre para este propósito, destacando a *Quanta Plus* o al editor *Kate*. En Windows, la mayoría de los editores con prestigio son shareware {nota pie pagina: El software shareware, a diferencia del freeware, se caracteriza por distribuirse bajo una licencia de evaluación que caduca al cabo de un determinado tiempo, siendo necesaria la adquisición de una licencia para poder seguir trabajando.}, cabiendo destacar a *Adobe Dreamweaver*.

---

45 En el libro de *Instalación y Mantenimiento de Servicios de Internet* de los mismos autores puede encontrarse una información más detallada junto con ejemplos sobre los Scripts CGI.

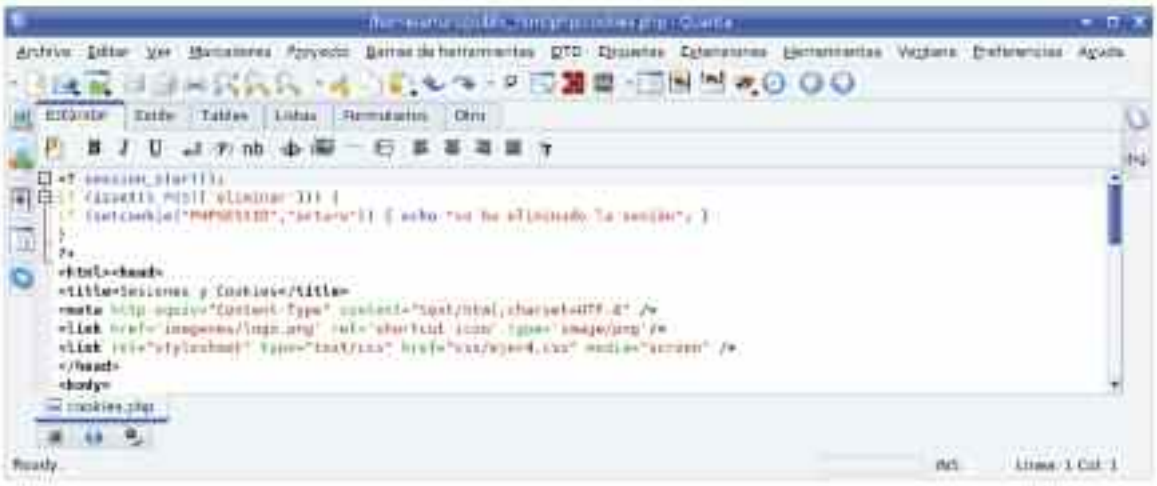


Figura 7.2. Los editores HTML/PHP nos facilitan la creación de documentos Web

3. LENGUAJE PHP

El lenguaje PHP es uno de los lenguajes de programación más usado en el mundo. Entre las características más reseñables de este lenguaje de programación frente a otros cabría señalar las siguientes:

— Lenguaje de programación *pensado expresamente para el desarrollo de sitios Web dinámicos*. Como ya se ha comentado en la introducción al capítulo, antes de que fueran desarrollados este tipo de lenguajes de programación, la única opción era desarrollar scripts CGI mediante la utilización de alguno de los lenguajes de programación disponibles (C, Perl, Pascal, ShellScripts, etc.), en los cuales se detectaban multitud de carencias, siendo necesario muchísimas líneas de código para implementar pequeñas aplicaciones, que con los actuales lenguajes de programación se pueden realizar de una forma mucho más sencilla.

— *Embebido en HTML*. El código PHP es introducido en los documentos HTML de manera directa mediante la utilización de una etiqueta especifica, **<?** *código PHP* **?>** o **<?php** *código PHP* **?>**. Para advertir al servidor que nuestros documentos hacen uso de este tipo de etiquetas tan sólo será necesario modificar su extensión a "*.**php**", lo que hará que el servidor Web, antes de servir estas páginas a un cliente sean inspeccionadas en búsqueda de estas etiquetas, ordenar a su interprete PHP que ejecute el código, y su resultado en formato text/html, entregarlo al cliente (*ver figura X*).

— *Software Libre*. Al distribuirse bajo licencia GPL (*Licencia Pública General*), esta disponible su código fuente, pudiendo ser modificado y mejorado por la comunidad, para ser usado bajo la finalidad que uno requiera sin la necesidad de compra de ninguna licencia. Esto además favorece que los errores que se detecten por la comunidad de usuarios sean más fácilmente resueltos. Otros lenguajes de programación como ASP (*Active Server Pages*), no son de uso libre, aspecto que van acusando en popularidad, pasando a ser una segunda alternativa en el desarrollo de aplicaciones Web frente a PHP. En el caso concreto de ASP, propiedad de Microsoft, ha sido desarrollado para que su código sea ejecutado bajo el servidor Web del mimo propietario, IIS (*Internet Information Server*), lo cual aún agrava más su elección.

— Multiplataforma y alta portabilidad. Su interprete esta disponible para multitud de sistemas operativos como GNU/Linux, Microsoft Windows, UNIX o MacOS, garantizando que los programas o scripts realizados en PHP bajo cualquiera de estas plataformas sea fácilmente portable a otra.

— Modular. Dispone de multitud de módulos compuestos por variopintas funciones que permiten llevar a cabo todo tipo de tareas en PHP. Existen módulos para el tratamiento de

imágenes, manipulación de ficheros, generación de documentos PDF, conexión con bases de datos o comunicación con todo tipo de servicios (FTP, SMTP, etc.), entre otras muchas funcionalidades.

— Se integra a la perfección junto al software servidor Web Apache y el gestor de bases de datos MySQL, siendo una solución perfecta para el desarrollo de las actuales sitios Web 2.0. Un ejemplo de ello es Xampp.

— La curva de aprendizaje del lenguaje PHP se caracteriza porque con muy poco esfuerzo , en poco tiempo, es posible desarrollar aplicaciones Web de mediana envergadura, gracias a que su sintaxis es muy clara, directa e intuitiva.

Si a todo lo anterior le sumamos la gran cantidad de documentación, tutoriales y manuales disponibles en la red, hacen de PHP una de las elecciones más habituales frente a otros lenguajes de programación como son ASP o JSP.

A continuación se describirá brevemente el lenguaje de programación PHP para poder tener una visión rápida de como usarlo.

3.1. Comentarios en PHP

Para introducir comentarios dentro de los fragmentos de código o scripts PHP, que ayuden a su posterior mantenimiento se hará uso de los siguientes caracteres:

— **//** *comentario*: Comentario de una sola línea.

— **/*** *comentarios* ***/**: Comentarios de varías líneas.

Según lo anterior, aquellas líneas que estén precedidas por "**//**", o se encuentre entre "**/*** ***/**", serán ignoradas por el intérprete de PHP.

3.2. Tipos de variables y constantes en PHP

Las variables PHP se distinguen por estar precedidas del carácter dólar, "$". Este distintivo permite al interprete PHP reconocer fácilmente donde se hace uso de ellas. En relación a aspectos vistos en otros capítulos, a diferencia de JavaScript, PHP permite hacer uso de variables no tipificadas, sin distinguir entre variables de tipo entero (*Integer*), real (*Double*) o cadena de caracteres (*String*). Esto significa que al definir una variable no es necesario indicar el tipo de variable en función del tipo de dato que va a ser almacenado en ella. Es decir, podemos asignar un numero entero a una variable, y a continuación, a esa misma variable una cadena de caracteres.

Otras variables PHP interesantes son los vectores o *arrays*, los cuales nos permiten crear tablas de valores haciendo referencia a cada uno de ellos mediante el nombre del *array* y su indice o posición. Ejemplos:

Tipo Variable	Ejemplos
Number Double String	$var = 12345678; $var = 12345.678; $var = "12345678";
Array	$lista = **array** ("alumno1" => 5.2,"alumno2" => 3.6,"alumno3" => 8.4, ...); $lista['alumno4'] = 7.1; $nombre = "alumno5"; $lista[$nombre] = 2; $bidimensional[$fila][$columna] = "Europa";

En el caso de querer definir una constante, será necesario hacer uso de la función PHP **define**(“nombre”, “valor). Por ejemplo, si quisiéramos definir una constante para relacionar la cotización actual entre el euro y el dolar:

```
<? define (“EuroDolar”,1.3523);
define (“DolarEuro”,1/1.3523); ?>
```

¡¡Importante!! PHP distingue entre mayúsculas y minúsculas en la definición y utilización de variables ($a › $A). Además, toda línea de código PHP debe terminar en el carácter punto y coma, “**;**”.

3.3. Operadores aritméticos, de comparación, lógicos, incremento y concatenación

Tipo Operador	Operadores	Ejemplos
Aritméticos	**+ - * /**: suma, resta, producto, división. (**int**)(n1 / n2): cociente entero. n1 **%** n2: resto de la división. **sqrt**(n): raíz cuadrada. **pow**(n1,n2): potencia ($num1^{num2}$). **round**(n): redondeo. **floor**(n): redondeo por defecto. **ceil**(n): redondeo por exceso.	$n1 = 9; $n2 = (**int**)($n1 / 2); $n3 = ($n1 + 30.4) * $n2; $n4 = **round** ($n1 / **sqrt** ($n3));
Comparación	==, !=: igual que, distinto que. <, >, <=, >=: menor que, mayor que, menor o igual que, mayor o igual que.	$edad >= 18 $nombre == “Maria”
Lógicos	&&, ||, !: AND, OR y negación.	($edad >= 18) && ($nombre == “Maria”)
Incremento	++, —: incrementa y decrementa en una unidad. +=n, -=n: incrementa y decrementa en n unidades-	++$edad; $edad++; $sueldo+=1000;
Concatenación	.: concatena cadenas de caracteres.	$saludo = “Bienvenido “.$nombre;

3.4. Funciones PHP más importantes

El lenguaje PHP se caracteriza por disponer de funciones asociadas a módulos que permiten realizar prácticamente cualquier tipo de tarea programada. Tratar de listar aquí todas las funciones disponibles sería impensable. Con la única pretensión, de conocer las más básicas y habituales, podrían resumirse de la siguiente forma:

Impresión de mensajes por pantalla:	
echo *mensaje*;	Formatea el mensaje indicado en formato *text/html*, cuyo resultado será entregado al cliente Web como respuesta.

Ejemplos:

```
$nombre = “Juanjo Martín”;
$premio = 10000.82;
echo “Enhorabuena Sr. “.$nombre.” ha ganado “.$premio;
echo “<br />Redondeando son”.round($premio);
```

¡¡Observación!! Para poder probar todos los ejemplos que se muestran, configura en apache un nuevo VirtualHost editando el fichero httpd.conf, e inserta en su pagina de inicio (*.php) todos los trozos de código PHP que desees. Recuerda que el código PHP debe ir encerrado entre las etiquetas <? *codigo-php;* ?> o <?php *codigo-php;* ?>.

```
<VirtualHost *>
  ServerName www.ejemplos.php # Debe definirse en el fichero hosts
  DocumentRoot /opt/lampp/htdocs/ejemplosphp # Directorio Raíz en GNU/Linux
  DocumentRoot c:/xampp/htdocs/ejemplosphp # Directorio Raíz en Windows
  DirectoryIndex misejemplos.php  # Página de inicio del sitio Web
</VirtualHost>
```

Tras reiniciar el servicio Apache/Xampp, podrás ver los resultados poniendo en la barra de direcciones de tu navegador preferido:

http://www.ejemplos.php o http://www.ejemplos.php/misejemplos.php.

Funciones de comprobación de variables:	
isset ($var)	Comprueba si la variable indicada existe. Devuelve un valor booleano (*1 o 0*, *true o false*).
unset($var)	Elimina la variable indicada. Devuelve un valor booleano informando de si hubo éxito.
is_int() **is_float**() **is_string**() **is_array**()	Funciones que permiten comprobar si el valor indicado es un entero, un real, una cadena de caracteres o un *array*. Devuelven un valor balor booleano.

Ejemplos:

```
$saludo = "hola";
echo "<br />Comprobamos si saludo existe: ".isset($saludo)." y si es string ".is_string($saludo);
unset($saludo);
echo "<br />Comprobamos si la variable saludo existe: ".isset($saludo);
```

¡¡Atención!! Las funciones booleanas, junto con alguna estructura condicional e iterativa (*if/else/elseif, while, foreach, etc.*) que veremos más adelante, son utilizadas para controlar el flujo de ejecución de las instrucciones dentro de los bloques de código o scripts PHP que incluyamos en nuestros documentos Web. Puede observarse que el valor devuelto por estas es 1 (*true*) en caso afirmativo o un valor vacío o nulo (*false*) en caso contrario.

Funciones que trabajan con cadenas de caracteres:	
strlen (cadena); **strolower** (cadena); **stroupper** (cadena);	Devuelven la longitud de la cadena, la cadena en minúsculas, o mayúsculas respectivamente.
ucfirst(cadena); **ucwords**(cadena);	Devuelven la cadena con el primero de sus caracteres en mayúsculas, o el primero de cada una de las palabras que lo forman en mayúsculas.
strcmp(cad1,cad2);	Compara las dos cadenas de caracteres pasadas como parámetros, y devuelve un valor nulo en caso de ser idénticas. En caso contrario, devuelve un -1, en caso de que la primera cadena le preceda alfabéticamente, o un 1, si es al revés.

strpos(cadena,patrón); **strrpos**(cadena,patrón);	Buscan el patrón en la cadena indicada, y devuelve la primera posición en que fue encontrado. En el caso de no encontrarse devolverá *false*. Dependiendo de si la búsqueda la queremos hacer de principio a fin o al revés (*reverse*), utilizaremos *strpos()* o *strrpos()* respectivamente. Destacar que para saber si se encontró o no el patrón especificado se utiliza un condicional haciendo uso de los operadores === o !==.
substr(cadena,n,m); **substr**(cadena,m);	Extrae *m* caracteres a partir de la posición indica por *n*. Si *n* es positivo la posición se cuenta de izquierda a derecha, y si negativo al revés. En caso de no indicar n, los m caracteres se extraerán contados desde el principio o final de la cadena, dependiendo de si es positivo o negativo su valor.
explode (sep,cadena,n); **implode** (sep,*array*);	La función *explode()* devuelve un *array* formado por un máximo de **n** posiciones resultado de trocear la cadena indicada por aquellas posiciones donde se encuentre el carácter separador indicado (*sep*). Por el contrario, *implode()* genera una cadena formada tras unir los elementos que forman parte del *array* indicado, separándolos por el carácter *sep*.
chop (cadena);	Elimina los espacios en blanco al final de la cadena.

Ejemplos:

```
$fichero = “documento.php”;
$nombre_fichero = substr ($fichero,0,strlen($fichero) – 4);
$extension = substr ($fichero,-3);
echo “<br />El fichero “.$fichero.” tiene como nombre “.$nombre_fichero.” y extensión “.$extension;
echo “<br />Mi nombre es “.ucwords (“arturo martin romero”);
$trozos = explode (“:”,”arturo:martin:romero”,3);
echo “<br />El primer trozo de la cadena anterior es: “.ucfirst($trozos[0]);
if ( strpos ($trozos[0], “art”) !== false ) { echo “<br />El primer trozo contiene la cadena art”; }
```

Funciones con *arrays*:	
count($array)	Indica el número de elementos que componen el vector.
in_array(valor,$array)	Función booleana que devuelve true o false (1 o 0), dependiendo de si encuentra o no el valor en el array pasados como parámetros a la función.
unset($array[pos]);	Borra el elemento del *array* de la posición indicada. En caso de no indicar posición elimina todo el *array*. Devuelve un valor booleano en función de si tuvo éxito o no.
sort($array) **rsort**($array) **ksort**($array) **krsort**($array) **reset**($array)	Funciones de ordenación de los elementos de un *array*. La función *sort()* ordena alfabeticamente los elementos del *array*. Por contra, *rsort()* lo ordena de manera inversa (*reverse*). Las funciones *ksort()* y *krsort()* ordenan igualmente el *array* pero por índice o *key*. Resultan útiles cuando los índices de los vectores no son numéricos. Tras realizar la ordenación de todo vector es conveniente resetear el puntero interno del *array* mediante la función *reset()*.
array_push() **array_pop**()	La función *array_push()* añade nuevos elementos al final del *array*, mientras que *array_pop()*, extrae su último elemento eliminándolo del *array*.

Ejemplos:

```
$lista1 = array (“mara”,”juanjo”,”hugo”);
echo “El número de elementos de la lista es: “.count($vector1);
sort($lista1);
reset($lista1);
echo “El último elemento de la lista ordenada es: “.array_pop($lista1);
echo “El número de elementos de la lista ahora es: “.count($vector1);
```

```
unset($lista1[1]);
echo "Tras borrar el segundo elemento de la lista, el numero de elementos es: ".count($vector1);
array_push ($lista1,"arturo","andrea");
echo "Tras añadir 2 nuevos elementos el número total es: ".count($vector1);
$lista2 = array ("b" => "mara", "a" => "juanjo", "c" => "hugo");
ksort ($lista2);
reset ($lista2);
echo "Tras la ordenación, el último elemento de la lista asociativa es: ".array_pop($lista2);
unset($lista2['a']);
echo "Tras extraer un elemento, y borrar el asociado al índice a quedan: ".count($vector1);
```

Funciones con fechas y horas:	
date(*param*)	Suministra información sobre la fecha y hora actual en el servidor. Es necesario pasarle al menos un parámetro correspondiente al dato a conocer: **l** (*nombre completo del día de la semana*), **j** (*dia del mes*), **n** (*mes*), **Y** (*año con cuatro dígitos*), **H** (*hora en formato 24h*), **i** (*minutos*), **s** (*segundos*), etc.
time()	Devuelve el número de segundos transcurridos desde el 1 de enero de 1970.
mktime() **strftime**() **setlocale**()	Al igual que *date*(), *mktime*() nos permite obtener la fecha y hora del servidor, siendo necesaria la función *strftime*() para darle un formato. Para establecer el formato es necesario hacer uso de una lista de variables predeterminadas: **%A** (*nombre completo del día de la semana*), **%d** (*día del mes*), **%m** (*mes*), **%Y** (*año con cuatro dígitos*), **%H** (*hora en formato 24h*), **%M** (*minutos*), **%S** (*segundos*), etc. Por defecto los nombres se suministran en inglés. En caso de que queramos que se muestren en español, será necesario utilizar la función *setlocale*().

Ejemplos (fecha en formato "*La fecha actual es 6-4-2009.10:11*"):

```
echo "<br />La fecha actual es ".date(j)."-".date(n)."-".date(Y)."."date(H).":".date(i);
$fecha = mktime();
setlocale (LC_TIME,"es_ES");
echo strftime ("<br />La fecha actual es %A, %d-%m-%Y.%H:%M",$fecha);
```

Funciones para el manejo de ficheros:	
$puntero=**fopen**(*ruta*,*modo*)	Abre el el fichero especificado en *ruta* en el *modo* indicado: • **r**, lectura. Sitúa el puntero del fichero al comienzo del mismo. • r+, lectura y escritura. Sitúa el puntero al comienzo. • w, sólo escritura. Sitúa el puntero del fichero al comienzo eliminando su posible contenido. Si no existe trata de crearlo (*deben concederse permisos*). • w+, lectura y escritura. Sitúa el puntero del fichero al comienzo eliminando su posible contenido. Si no existe trata de crearlo. • a, sólo escritura. Sitúa el puntero del fichero al final (*añade*). Si no existe trata de crearlo. • a+, lectura y escritura. Sitúa el puntero del fichero al final (*añade*). Si no existe trata de crearlo. Devuelve un puntero o manejador del fichero para poder hacer referencia a él: <? $puntero = **fopen**("/tmp/mifichero.txt","a+"); ?>
fclose($puntero)	Cierra un fichero que previamente haya sido abierto.
fread($puntero,*cantidad*) **fwrite**($puntero,*texto*)	Funciones utilizadas para la lectura y escritura en ficheros. La primera requiere la especificación de la cantidad de bytes a leer, y la segunda, el texto a añadir.

ftell($puntero) **fseek**($puntero,*posicion*) **rewind**($puntero)	Funciones relacionadas con el puntero de lectura y escritura de ficheros. *ftell*() informa de la posición del puntero, *fseek*() lo sitúa en la posición indicada y *rewind*() lo coloca al comienzo del fichero, respectivamente.
filesize(*ruta*)	Devuelve el tamaño del archivo especificado.
$array=**file**(*ruta*)	Devuelve una lista o *array*, donde cada una de sus componentes almacena cada una de las líneas del fichero especificado en ruta. No requiere su previa apertura, *fopen*(), ni posterior cierre, *fclose*().
file_exists(*ruta*)	Informa con un valor booleano (*true o false, 1 o 0*) si el fichero indicado existe.
is_uploaded_file(nombre) **move_uploaded_file**(*origen, destino*)	Informa con un valor booleano si el archivo especificado fue subido al servidor a través de un formulario HTML vía POST. Mueve un fichero subido al servidor al directorio especificado.
copy(*fichero1*,*fichero2*) **rename**(*fichero1*,*fichero2*) **unlink**(*ruta*)	Funciones para copiar, renombrar y borrar archivos.
disk_total_space (*dir*) **disk_free_space**(*dir*)	Devuelve el tamaño en bytes del directorio indicado. Devuelve el número de bytes de espacio libre dentro de la partición del disco asociada al directorio pasado como parámetro.

Ejemplos :

```
$espacio_libre = disk_free_space ("/"); // disk_free_space("C:"); en Windows
echo "<br />El espacio libre disponible es ".$espacio_libre;
$fichero = "/opt/lampp/htdocs/web/alumnos.txt";
if ( file_exists ($fichero) ) {
echo "<br />El tamaño del fichero ".$fichero." es ".filesize ($fichero). "bytes";
echo "<br />Las lineas que lo componen son:";
$lineas = file ($fichero);
foreach ( $lineas as $num_linea => $contenido_linea )
{ echo "<br />".$num_linea.": ".$contenido_linea; }
if ( unlink ($fichero) ) { echo "<br />El fichero ".$fichero." ha sido borrado ..."; }
}
```

Junto a la lista interminable de funciones PHP disponibles, el programador puede diseñar sus propias funciones, tan como sucedía en JavaScript:

```
function nombre_función (parámetros) {
  instrucciones;
  return $resultado;
}
```

Estas funciones pueden ser definidas en un documento independiente, con la finalidad de poder ser reaprovechadas por otros documentos, favoreciendo de esta forma la reutilización del código PHP, importándolo mediante la directiva **include**.

Ejemplo:

```
// Contenido del documento Web
<? include "mis_funciones.php"; ?>
<html><head>...</head><body> ... <? echo "<br />".fechayhora("2"); ?> ... </body></html>
```

```
<? // Contenido del fichero mis_funciones.php
function fechayhora ($tipo_formato) {
$fecha = mktime();
setlocale (LC_TIME,"es_ES");
if ($tipo_formato == "1")
{ return ( strftime ("Hoy es %A %d - %B - %Y",$fecha) ); }
if ($tipo_formato == "2")
{ return ( strftime ("Son las %H:%M:%S del %d - %B - %Y",$fecha) ); }
} ?>
```

3.5. Estructuras de control y bucles en PHP

Las estructuras de control y bucles iterativos en PHP son equivalentes a las disponibles en otros lenguajes de programación como las vistas en JavaScript.

— **if elseif else**:

Estructura de control que ejecuta un conjunto de instrucciones en función de si la condición indicada se cumple o no.

```
if ( condición ) { instrucciones; }
elseif (condición)  { instrucciones; }
else { instrucciones; }
```

Ejemplo:

```
$mipassword="12345";
if ( strlen($mipassword) <= 6) { echo "La contraseña introducida es demasiado corta."; }
```

— **switch case**:

Estructura de control condicional múltiple. Encauza el flujo de ejecución del código PHP hacía el caso que coincida con la expresión planteada.

```
switch (expresión) {
case valor1: { instrucciones;  break; }
case valor2: { instrucciones;  break; }
...
default: { instrucciones; }
}
```

Ejemplo:

```
$puesto="secretario";
switch ($puesto) {
  case "director": { $sueldo = 2000; break; }
  case "secretario": { $sueldo = 1800; break; }
  case "administrativo": { $sueldo = 1500; break; }
  default: { echo "<br />El puesto de ".$puesto." no esta dado de alta."; }
}
echo "El puesto de ".$puesto." tiene un sueldo de ".$sueldo." euros.";
```

— **foreach**:

Bucle iterativo pensado para la exploración de las componentes de un vector, *array* o lista de valores. En concreto, da tantas vueltas al bucle como componentes tiene el *array*, asignando en cada una de ellas a las variables índice y valor, los correspondientes valores que tiene este.

```
foreach ($array as $indice => $valor) { instrucciones; }
```

Ejemplo:

```
$lista = array (“alumno1” => 5.2,”alumno2” => 3.6,”alumno3” => 8.4);
echo “Las notas de los alumnos son: “;
foreach ($lista as $indice => $valor) {
  echo “<br />La componente del vector “.$indice.” tiene el valor: “.$valor.”.”;
}
```

— **while** y **do while**:

Estos bucles ejecutan el código que encierran hasta que deja de cumplirse la condición. La diferencia entre *while* y *do while*, es que en el primer caso solo se entrará al bucle y se ejecutarán las instrucciones PHP que incluya en el caso de que se cumpla la condición, mientras que en el segundo caso, se ejecutará al menos una vez, y si tras cumplir la condición esta se cumple se volverán a ejecutar hasta que deje de cumplirse.

```
while (condición) { instrucciones; }
do { instrucciones } while (condición);
```

Ejemplo:

```
$lista_participantes = array (“Julia Gonzalez”,”Alberto Herrero”,”Carla Diaz”);
echo “<br />La lista de participantes es:”;
$contador = 0;
while ( $contador < count($lista_participantes))
{ echo “<br />El índice es “.$contador.” y su valor “.$lista_participantes[$contador].”.”;
  $contador++;
}
```

— **for**:

Bucle iterativo que ejecuta las instrucciones que encierra mientras la condición de finalización no se produzca. Hasta cumplir dicha condición, partiendo de una condición inicial, tras dar una vuelta al bucle se ejecutará la instrucción de iteración hasta que se verifique.

```
for (condición inicial; condición de finalización del bucle; instrucción tras cada iteración)
{ instrucciones; }
```

Ejemplo:

```
$lista_participantes = array (“Julia Gonzalez”,”Alberto Herrero”,”Carla Diaz”);
echo “<br />La lista de participantes es:”;
for ($contador = 0; $contador < count($lista_participantes); $contador++)
{ echo “<br />El indice es “.$contador.” y su valor “.$lista_participantes[$contador].”.”; }
```

4. FORMULARIOS HTML Y PHP

Tras ver en el primer capítulo del libro prácticamente todos los elementos HTML que pueden componer un documento Web, llega ahora el momento de introducir algunos de los que dejemos pendientes, los formularios HTML. Estos son la herramienta clásica utilizada para poder enviar información desde un cliente a un servidor Web. Aunque su uso más habitual es el de interrogar al usuario para obtener información en operaciones de registro, también son utilizados en tiendas virtuales a través de Internet[46] , compra de billetes y reserva de estancias vía Web[47], aplicaciones de correo electrónico vía Web (*Webmail*), o ventanas de dialogo para la ejecución de tareas en el lado del servidor, entre otras utilidades. Un ejemplo de un formulario muy conocido por todo el mundo es el utilizado por *Google*, donde mediante una caja de introducción de texto y un botón de envió, el usuario puede realizar búsquedas de información personalizadas.

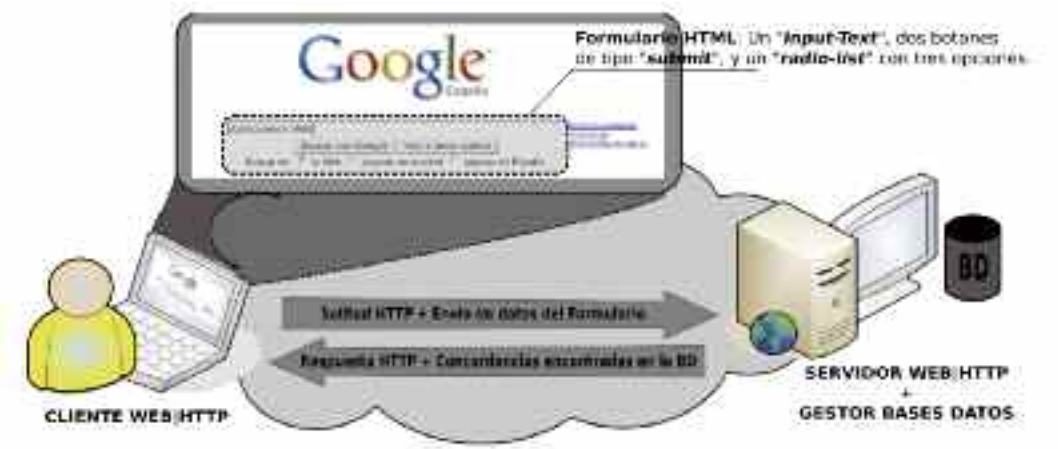


Figura 7.3. Google es el ejemplo más cotidiano del uso de formularios HTML para comunicarnos con un servidor Web

Como puede observarse en la figura 7.3, tras introducir las palabras claves de búsqueda en el formulario de la página de inicio de *Google*, "*http://www.google.es*", estas son enviadas al servidor, para que este busque en sus bases de datos ubicaciones en Internet que las contengan, reflejando la importancia de los formularios en la comunicación entre cliente y servidor.

En definitiva, para que esta comunicación entre una aplicación cliente (por ejemplo, Mozilla Firefox) y un aplicación servidora (por ejemplo, Apache) sea posible, es necesario que el cliente haga uso de un documento Web que incorpore un formulario HTML, y el servidor una aplicación, programa o script que este preparado para recoger los datos enviados por el formulario desde el cliente, y actuar en consecuencia. Estas aplicaciones del lado del servidor inicialmente fueron *scripts CGI*[48] programados mediante la utilización de lenguajes típicos de programación como *C* o *Perl*, aunque actualmente son utilizados lenguajes desarrollados explícitamente para la implementación de aplicaciones Web como *PHP*, *ASP* o *JSP*.

---

46 La compra de libros, material informático o deportivo, son tan sólo algunos ejemplos del gran abanico de productos que pueden adquirirse a través de Internet.

47 Adquisición de billetes de tren, avión o reserva de hoteles y casas rurales vía Web, es una de las formas más cómodas de prepararse uno mismo las vacaciones.

48 En el libro de *Servicios de Internet* de estos mismos autores, puede encontrarse información más detallada sobre scripts CGI. Aquí únicamente se tratarán scripts PHP.

La etiqueta HTML utilizada para introducir un formulario en el documento Web es **<form></form>**, la cual encierra todos los elementos que formarán parte de él:

```
<form id class style lang|xml:lang name action method accept enctype target >
  Elementos del Formulario
</form>
```

En relación a sus atributos, "***id***" identifica al formulario para que pueda ser referenciado desde los scripts *JavaScript* definidos en el propio documento (*<script> </script>*), "***class***" lo clasifica para poder asignarle un estilo definido en alguna de las hojas de estilo CSS que tenga asociadas el documento, "***style***" nos permite asignar una lista de propiedades de estilo CSS separadas estas por ";", "***lang|xml:lang***" establece el lenguaje utilizado en el formulario, destacando al resto su importancia y por ser algunos de ellos propios de esta etiqueta:

— ***name***: etiqueta al formulario permitiendo ditinguirlo entre los distintos formularios que se pueden definir dentro de un mismo documento Web. Aunque utilizado en *HTML*, el estándar *XHTML 1.1 strict* ha desaprobado este atributo.

— ***action***: indica cual será la aplicación dentro del servidor que recibirá los datos introducidos en el formulario. En caso de asignar un valor vacío a este atributo se estará omitiendo que el destinatario de los datos será la propia aplicación o documento Web que contiene el formulario.

— ***method***: indica cual será el método de envió de los datos introducidos en el formulario al servidor. Existen dos posibilidades, ***get*** o ***post***. El método ***get*** manda los datos adjuntándolos a la propia URL (*Uniform Resource Locator*, Localizador Uniforme del Recurso) de la petición HTTP al servidor, mediante la siguiente sintaxis, por ejemplo, "*http://www.midominio.es/script.php?name1=value1&name2=value2&...*", donde el carácter "?" separa la URL de los datos, name1=value1, name2=value2, etc., los nombres de los elementos que componen el formulario, junto a su valor, separados por el carácter "&". El método post manda los datos junto al cuerpo del formulario haciendo uso de la misma sintaxis anterior, "*name1=value1&name2=value2&...*".

— ***accept***: filtra los tipos de archivos que pueden ser subidos del cliente al servidor a través de un formulario. Solo serán aceptados la lista de tipos asociada a este atributo. Por ejemplo, si quisiéramos subir una imagen al servidor, restringiendo los formatos a jpg o png, asignaríamos el siguiente valor *accept="image/jpg,image/png"*.

— ***enctype***: Especifica el tipo de contenido de la información enviada, cuando el valor del atributo ***method*** es *post*. Por defecto, su valor es *application/x-www-form-urlencoded*. En el caso de que vayamos a enviar archivos mediante el formulario su valor deberá ser *multipart/form-data*.

— ***target***: En el caso de que el diseño del documento Web este basado en frames o marcos, este atributo indicará el marco donde se cargará la respuesta del servidor ante los datos enviados por el formulario.

A continuación, se mostrarán los distintos campos o elementos HTML que pueden formar parte de un formulario HTML, los eventos JavaScript que se les puede asociar y la forma en que los datos introducidos por el usuario en él, son recogidos en el equipo servidor Web mediante una aplicación en PHP.

4.1. Elementos HTML de un formulario

En relación a los elementos HTML más importantes que pueden formar parte de un formulario, destacaríamos los siguientes:

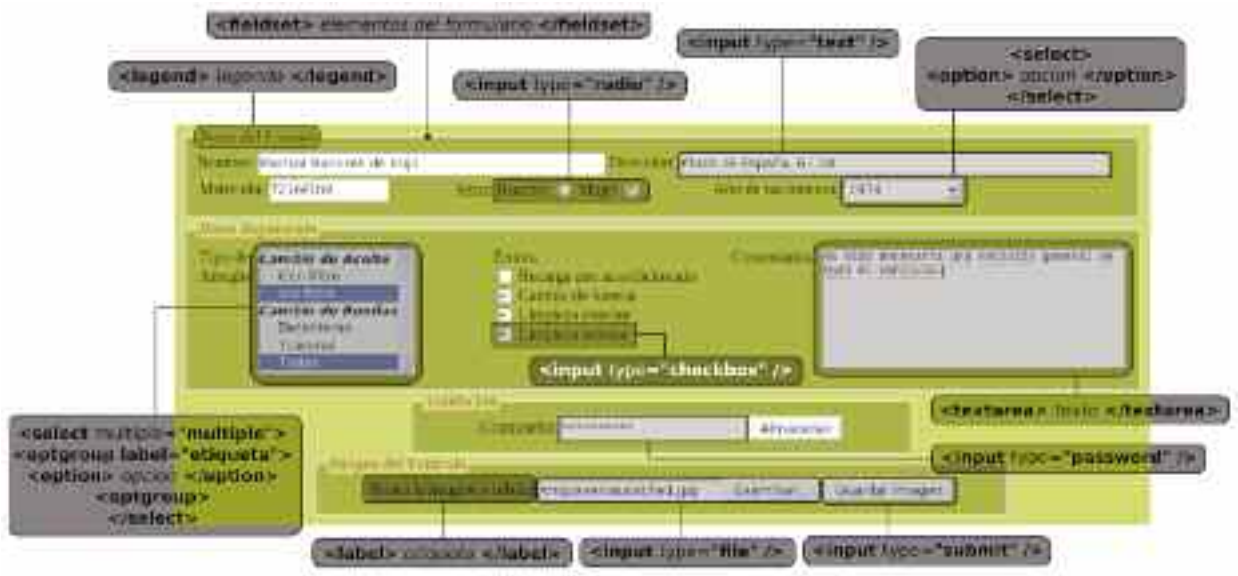


Figura 7.4. Elementos de un formulario

1. Caja de texto o "*Input-Text*", **<*input type="text"* />**: permite la introducción de un texto o cadena de caracteres.

> **<input type="text"** ***name id class style title lang size maxlength value readonly disabled accesskey tabindex*** **/>**

En relación a sus atributos, los de asignación general a todo elemento del formulario, serán descritos posteriormente al final del presente apartado. Entre los que le son propios destacar:

— ***size*** indica el tamaño de la caja de texto, medido como la cantidad de caracteres introducidos que serán visibles.

— ***maxlength*** indica la máxima cantidad de caracteres que van a poderse introducir. Este parámetro no tiene porque coincidir en valor con size.

```
<input type="text" name="dni" size="12" maxlength="9" />
```

2. Caja de texto para contraseñas o "*Input-Password*", **<*input type="password"* />**: igual que un "*Input-Text*", pero los caracteres introducidos no serán legibles por el usuario al verse sustituidos por el asteriscos, "*". Esta pensado para ser utilizado en formularios donde la introducción introducida sea confidencial.

> **<input type="password"** ***name id class style title lang size maxlength value readonly disabled accesskey tabindex*** **/>**

3. Área de texto o "*Text-Area*", **<*textarea*></*textarea*>**: permite introducir un texto al usuario, al igual que a "*Input-Text*", pero pudiendo constar de múltiples líneas.

> **<textarea** ***name id class style title lang rows cols readonly disabled tabindex*** **>**
> **</textarea>**

En relación a sus atributos, los de asignación general a todo elemento del formulario, serán descritos posteriormente al final del presente apartado. Entre los que le son propios destacar a ***rows*** y ***cols*** por ser encargados de determinar el tamaño del área de texto (*cols·rows, anchura·altura*):

— ***rows*** indica el número de líneas de texto que serán visibles.

— ***cols*** indica el número de caracteres que serán visibles por línea.

```
<textarea name="comentario" cols="60" rows="5">—escribe aquí tus comentarios—<
/textarea>
```

4. Opciones de selección múltiple o "*Check-Box*", ***<input type="checkbox" />***: también llamadas casillas de verificación, permiten encuestar al usuario pudiendo asignar uno o más valores a una determinada cuestión.

```
<input type="checkbox" checked name id class style title lang value readonly disabled
accesskey tabindex />
```

Por defecto, entre las opciones mostradas estarán sin marcar (*no checked*). En caso de querer que alguna de las opciones se encuentre seleccionada, será necesario hacer uso del atributo booleano **checked**[49]. Estos atributos booleanos, para ser compatibles con las restricciones del estándar *XHTML*, en su definición es necesario asignarles un valor, que se corresponde con el propio nombre del atributo, *nombre-atributo="nombre-atributo"*. En cuanto al resto de sus atributos, al ser de asignación general a prácticamente todos los elementos que pueden conformar un formulario, serán explicados al final del presente apartado.

```
Indica que libros has leido: <br />
<input type="checkbox" name="libro1" checked="checked" />I. Newton (lectura obligada) <br />
<input type="checkbox" name="libro2" />M. Faraday <br />
...
```

En el caso de que el atributo **name** sea el mismo para un conjunto de opciones, será posible asignar más de un valor al mismo campo del formulario, formando un vector o *array* de valores. Para informar de ello al servidor que se le envían los datos, el nombre asignado al atributo **name** deberá terminar en corchetes "**[]**".

```
Indica que libros has leido: <br />
<input type="checkbox" name="libro[]" value="newton" checked="checked" />I. Newton (lectura
obligada) <br />
<input type="checkbox" name="libro[]" value="faraday" />M. Faraday <br />
...
```

De todas las opciones mostradas solo serán enviadas al servidor aquellos campos del formulario junto con su valor asociado que hayan sido seleccionados. Si no se ha indicado un valor al campo del formulario a través de su atributo **value**, se enviará simplemente un valor "**on**".

---

49 En el estándar *HTML*, un atributo booleano, se considera verdadero, en caso de indicarse, y falso, en en caso de uno indicarse en la definición de la etiqueta *HTML* a la que afecta.

5. Opciones de selección única o "*Radio-List*", **<*input type*="radio" />**: al igual que las casillas de verificación anteriores, permiten encuestar al usuario, permitiendo asignar únicamente un único valor a una cuestión. Todas las opciones de este tipo se caracterizan por tener el mismo **name**.

```
<input type="radio" checked name id class style title lang value readonly disabled accesskey tabindex />
```

6. Lista de selección de opciones, "*List-Select*" o "*Combo-Box*", **<*select*><*option*></*option*></*select*>**: consiste en un caja de dialogo desplegable que permite elegir entre diferentes opciones. Cada una de estas opciones irá encerrada por la etiqueta *HTML* **<*option*></*option*>**, pudiendo ordenar y clasificar las distintas opciones en grupos mediante la etiqueta *HTML* **<*optgroup*></*optgroup*>**.

```
<select name id class style title lang size multiple disabled accesskey tabindex>
  <optgroup label id class style title lang disabled>
  <option value id class style title lang selected disabled label></option>
  </optgroup>
</select>
```

Entre las opciones mostradas en la lista, por defecto, solo será posible seleccionar una. En caso de querer permitir que el usuario pueda seleccionar más de una opción, será necesario hacer uso del atributo booleano **multiple**[50]. Estos atributos booleanos, para ser compatibles con las restricciones del estándar *XHTML*, en su definición es necesario asignarles un valor, que se corresponde con el propio nombre del atributo, *nombre-atributo="nombre-atributo"*. Al igual que ocurría con los Check-Box, en el caso de habilitar la opción **multiple**, los valores seleccionados formarán un vector o *array*, siendo necesario informar explícitamente de ello al servidor Web al que se le envían los datos del formulario añadiendo al final del nombre asignado al atributo **name** los corchetes "**[]**".

```
Selecciona los lenguajes de programación que conoces:
<select name="lenguajes" multiple="multiple">
  <optgroup label="Estructurados">
        <option value="C" selected="selected">Lenguaje C</option>
        <option value="pascal">Pascal</option>
        ...
  </optgroup>
  <optgroup label="Orientados a objetos">
        ...
  </optgroup>
</select>
```

Entre los atributos restantes podrían destacarse los siguientes:

— **size**: Determina el número de opciones de la lista que serán visibles. Por defecto, en listas no múltiples tan sólo será visible la opción seleccionada, mientras que en las múltiples serán visibles todas.

50 En el estándar *HTML*, un atributo booleano, se considera verdadero, en caso de indicarse, y falso, en en caso de uno indicarse en la definición de la etiqueta *HTML* a la que afecta.

— **selected**: Indica cuál de las opciones de la lista se encontrará seleccionada por defecto, pudiéndose asignar a más de una opción en caso de que tenga habilitado la opción **multiple**. Al igual que el atributo **multiple** es booleano, por lo que será necesario asignarle como valor el propio nombre del atributo según las restricciones *XHTML*.

— **value**: Indica el valor que será enviado al servidor en caso de que su opción asociada, <*option*></*option*> sea seleccionada. En caso de que no se le asigne un valor, por defecto, se adoptará como tal el texto encerrado entre las etiquetas <*option*></*option*>.

— **label**: Etiqueta al conjunto de opciones agrupadas por la etiqueta *HTML* <*optgroup*></*optgroup*>.

7. Campo oculto o "*hidden*", ***<input type="hidden" />***: elemento del formulario no visible por el usuario que permite mandar datos al servidor de manera transparente para el usuario.

```
<input type="hidden" name id class style title lang value />
```

Puede resultar útil para mandar información del cliente sin que sea necesario de que este la muestra explícitamente, como puede ser la dirección IP del equipo cliente, la fecha u hora local a la que se envían los datos, etc. En el ejemplo siguiente, se envía la fecha local sin formatear mediante la ayuda del evento JavaScript *onSubmit* y la función *Date()*. Este y otros eventos relacionados con los formularios se verán en el próximo apartado.

```
<form name="f1" method="post" action="http://www.dominio.com/destinatario.php"
onSubmit="document.getElementById('fecha').value = Date()">
<input type="hidden" name="fecha" id="fecha" />
...
</form>
```

En cuanto a sus atributos, por ser de uso general por todos los elementos del formulario, serán explicados al final del presente apartado.

8. Selección de archivos en el equipo cliente, ***<input type="file" />***: permite la subida del archivo seleccionado al servidor.

```
<input type="file" name id class style title lang value size readonly disabled accesskey
tabindex />
```

Debido al volumen de información que supone esta tranferencia de ficheros, es obligatorio enviarlo mediante el método *post*, y asignar al atributo *enctype* de la etiqueta *form* el valor *multipart/form-data*.

```
<form name="f1" method="post" action="destinatario.php" enctype="multipart/form-data">
<input type="file" name="fichero" />
...
</form>
```

La forma en que es tratado el archivo recibido por el servidor, se mostrará posteriormente mediante la realización de ejercicios prácticos. En cuanto a sus atributos, por ser de uso general por todos los elementos del formulario, serán explicados al final del presente apartado.

¡¡Importante!! Para poder subir archivos al servidor desde un formulario HTML debe estar habilitada esta opción *file_uploads* en el fichero de configuración *php.ini*. Este lo podemos encontrar en */opt/lampp/etc/php.ini* en GNU/Linux o *c:\xampp\php\php.ini* en Microsoft Windows.

file_uploads = On

En este mismo fichero podemos encontrar las directivas *upload_tmp_dir* y *upload_max_filesize*, encargadas de decidir el directorio donde serán almacenados temporalmente los archivos subidos al servidor y el tamaño máximo permitido para estos.

upload_tmp_dir = ruta_directorio
upload_max_filesize = max_tamaño_fichero

9. Inicialización o "*reset*" del formulario, ***<input type="reset" />***: botón o pulsador que restablece a su valor inicial todos los campos que forman parte del formulario.

```
<input type="reset" name id class style title lang value disabled accesskey tabindex />
```

El texto que aparecerá sobre el botón será el valor asignado al atributo **value**. En cuanto a sus atributos, por ser de uso general por otros elementos del formulario, serán explicados al final del presente apartado.

10. Envío de datos del formulario, ***<input type="submit" />*** o ***<input type="image" />***: botón o pulsador que envía los datos introducidos por el usuario en el formulario al destinatario indicado en el atributo de la etiqueta *HTML form*, *action*, haciendo uso del método asignado a *method*. La diferencia entre **type** *submit* e *image*, es que el segundo permite mostrar la imagen indicada en su atributo **src** en el botón.

```
<input type="submit" name id class style title lang value disabled accesskey tabindex />
<input type="image" src name id class style title lang value disabled accesskey tabindex />
```

El texto que aparecerá sobre el botón será el valor asignado al atributo **value**. En cuanto a sus atributos, por ser de uso general por otros elementos del formulario, serán explicados al final del presente apartado.

```
<form name="f1" method="post" action="http://www.dominio.com/destinatario.php">
...
<input type="submit" name="boton" value="Enviar Datos" />
</form>
```

11. Botones o "*button*", ***<button type="submit|reset|button"></button>***: introduce en el formulario botones de envió de datos (*submit*), inicialización (*reset*) u otra función definida mediante JavaScript, con un aspecto más personalizado que los anteriores, al poder encerrar texto, imágenes u objetos. Utilizado habitualmente en *HTML*, esta desaconsejado por el estándar *XHML 1.1 strict*.

```
<button type name id class style title lang value  disabled accesskey tabindex>
</button>
```

A modo de ejemplo, si quisiéramos mostrar en el botón unos textos e imágenes haríamos lo siguiente:

```
<button type="submit" name="boton1">
<p style="text-align:center;">Enviar<br /><img src="logo.gif" /><br />Datos</p>
</button>
```

En cuanto a sus atributos, por ser de uso general por otros elementos del formulario, serán explicados al final del presente apartado.

12. Etiqueta asociada al campo del formulario, ***<label> </label>***. Proporciona información al usuario para que sepa que tipo de datos son requeridos para la cumplimentación del formulario.

```
<label id class style title lang for accesskey />
```

En relación a sus atributos, los de asignación general a todo elemento del formulario, serán descritos posteriormente al final de este apartado. Entre los que le son propios destacar a for, al ser el atributo encargado de asociar la etiqueta a un campo del formulario, asignándole como valor el identificador (*id*) que tenga asignado ese campo. De esta forma, la etiqueta tendrá asociado un evento JavaScript que hará que al pinchar sobre ella, sea redireccionado el foco sobre ese campo en cuestión.

```
<label for="id-campo-formulario" />
```

13. Agrupación de varios elementos de un formulario, ***<fieldset> </fieldset>***. Como puede observarse en la figura X, este elemento HTML permite delimitar mediante un recuadro a un conjunto de elementos de un HTML por temática o cualquier otro criterio, para que le sea más fácil e intuitivo al usuario su cumplimentación.

```
<fieldset id class style title lang accesskey />
```

14. Título de un *fieldset*, **<legend></legend>**. Informa al usuario que cumplimenta el formulario sobre los campos del formulario a rellenar que se encuentran agrupados por el *fieldset*.

```
<fieldset id class style title lang accesskey />
```

Por último, se explicará la función de los atributos más genéricos que puedan asignarse a los campos que pueden componer un formulario:

— **name**: asigna un nombre identificador al campo del formulario. Es uno de los atributos más importantes, al ser utilizado por las aplicaciones del servidor para reconocer la información que les llega desde el cliente. Es decir, tras rellenar el formulario y pulsar sobre el botón *submit*, se envían los datos al servidor Web mediante el método *get* o *post* acompañando a cada uno de los datos su correspondiente identificador: name1=value1 &name2=value2&...

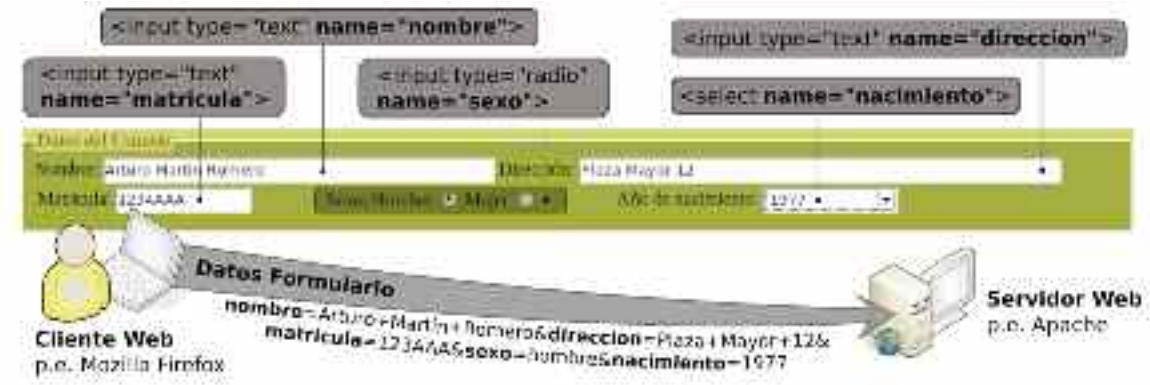


Figura 7.5. El valor asignado al atributo name del campo del formulario acompaña a los datos para poder ser renocidos por el servidor Web

— **id**=”identificador”: identifica al campo HTML del formulario para ser referenciado desde JavaScript. Puede consultarse el capítulo de JavaScript para saber más sobre este atributo.

— **class**=”nombre”: etiqueta al campo HTML del formulario para ser referenciado desde una hoja de estilos CSS. Puede consultarse el capítulo referente a hojas de estilo CSS para saber más sobre este atributo.

— **title**=”texto”: indica que texto será mostrado por el cliente Web al situar el ratón sobre el campo del formulario.

— **style**=”propiedad-css:value;...”: nos permite asignar propiedades de estilo CSS de manera personalizada a un campo del formulario. Recordar que el estándar XHTML no aconseja hacer uso de este atributo para la asignación de un estilo, sino la utilización de hojas de estilo CSS. Puede consultarse el capítulo del libro referente a hojas de estilo CSS para saber más sobre este atributo.

— **lang|xml:lang**=”idioma”: informa al navegador del lenguaje utilizado en edición del documento.

— **value**=”valor”: asigna un valor inicial al campo del formulario. En el caso de los *Input-Text*, *Input-Password* o *Text-Area*, este se mostrará pudiendo ser modificado. Tras pulsar sobre el botón *submit*, tal como se puede observar en la figura X, estos valores junto a los nombres asignados a cada uno de los campos serán enviados al servidor Web.

— **accesskey**=”tecla”: asigna una tecla de acceso al campo del formulario. Dependiendo del cliente Web[51], tras pulsar la combinación de teclas *ALT+SHIFT+AccessKey* el foco de cumplimentación del formulario se dirigirá hacia él.

— **tabindex**=”número”: especifica el orden en que se irán explorando los campos del formulario mediante la tabulación. En caso de que dos campos del formulario tengan asignado el mismo número de orden, se seguirá el orden en que aparecen en el documento.

— **disabled**=”disabled”: deshabilita el campo del formulario, evitando que el usuario pueda modificarlo. El valor asignado a estos campos no serán enviados con el resto de los datos del formulario al servidor Web. Al tratarse de un atributo booleano será necesario asignarle como valor el propio nombre del atributo, para cumplir con las restricciones XHTML, al no permitirnos indicar un atributo sin tener un valor asignado.

— **readonly**=”readonly”: Al igual que el atributo anterior, impide que el usuario pueda realizar cambios en el campo del formulario, diferenciandose, en que este si será enviado junto al resto de datos al servidor.

---

51 La activación de la tecla de acceso depende de si el cliente Web es *Mozilla Firefox, Internet Explorer*... debiendo pulsar la combinación *ALT+SHIFT+AccessKey, ALT+AccessKey*...

Ejercicio práctico n.º 1

Diseña un documento Web *XHTML* llamado "*formularios.php*", compuesto por dos formularios *HTML*, correspondiente a una aplicación de gestión asociada a un supuesto taller de vehículos, de tal forma que presente un aspecto similar al siguiente[52]:

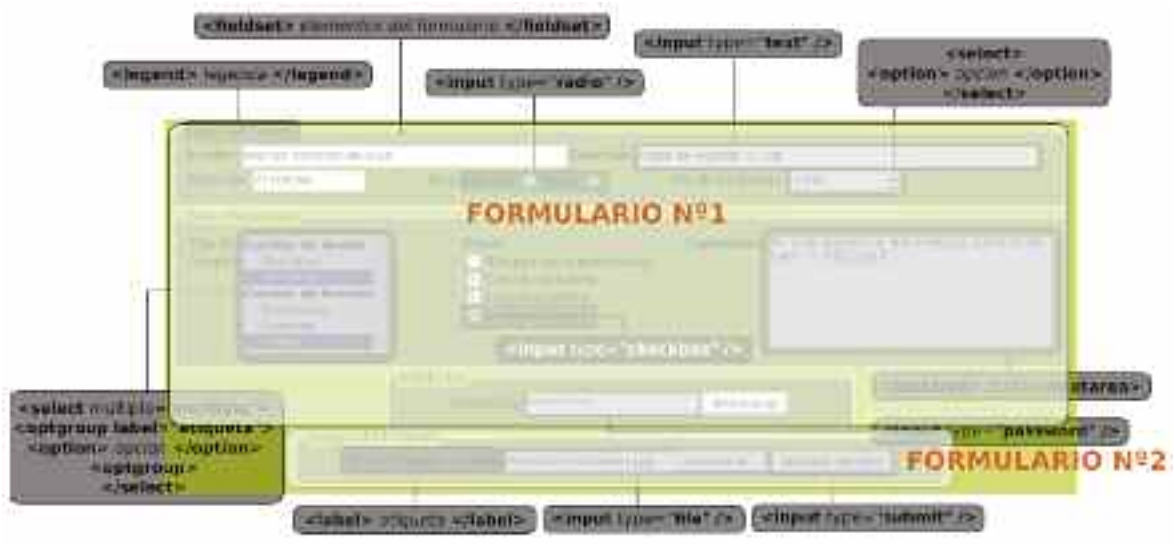


Figura 7.6. Formularios Web del ejercicio práctico

El primer formulario estará compuesto por tres *fieldset*:

I) Datos del usuario (nombre, dirección, matrícula, sexo y año de nacimiento) compuesto por tres *input-text*, un *input-radio-list* y un *select-list*;

II) Datos de la reparación (tipo de arreglo, extras y comentarios) compuesto por un *select-list-multiple*, un *input-check-box-list* y un *text-area*;

III) Validación (contraseña y botón de submit) compuesto por un *input-password* y un *input-submit*, tal como puede observarse en la figura 7.6. El segundo formulario lo forma un *fieldset* con dos elementos HTML, un *input-file* y un *input-submit*. Además es aconsejable seguir las siguientes pautas:

a) Todos los campos del formularios irán acompañados de una etiqueta, <*label*></*label*>, de tal forma que al situar el ratón sobre ella se muestre un título informativo que indique como cumplimentar el campo del formulario, <*label title="texto"*>, y que al pinchar con el ratón sobre ella sitúe el foco sobre dicho campo, <*label for="id-campo-formulario"*>.

b) El *select-list* o *combo-box* del formulario encargado de mostrar una lista con los años de nacimiento se generará de manera automática con JavaScript mediante una función definida en el archivo *formulario.js* asociado al documento *XHTML*, *function crearlista()*. El listado de edades estará comprendido entre 1950 y el año actual.

c) La organización de los elementos del formulario dentro de cada *fieldset* se realizará mediante la ayuda de una tabla *HTML*.

52 Con la finalidad de presentar todos los elementos *HTML* que pueden intervenir en un formulario, a modo de ejemplo, se ha pensado en una aplicación de gestión de un supuesto taller encargado de recopilar cierta información que puede resultar útil.

Solución ejercicio n.º 1

Antes de dar la solución, configuraremos Apache para dar servicio al sitio Web que contendrá los documentos de los distintos ejercicios prácticos que forman parte del presente capítulo:

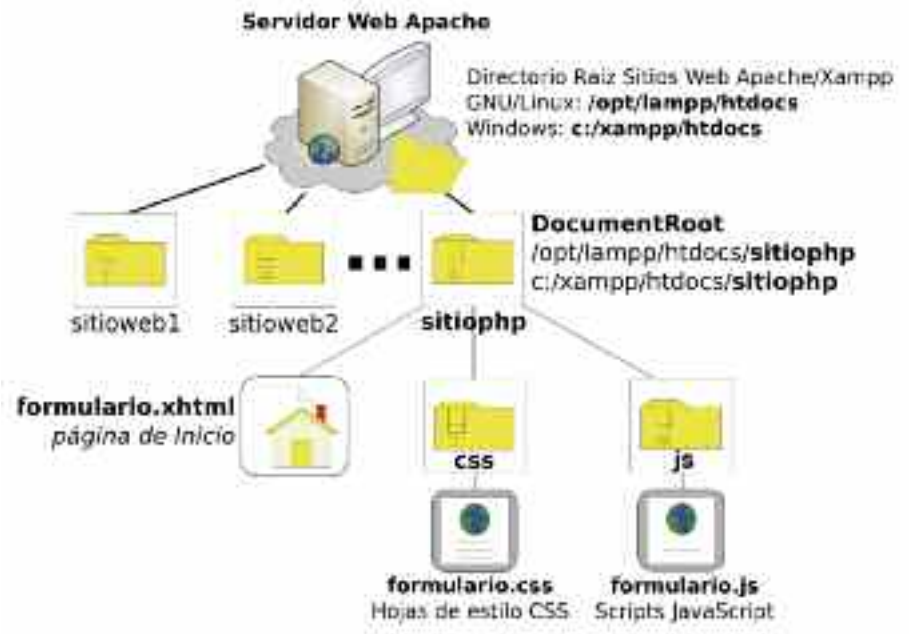


Figura 7.7. Esquema del sitio Web dentro sistema de ficheros para el presente capítulo

```
# Contenido del fichero de configuración de Apache: httpd.conf
DocumentRoot /opt/lampp/htdocs # Directorio Raíz de Xampp en GNU/Linux
DocumentRoot c:/xampp/htdocs # Directorio Raíz de Xampp en Microsoft Windows
ServerName localhost # Nombre del equipo servidor Xampp
Listen 80 # Puerto de escucha de Apache por defecto
NameVirtualHost 127.0.0.1 # Necesario si da servicio a más de un sitio Web bajo esa IP
Include /opt/lampp/etc/extra/sitiophp.conf # Fichero auxiliar de configuración en GNU/Linux
Include c:/xampp/apache/conf/extra/sitiophp.conf # Fichero auxiliar de configuración en
Windows
...
```

```
# Contenido del fichero auxiliar de configuración de Apache: sitiophp.conf
<VirtualHost *> # Atenderá peticiones HTTP a www.sitiophp.es bajo cualquier IP
  ServerName www.sitiophp.es # Nombre de dominio del sitio Web
  DocumentRoot /opt/lampp/htdocs/sitiophp # Directorio Raíz en GNU/Linux
  DocumentRoot c:/xampp/htdocs/sitiophp # Directorio Raíz en Windows
</VirtualHost>
```

```
# Añadir la siguiente Resolución en el fichero hosts: Direccion IP -> Nombre Equipo
127.0.0.1 www.sitiophp.es
```

Para cumplir las restricciones del estándar *XHTML*, separaremos los contenidos del documento de su presentación (hoja de estilos CSS), definiendo los estilos aplicados al documento en un fichero independiente, *formulario.css*, ubicado en una subcarpeta *css*, que será importado por el documento. Por los mismos motivos, las funciones JavaScript utilizadas en el documento serán definidas en un documento independiente que importaremos, *formulario.js*, ubicado en una subcarpeta *js*.

1. El documento XHTML *formulario.xhml* podría ser de la siguiente forma:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>Taller Mecánico</title>
<link href='imagenes/logo.png' rel='shortcut icon' type='image/png'/>
<!— Importamos la hoja de estilos CSS asociada al documento XHML —>
<link rel="stylesheet" type="text/css" href="css/formulario.css" media="screen" />
<!— Importamos el archivo JavaScript donde se definirán las funciones utilizadas en el documento —>
<script type="text/javascript" src="js/formulario.js"></script>
</head>
<body>
<h2 class="titulo"> TALLER MECANICO </h2>
<!— Primer Formulario.  Fieldset Datos del Usuario  —>
<form method="post" action="">
<fieldset>
<legend>Datos del Usuario</legend>
<label for="nombre" title="Introduce tu nombre y apellidos">Nombre:</label>
<input type="text" name="nombre" id="nombre" size="40" maxlength="40" />
<label for="direccion" title="Introduce tu dirección">Dirección:</label>
<input type="text" name="direccion" id="direccion" size="50" accesskey="k" /><br />
<table><tr><td class="columna">
<label for="matricula" title="Introduce la matricula del vehículo">Matricula:</label> <input type="text" name="matricula" id="matricula" size="12" />
</td>
<td class="columna">
<label for="hombre" title="Selecciona tu sexo">Sexo:</label>
<label for="hombre" class="sexo">Hombre</label>
<input type="radio" name="sexo" value="hombre" id="hombre" />
<label for="mujer" class="sexo">Mujer</label>
<input type="radio" name="sexo" value="mujer" id="mujer" />
</td>
<td class="columna">
<label for="edad" title="Selecciona el año de nacimiento">Año de nacimiento:</label>
<!— la lista de edades será generada con JavaScript, formulario.js: function crearlista() —>
<select name="edad" id="edad" class="etiqueta" onclick="crearlista(this)">
<option>—seleccionar—</option>
</select>
</td></tr></table></fieldset>
<!— Fieldset de Datos de Reparación. Primer Formulario.  —>
<fieldset>
<legend>Datos Reparación</legend>
<table border="0"><tr>
<td valign="top" title="Selecciona los arreglos realizados">Tipo de Arreglo:</td>
<td class="columna">
<select name="tipo[]" multiple="multiple">
<optgroup label="Cambio de Aceite">
<option>Con filtro</option><option>Sin filtro</option></optgroup>
<optgroup label="Cambio de Ruedas">
```

```
<option>Delanteras</option><option>Traseras</option><option>Todas</option></optgroup>
</select></td>
<td valign="top" class="columna">
<label title="Selecciona los arregos extras">Extras:</label><br />
<input type="checkbox" name="extra[]" value="recarga" /> Recarga aire acondicionado<br />
<input type="checkbox" name="extra[]" value="bateria" /> Cambio de bateria<br />
<input type="checkbox" name="extra[]" value="limpiezaext" /> Limpieza exterior<br />
<input type="checkbox" name="extra[]" value="limpiezaint" /> Limpieza interior</td>
<td valign="top">
<label for="comentarios" title="Introduce información complementaria sobre el arrglo">Comentarios:</label></td>
<td valign="top" class="columna"><textarea name="comentarios" id="comentarios" rows="7" cols="40">—Sin comentarios—</textarea></td></tr></table></fieldset>
<!— Fieldset de Validación. Primer Formulario. —>
<fieldset class="validacion">
<legend>Validación</legend>
<label for="pass" title="Introduce la contraseña de validación">Contraseña: </label><input type="password" name="pass" id="pass" /> <input type="submit" name="boton" value="Almacenar" title="Al pulsar serán enviados los datos introducidos al servidor" />
</fieldset></form>
<!— Segundo Formulario. Fieldset para la subida de la imágen. —>
<form method="post" enctype="multipart/form-data" action="">
<fieldset class="imagen">
<legend>Imagen del Vehiculo</legend>
<label for="imagen" title="Añade la imagen del vehículo">Busca la imagen y súbela:</label>
<input type="file" id="imagen" name="imagen" />
<input type="submit" name="boton" value="Guardar Imagen" title="Al pulsar será enviada la imagen al servidor para ser almacenada" />
</fieldset></form></body></html>
```

2. La hoja de estilos CSS *formulario.css* podría ser la siguiente:

```
body { background-color: GreenYellow; padding: 2cm; padding-top: 1cm; }
h2.titulo { border: gray solid 2px; background-color: white; padding: 5px; text-align: center; color: tomato; width: 50%; margin-left: auto; margin-right: auto; }
fieldset {margin-left: auto; margin-right: auto; background-color: OliveDrab;}
legend {font-style: italic; font-weight: 600; color: Brown; background-color: GreenYellow;}
td.columna { width: 33%; }
fieldset.validacion { width: 50%; text-align: center; }
fieldset.imagen { width: 70%; text-align: center; }
```

3. La función JavaScript *crearlista()* encargada de generar la lista con las edades comprendidas entre el año 1950 y el año actual situada en el fichero *formulario.js* podría ser de la siguiente forma:

```
// Contenido del archivo formulario.js
function crearlista(lista){
  if (lista.length == 1) // Comprobamos si la lista solo tiene la opción —seleccionar—
```

```
    {       var fecha = new Date();  // Creamos una variable de tipo fecha
            var contador = 1950;  // Inicializaremos la lista en el año 1950
            while (contador <= fecha.getFullYear())  // Comparamos con el año actual
            {       var opcion = document.createElement('option'); // Creamos una nueva
                        opción
                    opcion.text = contador.toString();
                    lista.add(opcion,null);  // Añadimos la nueva opción a la lista
                    contador++;
            }
    }
}
```

Como puede observarse en el contenido del documento *XHTML*, la función *crearlista()* es llamada al pinchar con el ratón, *onclick="crearlista(this)"*, pasándole como parámetro la propia lista. Para evitar que cada vez que el usuario pinche el ratón se llene la lista repetidas veces con el mismo contenido, lo primero que hace la función es comprobar que la lista tan solo tiene la primera opción (*lista.length == 1*), <*option*>—*seleccionar*—</*option*>, declarada en el cuerpo del propio documento. En caso de ser así, mediante la ayuda de un bucle y un contador, se va rellenando la lista mediante el método *add*, *lista.add(opcion,null)*, al cual se le pasa la nueva opción y el valor *null* para que la añada al final[53].

Otra posibilidad hubiera sido cargar la lista en el momento en que se carga el documento en el cliente Web, mediante la ayuda del evento *JavaScript OnLoad()*, definiéndolo en la etiqueta *body* , <*body onload="crearlista()"*>.

Las complicaciones anteriores para crear la lista es debido a las restricciones que presentan los documentos *XHTML* y las nuevas versiones de los documentos HTML, que no permiten utilizar las etiquetas <*script*></*script*>, y por tanto, código *JavaScript*, en el cuerpo del documento (*body*) de manera estricta, solo pudiendo ser invocadas las funciones *JavaScript* ante un evento declarado en una etiqueta HTML. En el caso de que el documento HTML no requiriera ser validado por el W3C, podría crearse la lista de una forma tan sencilla como la siguiente:

```
<select name="edad" id="edad">
<option>—selecciona—</option>
<script type="text/javascript">
var fecha = new Date();
var inicial = 1950;
while ( inicial <= fecha.getFullYear() )
{ document.write("<option>" + inicial + "</option>"); inicial++; }
</script>
</select>
```

53 En el caso de que el navegador sea Internet Explorer es posible que el método add no funcione de esta forma. En ese caso, quitar el segundo de los parámetros, *lista.add(opcion)*.

4.2. Eventos Javascript relacionados con formularios

Cumplimentando al capítulo asociado a *JavaScript*, a continuación se presentarán los eventos *JavaScript* que están relacionados con los formularios *HTML*, junto a un ejercicio práctico que muestra su utilización:

— **onreset** - **onsubmit**: Eventos que se desencadenan al pulsar sobre los botones del formulario de tipo *reset* o *submit* respectivamente. Sólo pueden ser declarados dentro de la etiqueta *HTML* <*form*></*form*>.

```
<form method="post" action="destinatario.php" onreset="return funcionJavaScript()"
onsubmit="return funcionJavaScript()"> ... </form>
```

Tras la detección del evento puede ejecutarse una función JavaScript que devolverá (*return*) un valor booleano, *true* o *false*, el cual le indicará al formulario si deben o no resetearse o enviarse los datos del formulario respectivamente.

— **onselect** - **onchange**: Eventos que se desencadenan al seleccionar o modificar el texto contenido en un *input-text*, *text-area* o *select-list*. Ambos sólo pueden ser declarados en este tipo de elementos *HTML*.

```
<input type="text" onselect="funcionJavaScript()" onchange="funcionJavaScript()" />
```

En relación al evento **onchange**, no confundir con los eventos JavaScript *onkeydown*, *onkeypress* y *onkeyup*. Mientras estos últimos se desencadenan en el momento en que el usuario pulsa una tecla (*pulsar, mantener pulsado y despulsar respectivamente*), produciendo una modificación del contenido del campo del formulario, esto no conlleva un evento **onchange**. Este evento se produce cuando tras recibir el foco, el usuario modifica su valor, siendo detectado el cambio al perder el foco.

— **onfocus** - **onblur**: Eventos que se desencadenan al recibir o perder el foco el elemento del formulario donde es declarado respectivamente. Pueden ser declarados en cualquier tipo de elemento *HTML* que contenga el formulario. A diferencia de los eventos anteriores, su utilización no es exclusiva en formularios *HTML*, pudiendo asignarse a enlaces <*a*></*a*>, tablas <*table*><*tr*><*td*></*td*></*tr*></*table*>, títulos <*h1*|*2*|...></*h1*|*2*|...>, etc.

```
<textarea cols="80" rows="10" onfocus="funcionJavaScript()" onblur="funcionJavaScript()">
</textarea>
```

Ejercicio práctico n.º 2

Continuando con el ejercicio práctico anterior, añade los siguientes eventos *JavaScript* a modo de ejemplo:

a) Al recibir el foco de cumplimentación (*onfocus*) los campos del formulario de tipo *input-text, input-password* y *text-area*, cambiarán su color de fuente (*p.e. color="red"*) y de fondo (p.e. *backgroundColor="yellow"*) a uno que llame más la atención del usuario. Al perder del foco (*onblur*), deberá volver a su estado anterior.

Figura 7.8. Al recibir el foco cambiará el color de fuente y de fondo del campo del formulario

b) Comprobar que tras cumplimentar (*onchange*) el campo del formulario asociado a la matricula del vehículo, el dato introducido esta compuesto por una longitud de 7 caracteres y en letras mayúsculas. En caso de no ser así, si la longitud es menor se mostrará un mensaje de alerta (*alert*) advirtiendo de ello, y en caso de que este en minúsculas directamente se modificará a mayúsculas (*toUpperCase()*).

Figura 7.9. Mensaje de alerta informando al usuario de que la matricula debe constar de 7 caracteres

c) Al pulsar sobre el botón submit del primer formulario (*onsubmit*), debe comprobarse que los campos del formulario *input-text* asociado a la matricula, contiene 7 caracteres, y el *input-password* correspondiente a la contraseña de validación no se encuentre vacío. En caso contrario, mostrar una ventana de diálogo (*alert*) al usuario indicándole la obligatoriedad de su cumplimentación para poder ser enviados los datos al servidor, situando el foco en el correspondiente campo del formulario. De manera similar comprobar los campos del formulario *input-text* asociados al nombre y la dirección, mostrando un mensaje de confirmación (*confirm*) al usuario en caso de que estén vacíos, consultándole si aún así desea enviar los datos del formulario.

d) Incluir un campo oculto (*type="hidden"*) en el formulario, de tal forma que se envíe al servidor (*onsubmit*), junto al resto de los datos, la fecha y hora local (*Date()*) en que se cumplimento con el siguiente formato *"día-mes-año/hora:minutos"*.

Solución ejercicio n.º 2

Para dar solución al ejercicio práctico propuesto modificaremos el fichero *XHTML* del ejercicio práctico Nº1 (p.e. *formulario.xhtml*) siguiendo los siguientes pasos:

1.º) Dentro del *<head></head>* del documento *XHMTL* incluiremos la etiqueta *<script></script>* informando al navegador o cliente Web de la utilización de *JavaScript*. Para evitar problemas con las restricciones impuestas por el estándar *XHTML*, evitaremos la introducción de código *JavaScript* en el propio documento, mediante la importación (atributo *src*) de un fichero externo (por ejemplo, *formulario.js*) ubicado en un subdirectorio llamado *js* (*src="js/formulario.js"*) donde serán definidas las distintas funciones *JavaScript* invocadas desde el documento.

```
<head>
...
<script type="text/javascript" src="js/formulario.js"></script>
</head>
```

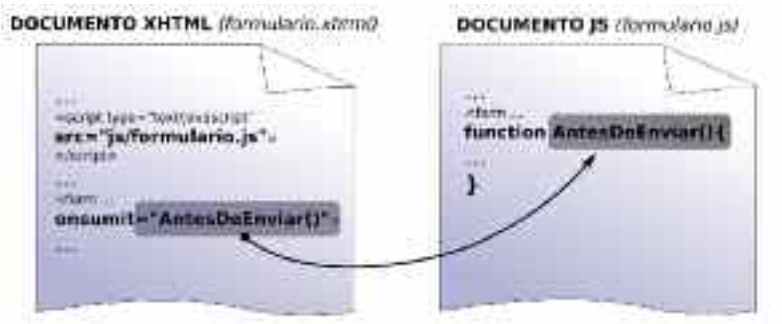


Figura 7.10. Las funciones JavaScript serán definidas en un documento importado independiente

2.º) Para cumplir con el requerimiento de diseño (a) será necesario declarar en todos los campos del formulario de tipo *input-text, input-password* y *text-area*, los eventos *onfocus* y *onblur* con sus llamadas a las funciones *JavaScript* asociadas (*RecibirFoco() y PerderFoco()*).

```
<input type="text" name="nombre" id="nombre" size="40" maxlength="40"
onfocus="RecibirFoco(this)" onblur="PerderFoco(this)" />
```

Como puede observarse, en la llamada a las funciones se envía como parámetro el objeto *this*, informando a la función del elemento o campo del formulario en concreto que esta realizando la llamada. En relación al contenido de las funciones JavaScript definidas en el archivo *formulario.js*, con la finalidad de llevar a cabo las funciones requeridas, será el siguiente:

```
// Contenido del archivo formulario.js: funciones RecibirFoco() y PerderFoco().
function RecibirFoco(campoformulario) {
  campoformulario.style.backgroundColor="yellow";
  campoformulario.style.color="red";
}
function PerderFoco(campoformulario) {
  campoformulario.style.backgroundColor="white";
  campoformulario.style.color="black";
}
```

La función *RecibirFoco()*, modifica el color de fondo y de fuente a amarillo y rojo respectivamente, devolviéndole sus valores iniciales, blanco y negro, la función *PerderFoco()*.

3.º) Para cumplir con el requerimiento de diseño (b), en primer lugar será necesario declarar el evento *onchange* en el *input-text* asociado a la matricula.

```
<input type="text" name="matricula" id="matricula" size="12" onfocus="RecibirFoco(this)"
onblur="PerderFoco(this)" onchange="ComprobarMatricula(this.value)" />
```

Como puede observarse, al provocarse el evento *onchange* se invocará a la función *JavaScript ComprobarMatricula()*, pasándole como parámetro el valor introducido en el propio *input-text*, *this.value*. A continuación, en el fichero *formulario.js* se añadirá la función anterior:

```
// Contenido del archivo formulario.js: función ComprobarMatricula().
function ComprobarMatricula(valor) {
```

```
    document.getElementById("matricula").value=valor.toUpperCase();
    if (valor.length != 7)   // Comprueba si el número de caracteres es 7
    {
    alert ("La matrícula "+ valor +" debe constar de 7 caracteres");
    }
}
```

La función *ComprobarMatricula()* comienza poniendo en letras mayúsculas el contenido del campo del formulario asociado a la matrícula, *valor.toUpperCase()*, y continua comprobando si el número de caracteres son siete, para en caso contrario advertir de ello mediante un mensaje de alerta, *alert()*.

4.º) Para cumplir con el requerimiento de diseño (c), declararemos en primer lugar el evento *onsubmit* en la etiqueta HTML *<form></form>*:

```
<form method="post" id="form1" action=""" onsubmit="return AntesDeEnviar(this)">
```

Como puede observarse, al pulsar sobre el botón submit del formulario se invocará a la función *AntesDeEnviar()* pasándole como parámetro el propio formulario. En función del valor booleano, *true* o *false*, devuelto por esta función, *return*, serán finalmente enviados los datos del formulario al servidor. El contenido de la función podría ser el siguiente:

```
// Contenido del archivo formulario.js: función AntesDeEnviar().
function AntesDeEnviar(formulario) {
  if (formulario.matricula.value.length != 7) // Comprueba si la martricula tiene 7 letras
  {
  alert ("Para que los datos del formulario sean enviados al \nservidor la matrícula " + formulario.matricula.value + " debe tener al menos 7 caracteres");
  formulario.matricula.focus(); // Envia el foco al input-text matricula
  return false; // Indica que aún no se envíen los datos del formulario
  }
  if (formulario.pass.value == "") // Comprueba si el input-password contraseña esta vacio
  {
  alert ("Para que los datos del formulario sean enviados al \nservidor debes introducir una contraseña de validación.");
  formulario.pass.focus();  // Envia el foco al input-password
  return false; // Indica que aún no se envíen los datos del formulario
  }
  if ((formulario.nombre.value == "") || (formulario.direccion.value == ""))
  {
  respuesta=confirm ("El campo Nombre o Dirección están vacíos ...\nQuieres seguir mandando los datos del formulario o rellenarlos:");
  if (respuesta == false)
  {
            if (formulario.nombre.value == "")
            { formulario.nombre.focus(); return false; }
            if (formulario.direccion.value == "")
            { formulario.direccion.focus(); return false; }
  }
  }
  return true;  // Validamos el envio de los datos del formulario al servidor.
}
```

La función empieza comprobando si el contenido del campo asociado a la matricula del vehículo tiene siete caracteres. En caso contrario, se mostrará un mensaje de alerta, *alert()*[54], se situará el foco sobre el *input-text* correspondiente para su correcta cumplimentación, cancelándose el envío de los datos del formulario, *return false*.

¡¡Importante!! Al encontrar una sentencia "***return valor;***" en una función, el intérprete de *JavaScript* del cliente Web, deja de interpretar el resto de código que forma parte de la función, y devuelve el valor indicado al evento que la invocó.

En el caso de que la matrícula sea correcta, se comprobará si el contenido del campo asociado a la contraseña de validación esta vacío. Si es así, se mostrará un mensaje de alerta, situando el foco sobre el *input-password* y cancelando el envío de los datos del formulario al servidor.

Por último, tras cumplimentar correctamente los campos anteriores, se comprueba si el nombre o dirección están vacíos. En tal caso, mediante un mensaje de confirmación, *confirm*, se le dará la opción al usuario de decidir si a pesar de ello quiere enviar los datos al servidor o rellenar estos campos antes, situando el foco de cumplimentación en el campo que se encuentre vació en el caso de decantarse por la segunda opción.

4.º) Para cumplir con el requerimiento de diseño (d), añadiremos al formulario un campo oculto *hidden*, cuyo valor será asignado en el momento del envió de los datos del formulario al servidor.

```
<form method="post" id="form1" action="" onsubmit="return AntesDeEnviar(this)">
<input type="hidden" name="fecha" id="fecha" />
... <!— Resto de los campos del formulario —>
</form>
```

Esta asignación se realizará en la función *JavaScript AntesDeEnviar()*, justo antes de validar el envío de los datos, "*return true;*", haciendo uso de los métodos del objeto JavaScript Date(), getDate(), getMonth(), getFullYear(), getHours(), getMinutes(), para la obtención del día, mes, año con cuatro dígitos, hora y minutos del equipo cliente:

```
// Contenido del archivo formulario.js: función AntesDeEnviar().
function AntesDeEnviar(formulario) {
  ... // Resto del contenido de la función AntesDeEnviar() mostrado en el apartado anterior.
  var fechahora = new Date(); // Declaramos una variable de tipo fecha.
  document.getElementById("fecha").value = fechahora.getDate() + "-" +
fechahora.getMonth() + "-" + fechahora.getFullYear() + "/" + fechahora.getHours() + ":" +
fechahora.getMinutes(); // Asignamos la fecha con formato "día-mes-año/hora:minutos"
  return true;      // Validamos el envio de los datos del formulario al servidor.
}
```

54 El carácter especial "**/n**" es utilizado en los diálogos *alert(), confirm()*, o *prompt()* para provocar un salto de línea en el mensaje visualizado.

4.3. Recepción de datos del formulario en PHP

Una vez enviados (*submit*) los datos del formulario HTML por el método POST o GET al servidor vía protocolo HTTP, el servidor los recibe y acepta almacenándolos en las siguientes variables:

— **$_GET**: Variable de tipo vector o *array* encargada de almacenar la lista de valores enviados por el método GET.

— **$HTTP_GET_VARS**[‘*name* campo formulario’] o **$_GET**[‘*name* campo formulario’]: Almacenan los valores (*value*) asignados a cada uno de los campos del formulario identificados por su atributo *name* enviados por el método GET.

— **$_POST**: Variable de tipo vector o *array* encargada de almacenar la lista de valores enviados por el método POST.

— **$HTTP_POST_VARS**[‘*name* campo formulario’] o **$_POST**[‘*name* campo formulario’]: Almacenan los valores (*value*) asignados a cada uno de los campos del formulario identificados por su atributo *name* enviados por el método POST.

¡¡Sugerencia!! Para mostrar en el navegador o cliente Web la lista de todos los datos contenidos en el *array* **$_SERVER** o **$_POST** podría ejecutarse el siguiente script PHP incrustandolo dentro del documento Web destinatario de los datos del formulario (*action=”destinatario.php”*):

```
<? echo “####### Todos los datos enviados por el formulario vía GET son: #######<br />”;
foreach ($_GET as $nombre_campo_formulario => $valor)
{ echo “$nombre_campo_formulario: $valor<br />”; }
echo “####### Todos los datos enviados por el formulario vía POST son: #######<br />”;
foreach ($_POST as $nombre_campo_formulario => $valor)
{ echo “$nombre_campo_formulario: $valor<br />”; } ?>
```

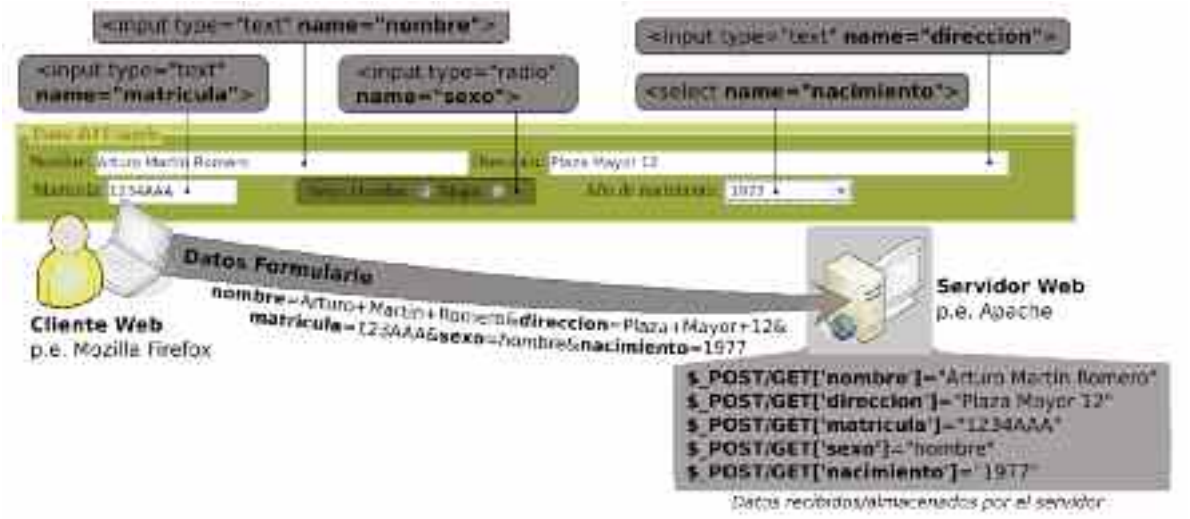


Figura 7.11. Forma en que son almacenados los datos del formulario en el servidor

— **$_FILES**: Variable de tipo vector o *array* encargada de almacenar la lista de todos los archivos enviados desde un formulario, <*input type=”file” name=”...”* />, al servidor.

— **$HTTP_POST_FILES**[‘*name* campo formulario’] o **$_FILES**[‘*name* campo formulario’]: Almacena el contenido de cada uno de los ficheros enviados al servidor identificado por su atributo *name*.

- **$_FILES**[‘*name* campo formulario’][‘**name**‘]: nombre del archivo seleccionado en el cliente y subido al servidor.
- **$_FILES**[‘*name* campo formulario’][‘**tmp_name**‘]: nombre temporal asignado al archivo recibido por el servidor que nos permite hacer referencia a él.
- **$_FILES**[‘*name* campo formulario’][‘**type**‘]: información sobre el tipo de fichero subido al servidor. Valdrá *image/jpeg*, *image/png*, *image/gif*, ... en el caso de tratarse de una imagen, *application/x-shockwave-flash*, si es un objeto Flash SWF, *application/zip*, para archivos comprimidos ZIP, *application/octet-stream*, si es una presentación PPS, *text/html*, si es un documento HTML, etc.
- **$_FILES**[‘*name* campo formulario’][‘**size**‘]: información sobre el tamaño del archivo subido al servidor.

Junto a los datos del formulario, existen otros datos que pueden resultar interesantes y que son intercambiados en la comunicación HTTP entre el cliente y el servidor, los cuales se encuentran almacenados en la variable superglobal **$_SERVER**[55] pudiendo ser consultada en PHP cuando sea necesario:

— **$_SERVER**: Variable de tipo vector o *array* que contiene información sobre el la solicitud HTTP realizada por el cliente al servidor. Entre los datos más importantes que contiene cabría resaltar:

- **$_SERVER**[‘**SERVER_SORTWARE | SERVER_NAME | SERVER_ADDR | SERVER_PORT | SERVER_PROTOCOL | REQUEST_METHOD | DOCUMENT_ROOT**‘]: Informa de características del servidor como el software utilizado (p.e. Apache), el nombre del equipo servidor, su dirección IP, el puerto por el que ofrece el servicio, el protocolo utilizado o el método de envío de la información.
- **$_SERVER**[‘**REMOTE_ADDR | REMOTE_PORT**‘]: Informa de la dirección IP del equipo cliente y el puerto por el que realiza la solicitud.

Por ejemplo, para mostrar por pantalla una tabla HTML con la dirección IP del equipo cliente y servidor que se comunican vía HTTP introduciríamos el siguiente código PHP:

```
<table><tr><td>Dirección IP Servidor</td><td>Dirección IP cliente</td></tr>
<tr><td><? echo $_SERVER['SERVER_ADDR']; ?></td><td><? echo
$_SERVER['REMOTE_ADDR']; ?></td></tr></table>
```

¡¡Sugerencia!! Para conocer toda la información contenida en el array $_SERVER podría ejecutarse el siguiente script PHP:

```
<?
echo "####### TODAS LAS VARIABLES SERVER SON: #######";
foreach ($_SERVER as $nombre_variable => $valor)
{ echo "$nombre_variable: $valor<br>"; }
?>
```

55 Se consideran a una variable superglobal cuando su valor asignado puede ser consultado desde cualquier ámbito de un documento PHP sin necesidad de una declaración previa.

Ejercicio práctico n.º 3

Siguiendo con los ejercicios prácticos anteriores, *formulario.xhtml*, crea un nuevo documento PHP, *presenta-datos.php*, encargado de recoger los datos del formulario (*action="presenta-datos.php"*) enviados desde el cliente vía post (*method="post"*) al servidor, de tal forma que los organice en una tabla HTML, tal como muestra la siguiente figura:

Figura 7.12. Representación de la comunicación entre cliente y servidor en el ejercicio práctico n.º 3

Los requisitos a cumplir podrían ser los siguientes:

a) El destinatario de los datos del formulario será un documento PHP localizado dentro del servidor en el mismo directorio se encuentra el documento *XHTML formulario.xhtml*. Por ejemplo, *presenta-datos.php*.

```
<form action="presenta-datos.php" method="post" id="form1" onsubmit="return AntesDeEnviar(this)"> ... </form>
```

b) En el primer formulario del documento *formulario.xhtml* será necesario introducir una contraseña valida (*por ejemplo,* "mipass") para la validación de los datos en el servidor.

```
<? if ($_POST['boton'] == "Almacenar" && $_POST['pass'] == "mipass" ) { ... } ?>
```

c) Los datos del primer formulario recogidos por *presenta-datos.php* serán organizados en una tabla con tres columnas: datos personales, datos reparación y comentarios. Junto a ellos se mostrará la dirección del equipo cliente que los envió y el método de envío.

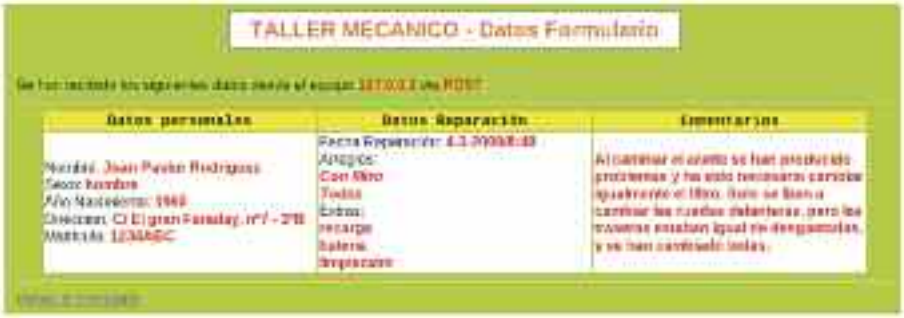


Figura 7.13. Los datos enviados al servidor serán organizados en una tabla

d) Las imágenes subidas al servidor desde el segundo formulario serán almacenadas en un subdirectorio llamado "*img-coches*":

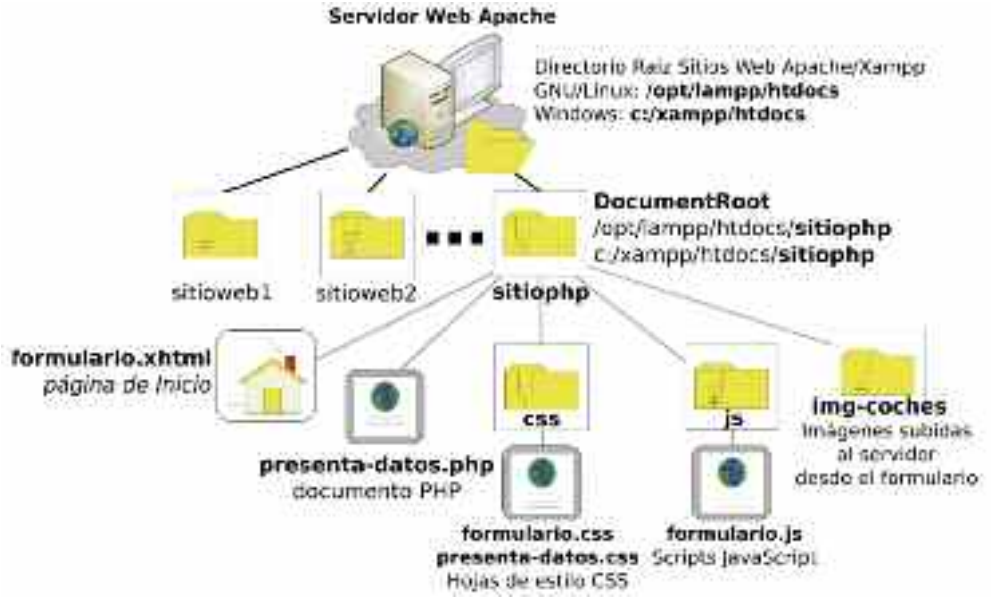


Figura 7.14. Esquema organizativo del sitio Web

¡¡Importante!! Para que los archivos subidos al servidor puedan ser almacenados en el subdirectorio *img-coches* es necesario asignar permisos de escritura al servicio Apache sobre él, en caso contrario el servidor nos devolverá un mensaje de error.

Figura 7.15. En el caso de que la cuenta de usuario del sistema bajo la que funciona Apache no tenga permisos de escritura sobre el directorio donde deseamos almacenar la imagen se nos advertirá de ello

Para ello, es necesario saber cual es la cuenta de usuario del sistema que Apache suplanta al dar el servicio. Esta información la podemos encontrar consultando el fichero de configuración de Apache **httpd.conf** (*/opt/lampp/etc/httpd.conf en GNU/Linux o c:/xampp/apache/conf/httpd.conf en Windows*), a través de la directiva **User**. Al buscarla advertiremos que esta directiva de configuración sólo esta disponible en GNU/Linux.

En el caso de Windows, si Apache se inicia como un servicio más del sistema, por defecto, será la cuenta de usuario "LocalSystem" la encargada de su gestión, con privilegios suficientes, tal vez excesivos, para realizar correctamente el almacenamiento de las imágenes. Si quisiéramos restringir estos permisos, se puede asignar otra cuenta configurando el servicio Apache desde "Administración de Equipos" (pinchando con el botón dere-

cho del ratón sobre “Mi PC”, seleccionando “Administrar”, o desde el panel de control en “Herramientas Administrativas) tal como se muestra en la figura 7.16:

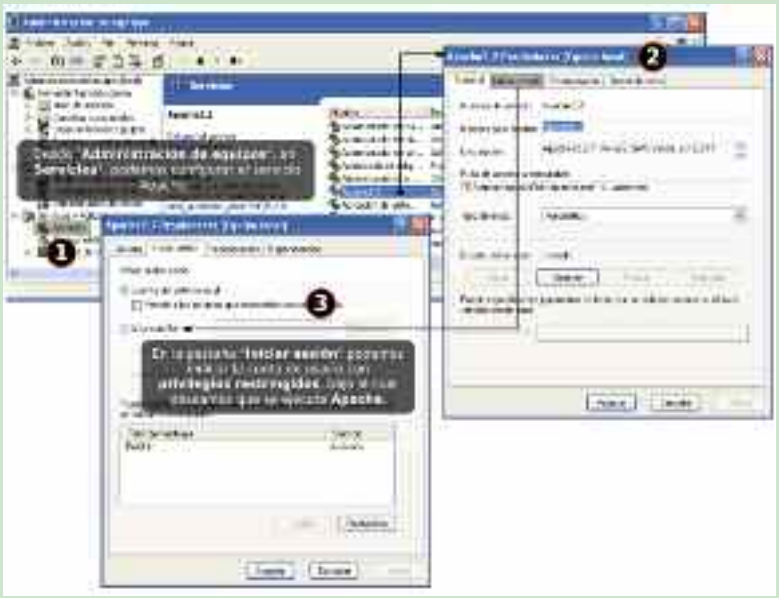


Figura 7.16. Forma de restringir los privilegios con que funciona Apache en Windows

En el caso de GNU/Linux, la directiva User en Apache bajo Xampp tiene asignarda la cuenta de usuario “nobody”. Esto implica que será necesario asignar permisos de escritura a “nobody” sobre el directorio “img-coches” modificando la lista de control de acceso al directorio (ACL) o modificando los permisos del sistema. La forma más sencilla, con el riesgo que eso conlleva sería asignar permisos de escritura a todo usuario sobre ese directorio:

```
[root@linux]# chmod o+w /opt/lampp/htdocs/sitiophp/img-coches
```

Una vez subida la imagen al servidor, este comprobará su correcto estado, devolviendo un mensaje al cliente que informe de ello, junto con la imagen subida. En caso de que hayan producido problemas se informará igualmente de ello.

Figura 7.17. Comunicación entre cliente y servidor en el segundo formulario

Solución ejercicio n.º 3

Para dar solución al ejercicio propuesto realizaremos las siguientes actuaciones:

1. Modificaremos el documento *formulario.xhtml* realizado en los ejercicios prácticos anteriores para indicar de manera explicita quien es el destinatario de los datos enviados por ambos formularios (*action="preparar-datos.php"*).

```
<form method="post" action="presenta-datos.php" ...>
```

2. Crearemos el documento PHP presenta-datos.php. Su contenido podría ser el siguiente:

```
<html><head><title>Taller Mecánico</title>
<meta http-equiv="Content-Type" content="text/html;charset=UTF-8" />
<link href='imagenes/logo.png' rel='shortcut icon' type='image/png'/>
<!— Importamos las hojas de estilo que serán utilizadas por el documento —>
<link rel="stylesheet" type="text/css" href="css/formulario.css" media="screen" />
<link rel="stylesheet" type="text/css" href="css/presenta-datos.css" media="screen" />
</head>
<body>
<h2 class="titulo"> TALLER MECANICO - Datos Formulario </h2>
<!— Comprobamos si el botón de submit pulsado es del primer formulario y la contraseña —>
<? if ($_POST['boton'] == "Almacenar" && $_POST['pass'] == "mipass" )
{ echo "<hr />Se han recibido los siguientes datos desde el equipo <span class='dato'>".$_SERVER['REMOTE_ADDR']."</span> vía <span class='dato'>".$_SERVER['REQUEST_METHOD']."</span><hr />"; ?>
<table> <tr class="cabecera-tabla"><td>Datos personales</td><td>Datos Reparación</td>
<td>Comentarios</td></tr>
<tr class="cuerpo-tabla">
<td> <? echo "Nombre: <span class='dato'>".$_POST['nombre']."</span>";
echo "<br />Sexo: <span class='dato'>".$_POST['sexo']."</span>";
echo "<br />Año Nacimiento: <span class='dato'>".$_POST['edad']."</span>";
echo "<br />Direccion: <span class='dato'>".$_POST['direccion']."</span>";
echo "<br />Matricula: <span class='dato'>".$_POST['matricula']."</span>"; ?> </td>
<td><? echo "Fecha Reparación: <span class='dato'>".$_POST['fecha']."</span>";
if (isset($_POST['tipo']))
{ echo "<br />Arreglos:";
  foreach ($_POST['tipo'] as $indice => $valor)
  { echo "<br /><span class='dato arreglo'>".$valor."</span>"; }
}
if (isset($_POST['extra']))
{ echo "<br />Extras:";
  foreach ($_POST['extra'] as $indice => $valor)
  { echo "<br /><span class='dato'>".$valor."</span>"; }
} ?></td>
<td><? echo "<span class='dato'>".$_POST['comentarios']."</span>"; ?></td></tr></table><? }
?>
<!— Comprobamos si el botón de submit pulsado es del segundo formulario —>
<? if ( $_POST['boton'] == "Guardar Imagen") {
  if (is_uploaded_file($_FILES['imagen']['tmp_name']))
  {
```

```
  echo "<hr />Se ha recibido el archivo <span class='dato'>".$_FILES['imagen']['tmp_name']."</span> con nombre <span class='dato'>".$_FILES['imagen']['name']."</span> y tipo de imagen <span class='dato'>".$_FILES['imagen']['type']."</span> desde el equipo <span class='dato'>".$_SERVER['REMOTE_ADDR']."</span> vía <span class='dato'>".$_SERVER['REQUEST_METHOD']."</span><hr />";
  $ruta="img-coches/".$_FILES['imagen']['name'];
  if (move_uploaded_file($_FILES['imagen']['tmp_name'],$ruta))
  {
  echo "<p class='resultado'> La imagen <span class='dato'>".$_FILES['imagen']['name']."</span> se guardo perfectamente en <span class='dato'>".$ruta."</span>"; ?>
<br /><br /><img src="<? echo $ruta; ?>" class="vehiculo" /></p>
  <? }
  else { echo "<p class='resultado'>Se han producido problemas al tratar de guardar la imagen en ".$ruta."</p>"; }
}
else
{ echo "El archivo se ha recibido correctamente en el servidor...."; }
} ?><hr /><a href="formulario.xhtml">Volver al formulario</a></body></html>
```

¡¡Importante!! Las comillas dobles utilizadas en la asignación de un valor a un atributo HTML se convierten en simples en el caso de estar encerrado por otras dobles correspondientes a un comando PHP: <? echo "<p class='parrafo1'> ...</p>"; ?>

En relación al documento PHP anterior aclarar los siguientes aspectos:

— Para distinguir cual de los botones de *submit* ha sido pulsado, correspondientes a los dos formularios definidos en el documento *XHTML*, al tener los dos el mismo *name* asignado (*name="boton"*), comprobamos si el valor asignado (*value*) se corresponde con el valor recogido por la variable del vector **$_POST**['*name-boton-submit*']:

```
if ($_POST['boton'] == "Almacenar" && $_POST['pass'] == "mipass" ) { ... }
if ( $_POST['boton'] == "Guardar Imagen") { ... }
```

En el caso de que los botones hubieran tenido diferente *name* asignado (*por ejemplo, name="boton1" y name="boton2"*), tan sólo hubiera sido necesario comprobar con la función *isset()* si el vector **$_POST** ha recibido algún valor del formulario de un campo con el nombre correspondiente a cada botón:

```
if ( isset($_POST['boton1']) && $_POST['pass'] == "mipass" ) { ... }
if ( isset($_POST['boton2']) ) { ... }
```

— Tras comprobar que se ha pulsado el botón *submit* asociado al primer formulario (***$_POST****['boton']* == *"Almacenar"*) y que la contraseña introducida es correcta (***$_POST****['pass']* == *"mipass"*), se presenta una tabla HTML con los datos introducidos en el formulario. Respecto a estos, destacar el tratamiento de los datos recibidos correspondientes a la lista de selección múltiple, <*select name="tipo[]" multiple="multiple"*>, y los check-box, <*input type="checkbox" name="extra[]"* ... />. En ambos casos, al poder

recibirse varios datos bajo el mismo *name*, son gestionados por los vectores **$_POST**[‘tipo’] (***$_POST[‘tipo’][0], $_POST[‘tipo’][1],*** ...) y **$_POST**[‘extra’] (***$_POST[‘extra’][0], $_POST[‘extra’][1],*** ...).

```
if (isset($_POST[‘tipo’]))
{ echo “<br />Arreglos:”;
  foreach ($_POST[‘tipo’] as $indice => $valor)
  { echo “<br /><span class=’dato arreglo’>”.$valor.”</span>”; }
}
if (isset($_POST[‘extra’]))
{ echo “<br />Extras:”;
  foreach ($_POST[‘extra’] as $indice => $valor)
  { echo “<br /><span class=’dato’>”.$valor.”</span>”; }
}
```

Se comprueba en primer lugar si se ha seleccionado alguna de las opciones mediante *isset()*, ***isset($_POST[‘tipo’])*** y ***isset($_POST[‘extra’])***, y posteriormente se recorren los vectores mediante un bucle *foreach*.

— En el caso de que el botón de submit pulsado sea el del segundo formulario, ***$_POST[‘boton’] == “Guardar Imagen”***, se comprobará mediante la función ***is_uploaded_file()*** si el archivo seleccionado a través del campo del formulario *<input type=”file” name=”imagen” id=”imagen” />* se ha subido (*upload*) correctamente. Advertir, que una vez subido al servidor, el archivo se almacena temporalmente en él como ***$_FILES[‘imagen’][‘tmp_name‘]***.

```
if (is_uploaded_file($_FILES[‘imagen’][‘tmp_name‘])) { ... }
```

En caso afirmativo, mediante la función ***move_uploaded_file()***, el fichero subido es reubicado al subdirectorio *img-coches* creado para ello, bajo su nombre original, ***$_FILES[‘imagen’][‘name‘]***.

```
$ruta=”img-coches/”.$_FILES[‘imagen’][‘name‘];
if (move_uploaded_file($_FILES[‘imagen’][‘tmp_name‘],$ruta)) { ... }
```

— El final de documento *PHP* se corresponde con un enlace *HTML* de vuelta al documento *formulario.xhtml*.

```
<a href=”formulario.xhtml”>Volver al formulario</a>
```

¡¡Importante!! Advierte que el código PHP del documento es ejecutado por el servidor siendo transparente para el cliente Web o navegador. Es decir, si observas el código fuente de la página PHP que se carga en el navegador[56], podrás comprobar que no hay rastro de ninguna línea de código PHP, ya que el servidor se encargo de ejecutarlo, y devolver al cliente el resultado de su ejecución mediante etiquetas HTML.

56 En la mayoría de los navegadores (por ejemplo, *Mozilla Firefox*), una vez cargado un documento Web, pinchando con el botón derecho del ratón sobre él es posible encontrar una opción similar a “Ver el código fuente de la página” que nos permitirá conocer el código interpretado por el navegador.

3. En cuanto a las hojas de estilo utilizadas por el documento PHP, *formulario.css*, fue creada para el documento *XHTML formulario.xhml*, y *presenta-datos.css*, es utilizada para asignar determinadas propiedades de estilo a las tabla encargada de presentar los datos recibidos por el formulario y sus contenidos. Un ejemplo podría ser:

```
/* Contenido de la hoja de estilos CSS presenta-datos.css */
body { font-family: helvetica, sans-serif; }
table { margin-left: 3%; margin-right: 3%; border: 1px; }
tr.cabecera-tabla { background-color: yellow; font-family: courier, monospace; text-align:
center; font-weight: 800; }
tr.cuerpo-tabla { background-color: white; }
td { width: 30%; }
span.dato { color: red; font-weight: 800; }
.arreglo { font-style: italic; }
.centrado { text-align: center; }
p.resultado { text-align: center; }
img.vehiculo { width: 400px; height: 300px; padding: 5px; border: 1px solid gray; back-
ground-color: white; }
```

5. SESIONES Y COOKIES EN PHP

En el intercambio de datos entre el cliente y el servidor mediante formularios puede resultar interesante almacenar determinada información para evitar tener que solicitárla repetidas veces. Esta información suele ser utilizada por el servidor para personalizar los contenidos Web que suministra al cliente. Un ejemplo muy típico, son los servicios de correo electrónico vía Web (*WebMail*) o las llamadas redes sociales de Internet, donde tras registrarse mediante un pequeño formulario (*login y password*), puede navegarse por el sitio Web, bajo contenidos totalmente personalizados, sin tener que volver a autenticarse durante toda la sesión.

¡¡Aclaración!! Una sesión es el intervalo de tiempo transcurrido desde que un usuario ingresa en la página de inicio de un determinado sitio Web, hasta que lo abandona. Además, dentro de un sitio Web pueden establecerse diferentes sesiones, tal como ocurre en sitios Web bancarios, en tiendas virtuales o cualquiera que este basado en un sistema gestor de contenidos (CMS, *Content Management System*), donde se accede inicialmente bajo una sesión de invitado, pudiendo registrarse, pasando a iniciarse una nueva sesión con unos privilegios diferentes.

En PHP se dispone de las llamadas variables de sesión, organizadas bajo el vector $_SESSION, el cual actúa de manera similar a los campos ocultos de los formularios, <*input type="hidden"* />, donde de manera transparente para el usuario, se almacenan valores que se van pasando de una página a otra durante la navegación de un sitio Web mientras no se termina con la sesión abierta.

— $_SESSION: Variable de tipo vector o *array* que permite almacenar información durante la sesión establecida por un usuario al visitar el sitio Web de un determinado dominio que podrá ser consultada desde cualquiera de las páginas que se visiten dentro de la misma sesión.

Es decir, las variables de sesión nos permiten guardar determinada información que se haya solicitado al usuario a través de algún formulario, y que puede ser interesante tenerla presente durante la navegación del resto de páginas que componen el sitio Web:

```
<? session_start(); ?>
...
<form method="post" action="">
Tu estilo de música preferido es: <input type="text" name="estilo" size="15" />
<input type="submit" name="boton" value="enviar"></form>
<? if ( $_POST['boton'] == "enviar")
{ $_SESSION['estilomusical']=$_POST['estilo']; } ?>
```

En el ejemplo anterior, una página PHP mediante un formulario HTML pregunta al usuario por sus gustos musicales, y la respuesta indicada es enviada al servidor y almacenada en una variable de tipo sesión, **$_SESSION**[*'estilomusical'*], pudiendo ser utilizada esta información para personalizar el resto de las páginas por las que navegue dentro del sitio Web en función del gusto declarado. Como puede observarse, para poder hacer uso de estas variables de sesión es imprescindible introducir como primera línea de todos los documentos que componen el sitio Web **<? session_start(); ?>**, función que se encarga de inicializar la sesión y mantenerla durante la navegación del sitio Web, transmitiendo el contenido del vector **$_SESSION** de una página a otra.

Entre las funciones PHP asociadas a la gestión de sesiones podrían destacarse las que figuran en la tabla:

Función PHP	Descripción
session_start()	Inicia y mantiene la sesión establecida por un usuario, permitiendo almacenar información en el *array* $_SESSION. Es necesario colocarla al principio del documento, anes de cualquier etiqueta HTML o línea en blanco.
session_destroy()	Destruye toda la información almacenada relacionada con la sesión actual.
session_unset()	Elimina y libera el espacio ocupado por todas las variables de sesión registradas. Si lo que se quiere es eliminar una única posición del array o variable de sesión haremos uso de la función *unset*: *<? unset ($_SESSION['nombre_variables']); ?>* Por el contrario para saber si una variable existe se hace uso de la función *isset.*
session_id()	Devuelve el identificador de sesión (*SID, IDentifier Session*) compuesto por 32 caracteres.
session_save_path()	Devuelve la ruta del directorio dentro del servidor donde es almacenada toda la información relacionada con la sesión. Esta ruta es configurable editando el fichero */opt/lampp/etc/php.ini* en GNU/Linux o *c:/xampp/php/php.ini* en Windows, modificando la línea referente a "**session. save_path**=*ruta*".

Junto a las variables de sesión, **$_SESSION**, también es importante destacar a las cookies, **$_COOKIE**. Estas últimas se corresponden con pequeños ficheros de información que se almacenan en el equipo cliente a petición del servidor, que permiten almacenar información que puede ser consultada por el servidor tanto en la sesión activa como en futuras sesiones, siendo el navegador Web el responsable de su gestión[57].

57 En un mismo equipo cliente podemos tener instalados diferentes navegadores (Mozilla Firefox, Internet Explorer, Opera, etc.), los cuales gestionan sus propias cookies no siendo compartidas entre ellos.

¡¡Importante!! Aunque tradicionalmente las cookies han sido utilizadas de manera similar a las variables de sesión, para almacenar determinada información del usuario que permita facilitar el acceso a los contenidos del sitio Web, un uso actual de las cookies es para realizar *Business Intelligence*. Es decir, gracias a la información proporcionada por las cookies a los servidores Web, estos pueden hacer un estudio estadístico de nuestros hábitos de navegación (*frecuencia de acceso, tipos de contenidos consultados, horas de acceso, etc.*), permitiéndoles generar un perfil de usuario de gran utilidad para las empresas de publicidad. Un ejemplo muy cotidiano de ello es el portal de *youtube*, donde al iniciar sesión en este sitio Web se nos ofrece unos vídeos que están relacionados con los gustos manifestados en visitas anteriores al mismo portal. Esto es posible gracias a que las cookies le permiten conocer al servidor quien esta accediendo, y en base a la información que tiene almacenada el servidor en una base de datos sobre nuestra historia de accesos anteriores, que podría agradarnos más. Estas técnicas de *Business Intelligence* están siendo utilizadas hoy en día de manera masiva, y cada vez más, sin que nos demos cuenta, por todo tipo de empresas en la promoción de viajes, venta de coches, etc.

— **$_COOKIE**: Variable de tipo vector o *array* que permite consultar información en el equipo cliente fue almacenada previamente mediante la función *setcookie()*. Esta función es la función más importante en la gestión de cookies, y es utilizada para su creación y asignación de valor. Al igual que <*? **session_start();** ?*>, debe colocarse al principio del documento antes de cualquier etiqueta HTML o línea en blanco.

setcookie (*nombre_cookie, valor, tiempo_vida, path, dominio, https*);	
nombre_cookie	Nombre identificador de la cookie. Será utilizado para rescatar su valor.
valor	Valor que se asignará a la cookie identificada con *nombre_cookie*. Por ejemplo, si es servidor quisiera almacenar temporalmente en el equipo cliente el DNI suministrado a través de un formulario para por ser consultado cuando requiera durante la sesión establecida: **if** (**isset** (**$_POST**[‘DNI’])) { **setcookie** (“midni”,**$_POST**[‘DNI’]); } En el caso de que queramos inhabilitar una cookie, tan sólo será necesario hacer uso de la función *setcookie()* asignándole un valor vacío.
tiempo_vida	Indica cual será la fecha de caducidad de la cookie en segundos. Para su asignación se hace uso de la función PHP *time()*, la cual nos suministra la fecha actual[58], más el número de segundos de vida que queremos dar a la cookie. Si no se indica valor, la cookie expirará al finalizar la sesión. Por ejemplo si quisiéramos almacenar durante una semana la información asociada a un gusto declarado por un usuario, para poder ser consultado en futuras sesiones por el mismo servidor: **setcookie** (“miaficion”,**$_POST**[‘aficion’],**time()** + (3600 * 24 * 7));
path	Indica el directorio desde el cual un script PHP puede recuperar el valor asignado a una cookie. Por defecto, si no se indica valor, podrán consultar su valor los documento PHP que se encuentren en el mismo directorio que el que la genero

58 La función *time()* nos informa de la fecha actual en relación a las 00:00:00 del día 1 de enero de 1970 GMT, indicando el número de segundos transcurridos desde entonces.

	mediante *setcookie()*. Destacar los siguientes valores para este parametro: • "/": Desde cualquier documento PHP del servidor podrá ser consultado el valor asignado a la cookie. • "/nombre_directorio/": Solo podrá consultarse por los documentos PHP que se encuentren dentro del directorio especificado, o subdirectorios de este. • "/nombre_directorio/subdirectorio/": Solo podrá consultarse por los documentos que se localicen en el subdirectorio especificado.
dominio	Establece el espacio de nombres de dominio de equipos servidores que podrán consultar el valor asignado a la cookie. Por defecto, si no se indica valor, se le asignará el nombre de dominio asociado a la página: "http://www.midominio.es/mipagina.php" -> "midominio.es" Pudiendo acceder en ese caso desde www.midominio.es, web.midominio.es, webmail.midominio.es, etc.
https	Parámetro booleano (*0,1*) que establece que la cookie solo puede crearse y consultarse en el caso de que el protocolo que se utilize sea seguro: *HTTPS*, HTTP en modo seguro. Por defecto, si no se indica lo contrario, no requerirá seguridad, 0. Un ejemplo sería el siguiente: **setcookie** ("ncueta",**$_POST**['ncuenta'],**time()** + (3600 * 24), "", "", 1);

La función *setcookie()* devuelve un valor booleano informándonos de si la cookie se creo con éxito o si hubo, pudiendo ser comprobado este aspecto con una sentencia *if()*:

```
if ( isset ( $_POST['DNI'] ) ) {
  if ( !setcookie ("midni",$_POST['DNI']) )
  { echo "Problemas al crear la cookiee!!!"; }
}
```

Este aspecto es interesante tenerlo en cuenta ya que el almacenamiento de cookies en el equipo cliente es dependiente de la configuración de nuestro navegador. Por ejemplo, en el navegador Mozilla Firefox, desde el menú Editar, la opción Preferencias, podemos configurar su comportamiento frente a las cookies.

Figura 7.18. El navegador nos permite gestionar las cookies almacenadasen el equipo cliente

¡¡Observación!! Puede observarse en la lista de cookies registradas en el navegador que al trabajar con sesiones en PHP, se crea suna cookie con nombre **PHPSESSID** que contiene el identificador de sesión (SID) activa, la cual puede ser consultada mediante **$_COOKIE**['PHPSESSID']. Su tiempo de vida se corresponde con la duración de la sesión. Si esta cookie es eliminada la sesión se eliminará, y con ella todas las variables de sesión que tenga almacenadas el servidor. Por ello, al finalizar la sesión, la cookie expira y con ella las variables asociadas.

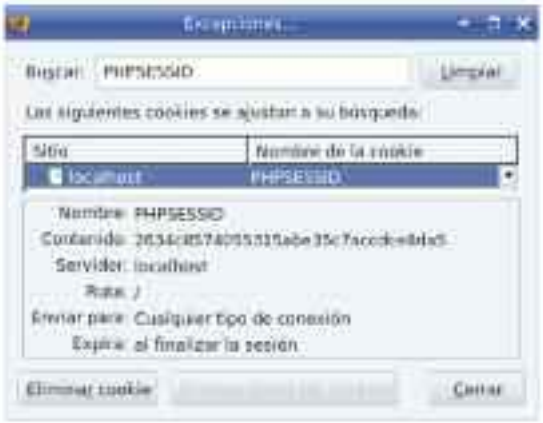


*Figura 7.19. PHPSESSID es una cookie creada por el servidor en el cliente al iniciar una sesión con <? **session_start()**; ?> que almacena el SID*

Ejercicio práctico n.º 4

Crea tres documentos PHP dentro del mismo sitio Web que has utilizado para el resto de ejercicios prácticos del presente capítulo y comprueba el funcionamiento de las variables de sesión y cookies, presentando las siguientes características:

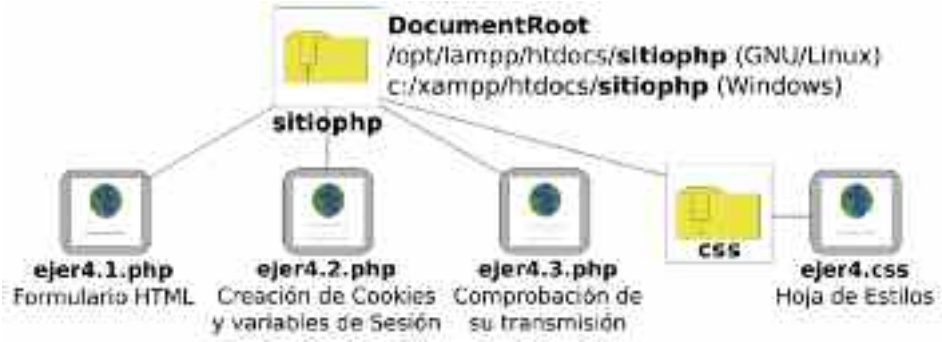


Figura 7.20. Estructura del sitio Web para el ejercicio práctico n.º 4

a) **ejer4.1.php**: Estará compuesto por un formulario HTML donde se le solicitará un nombre y un DNI. Vía post se enviarán estos datos al servidor, donde el documento PHP **ejer4.2.php** será el encargado de su gestión.

```
<form method="post" action="ejer4.2.php">
Nombre: <input type="text" name="nombre" size="20" />
DNI: <input type="text" name="DNI" size="10" maxlength="9" />
<input type="submit" name="boton" value="enviar" />
</form>
```

b) **ejer4.2.php**: El nombre recibido por el formulario del **ejer4.1.php** será almacenado en una variable de sesión, **$_SESSION**[‘nombre’], y el DNI en una cookie llamada “midni”. Para comprobar la transmisión de los datos introducidos a través de las variables de sesión y cookies durante la navegación del sitio Web, pudiendo ser utilizados por todas las páginas que lo componen, habrá un enlace al documento **ejer4.3.php** donde se hará uso de ellos.

```
<? session_start();
if (isset($_POST[‘boton’]))
{ if (isset($_POST[‘nombre’])) { $_SESSION[‘nombre’]=$_POST[‘nombre’]; }
  if (isset($_POST[‘DNI’])) {
            if (setcookie(“midni”,$_POST[‘DNI’],time() + 3600))
                    { echo “Problemas al crear la cookie!!!!”; }
  }
}
?>
```

c) **ejer4.3.php**: Comprobaremos que sin la necesidad de volver a solicitar los datos anteriores nombre y DNI mediante un nuevo formulario, están disponibles en a través de las variables de sesión y cookies.

```
<? session_start(); ?>
<? if (isset($_SESSION[‘nombre’]))
{ echo “<br />El nombre que introdujiste durante la sesión fue: “.$_SESSION[‘nombre’]; }
if (isset($_COOKIE[‘midni’]))
{ echo “<br />La cookie almacenada correspondiente al vale: “.$_COOKIE[‘midni’]; } ?>
```

Tal como se muestra en la solución del ejercicio, una vez creadas la variable de sesión y cookie, mediante el uso de enlaces de navegación entre las distintas páginas, puede observarse que los valores puedes ser utilizados desde cualquiera de los documentos.

Solución ejercicio n.º 4

Según las características anteriores el contenido completo de los documentos PHP podría ser el siguiente (la primera de las líneas de los documentos PHP antes de cualquier etiqueta HTML o línea en blanco será <? ***session_start();*** ?>, encargada de iniciar la sesión y mantenerla):

— **ejer4.1.php**:

```
<? session_start(); ?>
<html><head><title>Sesiones y Cookies</title>
<meta http-equiv=”Content-Type” content=”text/html;charset=UTF-8” />
<link href=’imagenes/logo.png’ rel=’shortcut icon’ type=’image/png’/>
<link rel=”stylesheet” type=”text/css” href=”css/ejer4.css” media=”screen” /></head>
<body><h2 class=”titulo”>SESIONES y COOKIES  - PAG Nº1</h2>
<fieldset><legend>Formulario - Datos Usuario</legend>
<form method=”post” action=”ejer4.2.php”>
Nombre: <input type=”text“ name=”nombre” size=”20” />
DNI: <input type=”text“ name=”DNI” size=”10” maxlength=”9” />
```

```
<input type="submit" name="boton" value="enviar" />
</form></fieldset>
<p class="variables">
<!— Durante la navegación se comprueba si las variables están disponibles —>
<? if (isset($_SESSION['nombre']))
{ echo "<br />El nombre que introdujiste durante la sesión fue: ".$_SESSION['nombre']; }
if (isset($_COOKIE['midni']))
{ echo "<br />La cookie almacenada correspondiente al vale: ".$_COOKIE['midni']; } ?></p>
<p class="enlaces"><a href="ejer4.2.php">Ir a la página 2</a>
<br /><a href="ejer4.3.php">Ir a la página 3</a></p></body>
```

— **ejer4.2.php**:

```
<? session_start();
if (isset($_POST['boton']))
{  if (isset($_POST['nombre'])) { $_SESSION['nombre']=$_POST['nombre']; }
   if (isset($_POST['DNI'])) {
            if (!setcookie("midni",$_POST['DNI'],time() + 3600))
                    { echo "Problemas al crear la cookie!!!!"; }
   }
}
?>
<html><head><title>Sesiones y Cookies</title>
<meta http-equiv="Content-Type" content="text/html;charset=UTF-8" />
<link href='imagenes/logo.png' rel='shortcut icon' type='image/png'/>
<link rel="stylesheet" type="text/css" href="css/ejer4.css" media="screen" /></head>
<body><h2 class="titulo">SESIONES y COOKIES - PAG Nº2</h2>
<p class="variables">
<? if (isset($_SESSION['nombre']))
{ echo "<br />El nombre que introdujiste durante la sesión fue: ".$_SESSION['nombre']; }
if (isset($_COOKIE['midni']))
{ echo "<br />La cookie almacenada correspondiente al vale: ".$_COOKIE['midni']; } ?></p>
<p class="enlaces"><a href="ejer4.1.php">Ir a la página 1</a>
<br /><a href="ejer4.3.php">Ir a la página 3</a></p></body>
```

— **ejer4.3.php**:

```
<? session_start(); ?>
<html><head><title>Sesiones y Cookies</title>
<meta http-equiv="Content-Type" content="text/html;charset=UTF-8" />
<link href='imagenes/logo.png' rel='shortcut icon' type='image/png'/>
<link rel="stylesheet" type="text/css" href="css/ejer4.css" media="screen" /></head>
<body><h2 class="titulo">SESIONES y COOKIES - PAG Nº3</h2>
<p class="variables">
<? if (isset($_SESSION['nombre']))
{ echo "<br />El nombre que introdujiste durante la sesión fue: ".$_SESSION['nombre']; }
if (isset($_COOKIE['midni']))
{ echo "<br />La cookie almacenada correspondiente al vale: ".$_COOKIE['midni']; } ?></p>
<p class="enlaces"><a href="ejer4.1.php">Ir a la página 1</a>
<br /><a href="ejer4.2.php">Ir a la página 2</a></p></body>
```

— **ejer4.css**:

```
body { background-color: GreenYellow; padding: 2cm; padding-top: 1cm; }
h2.titulo { border: gray solid 2px; background-color: white; padding: 5px; text-align: center;
color: tomato; width: 50%; margin-left: auto; margin-right: auto; }
fieldset {margin-left: auto; margin-right: auto; background-color: OliveDrab;}
legend { font-style: italic; font-weight: 600; color: Brown; background-color: GreenYellow;}
p.variables { background-color: brown; color: orange; border: 2px solid white; padding: 10px;
}
p.enlaces { text-align: center; background-color: yellow; width: 40%; margin: auto; border:
1px dotted brown; }
```

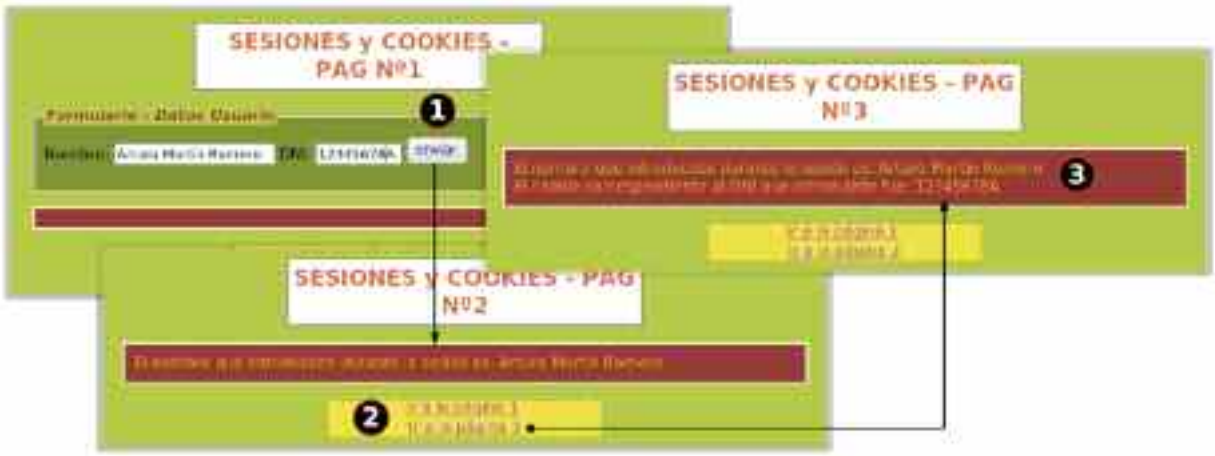


Figura 7.21. Capturas de pantalla del ejercicio práctico n.º 3

Tras la realización del ejercicio práctico anterior pueden comprobarse los siguientes aspectos:

1. Que las variables de sesión y cookies tras su creación, se propagan por todo el sitio Web pudiendo ser consultadas en cualquiera de las páginas que lo componen.
2. Que la cookie **PHPSESSID** queda registrada en el navegador tras iniciar una sesión, *<? **session_start()**; ?>*. Si eliminamos la cookie de nuestro navegador puede comprobarse que las variables asociadas a dicha sesión se eliminan.
3. Que la cookie "midni" queda registrada en el navegador Web con el valor asignado, y que expirará transcurrido el tiempo de vida indicado en *setcookie()*.

Figura 7.22. Comprobación del registro de la cookie en el navegador Mozilla Firefox

¡¡Observación!! Es importante advertir que la variable de sesión **$_SESSION** [‘nombre’] esta disponible en el mismo momento de su creación, al ser asignado su valor por el propio servidor al ejecutar el código PHP. Por el contrario, la cookie **$_COOKIE** [‘midni’] no esta disponible tras ejecutar la función setcookie(), ya que en primer lugar la función setcookie() realiza la petición de creación de dicha cookie al navegador del equipo cliente, y en caso exitoso, estará disponible a partir de ese momento.

Capítulo 8

PHP Y MySQUL

Tal como ya se ha ido advirtiendo en los últimos capítulos, la nueva Web 2.0 no se concibe sin la integración de una base de datos dentro del sitio Web. La gran cantidad de datos e información que contienen hace que sea necesaria la ayuda de algún sistema que facilite su gestión, permitiéndonos almacenarla de manera organizada y eficiente, y recuperarla lo más rápidamente posible. Al ser estas las características más básicas que cumple todo buen sistema gestor de bases de datos (*SDBM, System DataBase Management*), hace de estos el aliado perfecto para la gestión de nuestros sitios Web. Aunque los hay que presentan excelentes características, con mucho prestigio y solera, como ORACLE o Microsoft SQL Server, el coste de sus licencias ha provocado la búsqueda de otras alternativas dentro del software el software libre (*licencia GPL*), como MySQL.

Gracias al apoyo ofrecido por MySQL en la gestión de contenidos de sitios Web, convertido ya en un estándar de facto[59], le ha convertido en uno de los gestores de bases de datos más afamados en la actualidad. Entre sus características más importantes destacar:

— *Licencia GPL*: garantiza la libre distribución y modificación de su código fuente para los fines que el usuario crea convenientes sin tener que dar explicación a nadie. Esto hace que las bases de datos MySQL se utilicen hoy en día en prácticamente cualquier tipo de aplicación Web con fines muy diversos.

— *Eficiente comunicación con PHP*: operaciones como la recuperación (*select*), inserción (*insert*), actualización (*update*) y borrado (*delete*) de datos e información almacenados en una base de datos MySQL desde una aplicación Web haciendo uso de PHP son realizadas de una manera muy sencilla y eficiente. Esto se traduce en que sin apenas retardos, la información es obtenida por el servidor Web y entregada al cliente, siendo totalmente transparente para el usuario final la complejidad de los procesos involucrados.

Figura 8.1. Ejemplo de solicitud de una página de un sitio Web cuyos datos están gestionados por un SDBM

59 Los estándares de facto son aquellos que se establecen sin una planificación previa por parte de las instituciones de estandarización, cuando un determinado producto o modus operandi se extiende, llegando a ser considerado como la opción por defecto.

En relación a la ubicación del sistema gestor de bases de datos, puede localizarse en un equipo servidor independiente, o encontrarse instalado en la propia máquina local (*localhost*) donde también se encuentra configurado el servicio Web. En el presente capítulo, se asumirá esta segunda alternativa, al tener en cuenta que la herramienta software multiservicio instalada, Xampp, viene con el SDBM MySQL integrado.

1. CONTENIDOS DEL CAPÍTULO

El presente capítulo presenta la siguiente estructura:

— En primer lugar se indica el software necesario para la implementación de los ejemplos y ejercicios prácticos propuestos a lo largo del capítulo.

— A continuación se indica como preconfigurar el gestor de bases de datos MySQL y su herramienta de gestión Web phpMyAdmin, antes de trabajar con ambos.

— Se muestra como gestionar las bases de datos y tablas MySQL de una manera básica.

— Termina mostrando las funciones PHP que permiten interactuar con MySQL, con un ejercicio práctico que aclarará todo lo visto.

2. SOFTWARE DE DESARROLLO

Para la realización de los ejercicios prácticos que se proponen en el capítulo será necesario tener instalado y habilitado el servicio **MySQL**, su herramienta de gestión Web **phpMyAdmin**, y el módulo PHP que permite interactuar con bases de datos MySQL (*php-mysql*), todos ellos integrados en *Xampp*. Su comprobación dependerá del sistema operativo bajo el cual se ha instalado:

— Microsoft Windows: Al haberse instalado Xampp como una aplicación más del sistema, y formar parte de los servicios ofrecidos por este, su comprobación se hará consultando su estado desde *Panel de control* / *Herramientas Administrativas* / *Servicios*, o pinchando con el botón derecho del ratón sobre el icono de *Mi PC*, seleccionando *Administrar*, dentro de la interfaz de *Administración de equipos*, la opción *Servicios*:

Figura 8.2. Comprobación del estado del servicio MySQL en Windows

Pinchando con el botón derecho del ratón sobre el servicio *mysql* podremos gestionar su estado: *iniciar* / *detener* / *pausar* / *reiniciar*.

— GNU/Linux: Al no haberse instalado Xampp como una aplicación del sistema, no forma parte de los servicios que pueden encontrarse en */etc/init.d*. Por ello, la forma de comprobar su estado es ejecutando el script *lampp* pasando como parámetro *status*:

```
[root@linux]$ /opt/lampp/lampp status
Version: XAMPP for Linux 1.7
Apache is running.
MySQL is running.
ProFTPD is running.
```

En el caso de que los servicios integrados en Xampp bajo GNU/Linux no se encuentren activos (*running*) será necesario iniciarlos ejecutando el mismo script pasándole como parámetro *start* o *restart*.

```
[root@linux]$ /opt/lampp/lampp start|restart
Starting XAMPP for Linux ...
XAMPP: Starting Apache with SSL (and PHP5)...
XAMPP: Starting MySQL...
XAMPP: Starting ProFTPD...
XAMPP for Linux started.
```

Una vez comprobado que el servicio MySQL esta iniciado, puede comprobarse su herramienta de gestión Web *phpMyAdmin*. Para ello será necesario tener iniciado el servicio Apache dentro de Xampp, al ser este el que le da soporte. Después, tan sólo será necesario introducir en la barra de direcciones de nuestro navegador preferido la URL "*http://localhost/phpmyadmin/*".

Figura 8.3. phpMyAdmin es la herramienta de gestión vía Web de MySQL

En el caso de haber optado por la instalación de cada uno de estos paquetes software de manera independiente, en GNU/Linux suelen estar disponibles para su instalación en el DVD de la propia distribución, pero en el caso de estar trabajando con Microsoft Windows, será necesario descargarlos de la Web (*http://www.mysql.com*).

3. PRECONFIGURACIÓN DE MYSQL Y PHPMYADMIN

MySQL es un sistema gestor de bases de datos (SDBM) que puede ser configurado a través de la consola o una interfaz de comandos de nuestro sistema operativo, o vía Web, mediante *phpMyAdmin*. Por cualquiera de estas vías puede comprobarse que la cuenta de administrador de MySQL o usuario root puede administrarlo sin requerir de una contraseña o password. Esto implica graves problemas de seguridad para los datos almacenados en las bases de datos gestionados por MySQL, por lo que antes de empezar a trabajar es necesario asignar una contraseña al usuario root. Para ello seguiremos los siguientes pasos:

1. Abriremos una consola en nuestro sistema operativo y ejecutaremos la aplicación *mysql* que nos dará acceso a la interfaz de comandos de administración del SDBM, el cual se caracteriza por mostrar un *prompt* propio (*mysql>*):

```
[root@localhost]# /opt/lampp/bin/mysql (en GNU/Linux)
c:\xampp\mysql\bin> mysql.exe (en Windows)
mysql>
```

2. Toda la información relacionada con las cuentas de usuario MySQL, sus contraseñas, sus permisos y privilegios, las máquinas desde las que pueden acceder, etc. se encuentra en una base de datos que vuelve a llamarse igual que el propio SDBM, *mysql*. Por ello, para asignar una contraseña al root, será necesario en primer lugar indicar que queremos hacer uso de esa base de datos mediante el comando mysql *use*:

```
mysql> use mysql
Database changed
```

¡¡Observación!! Todas las bases de datos gestionadas por MySQL podemos encontrarlas en */opt/lampp/var/msql* en GNU/Linux, o *c:\xampp\mysql\data* en Microsoft Windows. Cada base de datos tiene asociado un directorio que contiene ficheros con información asociada a cada una de las tablas que la forman (*bases de datos del tipo MyISAM*). Destacar a los ficheros con extensión **.frm*, al ser estos los que contienen información sobre la estructura de las tablas, **.myd*, los datos almacenados en estos, y **.myi* sus índices. Entre las tablas que forman parte de la base de datos mysql destacar la tabla user (*user.frm, user.myd, user.myi*), la cual alberga los usuarios, sus contraseñas, permisos, etc.

Para mostrar las tablas que componen la base de datos mysql ejecutaremos el comando show tables:

```
mysql> show tables;
```

3. Modificaremos la contraseña del usuario *root* mediante la sentencia SQL *update*.

```
mysql> update user set Password=PASSWORD('password_root') where user='root';
Query OK, 2 rows affected (0,00 sec)
Rows matched: 2  Changed: 2  Warnings: 0
```

4. Refrescaremos los privilegios concedidos para surta efecto la configuración anterior, y saldremos de *mysql* (quit):

```
mysql> flush privileges;
Query OK, 0 rows affected (0,00 sec)
mysql> quit
```

5. Comprobamos la necesidad de introducir una contraseña para acceder a MySQL. Para indicar al ejecutable *mysql* que queremos acceder con la cuenta de usuario root, y que queremos que nos solicite la password, haremos uso de los parámetros *-u* y *-p* respectivamente:

```
[root@localhost]# /opt/lampp/bin/mysql (en GNU/Linux)
c:\xampp\mysql\bin> mysql.exe (en Windows)
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: NO)
```

```
[root@localhost]# /opt/lampp/bin/mysql -u root -p (en GNU/Linux)
c:\xampp\mysql\bin> mysql.exe -u root -p (en Windows)
Enter password:
```

Tras haber modificado la password del root, tampoco será posible acceder a la herramienta de gestión Web de MySQL *phpMyAdmin (http://localhost/phpmyadmin)*.

Figura 8.4. Tras asignar una password al usuario root de MySQL será necesario configurar phpMyAmin para poder acceder a esta herramienta de gestión Web

Para corregir este problema, tendremos que modificar el fichero de configuración de phpMyAdmin, *config.inc.php*. Este lo podremos encontrar en */opt/lampp/phpmyadmin/config.inc.php* en GNU/Linux y *c:\xampp\phpmyadmin\config.inc.php* en Microsoft Windows. En concreto, nos situaremos en la sección de este fichero asociada al tipo de autenticación, donde advertiremos que por defecto es de tipo *config*, lo que implica que para acceder hará uso de los valores asignados a las directivas *$cfg['Servers'][$i]['user']* y *$cfg['Servers'][$i]['password']* que aparecen en *config.inc.php*[60].

```
/* Authentication type and info */
$cfg['Servers'][$i]['auth_type'] = 'config';
$cfg['Servers'][$i]['user'] = 'root';
$cfg['Servers'][$i]['password'] = '';
$cfg['Servers'][$i]['AllowNoPasswordRoot'] = true;
```

Según esto, será necesario modificar la directiva *$cfg['Servers'][$i]['password']* asignándole la nueva password.

```
$cfg['Servers'][$i]['password'] = 'password_root';
```

También es posible crear una nueva cuenta de usuario con permisos y privilegios restringidos sobre determinadas bases de datos, y asignar el nombre de este y su password a las directivas anteriores, garantizando que vía Web solo puedan gestionarse determinadas operaciones.

```
$cfg['Servers'][$i]['user'] = 'nombre_usuario';
$cfg['Servers'][$i]['password'] = 'password_usuario';
```

60 La directiva $cfg['Servers'][$i]['AllowNoPasswordRoot'] permite acceder a phpMyAdmin como usuario root sin tener contraseña asignada tras la instalación de Xampp.

¡¡Ayuda!! Para crear una cuenta de usuario en MySQL con privilegios restringidos sobre determinadas bases de datos puede ejecutarse la siguiente sentencia SQL (***GRANT***) a través de una consola del sistema desde prompt MySQL:

mysql> GRANT *privilegios_asignados_usuario* **ON** *base_datos_MySQL.tablas_base_datos* **TO** *nombre_usuario@equipo_servidor* **IDENTIFIED BY** '*password_usuario*';

Por ejemplo, si quisiéramos crear una cueta de usuario MySQL con nombre *usuario1* y contraseña *password1* con privilegios suficientes para poder hacer consultas SQL de tipo *select*, *insert*, *update*, *create*, *alter*, *delete* o *drop* sobre cualquier tabla (*) de una base de datos llamada basedatos1 en el equipo local ejecutaríamos la siguiente sentencia:

mysql> GRANT *select,insert,update,create,alter,delete,drop* **ON** *basedatos1.** TO usuario1@*localhost* **IDENTIFIED BY** '*password1*';

En el caso en que queramos asignarle todos los privilegios sobre una determinada base de datos, podríamos simplificar la sentencia anterior ejecutando:

mysql> GRANT ALL PRIVILEGES ON *basedatos1.** TO usuario1@*localhost* **IDENTIFIED BY** '*password1*';

El problema que presenta la configuración anterior asignada a *phpMyAdmin* es que sólo podrá acceder vía Web para gestionar MySQL el usuario indicado en *$cfg['Servers'][$i]['user']*. En el caso de que queramos que pueda acceder cualquier cuenta de usuario para gestionar sus bases datos, deberemos cambiar el tipo de autenticación de *config* a *http*:

```
$cfg['Servers'][$i]['auth_type'] = 'http';
```

La autenticación de tipo http, hará que el navegador nos solicite un nombre de un usuario dado de alta en MySQL y su password asociada para acceder a phpMyAdmin, tal como se muestra en la figura 8.5.

Figura 8.5. Autenticación de tipo http en phpMyAdmin

4. GESTIÓN DE BASES DE DATOS Y TABLAS EN MYSQL

Para la gestión y administración de las bases de datos MySQL puede hacerse, como ya se ha comentado en el apartado anterior, a través de la línea de comandos o vía Web (*phpMyAdmin*). La forma más rápida y eficiente para crear (*create*) o eliminar (*drop*) una base de datos es haciendo uso del comando *mysqladmin*:

[root@localhost]# /opt/lampp/bin/**mysqladmin -u root -p create** *nombre_base_datos* (en *GNU/Linux*)

c:\xampp\mysql\bin> **mysqladmin.exe -u root -p create** *nombre_base_datos* (*en Windows*)

```
[root@localhost]# /opt/lampp/bin/mysqladmin -u root -p drop nombre_base_datos (en GNU/Linux)
c:\xampp\mysql\bin> mysqladmin.exe -u root -p drop nombre_base_datos (en Windows)
```

En el caso de querer crear o eliminar una base de datos desde el prompt del MySQL, haremos uso de las consultas SQL correspondientes:

```
mysql> create database nombre_base_datos;
mysql> drop database nombre_base_datos;
```

Y en el caso de querer hacer uso de la herramienta Web *phpMyAdmin*:

Figura 8.6. phpMyAdmin nos permite una gestión total de las bases de datos MySQL

En cuanto a la creación de las tablas dentro de una base de datos e inserción de datos en estas, aunque puede hacerse igualmente mediante la ayuda de sentencias SQL, resulta mucho más cómodo y eficiente hacer uso de *phpMyAdmin*.

5. CONEXIONES DESDE PHP SOBRE MYSQL

Para interactuar desde un documento Web PHP con las bases de datos creadas en MySQL haremos uso de las funciones PHP que vienen en el módulo php-mysql, entre las que cabría destacar:

— **mysql_connect** (*equipo_servidor, nombre_usuario, contraseña_usuario*): establece una conexión MySQL con el equipo servidor indicado, con los privilegios asignados a la cuenta de usuario introducida. Devuelve un descriptor de la conexión establecida que nos permitirá hacer referencia a ella desde otras funciones PHP, permitiendo distinguir entre diferentes conexiones que puedan establecerse desde un mismo documento Web.

Esta función suele acompañarse de la sentencia PHP **or die**, de tal forma que interrumpa la ejecución del código PHP en caso de detectarse un problema en el establecimiento de la conexión, mostrando un mensaje personalizado que informe de ello.

En el ejemplo siguiente se muestra como establecer una conexión con el equipo local (*localhost*), mediante la cuenta de usuario MySQL *usuario1*, con contraseña *password1*.

```
$descriptor_conexion_1 = mysql_connect ("localhost", "usuario1", "password1") or die ("Problemas en la conexión");
```

¡¡Observación!! Escribir la contraseña del usuario en el documento Web que haga uso de la función **mysql_connect**() no implica ningún riesgo de seguridad, ya que hay que recordar que el código PHP solo es leído e interpretado por el equipo servidor, mostrándose al cliente únicamente el resultado de su ejecución en formato text/html.

— **mysql_select_db** (*nombre_base_datos, descriptor_conexión_PHP-MySQL*): tras establecer una conexión con la función **mysql_connect**(), esta nos permite indicar con cual de las bases de datos existentes en el servidor MySQL queremos interactuar.

— **mysql_query** (*sentencia_SQL, descriptor_conexión_PHP-MySQL*): ejecuta una consulta (*query*) o sentencia SQL sobre la base de datos seleccionada con la función **mysql_select_db**(). Como resultado, dependiendo del tipo de consulta SQL, devuelve los registros obtenidos en caso de tratarse de una sentencia *select*, o un valor booleano (*true o false, 1 o 0*) informando del éxito o fracaso de su ejecución, en el caso de sentencias de tipo *insert*, *update*, *delete*, *drop*, etc.

— **mysql_num_rows** (*resultado_consulta_select*): informa del número de filas (*rows*) o registros devueltos tras la ejecución de una sentencia de tipo *select* lanzada por **mysql_query**().

En el ejemplo siguiente, se selecciona una supuesta base de datos creada en el servidor llamada *base_datos_1*, que contiene una tabla llamada *tabla1*, sobre la cual se realiza una consulta seleccionando todos sus registros ordenados por el campo de la tabla *nombre*.

```
mysql_select_db (“base_datos_1”, $descriptor_conexion_1);
$resultado = mysql_query (“select * from tabla1 order by nombre”, $descriptor_conexion_1);
$num_filas = mysql_num_rows ($resultado);
```

— **mysql_num_fields** (*resultado_consulta_select*): informa del número de campos (*fields*) que forman parte de cada una de las filas devueltas por una sentencia de tipo *select*.

— **mysql_result** (*resultado_consulta_select*, *numero_registro*, *campo_tabla*): informa del valor almacenado en el campo de la tabla del número de registro de la consulta *select* ejecutada por **mysql_query**() indicados como parámetros.

En el ejemplo que se muestra a continuación, tras realizar la consulta *select*, se genera una tabla HTML con tres columnas separando los valores de los supuestos campos *nombre*, *puesto* y *edad* en cada uno de los registros devueltos como resultado. Mediante la ayuda de un bucle (*for*) recorremos el *array $resultado* que contiene los registros seleccionados.

```
<table border=”1”><tr><td>NOMBRE</td><td>PUESTO</td><td>EDAD</td></tr>
<?
$num_filas = mysql_num_rows ($resultado);
for ( $contador=0; $contador < $num_filas; $contador++) {
echo “<tr>”;
echo “<td>”.mysql_result ($resultado,$contador,”nombre”).”</td>”;
echo “<td>”.mysql_result ($resultado,$contador,”puesto”).”</td>”;
echo “<td>”.mysql_result ($resultado,$contador,”edad”).”</td>”;
echo “</tr>”;
}
?>
</table>
```

— **mysql_close** (*descriptor_conexión_PHP-MySQL*): cierra una conexión establecida previamente mediante **mysql_connect**().

Ejercicio práctico n.º 1

Con la finalidad de probar todo lo anterior, crea una base de datos llamada *miweb*, que contenga una tabla *invitados* con seis campos, *nombre*, *dni*, *comunidad (comunidad autónoma)*, *sexo*, *edad* y *fotografía (ruta de la fotografía del usuario registrado)*. Para gestionarla se creará una cuenta de usuario (*login: adminweb, password: mipass*) con privilegios restringidos sobre ella, de tal forma que solo pueda ejecutar las cuatro sentencias SQL básicas *select*, *insert*, *delete* y *update*.

A continuación configura Xampp/Apache para dar servicio a un sitio Web (por ejemplo, *Servername: www.miweb.es, y DocumentRoot: htdocs/miweb*) cuya página de inicio (por ejemplo, *miweb.php*) sea un documento Web PHP que contenga cuatro formularios que permita llevar a cabo las cuatro sentencias SQL anteriores sobre la base de datos MySQL. Entre sus características destacar las siguientes:

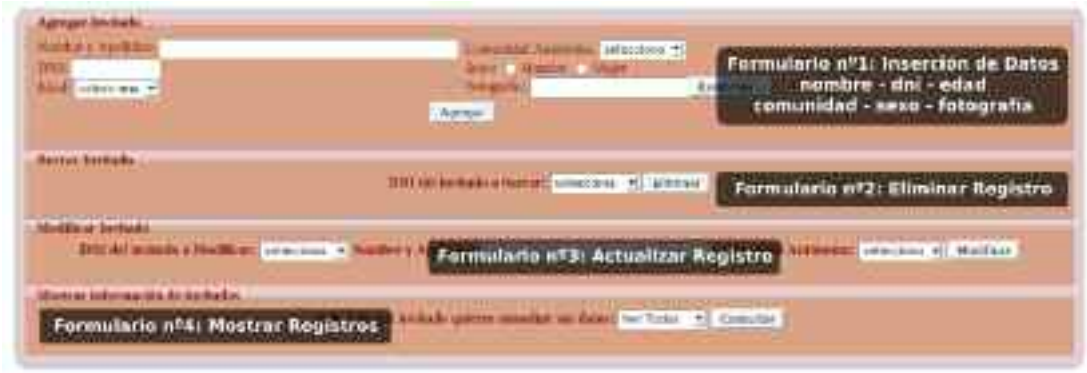


Figura 8.7. Aspecto del documento Web compuesto por 4 formularios HTML

¡¡Sugerencia!! Se aconseja configurar MySQL y Apache/Xampp (*ver solución del ejercicio práctico*) e ir implementando y probando cada uno de los formularios del documento uno a uno de manera independiente.

a) El primer formulario permitirá la inserción de datos en la tabla *invitados* de la base de datos *miweb*. Estará compuesto por dos *input-text* para el nombre y dni, dos *select* para la edad y comunidad autónoma, un *input-radio-list* para seleccionar el sexo, y un *input-file* para la fotografía. El destino de los los datos introducidos se enviarán mediante el método post al servidor (*method="post"*), siendo el propio documento *miweb.php* el destinatario (*action=""*[61]). La lista de edades comprendida entre 14 y 105, se generará mediante un bucle *for* en lenguaje PHP.

```
<form method="post" action="" enctype="multipart/form-data">
<table><tr><td class="datos">
Nombre y Apellidos: <input type="text" name="nombre" size="40" /> <br />
DNI: <input type="text" name="dni" size="10" maxlength="9" /> <br />
Edad: <select name="edad"><option selected="selected">selecciona</option>
<? for ($contador=14; $contador < 105; $contador++) { echo
"<option>".$contador."</option>";} ?>
</select>
```

61 Si el atributo *action* del elemento HTML *form* no tiene un valor asignado, se asume que el destinatario dentro del servidor de los datos del formulario es el propio documento PHP que lo contiene.

```
</td>
<td class="datos">
Comunidad Autónoma: <select name="comunidad"><option selected="selected">seleccio-
na</option>
<option>Andalucia</option>
<option>Aragón</option>
<option>Asturias</option>
<option> ... </option> <!— Resto de las comunidades autónomas —>
</select><br />
Sexo: <input type="radio" name="sexo" value="hombre" /> Hombre
<input type="radio" name="sexo" value="hombre" /> Mujer<br />
Fotografía: <input type="file" name="foto" />
</td></tr>
<tr><td class="agregar" colspan="2">
<input type="submit" name="boton" value="Agregar" />
</td></tr></table></form>
```

Para comprobar que la inserción de datos se realiza correctamente, tenemos la posibilidad de implementar el cuarto formulario del documento encargado de mostrar los registros almacenados en el servidor o consultar directamente el estado de la tabla *invitados* de la base de datos *miweb* a través de phpMyAdmin.

Figura 8.8. phpMyAdmin nos permite comprobar la inserción de datos realizada desde el primer formulario

b) El segundo formulario permitirá eliminar alguno de los registros insertados previamente en la base de datos. La selección de este registro se realizará mediante un campo *select* que mostrará la lista de todos los dni de los usuarios registrados en la base de datos. Para su obtención se ejecutará una sentencia SQL *select*: "**SELECT** *dni* **FROM** *invitados* **ORDER BY** *dni*".

```
<form method="post" action="">
DNI del invitado a borrar: <select name="seleccionado"><option>selecciona</option>
<?
$resultado = mysql_query ("SELECT dni FROM invitados ORDER BY dni",$con);
```

```
$num_filas = mysql_num_rows ($resultado);
for ( $contador=0; $contador < $num_filas; $contador++)
{ echo "<option>".mysql_result ($resultado,$contador,"dni")."</option>"; }
?>
</select>
<input type="submit" name="boton" value="Eliminar" />
</form>
```

c) El tercer formulario permitirá modificar los campos *nombre* y *comunidad* de alguno de los registros insertados previamente en el formulario. Mediante un *select* se seleccionará el *dni* del usuario del que se quiere modificar sus datos, junto con *input-text* para la introducción del nuevo nombre y otro *select* para la selección de la nueva comunidad autónoma.

```
<form method="post" action="">
DNI del invitado a Modificar: <select name="seleccionado"><option>selecciona</option>
<?
$resultado = mysql_query ("SELECT dni FROM invitados ORDER BY dni",$con);
$num_filas = mysql_num_rows ($resultado);
for ( $contador=0; $contador < $num_filas; $contador++)
{ echo "<option>".mysql_result ($resultado,$contador,"dni")."</option>"; }
?>
</select>
Nombre y Apellidos: <input type="text" name="nombre" size="30">
Comunidad Autónoma: <select name="comunidad">
<option selected="selected">selecciona</option>
<option>Andalucia</option>
<option>Aragón</option>
<option>Asturias</option>
<option> ... </option> <!— Resto de las comunidades autónomas —>
</select>
<input type="submit" name="boton" value="Modificar" />
</form>
```

d) El cuarto formulario permitirá consultar todos los datos introducidos en la base de datos, o el de uno de los usuarios registrados mediante la indicación de su *dni*. Un campo *select* permitirá seleccionar la primera opción (*ver todos*) o uno de los *dni* rescatados de la tabla *invitados*.

```
<form method="post" action="">
Indica de que invitado quieres consultar sus datos: <select name="seleccionado">
<option value="Todos">Ver Todos</option>
<?
$resultado = mysql_query ("SELECT dni FROM invitados ORDER BY dni",$con);
$num_filas = mysql_num_rows ($resultado);
for ( $contador=0; $contador < $num_filas; $contador++)
{ echo "<option>".mysql_result ($resultado,$contador,"dni")."</option>"; }
?>
</select>
<input type="submit" name="boton" value="Consultar" />
</form>
```

e) Con la finalidad de que el documento sea más legible, el código PHP asociado a cada una de las acciones que el servidor Web tiene que llevar a cabo en función del formulario que le envía los datos será definido en un documento independiente (*include funciones_miweb.php;*), formado por cuatro funciones PHP dependientes de las cuatro sentencias SQL a ejecutar: *agregar()*, *eliminar()*, *modificar()*, y *consultar()*. Advirtiendo que todos los botones de *submit* de los cuatro formularios que forman el documento *miweb.php* se denominan igual (*name="boton"*) diferenciándose en su *value,* para que el servidor reconozca cual ha sido pulsado, se utilizará una estructura de control *switch-case*, la cual se encargará de llamar a la función correspondiente.

```
<? if (isset($_POST['boton']))
{ switch ($_POST['boton']) {
        case "Agregar": agregar($con); break;
        case "Eliminar": eliminar($con); break;
        case "Modificar": modificar($con); break;
        case "Consultar": consultar($con); break;
  }
} ?>
```

Además, para poder ejecutar las sentencias SQL, será necesario establecer una conexión con la base de datos MySQL *miweb* haciendo uso de la cuenta de usuario *adminweb*. Todo ello, junto con la importación del documento PHP que contiene las funciones anteriores, irá colocado al comienzo del documento *miweb.php*.

```
<?
$con = mysql_connect ("localhost","adminweb","mipass") or die ("Problema de conexión...");
mysql_select_db ("miweb",$con);
include "funciones_miweb.php";
?>
```

f) La función **agregar**() definida dentro del documento PHP *funciones_miweb.php* recogerá los datos introducidos por el usuario en el primer formulario, más la variable de conexión con la base de datos pasada como parámetro, *$con*.

Antes de insertar los datos en la tabla *invitados* de la base de datos *miweb*, se comprobará que los campos del formulario *nombre y dni* no se han dejado vacíos, que se han seleccionado una *comunidad* y *edad*, y que se ha marcado un *sexo*. Simultáneamente se comprobará que el dni introducido este formado por nueve caracteres.

```
if ( $_POST['nombre'] != "" && strlen($_POST['dni']) == 9 && $_POST['comunidad'] != "selecciona" && $_POST['edad'] != "selecciona" && isset($_POST['sexo']) ) { instrucciones; }
```

En el caso de que se haya seleccionado y subido una fotografía al servidor, *is_uploaded_file()*, ésta será movida mediante *move_uploaded_file()* a un subdirectorio llamado *imagenes*, siendo renombrada mediante el valor del *dni* introducido. En el caso de no subir una fotografía o haberse producido un problema en su manipulación, se registrará que "*no hay foto*". Esta ruta del archivo, junto con el resto de los datos serán insertados finalmente en la base de datos:

¡¡Ayuda!! La sintaxis de una sentencia SQL de tipo INSERT es la siguiente:

"**INSERT INTO** *nombre_tabla* (*campo1,campo2,...*) **VALUES** ('*value1*', '*value2*', ...)";

"**INSERT INTO** *invitados* (*nombre,dni,sexo,comunidad,edad,fotografia*) **VALUES** ('*value1*', '*value2*', '*value3*', '*value4*', '*value5*', '*value6*')";

```
<? function agregar ($con) { // función agregar() definida dentro de funciones_miweb.php
if ( $_POST['nombre'] != "" && strlen($_POST['dni']) == 9 && $_POST['comunidad'] != "selecciona" && $_POST['edad'] != "selecciona" && isset($_POST['sexo']) )
{
  if (is_uploaded_file($_FILES['foto']['tmp_name']))
  {
        $extension = substr ($_FILES['foto']['name'],-3);
        $ruta="imagenes/".$_POST['dni'].".".$extension;
        if (!move_uploaded_file($_FILES['foto']['tmp_name'],$ruta))
        { $ruta="no hay foto"; }
  }
  else { $ruta="no hay foto"; }

  if (mysql_query ("INSERT INTO invitados (nombre,dni,sexo,comunidad,edad,fotografia) VALUES
('".utf8_decode($_POST['nombre'])."','".$_POST['dni']."','".$_POST['sexo']."','".utf8_decode($_POST['comunidad'])."','".$_POST['edad']."','".$ruta."')",$con))
  {     echo "Se ha agregado el usuario ".$_POST['nombre']." a la base de datos de invitados...";
        echo "<br />Puedes comprobar tus datos desde el formulario: Mostrar Información de Invitados.";
  }
  else { echo "Problemas al insertar los datos ..."; }
}
else { echo "Debes cumplimentar todos los campos ..."; }
} ?>
```

¡¡Observación!! Mucho cuidado con las comillas simples y dobles utilizadas en las sentencias SQL ejecutadas desde PHP. Hay que tener en cuenta que los valores a insertar en la tabla MySQL deben ir entre comillas simples, 'value', y que para concatenar el valor de una variable con una cadena de caracteres es necesario cerrar con comillas dobles y usar el carácter de concatenación (*un punto*, **.**), ".$variable."'.

Del código anterior, destacar la utilización de la función *utf8_decode()* para evitar problemas con la codificación de caracteres especiales como la ñ española y sus acentos a la hora de introducir los datos en la base de datos MySQL. Esta función convierte una cadena codificada en formato UTF-8 a ISO-8859-1, sistema de codificación por defecto de MySQL. Esta función no sería necesaria si el sistema de codificación utilizado por los documentos Web fuera directamente ISO-8859-1 modificando la etiqueta HTML <*meta* /> encargada de indicar el juego de caracteres, la cual trasladaría el problema de codificación al propio documento siendo necesario la utilización de caracteres especiales, "&|a|e|i|o|u|acute;" para los acentos, "ñ", para la ñ, etc.

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
```

Otra posibilidad sería editar el fichero *php.ini*, al ser este el encargado de determinar el comportamiento de los documentos PHP servidos por el servidor, introduciendo o descomentando la línea *default_charset = "iso-8859-1"*, la cual obligaría independientemente de la etiqueta anterior <*meta* /> a codificar los documentos Web en ISO-8859-1.

¡¡Importante!! Funciones PHP **utf8_decode**() y **utf8_encode**(): En el caso de que nuestros documentos Web estén codificados en UTF-8 será necesario convertir las cadenas de caracteres susceptibles de utilizar caracteres especiales (*ñ, acentos, etc.*) a ISO-8859-1, mediante **utf8_decode**(), para evitar problemas de codificación con MySQL. De igual forma, al rescatar los datos de nuestra base de datos será necesario hacer la recodificación en sentido contrario de ISO-8859-1 a UTF-8, mediante **utf8_encode**().

g) La función **eliminar**() definida dentro del documento PHP *funciones_miweb.php* eliminará el registro de la tabla invitados cuyo dni concuerde con el introducido en el segundo formulario.

¡¡Ayuda!! La sintaxis de una sentencia SQL de tipo DELETE es la siguiente:

"**DELETE FROM** *nombre_tabla* **WHERE** *condición*";
"**DELETE FROM** *invitados* **WHERE** *dni='value'*";

```
<? function eliminar($con) {  // función eliminar() definida dentro de funciones_miweb.php
if ( $_POST['seleccionado'] != "selecciona" )
{
  $resultado = mysql_query ("SELECT fotografia FROM invitados WHERE
dni='".$_POST['seleccionado']."'",$con);
  if (mysql_query ("DELETE FROM invitados WHERE
dni='".$_POST['seleccionado']."'",$con))
  {
        echo "Se ha borrado el invitado con dni ".$_POST['seleccionado'];
        if ( mysql_result ($resultado,0,"fotografia") != "no hay foto")
        { unlink (mysql_result ($resultado,0,"fotografia")); }
  }
  else { echo "Problemas al tratar de eliminar el usuario con ".$_POST['seleccionado']; }
}
}
?>
```

Como puede comprobarse por el código PHP, la función *eliminar()* comprueba en primer lugar si se ha seleccionado un *dni* (*opción diferente a la opción por defecto "selecciona"*). En ese caso, consultará la ruta de la fotografía subida al servidor (*"SELECT fotografia FROM invitados WHERE dni='".$_POST['seleccionado']."'"*), y tratará de eliminar tanto el registro de la base de datos (*"DELETE FROM invitados WHERE dni='".$_POST['seleccionado']."'"*), como el fichero asociado a la fotografía (*unlink (mysql_result ($resultado,0,"fotografia"))*). En el caso de que no se haya subido una fotografía al servidor, la *function agregar()* habrá almacenado en el campo fotografia "*no hay foto*", indicando en ese caso que no es necesario borrar mediante *unlink*() ningún fichero.

h) La función **modificar**() definida dentro del documento PHP *funciones_miweb.php* modifica los campos *nombre* y *comunidad* del registro cuyo *dni* coincide con el seleccionado en el *select* del tercer formulario.

¡¡Ayuda!! La sintaxis de una sentencia SQL de tipo UPDATE es la siguiente:

"**UPDATE** *nombre_tabla* **SET** *campo1='value1', campo2='value2',...* **WHERE** *condición*";
"**UPDATE** *invitados* **SET** *nombre='value1', comunidad='value2'* **WHERE** *dni='value'*";

```
<? // función modificar() definida dentro de funciones_miweb.php
function modificar($con) {
if ( $_POST['nombre'] != "" && $_POST['comunidad'] != "selecciona" )
{
  if (mysql_query ("UPDATE invitados SET nombre = "".utf8_decode ($_POST['nombre']).", comunidad = "".utf8_decode($_POST['comunidad'])." WHERE dni = "".$_POST['seleccionado']."",$con))
  { echo "El usuario ".$_POST['nombre']." se actualizo perfectamente..."; }
  else { echo "Problemas al llevar a cabo la actualización del registro!!!"; }
}
else { echo "Debes rellenar los campos nombre y comunidad!!!"; }
}
?>
```

Tras comprobar que todos los campos del formulario han sido completados (*nombre y comunidad*) se ejecutará la sentencia SQL de actualización sobre la base de datos (*"UPDATE invitados SET nombre = "".utf8_decode ($_POST['nombre']).", comunidad = "".utf8_decode ($_POST['comunidad']).' WHERE dni = "".$_POST['seleccionado']."'"*).

i) La función **consultar**() definida dentro del documento PHP *funciones_miweb.php* muestra los registros insertados en la tabla *invitados* de la base de datos *miweb*. En concreto, nos muestra todos los registros en el caso de haber seleccionado en el *select* del cuarto formulario "Ver Todos" (*value="Todos"*), o únicamente el del usuario cuyo *dni* coincide con el seleccionado.

Figura 8.9. Resultado de la consulta "Ver Todos"

¡¡Ayuda!! La sintaxis básica de una sentencia SQL de tipo SELECT es la siguiente:

"**SELECT** [**ALL**|**DISTINCT**] *campo1,campo2,...* **FROM** *nombre_tabla* **WHERE** *condición* **ORDER BY** *campo1* [**ASC**|**DESC**],*campo2* [**ASC**|**DESC**], ...";

"**SELECT** * **FROM** *invitados* **ORDER BY** *nombre*";

"**SELECT** * **FROM** *invitados* **WHERE** *dni='value'*";

```
<? // función consultar() definida dentro de funciones_miweb.php
function consultar($con) {
if ( $_POST['seleccionado'] == "Todos" )
{ $resultado = mysql_query ("SELECT * FROM invitados ORDER BY nombre",$con); }
else
{ $resultado = mysql_query ("SELECT * FROM invitados WHERE dni = '".$_POST['seleccionado']."'",$con); }
$num_filas = mysql_num_rows ($resultado);
echo "<table border='1' class='tabla_respuesta'> <tr class='cabecera_respuesta'> <td> NOMBRE </td> <td> DNI </td> <td> COMUNIDAD AUTÓNOMA </td> <td> EDAD </td> <td> IMAGEN </td>";
  for ( $contador=0; $contador < $num_filas; $contador++)
  {
  echo "<tr class='invitados'>";
  echo "<td>".utf8_encode (mysql_result ($resultado,$contador,"nombre")."</td>";
  echo "<td>".mysql_result ($resultado,$contador,"dni")."</td>";
  echo "<td>".utf8_encode (mysql_result ($resultado,$contador,"comunidad")."</td>";
  echo "<td>".mysql_result ($resultado,$contador,"edad")."</td>";
  if ( mysql_result ($resultado,$contador,"fotografia") != "no hay foto")
  { echo "<td><img src='".mysql_result ($resultado,$contador,"fotografia")."' width='100px' height='75px' /></td>"; }
  else { echo "<td> NO SE DISPONE DE FOTO</td>"; }
  echo "</tr>";
  }
echo "</table>";
}
?>
```

Como puede observarse en el código PHP, los datos devueltos como resultado de la consulta *select* son mostrados en el cliente Web organizados en una tabla. Destacar que los campos *nombre* y *comunidad*, al ser susceptibles de contener caracteres especiales (*letra ñ, acentos, etc.*), es necesario hacer uso de la función **utf8_encode**() para pasar del formato ISO-8859-1 con que fueron insertados (*repasar la function agregar()*) mediante la función **utf8_decode**(), a UTF-8.

j) Todos los formularios anteriores, y sus respuestas, serán encerrados por una etiqueta HTML *<fieldset></fieldset>* para mejorar su presentación.

Solución ejercicio n.º 1

Según todos los requisitos del ejercicio práctico anterior, una solución organizada podríamos dividirda en los siguientes pasos:

Paso n.º 1. Creación de la base de datos *miweb*, su tabla *invitados* y el usuario MySQL *adminweb* con contraseña *mipass*. Por comodidad, accediendo como usuario MySQL *root* se realizará todo mediante la interfaz Web *phpMyAdmin* tal como se muestra en las figuras siguientes.

Figura 8.10. Creación de la base de datos miweb *y tabla* invitados *desde phpMyAdmin*

Para evitar que existan usuarios registrados en la base de datos duplicados, al definir los campos de la tabla invitados, *dni* será declarado como clave primaria (ver figura 8.10).

¡¡Observación!! La elección de un adecuado motor de almacenamiento para nuestra base de datos en MySQL es fundamental para garantizar una eficiente ejecución de sentencias SQL. Entre los distintos motores disponibles, destacar a InnoDB y MyISAM. Sus características más importantes son las siguientes:

— InnoDB es transaccional: cuando un usuario trabaja sobre los registros de una tabla, bloquea la escritura sobre estos, hasta que no se confirme la transacción (*commit o rollback*), evitando que otros usuarios puedan manipularlos simultáneamente, garantizando la integridad, aislamiento y consistencia de los datos. MyISAM no es transaccional.

— InnoDB admite integridad referencial: permite definir claves ajenas entre campos de diferentes tablas. MyISAM no lo permite.

— InnoDB es el motor de almacenamiento recomendable cuando las sentencias predominantes son de tipo *insert* o *update*, frente a MyISAM, al presentar mayor rendimiento.

— MyISAM no requiere hacer comprobaciones de integridad, ni bloqueo sobre los registros, traduciéndose en una mayor velocidad de procesamiento y consulta de datos. Es recomendable frente a InnoDB cuando las sentencias predominantes son de tipo *select*.

Si tenemos en cuenta que la utilización de una base de datos en los actuales sitios Web, es la de elemento de organización y respaldo de los datos que forman parte de los contenidos de los documentos que los componen, la sentencia SQL predominante es la consulta (select) de datos para prepararlos para presentación al cliente.

Para la creación del usuario MySQL adminweb ejecutaremos una sentencia GRANT desde la pestaña SQL de phpMyAdmin. Esta sentencia creará al usuario al mismo tiempo que le concede privilegios restringidos (*select,insert,update,delete*) sobre la base de datos miweb:

GRANT select,insert,update,delete **ON** miweb.* **TO** adminweb@localhost **IDENTIFIED BY** 'mipass';
GRANT select,insert,update,delete **ON** miweb.invitados **TO** adminweb@localhost **IDENTIFIED BY** 'mipass';

Figura 8.11. Creación del usuario adminweb y concesión de privilegios mediante la sentencia SQL GRANT

Paso n.º 2. Configuración del servidor Apache/Xampp para dar servicio al sitio Web *www.miweb.es* (por ejemplo, *Servername: www.miweb.es, y DocumentRoot: htdocs/miweb*), y definimos localmente la resolución de su nombre de domino asociado (p.e. *www.miweb.es -> 127.0.0.1*).

```
# Contenido del fichero de configuración de Apache: httpd.conf
Listen 80 # Puerto de escucha de Apache por defecto, a través de cualquier dirección IP
DocumentRoot /opt/lampp/htdocs # Directorio Raíz de Xampp en GNU/Linux
DocumentRoot c:/xampp/htdocs # Directorio Raíz de Xampp en Microsoft Windows
ServerName localhost # Nombre del equipo servidor Xampp
NameVirtualHost 127.0.0.1 # Uso de la IP 127.0.0.1 en más de 1 sitio Web (VirtualHost)
Include /opt/lampp/etc/extra/ejercicio-practico-miweb-php.conf # Inclusión en GNU/Linux
Include c:/xampp/apache/conf/extra/ejercicio-practico-miweb-php.conf # Inclusión en Windows
... # Resto de las directivas de configuración del fichero httpd.conf
```

```
# Contenido del fichero auxiliar de configuración de Apache: ejercicio-practico-miweb-php.conf
<VirtualHost 127.0.0.1>
  ServerName www.miweb.es # Nombre de dominio del sitio Web 1
  DocumentRoot /opt/lampp/htdocs/miweb # Directorio Raíz del sitio Web en GNU/Linux
  DocumentRoot c:/xampp/htdocs/miweb # Directorio Raíz del sitio Web en Windows
  DirectoryIndex miweb.php
</VirtualHost>
```

```
# Contenido del fichero hosts: Direccion IP -> Nombre Equipo
127.0.0.1 localhost # Esta asociación se encuentra siempre por defecto en "hosts"
127.0.0.1 www.miweb.es
```

Paso n.º 3. Tras configurar MySQL y Apache/Xampp se crearán los documentos Web *miweb.php*, *funciones_miweb.php*, y *miweb.css*, haciendo uso de los trozos de código HTML y PHP presentados durante el enunciado del ejercicio practico.

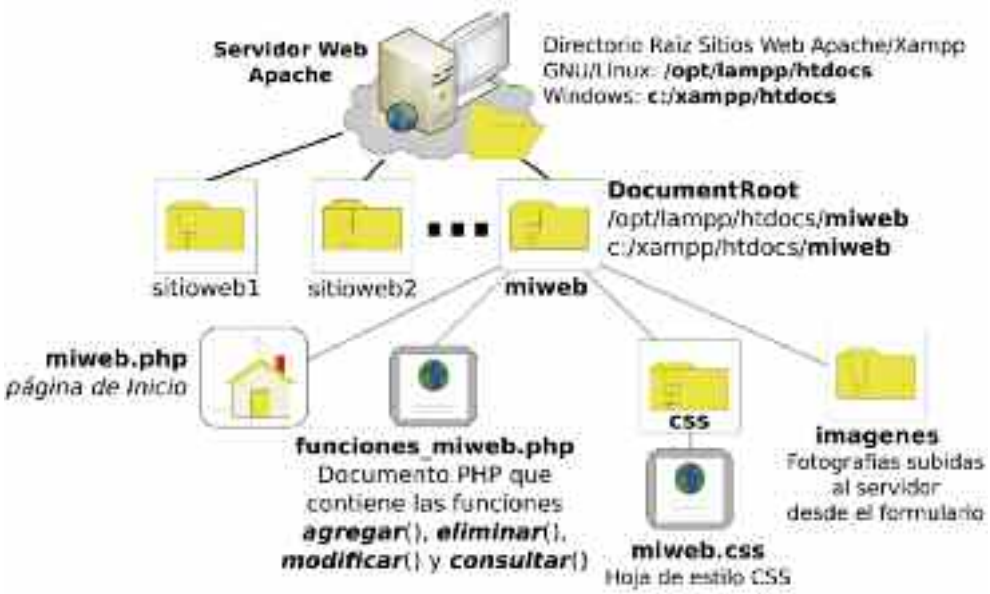


Figura 8.12. Estructura del sistema de ficheros del ejercicio práctico

— Contenido del documento *miweb.php*:

```
<?
$con = mysql_connect (“localhost”,”adminweb”,”mipass”) or die (“Problema de conexión...”);
mysql_select_db (“miweb”,$con);
include “funciones_miweb.php“;
?>
<html><head>
<meta http-equiv=”Content-Type” content=”text/html; charset=UTF-8“ />
<link href=’imagenes/logo.png’ rel=’shortcut icon’ type=’image/png’/>
<link rel=”stylesheet” type=”text/css” href=”css/miweb.css” media=”screen” />
<title>Gestion Web - PHP y MySQL</title>
</head>
<body>
<h2>BASE DATOS DE INVITADOS</h2>
<!— Antes de generar la respuesta HTTP en formato text/html, el servidor comprueba si se le
han enviado datos desde alguno de los formularios que lo forman —>
<? if (isset($_POST[‘boton’]))
{ echo “<fieldset class=’informacion’><legend>Respuesta Consulta</legend>”;
  switch ($_POST[‘boton’]) {
            case “Agregar”: agregar($con); break;
            case “Eliminar”: eliminar($con); break;
            case “Modificar”: modificar($con); break;
            case “Consultar”: consultar($con); break;
  }
  echo “</fieldset>”;
} ?>
<fieldset><legend>Agregar Invitado</legend>
<!— El formulario Nº1 tiene un campo input-file, por lo que enctype=”multipart/form-data“ —
>
<form method=”post“ action=”“ enctype=”multipart/form-data“>
<table border=”0”><tr><td class=”datos”>
```

```
Nombre y Apellidos: <input type="text" name="nombre" size="40"> <br />
DNI: <input type="text" name="dni" size="10" maxlength="9"> <br />
Edad: <select name="edad">
<option selected="selected">selecciona</option>
<? for ($contador=14; $contador < 105; $contador++)
{ echo "<option>".$contador."</option>";} ?>
</select>
</td><td class="datos">
Comunidad Autónoma: <select name="comunidad">
<option selected="selected">selecciona</option>
<option>Andalucia</option>
<option>Aragón</option>
<option>Asturias</option>
<option>...</option> <!— Resto de la lista de comunidades autónomas —>
</select><br />
Sexo: <input type="radio" name="sexo" value="hombre"> Hombre
<input type="radio" name="sexo" value="hombre"> Mujer<br />
Fotografía: <input type="file" name="foto" >
</td></tr><tr><td class="agregar" colspan="2">
<input type="submit" name="boton" value="Agregar"></td></tr></table>
</form></fieldset>
<!— Formulario Nº2 —>
<fieldset><legend>Borrar Invitado</legend>
<form method="post" action="">
DNI del invitado a borrar: <select name="seleccionado">
<option>selecciona</option>
<?
$resultado = mysql_query ("SELECT dni FROM invitados ORDER BY dni",$con);
$num_filas = mysql_num_rows ($resultado);
for ( $contador=0; $contador < $num_filas; $contador++)
{ echo "<option>".mysql_result ($resultado,$contador,"dni")."</option>"; }
?>
</select>
<input type="submit" name="boton" value="Eliminar" />
</form>
</fieldset>
<!— Formulario Nº3 —>
<fieldset><legend>Modificar Invitado</legend>
<form method="post" action="">
DNI del invitado a Modificar: <select name="seleccionado"><option>selecciona</option>
<?
$resultado = mysql_query ("SELECT dni FROM invitados ORDER BY dni",$con);
$num_filas = mysql_num_rows ($resultado);
for ( $contador=0; $contador < $num_filas; $contador++)
{ echo "<option>".mysql_result ($resultado,$contador,"dni")."</option>"; }
?>
</select>
Nombre y Apellidos: <input type="text" name="nombre" size="30">
Comunidad Autónoma: <select name="comunidad">
<option selected="selected">selecciona</option>
<option>Andalucia</option>
<option>Aragón</option>
```

```
<option>Asturias</option>
<option> ... </option> <!— Resto de las comunidades autónomas —>
</select>
<input type="submit" name="boton" value="Modificar" />
</form>
</fieldset>
<!— Formulario Nº4 —>
<fieldset><legend>Mostrar Información de Invitados</legend>
<form method="post" action="">
Indica de que invitado quieres consultar sus datos: <select name="seleccionado">
<option value="Todos">Ver Todos</option>
<?
$resultado = mysql_query ("SELECT dni FROM invitados ORDER BY dni",$con);
$num_filas = mysql_num_rows ($resultado);
for ( $contador=0; $contador < $num_filas; $contador++)
{ echo "<option>".mysql_result ($resultado,$contador,"dni")."</option>"; }
?>
</select>
<input type="submit" name="boton" value="Consultar" />
</form></fieldset>
</body></html>
```

— Contenido del documento funciones_miweb.php:

```
<?
// función agregar(): inserta nuevos registros en la tabla invitados.
function agregar ($con) {
if ( $_POST['nombre'] != "" && strlen($_POST['dni']) == 9 && $_POST['comunidad'] != "selecciona" && $_POST['edad'] != "selecciona" && isset($_POST['sexo']) )
{
  if (is_uploaded_file($_FILES['foto']['tmp_name']))
  {
            $extension = substr ($_FILES['foto']['name'],-3);
            $ruta="imagenes/".$_POST['dni'].".".$extension;
            if (!move_uploaded_file($_FILES['foto']['tmp_name'],$ruta))
            { $ruta="no hay foto"; }
  }
  else { $ruta="no hay foto"; }

  if (mysql_query ("INSERT INTO invitados (nombre,dni,sexo,comunidad,edad,fotografia) VALUES ('".utf8_decode($_POST['nombre'])."','".$_POST['dni']."','".$_POST['sexo']."','".utf8_decode($_POST['comunidad'])."','".$_POST['edad']."','".$ruta."')",$con))
  {         echo "Se ha agregado el usuario ".$_POST['nombre']." a la base de datos de invitados...";
            echo "<br />Puedes comprobar tus datos desde el formulario: Mostrar Información de Invitados.";
  }
  else { echo "Problemas al insertar los datos ..."; }
}
else { echo "Debes cumplimentar todos los campos ..."; }
```

```
}
// función eliminar(): elmina el registro indicado de la tabla invitados.
function eliminar($con) {
if ( $_POST['seleccionado'] != "selecciona" )
{
  $resultado = mysql_query ("SELECT fotografia FROM invitados WHERE dni='".$_POST['seleccionado']."'",$con);
  if (mysql_query ("DELETE FROM invitados WHERE dni='".$_POST['seleccionado']."'",$con))
  {
        echo "Se ha borrado el invitado con dni ".$_POST['seleccionado'];
        if ( mysql_result ($resultado,0,"fotografia") != "no hay foto")
        { unlink (mysql_result ($resultado,0,"fotografia")); }
  }
  else { echo "Problemas al tratar de eliminar el usuario con ".$_POST['seleccionado']; }
}
}
// función modificar(): actualiza los campos nombre y comunidad de los registros.
function modificar($con) {
if ( $_POST['nombre'] != "" && $_POST['comunidad'] != "selecciona" )
{
  if (mysql_query ("UPDATE invitados SET nombre = '".utf8_decode ($_POST['nombre'])."', comunidad = '".utf8_decode($_POST['comunidad'])."' WHERE dni = '".$_POST['seleccionado']."'",$con))
  { echo "El usuario ".$_POST['nombre']." se actualizo perfectamente..."; }
  else { echo "Problemas al llevar a cabo la actualización del registro!!!"; }
}
else { echo "Debes rellenar los campos nombre y comunidad!!!"; }
}
// función consultar(): muestra el registro o registros de la tabla invitados en una tabla HTML.
function consultar($con) {
if ( $_POST['seleccionado'] == "Todos" )
{ $resultado = mysql_query ("SELECT * FROM invitados ORDER BY nombre",$con); }
else
{ $resultado = mysql_query ("SELECT * FROM invitados WHERE dni = '".$_POST['seleccionado']."'",$con); }
$num_filas = mysql_num_rows ($resultado);
echo "<table border='1' class='tabla_respuesta'> <tr class='cabecera_respuesta'> <td> NOMBRE </td> <td> DNI </td> <td> COMUNIDAD AUTÓNOMA </td> <td> EDAD </td> <td> IMAGEN </td>";
  for ( $contador=0; $contador < $num_filas; $contador++)
  {
  echo "<tr class='invitados'>";
  echo "<td>".utf8_encode (mysql_result ($resultado,$contador,"nombre"))."</td>";
  echo "<td>".mysql_result ($resultado,$contador,"dni")."</td>";
  echo "<td>".utf8_encode (mysql_result ($resultado,$contador,"comunidad"))."</td>";
  echo "<td>".mysql_result ($resultado,$contador,"edad")."</td>";
  if ( mysql_result ($resultado,$contador,"fotografia") != "no hay foto")
  { echo "<td><img src='".mysql_result ($resultado,$contador,"fotografia")."' width='100px' height='75px' /></td>"; }
  else { echo "<td> NO SE DISPONE DE FOTO</td>"; }
  echo "</tr>";
  }
echo "</table>";
}
```

6. COPIAS DE SEGURIDAD DE MYSQL

Para finalizar el capítulo, se tratará un aspecto que es fundamental en el mantenimiento de todo sitio Web: realización de copias de respaldo. Si tenemos en cuenta que todos los datos de interés se encuentran almacenados en la base de datos, a continuación se mostrará la forma de hacer una copia de serguridad en MySQL mediante el comando **mysqldump**, /opt/lampp/bin/mysqldump en GNU/Linux o c:\xampp\mysql\bin\mysqldump.exe en Microsoft Windows. Este genera un fichero con todas las sentencias SQL necesarias para su restauración. Su sintaxis más básica es la siguiente:

```
mysqldump —opt —user=usuario_MySQL —password=contraseña nombre_base_datos
```

Donde las opciones *—user* y *—password* informan a **mysqldump** de la cuenta de usuario MySQL y la contraseña con la que se establecerá la conexión con el servidor, la cual deberá tener privilegios para llevar a cabo la copia de seguridad.

Para programar la realización de una copia de seguridad de la base de datos de una manera periódica, haremos uso de la herramienta software de la que disponga el sistema operativo para la realización de tareas automáticas. En el caso de GNU/Linux se recomienda hacer uso de **crontab**, y en Microsoft Windows su aplicación de tareas programadas.

a) *Crontab* en GNU/Linux. La configuración del servicio *crontab* pasa por crear un script que contenga el comando que nos permite hacer el backup de la base de datos y guardarlo en el directorio del servicio *crontab* correspondiente a la periodicidad con que queramos llevar a cabo la copia de seguridad. Resumiendo, los pasos a seguir serían los siguientes:

1. En primer lugar comprobaremos el estado del servicio *crontab*, *crond* (*demonio contrab*). Para ello ejecutaremos el siguiente comando:

```
[root@linux]# /etc/init.d/crond status
crond (pid 3023) está corriendo...
```

2. En el caso de que este parado el servicio será necesario iniciarlo ejecutando el comando "***/etc/init.d/crond start|restart***". Si por el contrario, el servicio *crontab* no estuviera disponible en nuestro equipo, sería necesario instalar el paquete *crontabs*[62]:

```
[root@linux]# urpmi crontabs
```

3. Tras comprobar que el servicio *crond* esta corriendo, comprobaremos que dentro del directorio del sistema */etc* existen varios directorios asociados a este servicio:

```
[root@linux]# ls /etc/cron*
cron.d/ cron.daily/ cron.hourly/ cron.monthly/ crontab cron.weekly/ cron.yearly/
```

4. Entre los directorios anteriores cabría recalcar "/etc/**cron.daily/**", "/ect/**cron.weekly/**" o "/etc/**cron.monthly/**", los cuales contienen los scripts que serán ejecutados por el servicio *crond* con una periodicidad diaria, semanal o mensual respectivamente. Según esto, tan

62 Dependiendo de la distribución de GNU/Linux con la que se este trabajando, la instalación de paquetes software se realiza mediante manejadores diferentes: *urpmi* en Mandriva, *apt-get* en Debian, *install rpm* en Red Hat, etc.

solo tendremos que crear un script que contenga el comando que lleva a cabo el backup de la base de datos, darle permisos de ejecución y colocarlo en el directorio correspondiente.

En el ejemplo que se muestra a continuación, se crea un script llamado *copia-seguridad*, que se ejecutará automáticamente de manera diaria, */etc/**cron.daily/***, el cual se encargará de que la lista de sentencias SQL de recuperación de la base de datos devuelta por el comando *mysqldump* se redireccionen sobre un fichero llamado *bd_miweb_[date].sql*[63].

```
[root@linux]# mkdir /var/copias_miweb/
[root@linux]# echo "mysqldump —opt —user=root —password=mipass miweb > /var/copias_miweb/bd_miweb_`date +%d%m%y`.sql" > /etc/cron.daily/copia-seguridad
[root@linux]# chmod +x /etc/cron.daily/copia-seguridad
```

Tras haber seguido todos los pasos anteriores, dispondrás diariamente de una nueva copia de seguridad de la base de datos MySQL *miweb*. En el caso de necesitar restaurar una de las copias, redireccionaremos la entrada MySQL desde el fichero **.sql* anterior, ejecutando el siguiente comando:

```
[root@linux]# /opt/lampp/bin/mysql —user=root —password=mipass miweb < bd_miweb_[fecha].sql
```

b) Tareas programas en Microsoft Windows. Estas pueden configurarse desde *Programas* / *Accesorios* / *Herramientas del sistema* / *Tareas programadas*.

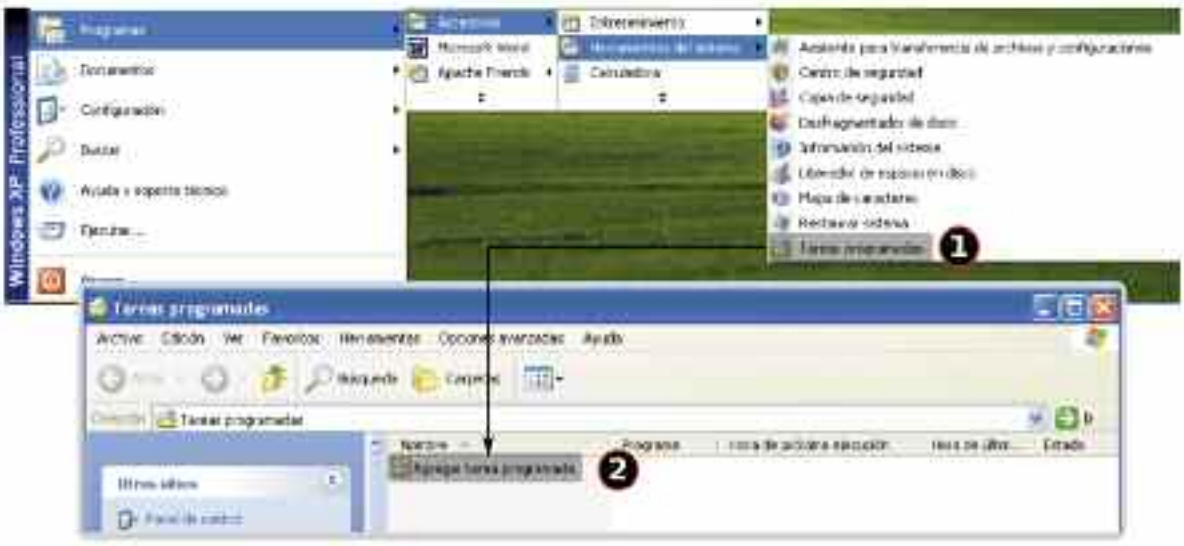


Figura 8.13. Tareas programadas en Windows

Al hacer un doble click sobre Agregar tarea programada, Windows nos mostrará un asistente que nos guiará sobre los diferentes pasos necesarios para la configuración de la tarea: 1) Elegimos el programa ejecutable que deseamos ejecutar periodicamente, 2) en el caso de que mysqldump no aparezca en la lista pincharemos en Examinar, y lo bucaremos dentro de c:\xampp\mysql\bin, 3) indicamos la periodicidad, 4) la cuenta de usuario del sistema con privi-

63 Para poder poner la fecha en que se hace la copia en el fichero SQL generado por el comando *mysqldump* es posible hacer uso del comando date.

legios para la ejecución de la tarea, 5) marcando la opción de ver opciones avanzadas, 6) editaremos el ejecutable añadiendo las opciones **—user**=*root* y **—password**=*mipass*, seguido del nombre de la base de datos, *miweb*, y el redireccionamiento sobre el fichero **.sql* que almacenará las sentencias SQL que permiten la recuperación de la base de datos: "**mysqldump —opt —user**=*root* **—password**=*mipass* miweb **>** *c:\xampp\copias_miweb\bd_miweb.sql*" (*será necesario crear el directorio copias_miweb*).

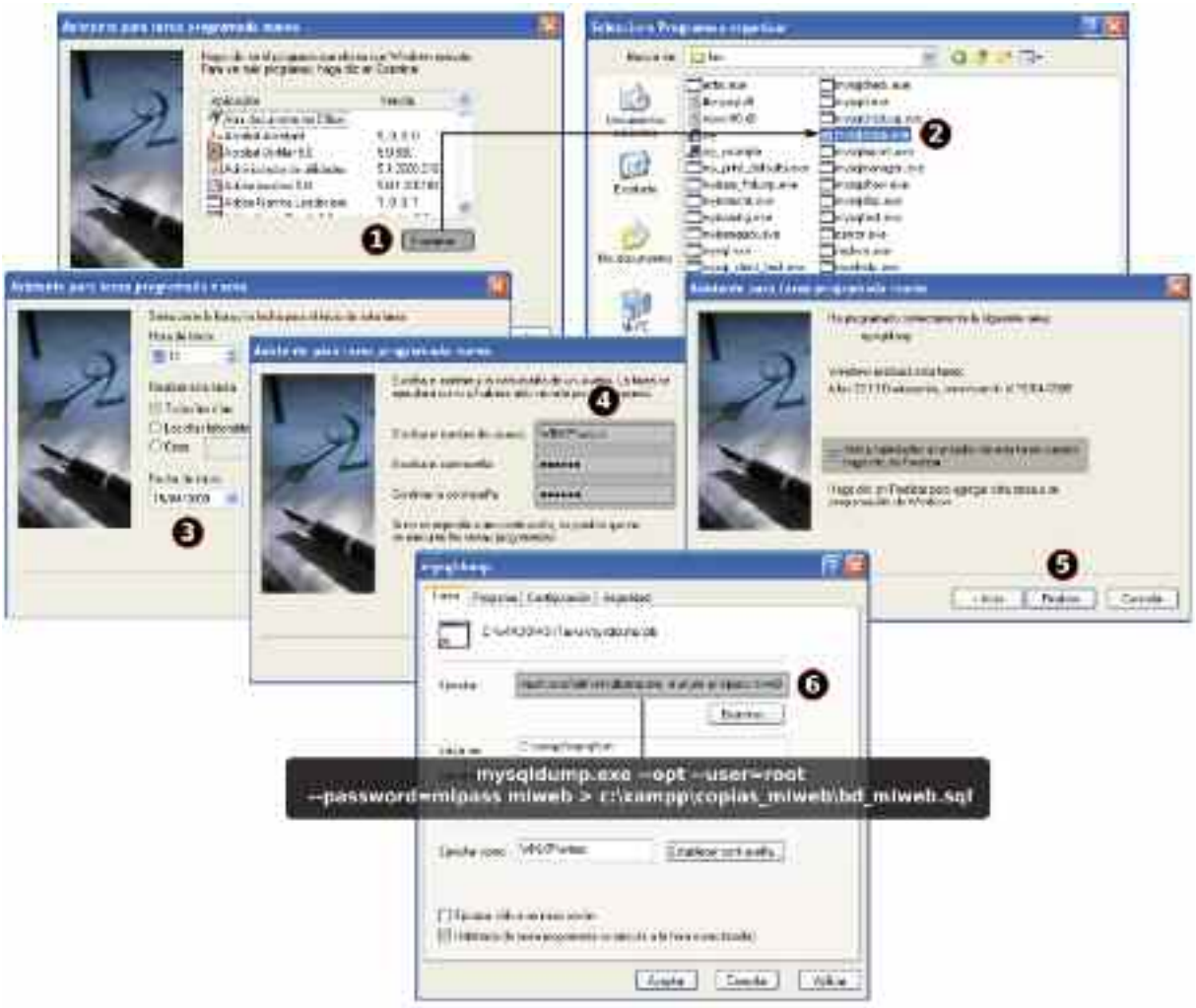


Figura 8.14. Pasos para la configuración de la tarea programada mysqldump en Windows

Al igual que bajo GNU/Linux, tras haber seguido todos los pasos anteriores, dispondrás diariamente de una nueva copia de seguridad de la base de datos MySQL *miweb*. En el caso de necesitar restaurar una de las copias, redireccionaremos la entrada MySQL desde el fichero **.sql* anterior, ejecutando el siguiente comando:

```
c:\xampp\mysql\bin> mysql.exe —user=root —password=mipass miweb < bd_miweb_[fecha].sql
```

¡¡Observación!! Lista el contenido del fichero de respaldo de la base de datos miweb **.sql*, y advierte la gran cantidad de sentencias SQL que se generan y que nos permiten restablecer la base de datos y todas sus tablas en caso de ser necesario.

Capítulo 9

PORTALES CON PYTHON. APACHE Y MOD_PYTHON EN ACCION

mod_python es un módulo para el servidor web Apache que permite la ejecución de programas python embebidas en el propio servidor. Aunque es posible ejecutar scripts CGI escritos en Python, mod_python esta pensado para sustituir a dichos scripts; va más allá y ofrece algunas ventajas sobre ellos:

— Una mayor velocidad de ejecución. Cada vez que se hace una conexión a un script CGI el servidor debe arrancar un nuevo proceso. Sin embargo con mod_python trás la ejecución del script el proceso no termina y por tanto en la siguiente conexión no debe ser arrancado de nuevo. En http://www.modpython.org/live/current/doc-html/intr-performance.html es posible encontrar un test comparativo en el que se muestra la velocidad en la atención de solicitudes de servicio.

— Mantiene datos almacenados en diferentes sesiones. Puesto que los procesos de ejecución iniciados al ejecutar scripts de Python no terminan al finalizar una conexión, los datos generados, proporcionados o calculados no se pierden, pudiendo ser almacenados y reutilizados en próximas conexiones.

— Los programas Python que mod_python ejecutará se almacenan en el servidor y por tanto podrán hacer casi cualquier cosa.

— Es capaz de comunicarse directamente con Apache. Apache procesa las peticiones de servicio por fases[64] a través de sus handlers; mod_python ofrece sus propios handlers, es decir funciones que procesan cada una de las fases y que pueden ser utilizadas en lugar de las que Apache ejecuta por defecto.

— Permite el uso Python embebido en las páginas web (PSP-Python Server Pages). Esto proporciona la posibilidad de realizar contenido dinámico similar a PHP-PHP Hypertext Preprocessor, ASP-Active Server Pages o JSP-Java Server Pages[65].

---

64 Por ejemplo, 1.ª fase: autentificar al usuario, 2.ª fase: comprobar si el usuario tiene acceso al contenido, 3.ª fase: mandar el contenido al usuario, 4.ª fase: hacer un log de la solicitud servida.

65 Para comprobar la velocidad de ejecución de páginas que contienen embebido código para crear contenidos dinámicos .psp y .php es posible acceder a las páginas **http://ralph.nuoj.com/alien.psp** y **http://ralph.nuoj.com/alien.php**. Ambas están destinadas a la creación de nombres extraños (un poco friki, ¿eh?). Se supone que ambas hacen lo mismo por lo que para comprobar la rapidez de ejecución bastaría con recargar ambas; esto regenerará la lista de nombres creados. Este simple test proporciona una idea acerca de la velocidad de ejecución de una página .psp.

1. INSTALACIÓN

mod_python se compila para trabajar en Apache como un DSO (Dynamic Shared Object). Aunque normalmente no es necesario hacer la compilación e instalación de manera manual (configure, make & make install) porque las principales distribuciones Linux ofrecen mod_python en forma de .rpm, .deb o similar. Por ejemplo en un distribución Mandriva (similar para las basadas en Red Hat) la instalación sería:

```
[root@mandriva]# urpmi mod_python
ftp://ftp.uni-
bayreuth.de/pub/linux/Mandrakelinux/official/2008.0/i586/media/contrib/release/apache-
mod_python-3.3.1-3mdv2008.0.i586.rpm
instalando apache-mod_python-3.3.1-3mdv2008.0.i586.rpm desde /var/cache/urpmi/rpms
Preparando...
########################################################################
    1/1: apache-mod_python
########################################################
[root@portatil]#
```

En una distribución basada en Debian:

```
debian# apt-get install libapache2-mod-python
Se instalarán los siguiente paquetes NUEVOS:   libapache2-mod-python
0 paquetes actualizados, 1 nuevos instalados, 0 para eliminar y 0 sin actualizar.
Necesito descargar 136kB de ficheros. Después de desempaquetar se usarán 590kB.

¿Quiere continuar? [Y/n/?] Y
Escribiendo información de estado extendido... Hecho
Des:1 http://ftp.de.debian.org lenny/main libapache2-mod-python 3.3.1-5 [136kB]
Descargados 136kB en 0s (150kB/s).
Preconfigurando paquetes ...
Seleccionando el paquete libapache2-mod-python previamente no seleccionado.
(Leyendo la base de datos ...132638 ficheros y directorios instalados actualmente.)
Desempaquetando libapache2-mod-python (de .../libapache2-mod-python_3.3.1-
5_amd64.deb) ...
Configurando libapache2-mod-python (3.3.1-5) ...

debian#
```

Tras la instalación y para poder utilizarlo, hay que decir a Apache que cargue el módulo. Esto se hace por medio de la directiva LoadModule en el fichero de configuración de Apache httpd.conf. Este fichero se encuentra dentro del directorio /etc. Utiliza el comando find para conocer su ubicación concreta en el caso de que la desconozcas:

```
[root@portatil]# find /etc/ -name httpd.conf
/etc/httpd/conf/httpd.conf
```

Edita el fichero y acude a la zona del mismo donde aparecen las directivas para cargar los módulos que Apache necesita. Esta parte podría tener una forma similar a:

```
                .
                .
                .
LoadModule dir_module modules/mod_dir.so
LoadModule imagemap_module modules/mod_imagemap.so
LoadModule actions_module modules/mod_actions.so
#LoadModule speling_module modules/mod_speling.so
LoadModule userdir_module modules/mod_userdir.so  # -> available in the apache-
mod_userdir package
LoadModule alias_module modules/mod_alias.so
LoadModule rewrite_module modules/mod_rewrite.so
                .
                .
                .
```

Observa el uso de LoadModule. La directiva hace un enlace con la librería .so (por ejemplo modules/mod_dir.so) y carga el módulo indicado (en el ejemplo sería dir_module) a la lista de los módulos activos. Observa también como se indica la ruta relativa al ServerRoot donde se encuentra la librería .so, así en el presente ejemplo, si la directiva ServerRoot es:

```
ServerRoot “/etc/httpd”
```

se sabe que la ubicación física de mod_dir.so es /etc/httpd/modules/mod_dir.so; no obstante conviene señalar que el directorio modules suele ser un enlace simbólico a un directorio que se ubica realmente dentro de la carpeta /usr/. Esto se puede comprobar de la siguiente forma:

```
[root@portatil]# ls -l /etc/httpd/

drwxr-xr-x 8 root root 4096 2008-02-20 15:51 conf/
drwxr-xr-x 2 root root 4096 2007-09-07 22:47 conf.d/
lrwxrwxrwx 1 root root   33 2008-01-08 16:57 extramodules -> ../../usr/lib/apache-extramodu-
les/
lrwxrwxrwx 1 root root   13 2008-01-08 16:57 lib -> ../../usr/lib/
lrwxrwxrwx 1 root root   19 2008-01-08 16:57 logs -> ../../var/log/httpd/
lrwxrwxrwx 1 root root   20 2008-01-08 16:57 modules -> ../../usr/lib/apache/
drwxr-xr-x 2 root root 4096 2008-11-25 22:49 modules.d/
```

En este caso se comprueba que el directorio /etc/httpd/modules/ realmente no existe y que es un enlace al directorio /usr/lib/apache/.

Por supuesto, lo que se acaba de mostrar y lo que se muestra a continuación es simplemente un ejemplo. El objetivo es encontrar la ubicación de la libreía de mod_python.so y cargarla en el httpd.conf, en cada distribución dicha librería puede tener ubicaciones diferentes y por tanto las siguientes líneas son sólo una referencia de los pasos a dar para la correcta instalación.

Lo importante, tras la instalación de mod_python, es saber donde se encuentra la librería mod_python.so que debe ser cargada en el httpd.conf. Para ello se puede hacer uso de nuevo del comando find:

```
[juanjo@portatil]# find /usr -name mod_python.so
/usr/lib/apache-extramodules/mod_python.so
```

Para encontrar el enlace simbólico en /etc/ se puede usar find con la opción -type l, tal como se muestra a continuación:

```
[root@portatil]# find /etc -type l -name *modules*
/etc/httpd/extramodules
/etc/httpd/modules
```

Utilizando en anterior listado se observa que /usr/lib/apache-extramodules/ tiene un enlace simbólico a /etc/httpd/extramodules/ y de nuevo se puede comprobar el contenido de ese directorio enlazado simbolicamente:

```
[root@portatil]# ls -l /etc/httpd/extramodules/

-rwxr-xr-x 1 root root  32152 2007-09-08 13:13 mod_php5.so*
-rwxr-xr-x 1 root root 115012 2007-09-09 00:52 mod_python.so*
```

Quedando probado donde se encuentra la librería mod_python.so con referencia a la ruta indicada en la directiva ServerRoot. Así en el fichero httpd.conf bastaría con añadir la línea:

```
LoadModule python_module extramodules/mod_python.so
```

2. EL PRIMER EJEMPLO

En primer lugar vamos a asegurarnos de que Apache funciona adecuadamente configurándolo de la forma habitual para que sirva un index.html como el mostrado a continuación:

```
<HTML>
<BODY LANG=”es-ES” DIR=”LTR”>
<P ALIGN=CENTER><FONT COLOR=”#ff0000”><FONT SIZE=16><SPAN STYLE=”back-
ground: #cccc00”>Funcionamiento Tradicional de Apache </SPAN></FONT></FONT></P>
</BODY>
</HTML>
```

Este contenido html es guardado en el fichero /var/www/html/webpython/index.html. Suponiendo que la máquina que está corriendo Apache tiene la dirección IP 192.168.10.150 y que la ubicación relativa de los archivos de configuración es con respecto a la ruta /etc/httpd/ bastaría con añadir las siguientes líneas al archivo httpd.conf:

```
ServerRoot “/etc/httpd”
Listen 192.168.10.150:80
ServerName 192.168.10.150:80
DocumentRoot “/var/www/html/webpython/”
```

Arrancamos (o rearrancamos) el servicio:

```
( [root@portatil]# apachectl start )

[root@portatil]# /etc/init.d/httpd restart

Starting httpd:                                    [  OK  ]
```

y en principio un navegador web nos mostraría algo similar a la figura 9.1 en pantalla.

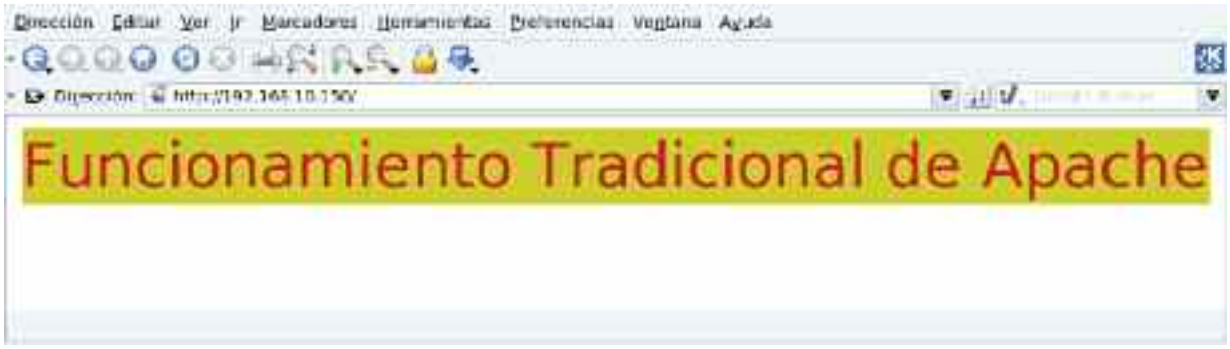


Figura 9.1. Prueba de funcionamiento de Apache

A continuación haremos funcionar a mod_python. En primer lugar debe añadirse al fichero httpd.conf información sobre el directorio que va a contener el programa python a ejecutar:

```
<Directory /var/www/html/webpython>
   AddHandler mod_python .py
   PythonHandler ejemplo1
   PythonDebug on
</Directory>
```

Las directivas asociadas al directorio /var/www/html/webpython que van condicionar el funcionamiento del servidor son:

1. AddHandler mod_python .py

Esta es una directiva de Apache que añade un handler para manejar archivos con una determinada extensión. En este caso, los archivos con extensión .py del directorio señalado serán manejados con mod_python cuando sean invocados desde el navegador. Sin esta directiva Apache no sabría como interpretar los archivos .py que tuviera el sitio web.

2. PythonHandler ejemplo1

Esta es una directiva de Apache que pertenece a mod_python. PythonHandler es uno de los manejadores para los archivos .py. Indica que los archivos de extensión .py se tratarán a través de un programa python llamado ejemplo1.py. Un programa python puede contener objetos (definidos a través de class) o funciones (definidas a través de def). Inicialmente, y tal como está expresado en el ejemplo, debe existir una función dentro de ejemplo1.py llamada handler[66] que se ejecutará al poner en el navegador como dirección un archivo con extensión .py dentro del directorio /var/www/html/ webpython.

3. PythonDebug on

Esta directiva está pensada para ser utilizada únicamente en la fase de desarrollo del sitio web. Cuando ocurre un error en el procesamiento de datos además de mandarlos al fichero log también lo hace al usuario del navegador a través de la pantalla.

66 Este nombre viene dado por el propio nombre del manejador, en este caso PythonHandler, escrito en minúsculas y con el prefijo Python eliminado. Por lo tanto PythonHandler se convierte en handler.

Una vez modificado el httpd.conf es necesario reiniciar Apache, tal y como se indicó anteriormente, para que los cambios surtan efecto. A continuación creamos el fichero python llamado ejemplo1.py, que será invocado por Apache:

```
from mod_python import apache

def handler(req):
    req.content_type='text/html'
        req.write('<P ALIGN=CENTER><FONT COLOR="#ff0000"><FONT SIZE=16><SPAN
STYLE="background: #cccc00">Funcionamiento con mod_python de Apache
</SPAN></FONT></FONT></P>')
    return apache.OK
```

¡¡Importante!! Cuando se escribe un programa Python hay que tener cuidado con los símbolos que se utilizan, dependiendo de la codificación utilizada: ISO 8859-1, UTF8, ... Por esta razón existen dos posibilidades: o se ajusta adecuadamente el tipo de codificación a utilizar o se recomienda no utilizar símbolos que cambien entre codificaciones, por ejemplo tildes o similar.

Por último escribe en el navegador la url apuntando a ejemplo1.py para obtener algo similar a lo mostrado en la figura 9.2.

Figura 9.2. Prueba de funcionamiento de Apache con mod_python

La descripción del anterior programa python sería la siguiente:

1. from mod_python import apache

En primer lugar se importa el objeto apache de la librería mod_python. Esto permite utilizar las funciones python asociadas al objeto apache.

2. def handler(req):

A continuación comienza la definición de la función handler que toma como argumento la solicitud en forma del objeto req (request). Si se desea definir la función con un nombre diferente, por ejemplo prueba_handler, debe ser indicado en la directiva PythonHandler, en el archivo htttpd.conf, escribiéndola de la siguiente forma: PythonHandler ejemplo1::prueba_handler.

3. req.content_type='text/html'

Esto indica que el contenido de la respuesta que se mandará al navegador y que el usuario visualizará está en formato html. Se podría también haber asignado texto plano como: req.content_type= 'text/plain'. Es muy importante asegurarse de que esta variable tiene un valor asignado antes de la utilización de req.write.

4. req.write('<P ALIGN=CENTER><FONT COLOR="#ff0000"><FONT SIZE=16> <SPAN STYLE="background: #cccc00">Funcionamiento con mod_python de Apache </SPAN></FONT></FONT></P>')

Esto sirve para indicar lo que se desea mostrar por pantalla al cliente. En este caso un texto en formato html que produce el resultado mostrado en la figura 9.2.

5. return apache.OK

Por último esta instrucción indica a Apache que todo el procesamiento se realizó correctamente. En este caso no hay mucha dificultad, pero en casos más complejos el procesamiento de información puede ser conducido a través de diferentes hilos de programación y cada uno devolver otros mensajes como apache.HTTP_FORBIDEN o apache.CONFLICT o alguna de las multiples respuestas que el servidor Apache está preparado para aceptar[67].

3. SEGUNDO EJEMPLO: EL MANEJADOR PUBLISHER

Puede comprobarse que en el anterior ejemplo, la llamada al contenido del programa ejemplo1.py a través del navegador podría haberse hecho con cualquier otro nombre, por ejemplo noexiste.py tal y como se muestra en la figura 9.3.

Figura 9.3. Gestión de archivos .py por parte de PythonHandler

Esto es debido a que en la configuración del servidor Apache ya fue especificado como manejador específico el script ejemplo1.py al escribir: PythonHandler ejemplo1. Al leer en la URL un fichero .py, tenga el nombre que tenga, el control pasa directamente al programa ejemplo1.py, que a su vez ejecutará la función handler contenida en él.

En este punto surge la siguiente pregunta: ¿Cómo pueden ser ejecutados diferentes programas o incluso diferentes funciones dentro de un programa?

Para hacer esto mod_python define lo que se denomina el manejador publisher (o publisher handler). Haciendo uso de esta herramienta se puede acceder a cualquier objeto, función o variable de un determinado programa. Un ejemplo muy sencillo se explica a continuación. Modifica el contenido de la directiva <Directory /var/www/html/webpython> en el archivo htttpd.conf con las siguientes líneas:

```
<Directory /var/www/html/webpython>
   SetHandler mod_python
   PythonHandler mod_python.publisher
</Directory>
```

67 Ver en http://httpd.apache.org/dev/apidoc/ las definiciones de constantes (Constant Definitions)

Al utilizar la directiva SetHandler se está obligando a utilizar mod_python para interpretar todos los archivos del directorio /var/www/html/webpython. A continuaciónm con la siguiente directiva se indica que el manejador será el publisher de mod_python.

Con esta configuración se podrá acceder de forma personalizada a los archivos contenidos en el directorio /var/www/html/webpython. Guarda en dicho directorio el siguiente archivo python con el nombre de respuesta.py:

```
from mod_python import apache
def mostrar(req,resp="mucho"):
    return "Estoy estudiando %s" %resp
```

Este programa define una función llamada "mostrar" que recibe dos argumentos el primero es la solicitud propia del servicio Apache (req) y la segunda es una variable llamada resp que toma por defecto el valor "mucho".

Al hacer la llamada a través de una url en el navegador web debe indicarse la función del correspondiente programa que debe ser invocada. Por ejemplo, http://192.168.10.150/ejemplo2.py/mostrar proporcionaría por pantalla lo mostrado en la figura 9.4.

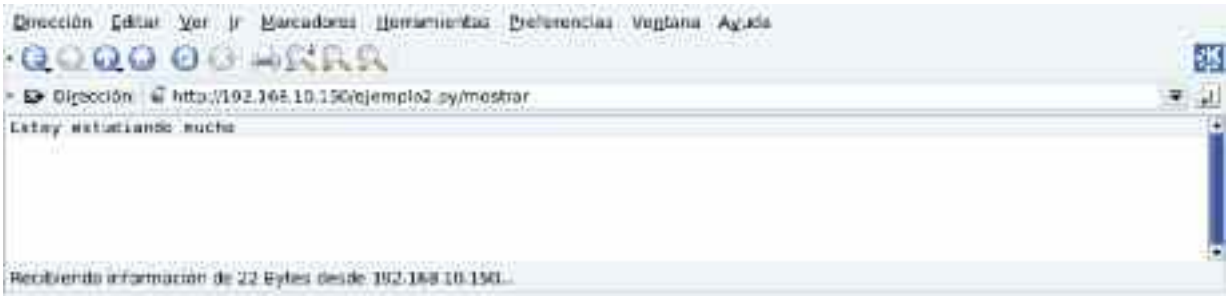


Figura 9.4. Ejemplo de funcionamiento del mod_python.publisher

Para ver una respuesta similar a la mostrada en la sección anterior deberíamos escribir la url: http://192.168.10.150/ejemplo1.py/handler.

El manejador publisher permite acceder directamente a las variables de un programa. El contenido de la variable resp podría modificarse tal y como se muestra en la figura 9.5[68].

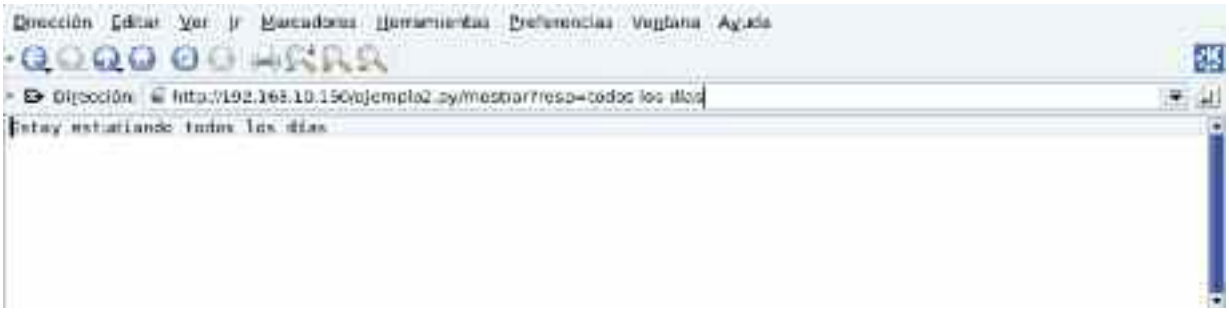


Figura 9.5. Acceso del mod_python.publisher a variables de un programa

Obsérvese que en este caso se está asignando a la variable resp la cadena de caracteres "todos los días".

68 Nótese que al no utilizar en el programa ninguna función definida internamente por mod_python no hubiera hecho falta la línea: from mod_python import apache.

4. TERCER EJEMPLO: RESPUESTA A UN FORMULARIO

En este ejemplo se utilizará mod_python para responder al envío de un formulario, por parte de un cliente web, a través de un correo electrónico. Para ello se hará uso del manejador publisher descrito en la sección anterior.

En primer lugar se debe configurar el archivo httpd.conf y después rearrancar Apache para que los cambios sean tenidos en cuenta. Siguiendo la misma línea que en ejemplos anteriores, la configuración puede realizarse para que únicamente tenga efecto sobre el directorio /var/www/html/webpython:

```
<Directory /var/www/html/webpython>
    AddHandler mod_python .py
    PythonHandler mod_python.publisher
</Directory>
```

Por otro lado se van a definir dos archivos: formulario.html y formulario.py. El primero de ellos mostrará el formulario al cliente web y el segundo responderá apropiadamente ante una recepción de datos proveniente de dicho formulario. El contenido de formulario.html sería:

```
<HTML>
<B>¡¡Importante!! Rellena todos los campos mostrados a continuación:</B>
<P>
<FORM ACTION="formulario.py/email" METHOD="POST">
  <FONT COLOR="#800000"> <B>Nombre:</B></FONT> <INPUT TYPE=TEXT
NAME="nombre">
  <FONT COLOR="#800000"> <B>Email:</B></FONT> <INPUT TYPE=TEXT
NAME="email">
      <P>
  <FONT COLOR="#800000"> <B>Comentario:</B></FONT>
  <TEXTAREA NAME="comentario" ROWS=3 COLS=25></TEXTAREA>
      <INPUT TYPE=SUBMIT VALUE="MANDAR">
      </P>
</FORM>
</P>
</HTML>
```

que proporcionaría una respuesta sobre el navegador similar a la mostrada en la figura 9.6.

Figura 9.6. Formulario propuesto para el tercer ejemplo

Obsérvese que la acción del formulario se destina a formulario.py/email, lo cual significa que la encargada de proporcionar una adecuada respuesta ante la recepción de los datos del formulario será una función denominada "email" contenida en el programa formulario.py. El contenido de este archivo podría ser:

```
import smtplib
WEBMASTER = "juanjo@cossio.net"
SMTP_SERVER = "smtp.cossio.net"
def email(req, nombre, email, comentario):
    if not (nombre and email and comentario):
        return "Debes completar todos los campos"
    # Creación del contenido del correo
    msg = ""\
Correo de: %s
Asunto: formulario
Dirigido a: %s
Mi comentario es el siguiente:
%s
Atentamente,
%s
"" % (email, WEBMASTER, comentario, nombre)

    conn = smtplib.SMTP(SMTP_SERVER)
    conn = smtplib.login('usuario','password')
    conn.sendmail(email, [WEBMASTER], msg)
    conn.quit()
    # A mostrar en pantalla para el usuario:
    salida = ""\
<html>
Estimado %s,<br>
Gracias por mandarnos tus comentarios, nos pondremos
en contacto contigo con la mayor brevedad posible.
</html>"" % nombre
    return salida
```

5. CUARTO EJEMPLO: AUTENTICACIÓN

Este ejemplo va a mostrar como usar mod_python para autenticar accesos a un contenido web. Por supuesto, esto es capaz de hacerlo Apache por sí mismo, pero nos permitirá ver otro posible uso de mod_python y de sus handlers. De hecho en este caso se necesitará utilizar el PythonAuthenHandler que a su vez requiere de un tipo de autenticación del tipo Basic, para poder decodificar la contraseña transmitida en base64. La configuración del httpd.conf requerida para el directorio asociado al sitio web sería:

```
<Directory /var/www/html/webpython>
    AddHandler mod_python .py
    PythonHandler miprograma
    PythonAuthenHandler miprograma
    PythonDebug on  #No es requerido pero ayuda a depurar los errores iniciales
    AuthType Basic
    AuthName "Area restringida a usuarios con permisos"
    require valid-user
</Directory>
```

El sitio web donde se requiere la autenticación está definido dentro del directorio /var/www/html/ webpython. A continuación se añade como handler a mod_python para aquellos ficheros con extensión .py. Posteriormente se definen dos handlers (PythonHandler y Python AuthenHandler) asociados al mismo programa python, **miprograma.py**. Esto es posible debido a que dependiendo del manejador llamado se ejecutarán diferentes funciones (o métodos) descritos en el programa **miprograma.py**; es decir cuando sea utilizado el PythonAuthenHandler la función llamada será **authenhandler**, y cuando sea utilizado el PythonHandler la función a ejecutar dentro de **miprograma.py** será **handler**. La utilización de **PythonDebug on** no sólo no es requerido, sino que en un funcionamiento normal del servidor debe evitarse su uso, ya que podría facilitar información privilegiada al usuario o cliente web.

AuthType Basic determina que la codificación http utilizada para la transmisión del la contraseña se hace en base64, la cual es la única que es capaz de codificar mod_python en el momento de escribir estas líneas. AuthName determina la frase que se imprimirá en el momento de intentar acceder al sitio web. Por último **require valid-user** es la directiva utilizada para indicar que sólo se permita el acceso si se utiliza un usuario válido, si no se utiliza cualquier usuario tendrá acceso.

Suponiendo que se desea validar al usuario **hugo**, que posee la contraseña **Ari02**, el contenido de **miprograma.py** podría ser:

```
from mod_python import apache

def authenhandler(req):
    contrase = req.get_basic_auth_pw()
    usuario = req.user
    if usuario == ‘’hugo” and contrase == ‘’Ari02”:
        return apache.OK
    else:
        return apache.HTTP_UNAUTHORIZED

def handler(req):
    req.content_type=’text/html’
        req.write(‘<P ALIGN=CENTER><FONT COLOR=”#ff0000”><FONT SIZE=16><SPAN
STYLE=”background: #cccc00”>Funcionamiento con mod_python de Apache
</SPAN></FONT></FONT></P>’)
    return apache.OK
```

Con esta configuración, al escribir la url http://192.168.10.150/ aparecerá en pantalla algo similar a lo mostrado en la figura 9.7.

Figura 9.7. Ventana de autenticación generada desde mod_python

A continuación se describe el funcionamiento del anterior programa python. En primer lugar se hace la definición de la función encargada de la autenticación:

```
def authenhandler(req):
```

Puesto que en el httpd.conf se ha escrito la directiva PythonAuthenHandler miprograma, el programa debe llamarse miprograma.py y la función python encargada de la autenticación igual que la directiva, pero en minúsculas y eliminando el término Python: PythonAuthenHandler —-> authenhandler.

```
    contrase = req.get_basic_auth_pw()
```

Al tratar de acceder al directorio protegido con la autenticación se ejecutará esta función donde inicialmente se define la variable **contrase**. La contraseña es mandada por el cliente web al hacer la solicitud de acceso con una codificación base64 (pulsar el botón aceptar de la ventana mostrada en la figura autenticación.eps) y recogida en el objeto **req**. mod_python acepta la solicitud a través de Apache y decodifica la contraseña a través de la función req.get_basic_auth_pw(). Una vez decodificada la almacena en la variable **contrase**.

```
    usuario = req.user
```

Como ya ha sido comentado, al pulsar **aceptar** en la ventana de autenticación el cliente devuelve al servidor la información proporcionada por el usuario y el objeto **req** contendrá en su propiedad **user** el nombre del usuario. Esta instrucción se encarga de almacenar en la variable **usuario** el nombre del usuario.

```
    if usuario == ''hugo'' and contrase == ''Ari02'':
```

Esta línea de código comprueba si el nombre de usuario y la constraseña (almacenadas anteriormente en las variables **usuario** y **contrase**) son las correctas, en cuyo caso ejecutará el contenido del **if**, que en python son aquellas líneas tabuladas con respecto a la línea de condición.

```
        return apache.OK
```

Este es la única instrucción tabulada con respecto al **if** y por tanto la que se ejecutará en caso de que el nombre de usuario y contraseña sean las correctas. Si es así, se devuelve a Apache un **apache.OK**. De esta forma Apache considerará que esta fase de solicitud está completa y deberá continuar con la siguiente fase, que será la lectura del **index.html** existente en el directorio si en la url no se indica nada o la ejecución del **handler** en caso de que la url contenga un archivo .py, tal y como se explicará posteriormente.

```
    else:
        return apache.HTTP_UNAUTHORIZED
```

En el caso de que el nombre de usuario o la contraseña no sean las esperadas se ejecuta esta parte del programa que indica a Apache que el usuario no está autorizado. En este caso Apache manda un mensaje al cliente web indicando la situación en la que se encuentra, tal y como se muestra en la figura no_autorizado.eps.

Tras la descripción del programa se van a comprobar los resultados de ejecución. Si son introducidos el nombre (hugo) y la constraseña (Ari02) adecuadamente se mostrará la pantalla el contenido del archivo index.html que Apache buscará en el directorio correspondiente, en este caso /var/ www/html/webpython. En la figura 9.8 izquierda se muestra la pantalla proporcionada por el navegador.

Si se escribiera en el navegador la url http://192.168.10.150/miprograma.py el comportamiento sería diferente. Inicialmente aparecería en pantalla el mensaje mostrado en la figura 9.7, y tras introducir el nombre y contraseña adecuados se mostraría por pantalla el contenido de la función handler contenida en miprograma.py, tal y como se muestra en la figura 9.8 derecha.

Figura 9.8. Diferentes resultados tras la autenticación

Debe observarse que la url podía haber sido http://192.168.10.150/noexiste.py o cualquier otro nombre de programa inventado con extensión .py, tal y como ya fue comentado en el apartado 2. SEGUNDO EJEMPLO: EL MANEJADOR PUBLISHER para obtener un resultado semejante al mostrado en la figura 9.8 derecha.

6. QUINTO EJEMPLO: ENVÍO DE COOKIES

Una de las características que un portal de Internet debe manejar adecuadamente son las cookies. Las cookies son porciones de información que el servidor web solicita sean almacenadas por el cliente a través del navegador. El navegador almacena esta información en el disco duro del ordenador de forma que cuando el cliente vuelva a visitar el sitio web que emitió la cookie, este pueda responder con las preferencias indicadas por el usuario, como fondos, diseño, colores... Las cookies también son utilizadas para guardar nombres de usuario y contraseñas en el acceso a determinados sitios web. Esto último presenta el problema de que si una persona identificada como un determinado usuario y almacena una cookie para entrar en una sitio con un determinado usuario y contraseña, cualquier otra persona que posteriormente se identifique con ese usuario también accederá; por eso se dice que la cookies no identifican personas sino usuarios del ordenador (no del sitio web) y navegadores (debido a que cada navegador almacenará sus cookies).

Un inconveniente que presentan la cookies es el de que pueden proporcionar información sobre los hábitos de un usuario. Por esta razón, los detractores de las cookies acusan a este sistema de restar privacidad a los usarios. En cualquier caso el uso o no de las cookies está controlado por el usuario del navegador web, que puede configurarlo para que no las acepte si así lo desea. A modo de ejemplo en los navegadores recomendados en este libro:

— En Mozilla basta con ir a **Editar** -> **Preferencias** -> pestaña **Privacidad** y hacer la selección deseada.

— En Opera ir a **Tools** -> **Preferences** -> Pestaña **Advanced** y seleccionar en el menú de la izquierda **Cookies** para seleccionar el comportamiento requerido.

Obsérvese que para comprobar el correcto funcionamiento de las siguientes configuraciones será necesario tener habilitado el uso de las cookies.

El uso de las cookies fue inventado por Netscape y mod_python usa la especificación publicada por Netscape para la creación, tratamiento, envío y recepción de cookies. Existen varios RFC (Request For Comments)[69] que señalan como debe implementarse el mecanismo de gestión de cookies, sin embargo en la práctica la mayoría de los navegadores usan la especificación dictada originalmente por Netscape, y esto implica que no son completamente compatibles con las RFCs. Por su parte mod_python se basa también en la especificación inicial de Netscape y por tanto es compatible con la mayoría de los navegadores actuales.

En este primer ejemplo se va a mostrar como se envía una cookie al cliente que entra en nuestro sitio web. Para reutilizar el código anterior se va a mantener la configuración de conexión segura explicada anteriormente con pequeñas modificaciones. La parte del httpd.conf que nos interesa tendría la siguiente estructura:

```
<Directory /var/www/html/webpython>
    AddHandler mod_python .py
    PythonHandler cookie
    PythonAuthenHandler cookie
    PythonDebug on  #Ayuda a depurar los errores iniciales
    AuthType Basic
    AuthName “Area restringida a usuarios con permisos”
    require valid-user
</Directory>
```

El único cambio introducido con respecto a la configuración de la anterior sección es la de que el programa python que contendrá las funciones **handler** y **authenhandler** será **cookie.py**. Así no es necesario borrar los programas que anteriormente se han utilizado y podrán quedar todos en el directorio /var/www/html/webpython junto a **cookie.py** que se describe a continuación.

```
from mod_python import Cookie, apache
import time

def authenhandler(req):
    contrase = req.get_basic_auth_pw()
    usuario = req.user
    if usuario == “hugo” and contrase == “Ari02”:
        return apache.OK
    else:
        return apache.HTTP_UNAUTHORIZED

def handler(req):
    galleta = Cookie.Cookie(‘primera’,’prueba de funcionamiento’)
    galleta.expires = time.time()+40000
    galleta.path = ‘/var/www/html/webpython’
    Cookie.add_cookie(req, galleta)
    req.write(‘Te hemos enviado una cookie.\n’)
    return apache.OK
```

69 Los RFC son documentos donde se explican los protocolos a usar en Internet sin que haya espacio para la duda en la implementación de los mismos. RFCs relacionados con las cookies son el RFC 2109, RFC 2964 y RFC 2965.

Veamos para qué sirve cada una de las líneas de código.

```
from mod_python import Cookie, apache
import time
```

La primera línea importa la clase Cookie y apache pertenecientes al módulo mod_python. Cookie es la clase que nos permite crear, enviar y recibir las cookies.

La segunda línea importa la librería **time**, que permitirá trabajar con fechas y tiempos en función del reloj del sistema. Ambas líneas de código cargan clases de diferentes librerías, la diferencia entre las dos formas radica en lo siguiente: La primera línea importa dos de las clases contenidas en mod_python y cuando vayan a ser utilizadas basta con escribir el nombre de la clase. La segunda línea importa todas las clases de la librería **time** y para emplear alguna de esas clases será necesario utilizar como prefijo el nombre de la librería. Esto son cuestiones de Python, pero posteriormente se dará otra pequeña explicación.

La función **authenhandler(req)** fue descrita en la sección anterior, página xxx. No se ha cambiado un ápice y por tanto no será explicada de nuevo.

```
def handler(req):
    galleta = Cookie.Cookie('primera','prueba de funcionamiento')
```

Tras la definición de la función **handler** se hace uso de la clase **Cookie** para crear una que es asignada a la variable **galleta**. Una cookie tiene una serie de parámetros como **name**, **value**, **version**, **path**, **domain**, **secure**, **comment**, **expires**, ... Estos parámetros configuran la cookie y son los datos enviados al navegador web para que sean almacenados. Con respecto a nuestro programa, la clase **Cookie** posee un método, llamado de la misma forma: **Cookie**. La estructura de este método es:

```
Cookie(name,value[,atributes])
```

Esto significa que **name** y **value** son obligatorios si se desea crear una cookie y el resto, listado anteriormente, no son estrictamente necesarios. En la línea de código del programa que estamos analizando únicamente han sido asignados el atributo **name** con el contenido **primera** y el atributo **value** con el contenido **prueba de funcionamiento**.

```
galleta.expires = time.time()+40000
galleta.path = '/var/www/html/webpython'
```

Estas dos líneas de código añaden atributos al objeto **galleta** creado anteriormente. Podrían haber sido añadidos en el anterior método utilizando argumentos keyword (indicando el atributo al que se hace referencia):

```
cookie = Cookie.Cookie('primera','prueba de funcionamiento', expires=time.time()+40000,
path='/var/www/html/webpython')
```

El orden de los atributos dados en forma de argumentos keyword es indiferente. El atributo **expires** indica que la cookie expirará una vez transcurridos 40000 segundos desde el momento del envío. A una cookie que tiene asignado un tiempo de expiración se denomina persistente porque aunque el navegador se cierre la cookie se mantendrá almacenada en la próxima sesión. Sin embargo cuando no se da un tiempo de expiración la cookie es borrada por el navegador una vez finalizada la sesión.

La función **time()** pertenece a la librería **time** anteriormente importada. Si la la librería se hubiese importado de la forma:

```
from time import time
```

Esto significa de la librería **time** importa la función (o método) **time**. Para asignar el tiempo de expiración de la cookie, la línea de código hubiera sido:

```
galleta.expires = time()+40000
```

donde no se usa el nombre de la librería como prefijo para llamar al método **time**. La otra línea de código se utiliza para asignar al atributo **path** un contenido que en este caso es el lugar donde está ubicada la página web, pero que podría haber sido cualquier cosa.

```
Cookie.add_cookie(req, galleta)
```

El método **add_cookie()** perteneciente a la clase **Cookie** permite enviar al navegador web la cookie creada. Tras la ejecución de esta línea el cliente web tendrá almacenada la cookie que se ha creado anteriormente.

```
req.write('Te hemos enviado una cookie.\n')
```

Esta línea de código sólo sirve para mandar un mensaje que será mostrado por el navegador.

```
return apache.OK
```

Finalmente esta instrucción, como en casos anteriores, sirve para indicar a Apache que esta fase de procesamiento a finalizado.

Para comprobar el funcionamiento, guarda el programa **cookie.py** en /var/www/html/webpython, así como la modificación del httpd.conf en su lugar correspondiente. Reinicia el servicio httpd:

```
[root@portatil]# /etc/init.d/httpd restart
```

y a continuación accede al sitio web que has creado. Tras introducir el nombre de usuario y contraseñas requeridos por la utenticación, se obtiene el mensaje de texto: 'Te hemos enviado una cookie.'

Por ejemplo, con Mozilla se puede comprobar que ha sido enviada yendo a **Editar** -> **Preferencias** -> icono **Privacidad** -> botón **Mostrar cookies**. El resultado es algo similar a lo mostrado en la figura 9.9.

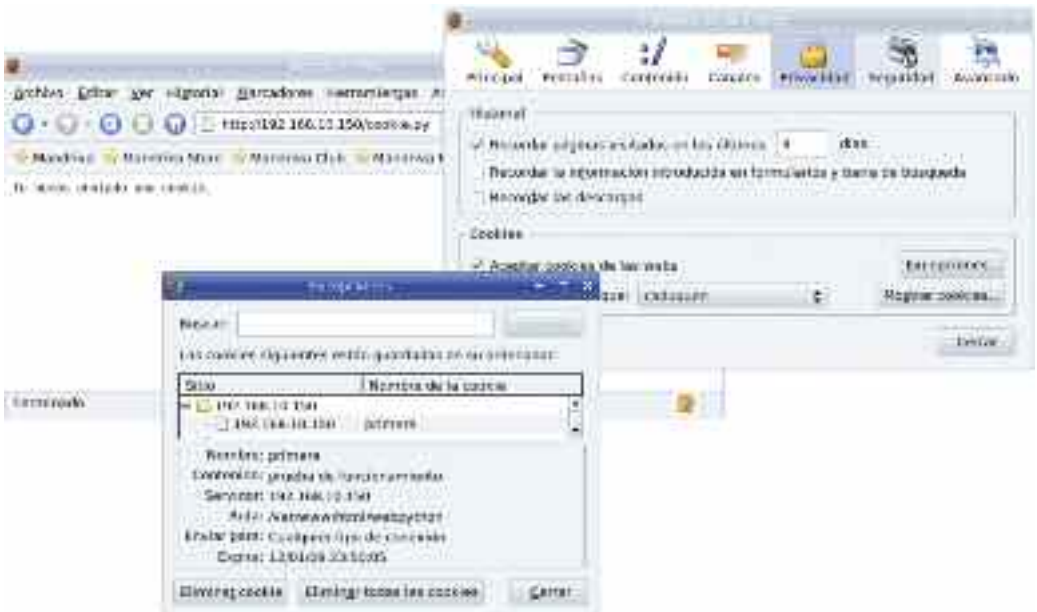


Figura 9.9. Comprobación de la recepción de la cookie

Comprueba cómo coinciden el nombre, contenido y fecha de expiración.

¡¡Importante!! Para probar las diferentes posibilidades de las configuraciones presentadas será necesario en ocasiones borrar las cookies que el servidor web ha enviado al cliente web debido a que no se producirá un reenvío de una cookie en el caso de que esta ya exista. Esto se hará de forma diferente en función del navegador que se esté utilizando, pero las diferencias son mínimas. Al visionar las cookies, tal y como se ha explicado y mostrado anteriormente a través de la figura 9.9 existirá una opción para eliminar las cookies.

7. SEXTO EJEMPLO: MANEJO DE COOKIES

Mandar cookies a un cliente para después no hacer nada más, tal y como se ha hecho en el ejemplo anterior, no tiene ningún sentido. El objetivo es recuperar la cookie que fue mandada a un cliente cuando este vuelve a visitar una página web determinada. Esto permite construir estadísticas, determinar la fidelidad de clientes y ofrecer servicios personalizados.

Además la información mandada con la cookie en el anterior ejemplo puede ser leída por cualquiera que interfiera la comunicación, así como por el propio cliente. Por esta razón generalmente es deseable mandar los datos encriptados y mod_python sabe como hacerlo.

En este ejemplo se va a mostrar como realizar esas tareas. Además en ocasiones se desea mandar más de un valor en una cookie, para ello mod_python serializa los datos, es decir convierte un conjunto de datos en una estructura autocontenida que en Python se denomina **marshalling** (marshalarizar).

La configuración del **httpd.conf** puede dejarse igual. En el caso de que no se desee cambiar el archivo **cookie.py** escrito para el ejemplo anterior bastaría con escribir en el **httpd.conf** lo siguiente:

```
<Directory /var/www/html/webpython>
    AddHandler mod_python .py
    PythonHandler cookie2
    PythonAuthenHandler cookie2
    PythonDebug on  #Ayuda a depurar los errores iniciales
    AuthType Basic
    AuthName “Area restringida a usuarios con permisos”
    require valid-user
</Directory>
```

Donde se indica que el programa python que contiene el código a ejecutar por los handlers es **cookie2.py**. Una vez hecho esto, el siguiente paso consiste en escribir dicho programa. El objetivo del mismo es el siguiente: cuando un usuario se conecte a nuestra página web se debe comprobar si tiene o no almacenada una cookie que atestigue anteriores conexiones. Si la tiene el programa devolverá los datos que en su día se mandaron y en el caso de que no la tenga (porque es la primera vez que entra en el sitio web) se mandará una. Con estas premisas el programa sería:

```
from mod_python import Cookie, apache

def authenhandler(req):
    contrase = req.get_basic_auth_pw()
    usuario = req.user
    if usuario == “hugo” and contrase == “Ari02”:
```

```
        return apache.OK
    else:
        return apache.HTTP_UNAUTHORIZED

def handler(req):
    galletas = Cookie.get_cookies(req, Cookie.MarshalCookie,secret='descifrador')
    if galletas.has_key('contador'):
        contadorcookie = galletas['contador']
        req.write('Se ha encontrado una cookie del tipo contador: %s\n\n' % str(contadorcookie))
        if type(contadorcookie) is Cookie.MarshalCookie:
                    req.write('El contenido decodificado de la cookie %s es: %s\n' %
(contadorcookie.name, contadorcookie.value))
        else:
          req.write('No se puede decodificar. Puede que la cookie no sea del tipo Marshal o que
la clave de cifrado sea incorrecta')
    else:
        ipaddress=req.get_remote_host(apache.REMOTE_NOLOOKUP,None)
        valor = {'IP': ipaddress , 'fecha': time.ctime()}
        Cookie.add_cookie(req, Cookie.MarshalCookie('contador', valor,'descifrador'))
        req.write('No existía una cookie de tipo contador y ha sido enviada una.\n')
    return apache.OK
```

Se explicará únicamente el método **handler(req)**:

```
galletas = Cookie.get_cookies(req, Cookie.MarshalCookie,secret='descifrador')
```

En esta línea se hace uso del método **get_cookies** perteneciente a la clase **Cookie**. Este método permite recuperar las cookies que habían sido almacenadas anteriormente por el navegador que se está utilizando para visualizar el contenido del sitio web **http:\\192.168.10.150\cookie2.py**. El conjunto de cookies es devuelta en forma de un diccionario Python[70]. La función necesita como argumentos:

— El objeto de solicitud pasado a la función **handler** y en este caso llamado **req**[71].

— El tipo de cookie que se desea recuperar. En este caso es una cookie del tipo Marshal (Marshal- Cookie). En mod_python se pueden definir tres tipos de cookies: normal que no necesitaría ser indicada en este método porque es la tomada por defecto, Signed (SignedCookie) que es una cookie normal encriptada y por último la Marshal que es una cookie encriptada con los datos serializados (un conjunto de datos se almacenan en un solo dato con un formato, en este caso, entendible únicamente por Python).

— El último argumento viene dado por la keyword **secret** que debe contener una cadena ASCII utilizada para descifrar los datos enviados con la cookie usando md5. Para poder realizar el descifrado, esta cadena debe ser la misma que se utilizó durante el cifrado.

```
    if galletas.has_key('contador'):
```

Como ya se ha comentado **galletas** es un diccionario y por tanto puede emplearse el método **has_key** asociado a cualquier diccionario Python para comprobar si existe la palabra **contador**,

70 Un diccionario Python es un tipo de datos, como lo son, por ejemplo, los tipos **int**, **float**, **complex**...

71 En principio podría tener cualquier nombre, pero la forma habitual e incluso recomendada por Gregory Trubetskoy, el creador de mod_python, es utilizar **req**.

que en este caso es el tipo de cookie que deseamos comprobar su existencia. La línea de código podría ser leída como: "Si dentro del diccionario **galletas** existe una palabra llamada **contador** haz lo que viene a continuación".

```
contadorcookie = galletas['contador']
```

Esta línea introduce en la variable **contadorcookie** la cookie del tipo contador existente en el diccionario **galletas**. **galletas** podría contener varias cookies, que no son sino las definiciones asociadas a las palabras que contiene el diccionario. Por tanto la palabra **contador**, perteneciente al diccionario, tiene asociada como definición a la propia cookie de tipo contador que ahora será almacenada en la variable **contadorcookie**. Esto significa que **contadorcookie** es un objeto resultante de una instanciación de la clase Cookie o de una subclase de ella[71].

```
req.write('Se ha encontrado una cookie del tipo contador: %s\n\n' % str(contadorcookie))
```

str(contadorcookie) es una función encargada de transformar la información contenida en el objeto **contadorcookie** en una cadena de caracteres. Por su parte **req.write()** es el método encargado de escribir un texto por pantalla. Observa el uso del operador %s encargado de introducir en la frase contenida en **req.write()** la cadena proporcionada por **str(contadorcookie)**, de la misma forma que es usado en C o C++. Puesto que esos datos estaban codificados, en la pantalla aparecerán de esa forma y serán ilegibles para el usuario.

```
if type(contadorcookie) is Cookie.MarshalCookie:
```

Para que el resultado de esta condición sea afirmativo (True) se deben cumplir dos cosas: La primera es que el objeto **contadorcookie** debe ser una instancia de la clase MarshalCookie. La segunda es que la cadena de descifrado dada en la función **get_cookies** a través del argumento keyword **secret** debe ser la misma que la que se dio en el momento del envío de la cookie.

```
req.write('El contenido decodificado de la cookie %s es: %s\n' %
(contadorcookie.name, contadorcookie.value))
```

Si el resultado de la anterior comparación es afirmativo se imprimen por pantalla los atributos **name** y **value** contenidos dentro de la cookie (objeto) **contadorcookie**. Ahora la información aparece por pantalla descifrada.

```
else:
  req.write('No se puede decodificar. Puede que la cookie no sea del tipo Marshal o que
la clave de cifrado sea incorrecta')
```

En el caso de que la la cookie almacenada en el objeto **contadorcookie** no hubiera sido una del tipo MarshalCookie o si la contraseña de cifrado no hubiese sido **descifrador** la condición a la que hace referencia este **else** habría sido falsa (False) y se ejecutaría el contenido del susodicho **else**. Este contenido indica que debe escribirse por pantalla el texto contenido en la función **req.write**.

```
else:
  ipaddress=req.get_remote_host(apache.REMOTE_NOLOOKUP,None)
```

71 Las subclases de Cookie pueden ser dos: SignedCookie o MarshalCookie. Ver manual de mod_python para más información.

Este **else** hace referencia a la condición **if galletas.has_key('contador'):**[72]. En el caso de que el diccionario **galletas** no contenga la palabra **contador** significará que el cliente web no tenía almacenada una cookie del tipo **contador** lo que conducirá a que el servidor web le mande una. Para ello y en primer lugar a través de el método **get_remote_host()**[73] se está solicitando la dirección IP del cliente que es almacenada en la variable **ipaddress**.

```
valor = {'IP': ipaddress , 'fecha': time.ctime()}
```

En la variable valor se ha almacenado un diccionario que contiene dos palabras: 'IP' que almacena el contenido de la variable **ipaddress**, es decir la dirección IP capturada por el método **get_remote_host()** anteriormente, y 'fecha' que contendrá lo que la función **time.ctime()** devuelva, que en este caso es la fecha y hora en el formato: **Día_semana Mes Día_mes hh:mm:ss año**

```
Cookie.add_cookie(req, Cookie.MarshalCookie('contador', valor,'descifrador'))
```

Esta línea utiliza el método **add_cookie()**, que tiene el cometido de enviar una cookie a un cliente web. Los argumentos son dos: El primero es **req**, que es el objeto de solicitud creado por mod_python al solicitar el cliente web el contenido de una página web al servidor Apache, y . el segundo es una instancia de una clase **Cookie** o subclase. Para este segundo argumento se ha proporcionado una instanciación de la clase **MarshalCookie**, la cual a su vez necesita al menos tres objetos: el nombre (name), en este caso **contador**, el valor (value), en este caso el contenido de la variable **valor** definido en la línea de código anterior, y la clave para el cifrado, en este caso **descifrador**.

```
    req.write('No existía una cookie de tipo contador y ha sido enviada una.\n')
return apache.OK
```

La primera de estas dos líneas pertenece al último **else** listado en el programa (obsérvese la tabulación con respecto al mismo) y sirve para indicar a por medio de la pantalla del navegador que ha sido enviada una cookie. Viendo la tabulación de la segunda línea se observa que pertenece al bloque contenido en el método **def handler(req):** y sirve para indicar a Apache que esta fase de procesamiento ha concluido.

Una vez visto este ejemplo es fácil implementar variaciones en las que el servidor web, a través de un programa Python, almacene en una base de datos las fechas de acceso y direcciones IP de diferentes clientes web. Así pueden ser controlados el número de accesos a una página web por parte de un cliente u otro tipo de aspectos.

8. SÉPTIMO EJEMPLO: USO DE PSP

PSP significa Python Server Pages y su usa radica en embeber código Python dentro de un documento HTML con el objetivo de crear páginas web dinámicas, al igual que hacen PHP, JSP o ASP.

En PSP hay cuatro tipo de elementos o entidades: El código, las expresiones, las directivas y los comentarios.

— Código: Es el contenido de las instrucciones Python que determinan la secuencia de operaciones capaces de generar una determinada salida. Dentro del documento HTML este código esta encerrado entre los símbolos <% y %>.

---

72 Obsérvese que este **else** tiene la misma tabulación que el **if** indicado. Como ya sabrá el lector, el contenido de un **if**, **else**, **for**, **def**... en Python viene dado por las instrucciones tabuladas con respecto a esos comandos.

73 La función o método **get_remote_host()** puede capturar otro tipo de información del cliente. Ver el manual de mod_python para ampliar esta información.

— Expresiones: Es un código Python que se convierte en una salida real en el cliente web. Esta salida es siempre un tipo string y dicho código está encerrado entre los símbolos <%= y %>.

— Directivas: Son instrucciones necesarias para el procesado PSP por parte de mod_python. Las directivas se delimitan por <%@ y %>.

— Comentarios: Permite la introducción de explicaciones a las líneas de código que están siendo creadas por el desarrollador. Se encuentran delimitadas por <%— y —%> y a diferencia de los comentarios HTML no son mandados al cliente web ya que el parser psp los elimina durante la fase de procesado.

¡¡Importante!! Recuerda que el código embebido en una página web, ya sea PSP, PHP, JSP o ASP es preprocesado en el servidor para ser convertido en código HTML. Este último es el enviado al navegador web. Un navegador web no entiende PSP.

Para que Apache pueda entender PSP a través de mod_python es necesario configurarlo adecuadamente. El httpd.conf debería contener lo siguiente en lo referente al directorio /var/www/html/webpython, que es con el que estamos haciendo todas las pruebas en este capítulo, al menos las siguientes líneas:

```
<Directory /var/www/html/webpython>
   AddHandler mod_python .psp
   PythonHandler mod_python.psp
</Directory>
```

Durante la fase de construcción del sitio web dinámico es conveniente tener activado el modo de depuración por lo que la configuración apache para el anterior directorio sería:

```
<Directory /var/www/html/webpython>
   AddHandler mod_python .psp .psp_
   PythonHandler mod_python.psp
   PythonDebug on
</Directory>
```

Observa que se ha añadido .psp_, lo que significa que mod_python será capaz de procesar este tipo de archivos. De hecho para ver el código python producido por el parser al leer un archivo .psp bastará con escribir en la url del navegador el mismo nombre de archivo con la extensión .psp_.

La aplicación que se va a desarrollar es sencilla, al igual que las anteriores. No se busca la funcionalidad sino aprender el funcionamiento y manejo de las herramientas. En este caso la página web mostrará por pantalla la fecha y la hora de conexión al servidor. La recarga de la página podrá hacerse desde un enlace existente en ella misma lo que posibilitará la actualización dinámica de los datos mostrados.

También contendrá un saludo que tendrá carácter dinámico ya que cambiará en función de una de las opciones ofrecidas.

Por último el archivo ilustra como usar el comando **for** para crear bucles. El archivo será llamado **index.psp** y tendrá un contenido html con python embebido tal y como se muestra a continuación:

```
<html>
<%
import time
%>
```

```
<h3>La fecha <a href="index.psp"> actual </a>  es: <%=time.strftime("%Y-%m-%d,
%H:%M:%S")%></h3>

<%
veces = 1
if form.has_key('nombre'):
    saludo = 'Hola %s, espero que te vaya bien' % form['nombre']
    if form['nombre'] == 'Carol': veces=3
else:
    saludo = 'Hola, seas quien seas.'
# fin del código
%>

Elige uno de estos nombres: <a href="index.psp?nombre=Ariadna"> Ariadna </a>, <a
href="index.psp?nombre=Hugo"> Hugo </a> o <a href="index.psp?nombre=Carol"> Carol
</a>

<h3><%= saludo %></h3>

<%
if veces != 1:
    for n in range(veces):
        #comentario python tabulado
%>
<h2> Has seleccionado a Carol </h2>
<%
# Fin del for
%>

Fin del contenido.
</html>
```

El resultado de la ejecución dará algo similar a lo mostrado en la figura 9.10. La pantalla de la parte superior izquierda de la figura muestra el resultado cuando se accede directamente al **index.psp**. La pantalla de la parte inferior derecha de la figura muestra el resultado tras pulsar sobre el enlace indicado como Hugo. Tras copiar el archivo y visionarlo en tu navegador podrás ver cual es el resultado al pulsar sobre los enlaces actual, Ariadna y Carol, aunque leyendo el código anterior ya podrías predecirlo.

Figura 9.10. Resultado de la invocación de la página index.psp

La descripción del código sería como sigue:

```
<html>
<%
import time
%>
```

La primera línea contiene la etiqueta que indica el comienzo del documento html. La segunda línea indica el comienzo de contenido python embebido y la cuarta línea la finalización del mismo. La tercera línea es el contenido python que simplemente indica la importación de la librería **time**, evidentemente porque posteriormente se hará uso de alguna de las funciones pertenecientes a dicha librería.

```
<h3>La fecha <a href="index.psp"> actual </a>  es: <%=time.strftime("%Y-%m-%d,
%H:%M:%S")%></h3>
```

Esta línea comienza con una etiqueta de cabecera html dentro de la cual se hace un hiper enlace a la página **index.psp** a través de la palabra **actual**. Puesto que el archivo que se está analizando se llama **index.psp**, pulsar sobre la palabra **actual** significará recargar la página. A continuación aparece código python embebido. Es un contenido que será mostrado por pantalla debido a que comienza con <%=, tal y como se explicó al comienzo de esta sección. Lo que se va a mostrar por pantalla es el resultado de la función **strftime()** perteneciente a la librería **time**. Se deduce fácilmente que el resultado de esta función es la fecha con el formato indicado como argumento de la misma.

```
<%
veces = 1
```

El símbolo <% indica de nuevo el comienzo de código python. La siguiente línea define una variable llamada **veces** a la cual se asigna un valor de 1. Es de suponer que esta variable será usada posteriormente.

```
if form.has_key('nombre'):
    saludo = 'Hola %s, espero que te vaya bien' % form['nombre']
    if form['nombre'] == 'Carol': veces=3
else:
    saludo = 'Hola, seas quien seas.'
# fin del código
%>
```

Esta parte del archivo realiza una operación condicionada. **form** es un objeto global definido en mod_python para ser usuado en el contexto de una página .psp. Es un diccionario python que recoge los datos proporcionados a través de la url del navegador. Esto se hace indicando la url y a continuación una cadena de variables que siguen a un signo de interrogación **?**. Por ejemplo la url:

```
http://192.168.10.150/index.psp?nombre=Ariadna&valor=5&mes=enero
```

Accedería a la página http://192.168.10.150/index.psp proporcionando datos a través de las variables **nombre**, **valor** y **mes**, que respectivamente contienen los valores **Ariadna**, **5** y **enero**. Esta información sería almacenada en el diccionario form = {'nombre': 'Ariadna' , 'valor': '5' , 'mes': 'enero'}.

Si se hubiera introducido un dato a través de la url, se comprobaría si el diccionario creado por mod_python con ese/esos dato/s contiene la palabra **nombre** en cuyo caso se guardaría en la variable **saludo** un cadena de texto donde se incrusta el valor asociado a **nombre**.

A continuación aparece un nuevo **if** incluido en el anterior. En el caso de que el contenido de **nombre** sea **Carol** se asigna a la variable **veces** un valor 3, es decir se modifica el valor dado originalmente.

La siguiente línea es un **else**, que pertenece al primer **if** ya que, al igual que él, no tiene tabulación. Por tanto en caso de que el diccionario **form** no contenga la palabra **nombre** a la variable **saludo** se le asigna otra cadena diferente.

La siguiente línea comienza por # lo que representa un comentario en código python. La misión de esta línea no es solamente comentar el código, además sirve para indicar el fin del contenido del anterior **else** ya que esta línea no está tabulada. Si no existiera, el código html que existiera a continuación y fuera del python embebido se interpretaría como parte del **else**.

```
Elige uno de estos nombres: <a href="index.psp?nombre=Ariadna"> Ariadna </a>, <a
href="index.psp?nombre=Hugo"> Hugo </a> o <a href="index.psp?nombre=Carol"> Carol
</a>
```

Esto se corresponde con código html que mostrará por pantalla un texto con tres hiper enlaces: Ariadna, Hugo, y Carol. Estos se encargarán de recargar la página **index.psp** añadiendo la variable nombre a la url con diferentes contenidos. De esta forma al pulsar sobre alguno de ellos el contenido de la página cambiará.

```
<h3><%= saludo %></h3>
```

Esta línea imprime con una cabecera de nivel 3 el contenido de la variable saludo, ya que el delimitador inicial es <%=.

```
<%
if veces != 1:
    for n in range(veces):
        #comentario python tabulado
%>
<h2> Has seleccionado a Carol </h2>
<%
# Fin del for
%>
```

Este python embebido (delimitadores <% y %>) es un ejemplo de creación de bucles por medio de **for**. Para empezar el **for** y su contenido se ejecutarán di la variable **veces** contiene un valor diferente de 1; en otras palabras en el caso de que se haya pulsado el hiper enlace Carol.

La línea del **for** se podría leer: "para valores[74] de n=0, n=1 y n=2 realizar las siguientes instrucciones tabuladas". Dentro del python embebido el único código que hay es un comentario y por tanto no repercutirá en la salida mostrada por el navegador, sin embargo este comentario sirve para indicar una línea de código tabulada para el **for**; el resto de líneas html que se encuentren a continuación se supondrán dentro del **for** hasta que aparezca una línea de código no tabulada.

74 En el caso de que **veces** no sea 3 sino que posee un valor cualquiera se leería: "para valores n=0, n=1, ..., n=veces-1 realizar las siguientes instrucciones tabuladas".

Por tanto la cabecera html de nivel 2 se encuentra dentro del **for** y será repetida 3 veces. A continuación se vuelve a embeber un comentario python cuya única misión es indicar al parser que finalizó el contenido del **for**, ya que este comentario no está tabulado.

```
Fin del contenido.
</html>
```

La primera de estas líneas es entendida como código html, pero que no estará contenida dentro del último **for** anteriormente descrito y está aquí para demostrarlo. Por último se cierra el documento html.

Como última práctica escribe la url: **http://192.168.10.150/index.psp_** o **http://192.168.10.150/index.psp_?nombre=Ariadna** para comprobar como mod_python devuelve por pantalla el código python generado si está la directiva **PythonDebug** a **on** en el archivo **httpd.conf**.

Capítulo 10

GESTOR DE CONTENIDOS JOOMLA

Un Sistema Gestor de Contenidos (CMS, *Content Management System*) es una aplicación Web que facilita el trabajo de creación de portal de Internet sin necesidad de tener conocimientos sobre lenguajes de programación (*PHP, Phyton, etc.*) o gestión de bases de datos (*sentecias SQL, phpMyAdmin, etc.*).

Compuestos por una doble interfaz Web (*front-end* y *back-end*), y apoyados por un gestor de bases de datos (*por ejemplo, MySQL*), permiten personalizar las plantillas Web prediseñadas disponibles, pudiendo modificar y cambiar los distintos elementos que las forman. En concreto, el *front-end* se corresponde con la parte de la aplicación accesible por los usuarios que contiene los contenidos que forman el sitio Web, mientras que *back-end* es la parte asociada a la administración de los contenidos visibles en el *front-end*.

Aunque existen multitud de CMS pensados para diferentes finalidades (Blogs, Wikis, eCommerce, etc.), se ha elegido a Joomla como ejemplo, al ser uno de los más populares, disponer de gran cantidad de plantillas y ofrecerse bajo licencia GPL. A lo largo del capítulo se proponen multitud de ejercicios prácticos que ayudarán a comprender la importancia de un CMS en la nueva Web 2.0.

1. CONTENIDOS DEL CAPÍTULO

El presente capítulo presenta la siguiente estructura:

— Tras indicar el software necesario, se explicará como preparar el servidor para la instalación de Joomla.

— Después pasaremos su instalación bajo Xampp/Apache.

— A continuación se explicará como modificar la plantilla Joomla, como crear cuentas de usuario Joomla y las diferencias entre ellas, las distintas formas de publicar contenidos en Joomla, y las distintas extensiones que pueden instalarse para aumentar su potencia.

— El capítulo contiene multitud de ejercicios prácticos que se recomiendan hacer en el mismo orden en que aparecen.

2. SOFTWARE DE DESARROLLO

Para la realización de los ejercicios prácticos que se proponen en el presente capítulo deberemos descargarnos el paquete Joomla de la red: "http://*www.**joomlaspanish**.org*". De las distintas versiones disponibles nos bajaremos la última que sea estable.

¡¡Importante!! En el caso de trabajar con Apache de manera independiente (*sin Xampp*) será necesario instalar un conjunto de módulos:

— **apache-mod_php**: módulo de Apache que le permite interpretar y ejecutar código PHP, lenguaje utilizado en el desarrollo de *Joomla*.

— **php-mysql**: módulo PHP que contiene las funciones necesarias para poder interactuar con bases de datos MySQL.

— **php-xml**: módulo PHP que contiene las funciones necesarias para poder crear y manipular ficheros XML. Muchos de los datos manejados por *Joomla* están en este formato.

— **php-gd**: módulo PHP que contine funciones útiles para el manejo de imagenes. Por ejmplo, Joomla hace uso de esta librería de funciones para la creación y gestión de galerías de imágenes Web.

— **php-zlib**: módulo PHP que contiene funciones útiles para el manejo de archivos comprimidos. Es necesario, ya que bajo Joomla se pueden instalar paquetes software (*componentes, módulos y plugins o también llamados mambots*) que por convenio se encuentran comprimidos.

3. PRECONFIGURACIÓN DEL EQUIPO SERVIDOR: APACHE Y MySQL.

Para poder instalar el gestor de contenidos *Joomla* en el servidor es necesario configurar previamente Apache, MySQL y opcionalmente, como veremos más adelante, el servicio FTP (ProFTPd en GNU/Linux y Filezilla en Microsoft Windows), para darle soporte.

— **Apache**: Joomla es una aplicación Web diseñada en lenguaje PHP que requiere de un servidor Web que haga de intermediario entre el usuario final (*cliente Web*) y él. Si hacemos uso de Apache y está dando servicio a más de un sitio Web, será necesario configurar un nuevo VirtualHost bajo la IP que tenga asignada el equipo servidor dentro de la Intranet[75] (por ejemplo, *192.168.1.1*), asignándole el nombre de dominio (p.e. *www.escarbapedal.com*) y directorio raíz (p.e. */opt/lampp/htdocs/joomla en GNU/Linux y c:\xampp\htdocs\joomla en Microsoft Windows*) deseados.

```
# Contenido del fichero de configuración de Apache: httpd.conf
Listen 80 # Puerto de escucha de Apache por defecto, a través de cualquier dirección IP
DocumentRoot /opt/lampp/htdocs # Directorio Raíz de Xampp en GNU/Linux
DocumentRoot c:/xampp/htdocs # Directorio Raíz de Xampp en Microsoft Windows
ServerName localhost # Nombre del equipo servidor Xampp
NameVirtualHost 127.0.0.1
NameVirtualHost 192.168.1.1
Include /opt/lampp/etc/extra/joomla.conf # Fichero auxiliar de configuración en GNU/Linux
Include c:/xampp/apache/conf/extra/joomla.conf # Fichero auxiliar de configuración en Windows
... # Resto de las directivas de configuración del fichero httpd.conf
```

75 Para que el sitio Web o portal *Joomla* sea accesible desde otros equipos de la red además del propio equipo local (*localhost*), haremos uso de la dirección IP que tenga asignada, aunque será necesario configurar un servidor DNS que resuelva el nombre de dominio del sitio Web, *www.escarbapedal.com -> 192.168.1.1*, o definir localmente esta resolución en el fichero hosts de todos los equipos clientes.

```
# Contenido del fichero auxiliar de configuración de Apache:  joomla.conf
<VirtualHost 192.168.1.1>
  ServerName www.escarbapedal.com # Nombre de dominio del sitio Web joomla
  DocumentRoot /opt/lampp/htdocs/joomla # Directorio Raíz del sitio Web en GNU/Linux
  DocumentRoot c:/xampp/htdocs/joomla # Directorio Raíz del sitio Web en Windows
</VirtualHost>
```

¡¡Importante!! Recuerda reiniciar el servicio para que los cambios en la configuración surtan efecto.

```
[root@linux]# /opt/lampp/lampp restart (reinicia todos los servicios Xampp en GNU/Linux)
C:\xampp> xampp_restart.exe (reinicia todos los servicios Xampp en Microsoft Windows)
```

```
# Contenido del fichero hosts: Direccion IP -> Nombre Equipo
127.0.0.1 localhost # Esta asociación se encuentra siempre por defecto en "hosts"
192.168.1.1       www.joomla.com
```

— **MySQL**: *Joomla* utiliza una base de datos para organizar los datos e información en diferentes tablas sin la cual no puede funcionar. Si hacemos uso del gestor MySQL tan sólo será necesario crear una base de datos (por ejemplo, *webescarbapedal*) y un una cuenta de usuario con todos los privilegios sobre ella (por ejemplo, *user=arturo y password=mipass*), siendo Joomla el encargado de crear toda la estructura de tablas necesarias durante su instalación. Para ello, desde la consola MySQL como usuario root:

```
[root@linux]# /opt/lampp/bin/mysql -u root -p
c:\xampp\mysql\bin> mysql.exe -u root -p
mysql> CREATE DATABASE webescarbapedal;
mysql> GRANT ALL PRIVILEGES ON webescarbapedal.* TO arturo@localhost IDENTIFIED BY
'mipass';
mysql> flush privileges;
mysql> quit
```

4. INSTALACIÓN DE JOOMLA

Para instalar Joomla en el equipo servidor descomprimiremos el documento (*zip, tar.gz, etc.*) descargado del sitio Web "*http://www.**joomlaspanish**.org*" correspondiente a la última versión disponible en la raíz del sitio Web configurado en Apache (por ejemplo, *htdocs/joomla*).

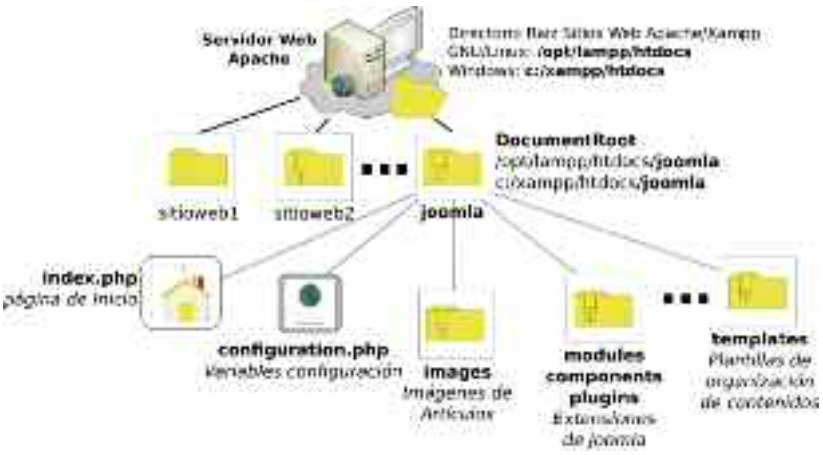


Figura 10.1. Estructura del sistema de ficheros Joomla

Tras la descompresión, si listamos el contenido del directorio raíz de Joomla podremos advertir la estructura de Joomla (ver figura 10.1), destacando los siguientes archivos y directorios:

— **installation**: directorio que contiene los scripts PHP necesarios para la instalación de Joomla. Una vez instalado será eliminada.

— **index.php**: página de inicio del sitio Web.

— **configuration.php**: contiene las variables de configuración global de Joomla. Almacenan los valores indicados durante la instalación de Joomla: $dbtype, el tipo de bases de datos utilizada (*mysql*), $host, el nombre del equipo servidor (*localhost*), $db, el nombre de la base de datos (*webescarbapedal*), $user, el usuario de la bases de datos (p.e. *arturo*), $password, su contraseña, etc.

— **images**: directorio por defecto encargado de almacenar las imágenes que serán utilizadas por los usuarios Joomla en la edición de sus artículos.

— **modules**, **components** y **plugins**: directorios encargados de organizar las distintas extensiones que se vayan instalando en Joomla.

— **templates**: almacena todas las plantillas disponibles instaladas durante la instalación de Joomla y posteriormente. La elección de la plantilla que sea más acorde con el tipo de contenidos que se van a publicar es fundamental para que nuestra Web presente un aspecto atractivo.

Para empezar con la instalación, pondremos en la barra de direcciones de nuestro navegador Web *http://* seguido del nombre de dominio asignado a la directiva *ServerName* en la configuración de Apache (p.e. *http://www.escarbapedal.com*). Si todo funciona como debiera, en el navegador nos aparecerá la pantalla de bienvenida de instalación de *Joomla*.

¡¡Advertencia!! Si el sistema operativo bajo el que va a instalarse *Joomla* es GNU/Linux, hay que recordar que Xampp/Apache no forma parte de lista de servicios dados de alta en el sistema (*/etc/init.d*), funcionando bajo la cuenta de usuario *nobody* con todas sus restricciones y escasos permisos asociados. Por ello, para que funcione correctamente, tras descomprimir Joomla, será necesario configurar el propietario de todos los documentos que forman Joomla (*chown*):

```
[root@linux]# mkdir /opt/lampp/htdocs/joomla
[root@linux]# mv Joomla-version-spanish.tar.gz /opt/lampp/htdocs/joomla
[root@linux]# cd /opt/lampp/htdocs/joomla
[root@linux /opt/lampp/htdocs/joomla]# tar xzf Joomla-version-spanish.tar.gz
[root@linux /opt/lampp/htdocs/joomla]# rm -f Joomla-version-spanish.tar.gz
[root@linux /opt/lampp/htdocs/joomla]# chown –R nobody /opt/lampp/htdocs/joomla
```

Paso n.º 1. Selección del idioma de instalación de *Joomla*

Figura 10.2. Pantalla de bienvenida del asistente de instalación de Joomla

A partir de este momento el asistente nos guiará siguiendo los pasos necesarios para una correcta instalación, avisándonos en el caso de que no se cumplan determinados requisitos imprescindibles para el correcto funcionamiento de *Joomla*. Tras seleccionar el idioma, el asistente chequeará el sistema y realizará una comprobación previa: versión de PHP (*>=v4.3*), dependencias necesarias (*módulos php-mysql, php-xml, php-zlib*), permisos de escritura sobre el directorio raíz del sitio Web y archivo de configuración (*configuration.php*), y valores asignados a las directivas de configuración de PHP en el archivo *php.ini*.

Paso n.º 2. Comprobación previa

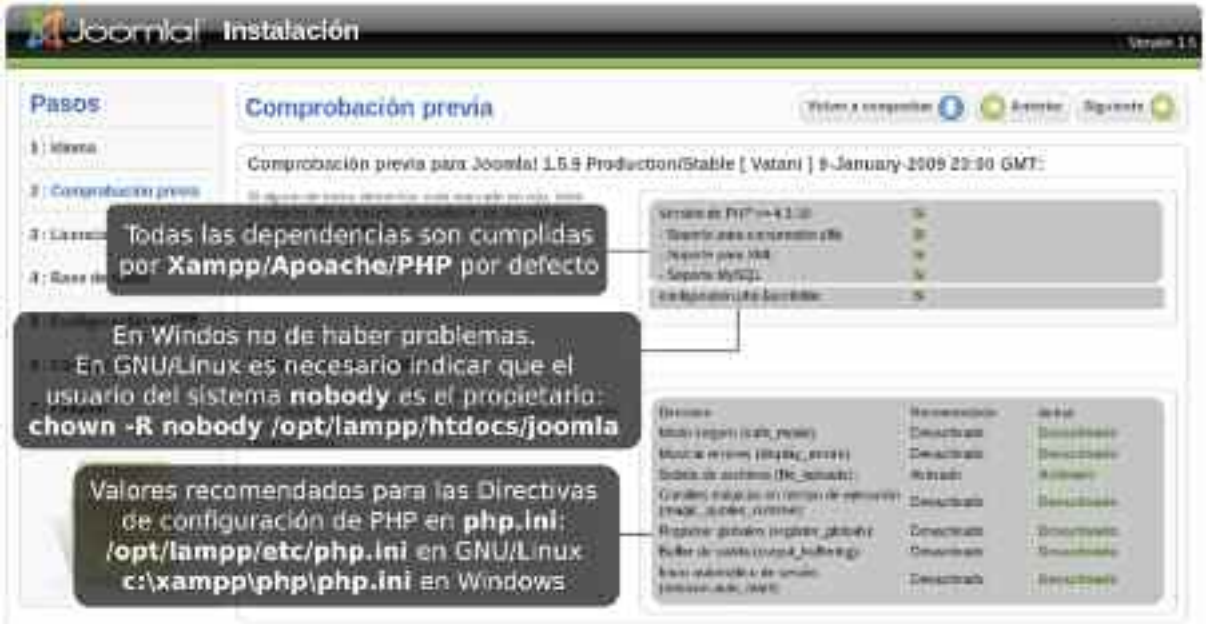


Figura 10.3. Comprobación de requisitos antes de la instalación

En Microsoft Windows no tienen porque aparecer problemas, al cumplir Xampp/Apache/PHP todas las dependencias requeridas, y no haber problemas de permisos al ser la cuenta de usuario del sistema LocalSystem la encargada de ofrecer el servicio. En cambio en GNU/Linux, al estar funcionando Xampp/Apache bajo la cuenta de usuario del sistema nobody {nota pie pagina: Recordar que en GNU/Linux la directiva *User* del fichero de configuración de Apache, httpd.conf, tiene asignado por defecto el valor nobody. Esta es una cuenta del sistema con permisos restringidos. Si quisiéramos que funcionase bajo otra cuenta de usuario del sistema deberíamos modificar la directiva anterior.}, ha sido necesario cambiar el propietario del directorio raíz del sitio Web (/opt/lampp/htdocs/joomla) para que no existan conflictos al tratar de escribir sobre sus contenidos.

```
[root@linux /opt/lampp/htdocs/joomla]# chown –R nobody /opt/lampp/htdocs/joomla
```

En cuanto a las recomendaciones, si el valor asignado a alguna de las directivas no es el recomendado, podemos editar el fichero */opt/lampp/etc/php.ini* en GNU/Linux o *c:\xampp\php\php.ini* en Windows, buscarla y modificar su valor. Las directivas que suelen no corresponderse con el valor aconsejado son *display_errors* y *register_globals*:

```
; Contenido de php.ini: Valores aconsejados para display_errors y register_globals
display_errors = Off
register_globals = Off
```

Tras modificar el fichero php.ini será necesario reiniciar el servicio Xampp/Apache/PHP para que surtan efecto los cambios:

```
[root@linux]# /opt/lampp/lampp restart (reinicia todos los servicios Xampp, GNU/Linux)
C:\xampp> xampp_restart.exe (reinicia todos los servicios iniciados Xampp, Windows)
```

Pasos n.º 3 y 4. Licencia y configuración de la base de datos para *Joomla*

Tras aceptar la licencia GNU/GPL de *Joomla*, configuraremos la conexión con la base de datos que se ha creado expresamente para este portal (*ver apartado 3*):

Figura 10.4. Configuración de la base de datos MySQL en Joomla

Paso n.º 5. Configuración del servicio FTP para Joomla

A continuación, nos preguntará si queremos configurar el servicio FTP para poder subir contenidos por esta vía a Joomla. Su habilitación es opcional. En caso de habilitarlo, será necesario configurar una cuenta de usuario FTP, cuya raíz de documentos coincida con la del sitio Web (*htdocs/joomla*), y permisos suficientes para poder escribir. Para recordar como se configura Filezilla en Microsoft Windows y ProFTP en GNU/Linux, se recomienda repasar el capítulo 4 asociado al servidor Web Apache, en su apartado 4.2.1, *publicación de contenidos vía FTP*.

Figura 10.5. Configuración de FTP en Joomla

Los pasos para configurar la cuenta FTP en FileZilla bajo Microsoft Windows se resumen en:

1) Desde el Xampp Control Panel accederemos a la interfaz de configuración del servidor FTP FileZilla.
2) Iremos a la ventana encargada de la gestión de usuarios FTP de FileZilla.
3) Crearemos una cuenta de usuario para la gestión FTP de Joomla (*por ejemplo, joomla*).
4) Le asignaremos la usuario anterior, *joomla*, el directorio raíz del sitio Web como Home.

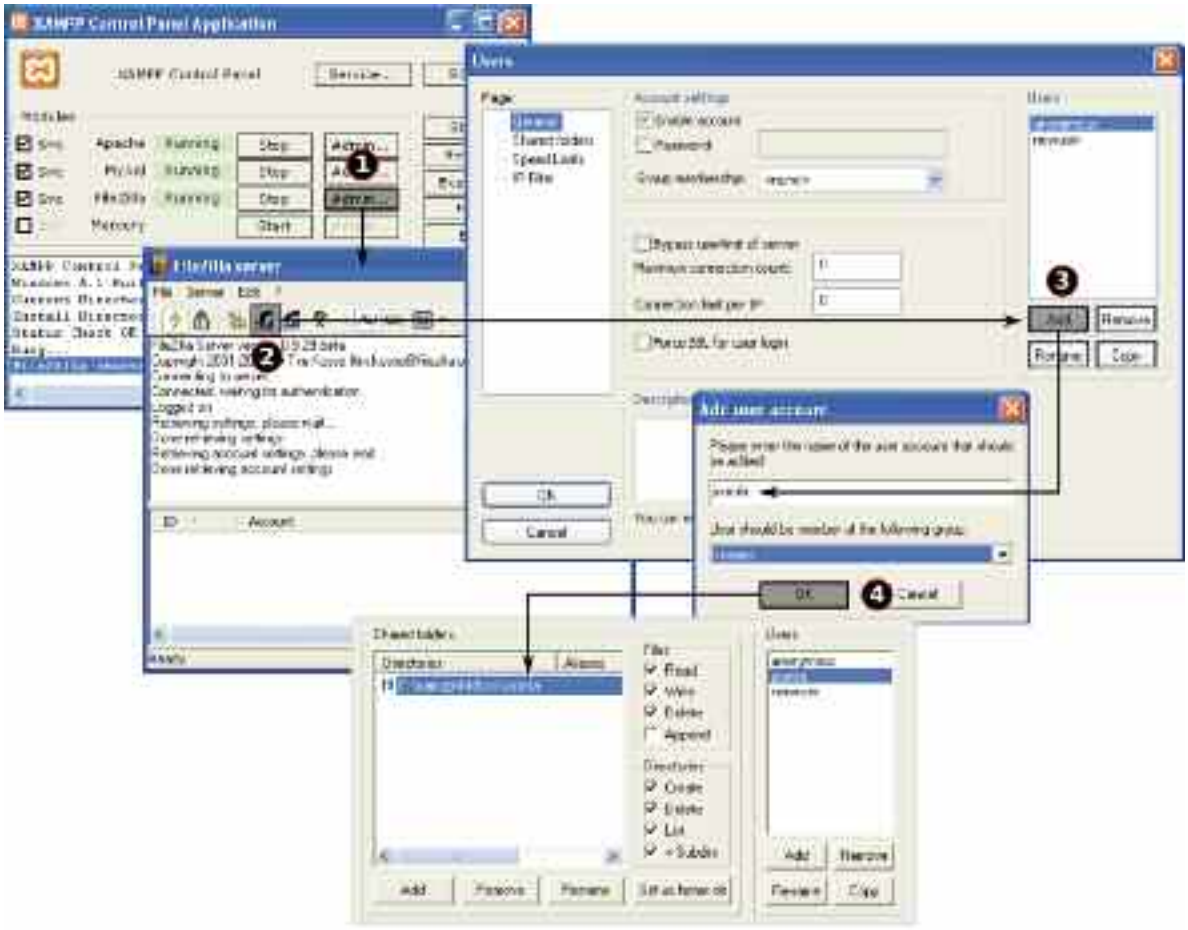


Figura 10.6. Configuración de la cuenta de usuario FTP para Joomla en FileZilla

En el caso de que nos encontremos bajo GNU/Linux, la cuenta de usuario FTP se creará como una cuenta más del sistema mediante *useradd*. Le asignaremos como directorio Home la raíz del sitio Web, *-d /opt/lampp/htdocs/joomla*, lo agregaremos al grupo de usuarios nobody para evitar conflictos con Xampp/Apache {nota pie pagina: Recordar que Xampp/Apache en GNU/Linux funciona bajo la cuenta de usuario del sistema *nodoby*, al tener asignado su directiva de configuración *User* en *httpd.conf* este usuario.}, *-g nobody*, y le asignaremos una shell falsa para evitar que pueda accederse mediante esta cuenta de usuario por otras vías como SSH, *-s /bin/false*. Después le asignaremos una contraseña mediante passwd, y le concederemos permisos de escritura sobre todo el sitio (*-R, recursivo*) mediante chown (*CHange OWNer, cambio de propietario*) y chmod (*CHange MODe, cambio de modo, de permisos*).

```
[root@linux]# useradd -d /opt/lampp/htdocs/joomla -g nobody -s /bin/false joomla
[root@linux]# passwd joomla
[root@linux]# chown -R nobody.nobody /opt/lampp/htdocs/joomla
[root@linux]# chmod -R g+w /opt/lampp/htdocs/joomla
```

Paso n.º 6. Configuración principal

Nos solicitará un nombre al sitio Web, que será utilizado por las plantillas de *Joomla* para personalizar sus títulos, una cuenta de correo electrónico para el administrador, y la contraseña del usuario *Joomla admin* con privilegios para gestionar el portal. Es recomendable instalar los datos de ejemplo que tiene *Joomla*, si es la primera vez que trabajamos con él, ya que tras la instalación nos permitirán hacernos una idea de como quedará nuestro sitio Web cuando hayamos introducido los nuestros.

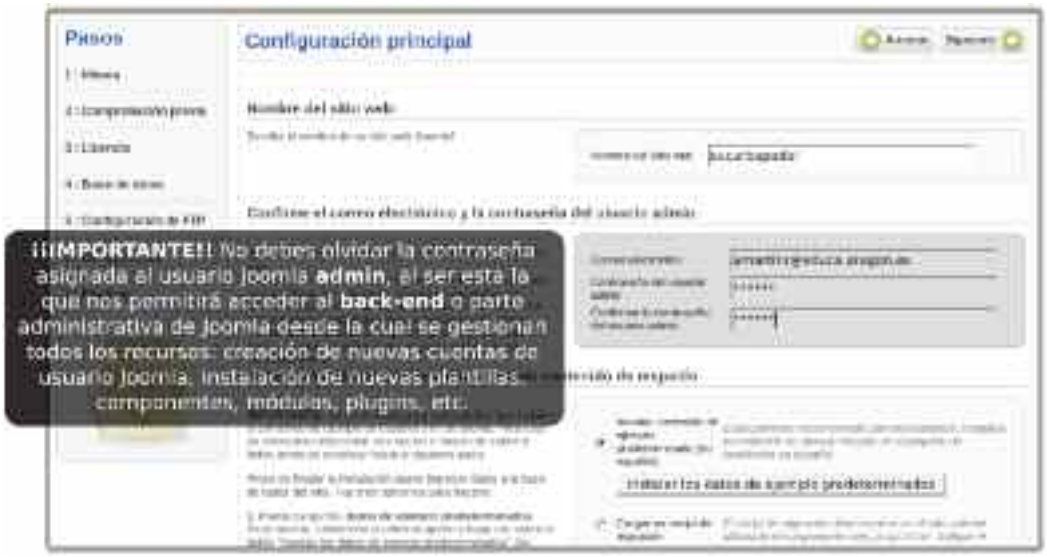


Figura 10.7. Datos de configuración general

Paso n.º 7. Final de la instalación

Para completar la instalación se nos pide que eliminemos el subdirectorio "**installation**" existente en el directorio raíz de *Joomla htdocs/joomla/installation*. Para que el borrado sea recursivo (el directorio con todo su contenido), y no nos pregunte el sistema si estamos seguros de eliminarlo, haremos uso de las opciones **-Rf** en GNU/Linux, y **/S /Q** en Microsoft Windows.

```
[root@linux /opt/lampp/htdocs/joomla]# rm -Rf installation
c:\xampp\htdocs\joomla> rmdir /S /Q installation
```

Figura 10.8. Pantalla final del asistente de instalación de Joomla

Tras la eliminación del directorio podremos acceder al portal Joomla correspondiente a nuestro sitio Web (**front-end**, *http://www.escarbapedal.com*), o a su parte administrativa (**back-end**, *http://www.escarbapedal.com/administrator*) desde los dos enlaces de la parte superior derecha (*ver figura 10.9*).

Figura 10.9. Aspecto del front-end de Joomla tras la instalación (plantilla JA_purity)

En los siguientes apartados se explicará la forma en que podemos personalizar los contenidos, módulos, header, etc. que se observan en el *front-end* de *Joomla* a través del *back-end*.

5. PLANTILLAS JOOMLA

El aspecto que presenta el sitio Web *Joomla* depende en gran medida de la plantilla (*template*) utilizada. En Internet podemos encontrar tantas plantillas como formas distintas se nos puedan ocurrer que hay a la hora de organizar los contenidos que forman parte de las páginas Web que componen el portal Joomla. Con la instalación de *Joomla* solo se incluyen unas pocas.

Para cambiar de plantilla, accederemos al **back-end**, *http://www.escarbapedal.com/administrator*, pincharemos sobre el menú *Extensiones*, y seleccionaremos la opción *Gestor de plantillas*.

Figura 10.10. Gestión de las plantillas en Joomla

Tal como se muestra en la figura 10.10, desde este gestor podemos seleccionar cualquiera de las plantillas disponibles y marcarla como predeterminada. Para ver el aspecto que presenta nuestro sitio Web con la nueva plantilla, deberemos acceder al **front-end**, *http://www.escarbapedal.com*. En el caso de que no nos guste ninguna de ellas, es posible buscar y descargar plantillas gratuitas (*free templates*) desde Internet poniendo en la barra de búsqueda de Google "*Download template joomla*", e instalarlas posteriormente en *Joomla* desde el menú *Extensiones*, opción *Instalar/Desinstalar*. Tras su instalación volveremos a la gestión de plantillas *Joomla* y seleccionaremos la nueva plantilla.

Ejercicio práctico n.º 1

Descarga alguna plantilla gratuita de la red, instala y comprueba el cambio de aspecto que presenta el sitio Web de Joomla.

Solución ejercicio n.º 1

1. Tras haber descargado una plantilla de muestra, Ingresa en el **back-end** (*http://www.escarbapedal.com/**administrator***) con la cuenta de usuario **admin** e instalala desde el menú *Extensiones*, opción *Instalar/Desinstalar*.
2. Desde el formulario *Subir paquete* del *Gestor de funciones*, buscar la plantilla descargada, súbela al servidor e instálala.

Figura 10.11. Pasos para la instalación de plantillas (y otras extensiones) en Joomla

¡¡Advertencia!! Puede sucedernos que al instalar una extensión (*componente, módulo, plugin, idioma, o plantilla*) nos aparezca un error advirtiéndonos de que la versión de Joomla para la cual fue diseñada no coincide con la versión disponible.

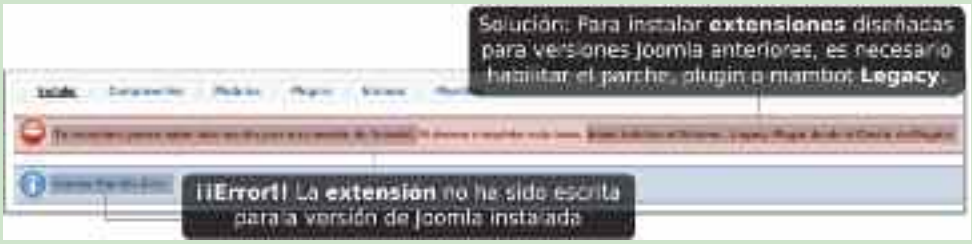


Figura 10.12. Problema al instalar extensiones diseñadas para versiones anteriores de Joomla

Para solucionar este problema, y poder instalar estas extensiones, es necesario parchear Joomla habilitando el plugin, también llamado mambot, Legacy. Este ya viene instalado con la instalación por defecto, pero se encuentra deshabilitado (*no publicado*). Para ello, desde el menú *Extensiones*, accederemos al *Gestor de plugins* para publicar Legacy. Después volveremos a tratar de instalar la extensión.

Figura 10.13. Habilitación/Publicación del plugin Legacy

3. Accede al Gestor de plantillas del menú Extensiones y selecciona como plantilla predeterminada la recién instalada.

4. Comprueba el cambio de aspecto del sitio Web Joomla accediendo a su parte pública, el **front-end** (por ejemplo, *http://www.escarbapedal.com*). Si no te gusta, prueba con otras.

Figura 10.14. Cambio de aspecto del sitio Web tras cambiar de plantilla

6. CUENTAS DE USUARIO JOOMLA

Con la finalidad de delegar privilegios de administración y publicación de contenidos a otros usuarios diferentes del administrador *admin*, Joomla nos permite crear otras cuentas de usuario con diferentes perfiles.

Figura 10.15. Gestor de usuarios Joomla

Estos perfiles de usuario se agrupan en dos grandes grupos: aquellos que tienen privilegios para acceder y gestionar Joomla desde el **back-end** (*parte administrativa de Joomla*), y los que solo pueden llevar a cabo operaciones de gestión desde el **front-end** (*parte de publicación de contenidos*).

— **Super Administrador**: Perfil de Joomla con todos los privilegios de administración. Desde el **back-end** puede instalar/desinstalar extensiones, modificar la plantilla del sitio Web, crear cuentas de usuario Joomla, etc.

— **Administrador**: Perfil de Joomla similar a la del super administrador con algunas restricciones, entre las que cabría señalar que no puede modificar la plantilla del sitio Web ni modificar sus parámetros de configuración global. Por contra, desde el **back-end** puede instalar/desinstalar extensiones, gestionar las cuentas de usuario, los componentes, los módulos, los plugins, los contenidos, etc.

— **Gestor**: Perfil de Joomla con acceso al back-end, pero con muchas limitaciones. Destacar que no puede gestionar los usuarios Joomla, ni instalar/desinstalar extensiones, ni modificar los parámetros de configuración global, entre otros aspectos. Esta pensada para que pueda gestionar los contenidos del sitio Web.

— **Publicador/Editor**: Perfiles de Joomla sin acceso al back-end, pero con privilegios para gestionar los contenidos publicados en el front-end y enviar nuevos artículos al servidor.

— **Autor**: Perfil de Jommla sin acceso al back-end, ni privilegios para poder gestionar los contenidos publicados en el front-end. Tan solo se esta permitido enviar artículos al servidor desde el front-end.

— **Registrado**:Perfil de Joomla con mayores restricciones. No tiene acceso al back-end. Desde el front-end no puede ni modificar los contenidos publicados, ni enviar nuevos artículos al servidor. Tan sólo puede iniciar sesión desde el front-end con privilegios para poder consultar contenidos que están restringidos a usuarios registrados.

Ejercicio práctico n.º 2

Ingresa en el **back-end** (*http://www.escarbapedal.com/**administrator***) con la cuenta de usuario **admin** y crea una nueva cuenta de usuario *Joomla* (por ejemplo, *usuario1*) con perfil de **publicador** o **editor**, y comprueba que no puede acceder al **back-end**, pero si puede acceder a la gestión de contenidos publicados en el **front-end** a través del formulario de acceso disponible en la página de inicio del sitio Web. Modifica los contenidos de alguno de sus artículos.

Solución ejercicio n.º 2

1. Desde el *gestor de usuarios* del **back-end** *Joomla*, crea la cuenta *publicador* (*por ejemplo, usuario1*).

Figura 10.16. Creación de una cuenta de usuario publicador

2. Cierra la sesión del usuario *admin*, y comprueba que la cuenta *usuario1* no puede acceder al **back-end**. Después, desde el formulario de acceso del **front-end** accede a la gestión de sus contenidos. Comprueba además que aparece un nuevo menú de usuario que permite crear nuevos artículos y enviarlos al servidor, para que el administrador de *Joomla* los publique desde el **back-end**.
3. Advierte que tras iniciar sesión como usuario publicador, en todos los artículos publicados en el las páginas Web que componen el **front-end** aparece un icono sobre su esquina superior izquierda sugiriendo su edición.

Figura 10.17. Edición de artículos desde el Front-End

4. Modifica el contenido de alguno de los artículos, guarda los cambios y comprueba que los cambios son visibles en la Web.

Ejercicio práctico n.º 3

Ingresa en el **back-end** con la cuenta de usuario **admin** y crea una nueva cuenta de usuario *Joomla* con perfil de **autor** (por ejemplo, *usuario2*). Comprueba que a diferencia de los usuarios publicadores o editores (*ejercicio práctico n.º 2*), estos no pueden modificar los contenidos publicados en el **front-end**, pero si pueden crear nuevos artículos (*menú de usuario*).

Ejercicio práctico n.º 4

Ingresa en el **back-end** con la cuenta de usuario **admin** y crea una nueva cuenta de usuario *Joomla* con perfil de **gestor** (por ejemplo, *usuario3*). Comprueba que al acceder nuevamente al **back-end** con el *usuario3* el panel de control ha cambiado, y advierte que no puede modificar la plantilla del sitio Web, ni gestionar los usuarios *Joomla*, ni instalar/desinstalar extensiones entre otras cosas.

Solución ejercicio n.º 4

1. Desde el *gestor de usuarios* del **back-end** *Joomla*, crea la cuenta *gestor* (*por ejemplo, usuario3*).

Figura 10.18. Perfil del usuario Joomla gestor

2. Cierra la sesión del usuario *admin*, e inicia sesión en el back-end con la nueva cuenta de usuario. Comprueba sus restricciones.

7. PUBLICACIÓN DE CONTENIDOS JOOMLA

Los contenidos que pueden publicarse en *Joomla* son básicamente de tres tipos:

1. Artículos o noticias redactadas por los usuarios *Joomla*. Para su edición, *Joomla* dispone de un editor Web similar a las herramientas ofimáticas que habitualmente utilizamos.

Figura 10.19. Aspecto del editor Web disponible en Joomla

Joomla organiza los artículos en categorías, y a su vez estas en secciones. Por este motivo, es conveniente antes de empezar a crearlos y publicar contenidos hacer un estudio del número de secciones y categorías dentro de estas que sería conveniente crear para que toda la información se encuentre perfectamente organizada.

La forma en que un usuario *Joomla* tiene para crear artículos depende del perfil asignado. En el caso de tener acceso al **back-end** (*superadministrador, administrador, gestor*), en el panel de control encontraremos dos iconos relacionados con ello: 1) *Añadir un nuevo artículo*, el cual nos abriría directamente el editor para su edición, y 2) *Gestor de artículos*, desde donde podemos crear, eliminar, publicar/despublicar, mover o copiar artículos.

Figura 10.20. Creación de artículos desde el back-end

Si por el contrario, el usuario tiene un perfil con acceso restringido al **front-end**, de tipo publicador, editor, o autor, puede crear artículos y enviarlos al servidor, tras iniciar sesión en el formulario de acceso del **front-end**.

2. Componentes. Nos permiten introducir multitud de tipos de contenido que serían imposibles de conseguir editando artículos. Hay componentes que agregan a nuestro sitio Web complejas galerías de imágenes (*Morfeo Show*), un explorador de archivos (*eXtplorer*), un calendario para eventos (*GCalendar*), o un gestor de almacenamiento de archivos (*Remository*), sin necesidad de programar nada. Su instalación se realiza exactamente igual que se hacia con las plantillas, y son gestionados desde un menú exclusivo que hay en el **back-end**, *Componentes*.
3. URL embebida o Wrapper. Nos permite reutilizar las páginas Web que hayamos diseñado bajo otra herramienta o plataforma de desarrollo Web (*Quanta Plus, Adive Dreamweaver, etc.*), embebiéndolas en el portal *Joomla* como si se se hubieran generado por el propio *Joomla*.

A continuación, mediante la realización de ejercicios prácticos trataremos de comprender las tres estrategias anteriores.

Ejercicio práctico n.º 5

Crea una nueva sección en Joomla (por ejmplo, *Rutas BTT*), y tres categorías dentro ella (por ejemplo, *Rutas Bajo Aragón, Rutas Nacionales y Rutas Extranjero*) para organizar los artículos que vayamos a crear.

Figura 10.21. Organización de artículos en Joomla

Solución ejercicio n.º 5

Ingresa en el **back-end** con la cuenta de usuario **gestor** (por ejemplo, *usuario3*) creada en el ejercicio práctico n.º 4, y pincha sobre el icono del panel de control *Gestor de secciones*. Desde allí crea la nueva sección.

Figura 10.22. Pasos para la creación de una nueva sección en Joomla

Vuelve al panel de control, pincha sobre el icono Gestor de categorías y crea tres dentro de la sección anterior.

Figura 10.23. Pasos para la creación de una nueva categoría en Joomla

Ejercicio práctico n.º 6

Ingresa en el **front-end** con la cuenta de usuario de perfil **autor** (por ejemplo, *usuario2*) creada en el ejercicio práctico n.º 3, crea un artículo dentro de alguna de las categorías anteriores (por ejemplo, *Rutas Nacionales*), dentro de la sección creada (por ejemplo, *Rutas BTT*) y envíalo al servidor. Para que este artículo aparezca en la página de inicio, selecciona la opción de edición correspondiente. Después comprueba que hasta que el gestor o administrador no acceda al *gestor de artículos* del panel de control del **back-end** de *Joomla* y no lo publique, no será visible en el sitio Web de Joomla.

Solución ejercicio n.º 6

1. Ingresa en el **front-end** (por ejemplo, *http://www.escarbapedal.com*) con la cuenta de usuario de perfil **autor** (por ejemplo, *usuario2*).
2. Pincha sobre la opción del menú de usuario Enviar Artículo.
3. Edita un artículo de ejemplo. Pon un título e introduce algún contenido. Cumplimenta el formulario de propiedades del artículo, indicando la sección y categoría a la que queremos que pertenezca, además de señalar la opción para que sea mostrado en la página de inicio.

Figura 10.24. Pasos para

4. Envía el artículo al servidor pinchando sobre el botón guardar. Comprueba que el artículo no aparece en la página de inicio. Un mensaje nos indica que para publicarse debe ser revisado por un administrador o gestor de Joomla.

Figura 10.25. Mensaje informativo de que el artículo ha sido enviado al servidor para ser revisado

¡¡Importante!! Tras terminar el ejercicio práctico comprueba que si el usuario que se hubiese registrado a través del formulario de acceso del **front-end** hubiera sido uno con perfil de administración, p.e. **admin**, *super-administrador*, podría publicar nuevos artículos y modificar los que muestra el **front-end** sin necesidad de una revisión posterior.

5. Ingresa en el **back-end** (por ejemplo, *http://www.escarbapedal.com/administrator*) con la cuenta de usuario **admin** (*superadministrador*), o de perfil administrador o gestor (por ejemplo, *usuario3*), accede desde el panel de control al gestor de artículos y comprueba que en la lista de artículos aparece el artículo enviado por *usuario2*. Publícalo y comprueba que el artículo ya aparece en la página de inicio del sitio Web (*front-end*).

Figura 10.26. Publicación de artículos enviados por un autor desde el back-end

Ejercicio práctico n.º 7

Accede al **back-end** con una cuenta de perfil **gestor** (por ejemplo, *usuario3*) y crea un artículo dentro de la misma sección y categoría que en ejercicio anterior, e indica que se muestre en la página de inicio. A diferencia del anterior, busca el parámetro del artículo relacionado con el *nivel de acceso*, y haz que sólo puedan consultar sus contenidos los usuarios registrados. Publícalo, y comprueba que el artículo solo se muestra en el **front-end** si se accede a través del formulario de acceso con una cuenta de usuario registrada en *Joomla*. Crea una nueva cuenta de usuario de perfil **registrado** (por ejemplo, *usuario4*) y comprueba que al registrarse desde el **front-end**, el artículo es accesible.

Solución ejercicio n.º 7

1. Ingresa en el **back-end** con la cuenta de usuario *usuario3* y crea un nuevo artículo desde el *gestor de artículos*, o pinchando sobre el icono *Añadir un nuevo artículo* del panel de control. Crea el artículo de igual forma que en el ejercicio práctico anterior, pero restringiendo el acceso a su contenido a usuarios registrados.

Figura 10.26. Creación de artículos desde el back-end

2. Comprueba que a pesar de que esta publicado el artículo no se muestra en el **front-end**.
3. Ingresa en el **back-end** con la cuenta de usuario **admin** y crea un nuevo usuario con perfil registrado (p.e. *usuario4*).
4. Con la cuenta de usuario anterior, registrate a través del formulario de acceso del **front-end** y comprueba que entonces se muestra el artículo.

8. INSTALACIÓN DE EXTENSIONES EN JOOMLA

Como puede advertirse a través de los ejercicios prácticos, *Joomla* es una herramienta Web muy versátil y fácil de utilizar que nos permite en muy poco tiempo crear un sitio Web atractivo. No obstante, si continuáramos configurando *Joomla* pronto echaríamos en falta multitud de funcionalidades que no están disponibles. Para suplir estas carencias la comunidad de usuarios Joomla ha desarrollado gran cantidad de paquetes software que pueden instalarse en *Joomla* aportándole distintas funcionalidades. A estos paquetes se les denomina generalmente extensiones.

Se entiende por extensiones todo tipo de añadidos que se realizan sobre la instalación base de Joomla cuya finalidad es aumentar su potencia tratando de suplir sus carencias. Además de la instalación de plantillas (*ejercicio práctico n.º 1*), en *Joomla* también pueden instalarse otro tipo de extensiones: componentes, módulos y plugins (*o mambots*). En "*http://extensions .joomla.org*" podemos encontrar una gran cantidad de ellos disponibles para ser descargados e instalados.

8.1. Instalación de plugins en Joomla

Los *plugins* son pequeños paquetes software que aportan funcionalidades muy simples a Joomla. También llamados *mambots*, pueden considerarse como parches del sistema que tratan de cubrir alguna de las muchas carencias que tiene la instalación básica de *Joomla*. Aunque hay una gran variedad, la mayor parte de ellos simplemente tratan de facilitar la labor de edición de artículos y mejorar su presentación. Para comprender mejor su utilidad se aconseja realizar el siguiente ejercicio práctico.

Ejercicio práctico n.º 8

Instala el plugin Xtypo para mejorar la presentación de los contenidos en los artículos. Este nos permite resaltar de una manera sencilla ciertos contenidos del documento mediante distintos tipos de diálogo Xtypo. Para probarlo, crea un nuevo artículo que contenga un diálogo Xtypo y un video de youtube.

Figura 10.28. Xtypo permite introducir diálogos atractivos en los artículos

Solución ejercicio n.º 8

1. Entra en "*http://extensions.joomla.org*", busca y descarga el plugin Xtypo. En el caso de que este disponible para diferentes versiones de *Joomla*, descarga aquel que más se ajuste más a versión instalada.

¡¡Advertencia!! Recuerda que es posible instalar extensiones que hayan sido diseñadas para versiones inferiores a la versión de tu Joomla si tienes habilitado el plugin Legacy (*revisar ejercicio práctico n.º 1*).

2. Ingresa en el **back-end** con la cuenta de usuario **admin** e instala Xtypo desde el *Gestor de extensiones*, menú *Extensiones*, opción *Instalar/Desinstalar*.

Figura 10.29. Pasos necesarios en la instalación de un plugin

3. A través del formulario Subir paqute del Gestor de extensiones busca el paquete descargado, sube el archivo al servidor e instálalo.
4. Por defecto, todo plugin que se instale en Joomla por defecto esta deshabilitado/despublicado. Para habilitarlo/publicarlo es necesario acceder al *Gestor de plugins* del menú *Extensiones*.
5. Para saber como utilizar un plugin, suelen venir con una ayuda accesible al pinchar sobre él en el gestor de plugins.

Figura 10.30. Instrucciones de utilización del plugin Xtypo

6. Como puede observarse, Xtypo, y la mayor parte de los plugin de contenido hacen uso de un tipo de etiquetas especiales parecidas al estandar HTML: {xtypo_alert} mensaje {/xtypo_alert}, para diálogos de alerta, {xtypo_info} mensaje {/xtypo_info}, para diálogos de información, etc. Para su utilización en un artículo, deberemos escribirlas tal cual, siendo visible su efecto sobre el front-end. Crea un nuevo artículo, y haz uso de alguna de ellas.

Figura 10.31. Uso de Xtypo en un artículo

7. Para añadir un video de youtube en nuestro artículo, copia el código HTML, <*object*></*object*>, que youtube nos suministra.

Figura 10.32. Código HTML suministrado por youtube para agregar el vídeo en nuestros artículos

8. Para agregar el código HTML anterior en el artículo es necesario pinchar sobre el icono de edición HTML del editor y copiarlo.

Figura 10.33. Forma de agregar código HTML en nuestros artículos

9. Guarda el artículo, publícalo en la página de inicio del sitio Web y comprueba el resultado.

8.2. Instalación de módulos en Joomla

Los módulos son los elementos utilizados en Joomla para componer las plantillas utilizadas en el **front-end** con la finalidad de garantizar una modularidad e independencia en los contenidos que forman parte del sitio Web. Esta característica es muy importante, ya que nos permite modificar un región del documento, agregando, quitando o substituyendo algún módulo, sin tener en cuenta al resto de elementos que componen el documento. Como puede observarse en la figura X, los módulos se distribuyen en torno al área de contenidos central donde se muestran los artículos que son publicados, por su parte izquierda (*posición* **left**), derecha (*posición* **right**), encima (*posiciones* **top**), debajo (*posición* **footer**), o en posiciones personalizadas desde la plantilla (**user1**, **user2**,**user3** y **user4**).

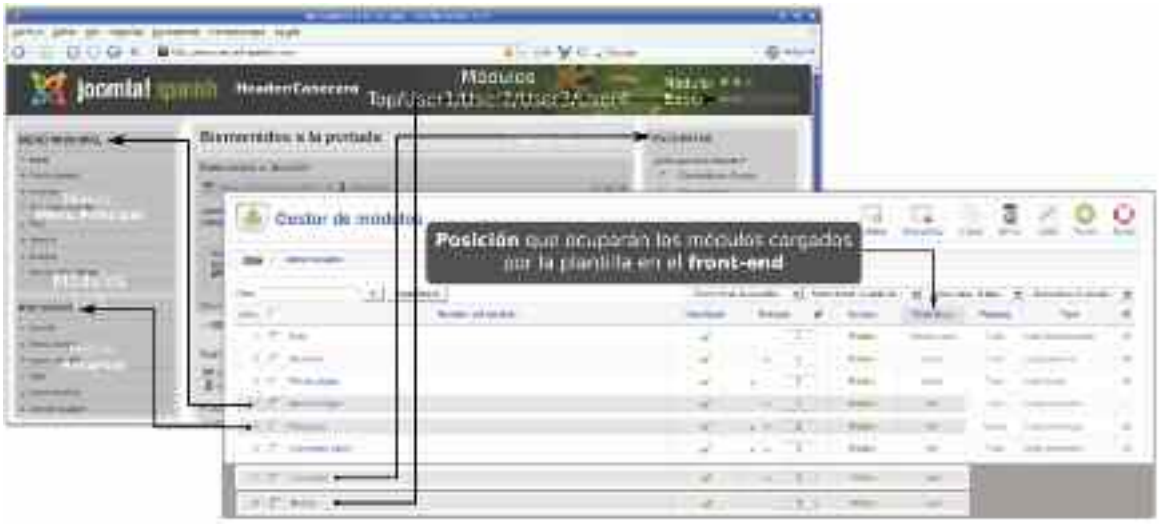


Figura 10.34. Distribución de los módulos dentro de la plantilla Joomla

Desde el *Gestor de módulos* en el menú *Extensiones*, podemos comprobar la lista de todos los módulos disponibles, y su posición dentro de la plantilla, tras la instalación de Joomla, pudiendo habilitarlos o deshabilitarlos.

¡¡Observación!! No todos los módulos que aparecen en la lista forman parte del sitio Web. Únicamente se cargan aquellos que son utilizados por la plantilla[76].

También podemos instalar nuevos módulos con la finalidad de mejorar el aspecto de nuestra Web y aportarle nuevas funcionalidades desde el *Gestor de extensiones*, menú *Extensiones*, opción *Instalar/Desinstalar*. Los hay de muy variados tipos: relojes, calendarios, reproductores de imagenes aleatorias, reproductores mp3, etc.

76 En la subcarpeta templates del directorio raíz de Joomla, *htdocs/joomla*, encontraremos todas las plantillas disponibles organizadas en diferentes carpetas. Entrando a la carpeta correspondiente a nuestra plantilla, podemos consultar el documento *index.php* para conocer los módulos que son cargados. También puede editarse desde el *Gestor de plantillas de Joomla*.

Ejercicio práctico n.º 9

Ingresa en el **back-end** con la cuenta de usuario **admin**, accede al *Gestor de módulos* desde el menú Extensiones, y deshabilita el módulo menú principal. Comprueba que al volver a cargar la página de inicio del sitio Web este ya no se muestra.

Ejercicio práctico n.º 10

Configura los módulos denominados "Ultimas Noticias" y "Popular", encargados de mostrar la lista de los últimos artículos creados en Joomla y los más visitados respectivamente de tal forma que cumplan los siguientes requisitos:

— Su nueva posición será left, debajo del menú principal.

— La lista de artículos solo mostrará los tres últimos publicados, y los tres más populares respectivamente.

— Sus nuevos títulos serán "Lo último!!" y "Lo más consulado".

— Sólo serán visibles desde la página de inicio. Al navegar por el sitio Web dejarán de mostrarse.

Figura 10.35. Aspecto de la Web tras la reubicación de los módulos

Solución ejercicio n.º 10

1. Desde el **back-end**, accede al *Gestor de módulos* en el menú *Extensiones*. En la lista de módulos disponibles buscar los denominados "Ultimas Noticias" y "Popular". Pincha sobre cualquiera de ellos para editar sus características. Tal como se muestra en la figura 10.36, edita su título, posición (*y orden en esa posición*), el número de artículos a listar, y las páginas del sitio Web donde queramos que se muestre.

Figura 10.36. Edición del módulo Últimas Noticias

2. Haz lo mismo con el otro módulo. Después visualiza el front-end y comprueba que los cambios de configuración han surtido efecto.

Ejercicio práctico n.º 11

Entra en "*http://extensions.joomla.org*" y descarga algún módulo calendario (por ejemplo, *Flash Dinamic Calender*), instálalo y ubícalo donde te parezca más razonable. De esta forma los usuarios que accedan al sitio Web podrán conocer fácilmente la fecha actual.

Ejercicio práctico n.º 12

Los menús de usuario son un módulo especial que tienen su propia herramienta de gestión. Compruébalo accediendo al módulo *Menú principal* desde el menú *Menús*. Edita el *Menú principal* despublicando todos sus items a excepción de *Inicio*. Después crea un nuevo item (por ejemplo, *Mi Artículo*) que enlace a un articulo que contendrá una imagen ubicada en la subcarpeta (por ejemplo, *images/naturaleza/foto1.jpg*) que crearemos dentro de la raíz de Joomla */htdocs/joomla*.

Figura 10.37. Item del menú principal asociado a un artículo

Solución ejercicio n.º 12

1. Despublica todos los items del Menú principal ha excepción de Inicio. Actualiza el **front-end** y comprueba su efecto.

Figura 10.38. Gestor de Items del menú principal

2. Crea un nuevo artículo dentro de alguna categoría pertenciente a la sección creada en los ejercicios prácticos. Para insertar una imagen desde el editor de artículos (por ejemplo, *images/naturaleza/foto1.jpg*) pincharemos en el icono asociado, e indicaremos

Figura 10.39. Pasos para la inserción de imágenes en nuestros artículos

3. Volviendo al *Gestor de items del menú* principal, crea un nuevo item del tipo *Artículos*.

Figura 10.40. Tipos de items

4. Por último configura los parámetros del nuevo item (por ejemplo, *Mi Artículo*) para que enlace con el artículo anterior.

Figura 10.41. Configuración del item del menú

5. Actualiza el **front-end** y comprueba el correcto funcionamiento.

Ejercicio práctico n.º 13

Crea un nuevo item (por ejemplo, *google*) en el menú principal de tipo URL embebida (Wrapper), y comprueba como podemos embeber URLs y documentos externos dentro de nuestro sitio Web, integrando el buscador de Google.

Figura 10.42. Resultado de embeber el buscador Google en nuestro sitio Web

Solución ejercicio n.º 13

Crea el item igual que en el ejercicio anterior, seleccionando tipo URL embebida, y configúralo para que la URL de enlace sea "*http://www.google.com*".

Figura 10.43. Configuración de un item de tipo Wrapper

Crea un subdirectorio llamado *misweb* en la raíz de *Joomla htdocs/joomla/misweb*, y copia allí todos los documentos HTML, XHTML, CSS, PHP, etc. que hayas diseñado durante la realización de los ejercicios propuestos a lo largo del libro y embébelos en el Joomla mediante la implementación de items de tipo wrapper. Si por ejemplo quisieramos embeber el ejercicio PHP-MySQL realizado en el capítulo X, tras copiar todos sus documentos asociados en el directorio anterior, la dirección del wrapper a configurar sería "*http://www.escarbapedal.com/misweb/miweb.php*".

Figura 10.44. Los item de tipo Wrapper nos permiten reutilizar todos los documentos Web realizados

8.3. Instalación de componentes en Joomla

Los componentes son las extensiones que aportan mayor funcionalidad a Joomla. Los hay que configuran dentro de nuestro sitio Web una completa galería de imágenes (Morfeo Show), un completo explorador de archivos (eXtplorer), etc. Para comprender mejor su utilidad instalaremos uno de ejemplo.

Ejercicio práctico n.º 14

Entra en "*http://extensions.joomla.org*", descarga el componente *eXtplorer* e instálalo desde el Gestor de extensiones (menú *Extensiones*, opción *Instalar/Desinstalar*).

Figura 10.45. Instalación del componente eXtplorer desde el gestor de extensiones

Después ves al menú Componentes y selecciónalo. Comprueba que se trata de una potente herramienta software que nos permite subir archivos al servidor, crear directorios y archivos, cambiar permisos, modificar contenidos, etc. Muy útil.

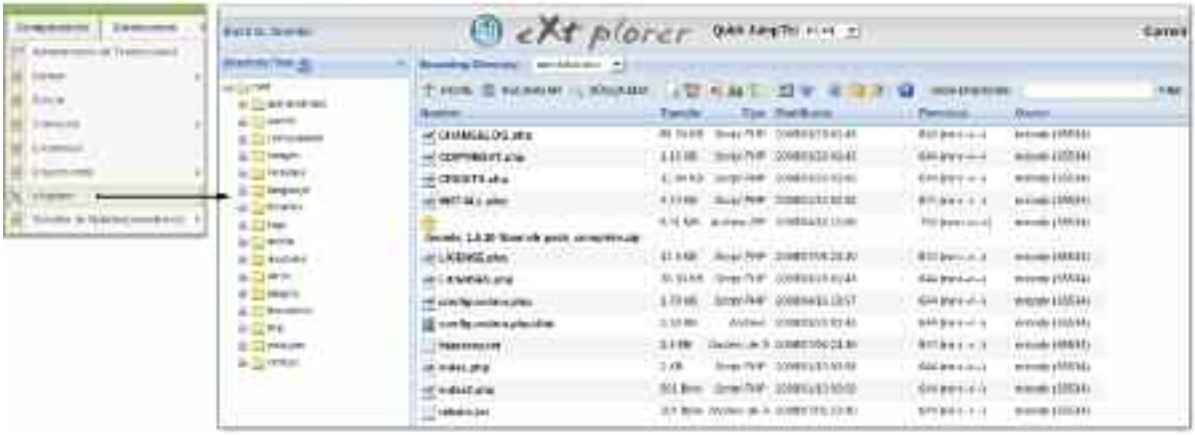


Figura 10.46. Aspecto de eXtplorer sobre Joomla

Ejercicio práctico n.º 15

Crea un nuevo ítem en el menú principal que permita acceder al subdirectorio "images" del sitio Web *Joomla* desde el **front-end** a través del componente eXtplorer, presentando las siguientes características:

— Acceso especial: el ítem del menú sólo será accesible por usuarios *Joomla* especiales (*autores, publicadores, ..., administradores*). Para el público en general y usuarios registrados, el ítem será transparente.

— Al pinchar sobre el ítem, el contenido del subdirectorio *images* (archivos y subdirectorios) se mostrará en una nueva ventana del explorador sin barra de navegación.

Solución ejercicio n.º 15

1. Desde el Gestor de Ítems del Menú principal crea un nuevo ítem (p.e. *Explorador*).
2. Selecciona el tipo de ítem del menú asociado al componente eXtplorer.
3. En el apartado de Detalles del ítem del menú, seleccionar nivel de acceso especial, y al hacer click, abrir dentro de Nueva Ventana sin barra de navegación. Después aplica y guarda los cambios para que estos surtan efecto.

Figura 10.47. Pasos para la creación de un ítem asociado al componente eXtplorer

4. Edita el fichero de configuración de eXtplorer ubicado en *htdocs/joomla/components/com_extplorer/configuration.ext.php*, y asigna los siguientes valores a sus variables:

```
$frontend_enabled = true;
$subdir = '/images';
```

Donde *$frontend_enabled* permite que el componente eXtplorer pueda usarse desde el **front-end**, y *$subdir* indica el subdirectorio de Joomla que será explorado.

5. Autentícate a través del formulario de acceso del **front-end** con una cuenta de usuario *Joomla* (por ejemplo, *admin*) y comprueba el acceso restringido y correcto funcionamiento del nuevo ítem.

Figura 10.48. Comprobación del comportamiento del ítem del menú principal

Ejercicio práctico n.º 16

Además de los componentes preinstalados en Joomla (servidor de noticias, encuestas, banners, etc.) existen otros muchos que nos permiten cubrir multitud de necesidades. Instala otros componentes disponibles para *Joomla* (por ejemplo, *http://extensions.joomla.org*) y comprueba su potencia. Se recomiendan los componentes *editor JCE*, galería de imágenes *Morfeo Show*, el calendario de eventos *GCalendar*, el administrador de comentarios de artículos *yvComment* o el gestor de archivos para subir y descargar archivos *Remository*.

Podrá comprobarse que muchos de estos componentes tienen asociados módulos y plugins permitiéndonos aprovechar su funcionalidad en la personalización de los módulos que componen las páginas del sitio Web y creación de artículos.

¡¡Observación!! Al crear un usuario en *Joomla* es posible asignarle cualquiera de los editores de textos para la creación de artículos disponibles. Sus diferencias radican en el aspecto, las presentaciones y barras de herramientas ofrecidas. El editor JCE se caracteriza por tener su propio gestor permitiéndonos personalizar su apariencia, agregar nuevas herramientas, etc.

Figura 10.49. Asignación del editor de artículos al crear un usuario

9. CABECERA DE JOOMLA

Otro aspecto fundamental en la personalización de nuestro sitio Web es configurar la cabecera o header de la plantilla, colocando una imagen u objeto (Flash o SVG) acorde a sus contenidos. Desde el *Gestor de plantillas*, menú *Extensiones*, podemos personalizar algunas de sus características, pero no la imagen a utilizar.

Figura 10.50. Edición de parámetros de la plantilla asignada al sitio Web Joomla

Aunque cada plantilla coloca el logo en su cabecera haciendo uso de estrategias diferentes, podremos aprovecharnos de matices comunes. Su modificación puede realizarse de diferentes formas:

1. La forma más sencilla de hacerlo, es substituyendo la imagen o logo que utiliza la plantilla por el nuestro. Si exploramos el contenido de nuestra plantilla dentro del subdirectorio *templates/nombre_plantilla*, p.e. mediante eXtplorer, encontraremos un subdirectorio llamado *images* que contiene todas las imágenes usadas por ella. Entre ellas podremos localizar el logo de la cabecera (p.e. *logo.png*) y reemplazarla por la nuestra con el mismo nombre, atendiendo especialmente a sus dimensiones.

¡¡Ayuda!! Para conocer las dimensiones del logo utilizado por defecto por nuestra plantilla es posible utilizar la herramienta software *identify* perteneciente al paquete *ImageMagick* disponible tanto para GNU/Linux como para Microsoft Windows.

```
> identify templates/nombre_plantilla/images/logo.png
templates/nombre_plantilla/images/logo.png PNG 350x80 350x80+0+0 PseudoClass 256c 8-bit
6.16406kb
```

Para redimensionar sin deformar, renombrar y convertir (*jpg -> png, gif -> jpg, etc.*) nuestra imagen/logo acorde al tipo y dimensiones anteriores (*la altura del logo debe respectarse por distribución de la plantilla*) puede hacerse uso de la herramienta software *convert* perteneciente al paquete *ImageMagick*, ejecutando alguno de los siguientes comandos:

```
> convert -resize 350x80 mi_imagen.jpg templates/nombre_plantilla/images/logo.png
> convert -resize x80 mi_imagen.jpg templates/nombre_plantilla/images/logo.png
```

2. Otra posibilidad es editar el documento Web *index.php* u hoja de estilos CSS (*css/template.css*) y modificar la ruta asociada a su logotipo por la correspondiente a la nuestra, o insertarla directamente haciendo uso de las etiquetas HTML <*img*> u <*object*></*object*> (por ejemplo, *objeto Flash*), pudiendo comentar <*!*— —> todo aquello que no queramos que aparezca.

Figura 10.51. Edición de la página de inicio de nuestra plantilla para insertar nuestro logo

¡¡Observación!! La ruta de la imagen u objeto a insertar como cabecera o header de la plantilla no es relativa al documento *index.php* que se esta editando, sino a la raíz Joomla.

Ejercicio práctico n.º 17

Consulta cuales son las dimensiones más apropiadas para la cabecera de la plantilla que utilizas y crea una imagen u objeto para reemplazar la que haya por defecto.

10. OTROS ASPECTOS A PERSONALIZAR EN JOOMLA

Joomla nos permite personalizar otros muchos aspectos que pueden resultar interesantes. Explorando por su panel de control, o editando los ficheros Web que lo componen, podremos solucionar todos aquellos problemas de apariencia que podamos detectar. Un aspecto que merece la pena mencionar es como configurar el título que se muestra por defecto en la página de inicio, "Bienvenidos a la portada", y la forma en que tiene esta de se organizar los artículos publicados. Todo esto es modificable accediendo a los parámetros de configuración de la página de inicio a través del ítem *Inicio* del menú principal, a través del *Gestor de ítems del menú* principal, del menú *Menús*.

Figura 10.52. Personalización de los parámetros de configuración de la página de inicio del sitio Web

Capítulo 11

INTRODUCCIÓN AL SERVICIO WEB SEGURO

Cuando alguien accede a nuestra página web, a través del puerto 80, toda la información que se mueve por la red no está encriptada. Esto significa que cualquier hacker que se encuentre con un sniffer en el segmento de red adecuado tiene la posibilidad de acceder a la información que viaja por él. Esta información podría ser de carácter privilegiado como nombres de usuario y contraseñas, lo que proporciona una gran vulnerabilidad para la empresa y las aplicaciones web que se puedan ejecutar.

Para solucionar este problema apareció el protocolo **https** que permite una comunicación web encriptada a través del puerto 443. Esta encriptación se realiza a través de protocolos SSL (secure sockets layer) y TLS (transport layer security). Cuando se utiliza esta solución, aunque las tramas que discurren por la red sean capturadas, el hacker no podrá leerlas a causa de la encriptación y por tanto la información transmitida estará segura.

Esta la razón por la cual toda transacción económica a través de la red se realiza mediante **https**: compra de artículos en los que debes proporcionar números de tarjetas de crédito, operaciones bancarias... Por tanto, si debes diseñar un sitio web para una empresa que hace negocios a través de Internet, es fundamental tener unos conocimientos básicos acerca de como se crean estos sitios seguros y ese es el objetivo fundamental de este capítulo.

1. CONTENIDOS DEL CAPÍTULO

En primer lugar se hace una introducción al cifrado en la transmisión de datos, describiendo el funcionamiento de las claves asimétricas. A continuación se explica que es una autoridad de certificación y como podemos crear una nosotros mismos para hacer pruebas. La siguiente y ultima sección teórica está dedicada a explicar como se configura Apache para dar un servicio seguro (protocolo https).

Como colofón final se presentan tres ejercicios. Los dos primeros tienen una dificultad baja y muestran como poner en funcionamiento un sitio web seguro y como combinar servicios seguros y no seguros con el mismo servidor. El tercero tiene una mayor complejidad y explica como hacer uso de Apache, SSL, Python y MySQL para crear sitios seguros y dinámicos.

2. SOFTWARE NECESARIO PARA REALIZAR LAS PRÁCTICAS PROPUESTAS

Todo el software utilizado en este capítulo es libre y disponible desde Internet. Está disponible tanto para la plataforma Windows como para la plataforma Linux. Necesitarás:

— Apache: Inicialmente con el módulo **ModSSL** y posteriormente, para que puedas realizar el último ejercicio propuesto, el módulo **mod_python**.
— OpenSSL: Es el módulo que te permite obtener las claves necesarias para hacer las transmisiones seguras con Apache, así como crear tu propia entidad de certificación.
— MySQL: El último ejercicio propuesto necesita trabajar con una base de datos para almacenar información.
— Python: Se va a utilizar PSP para crear un comportamiento dinámico de los sitios web propuestos. Además será necesaria la librería MySQLdb para poder acceder mediante programación a las bases de datos MySQL.

3. INTRODUCCIÓN A LA TRANSMISIÓN SEGURA: CLAVE ASIMÉTRICA

El objetivo de esta sección es proporcionar una ligera idea de lo que es una transmisión cifrada y como se produce. Existen dos formas básicas de cifrar una transmisión: cifrado simétrico y cifrado asimétrico.

El cifrado simétrico está basado en la utilización de una clave secreta que conocen los interlocutores y que es utilizada tanto para cifrar y descifrar. Este método tiene básicamente dos inconvenientes. El primero de ellos surge porque en algún momento la clave debe ser comunicada entre los interlocutores y durante este periodo podría ser interceptada. El segundo inconveniente se debe al número de claves necesarias para intercomunicar de forma segura a un número elevado de personas; para un grupo de n personas se necesitarían $n(n-1)/2$ claves diferentes. Esto conduce a graves problemas de mantenimiento y gestión segura.

El cifrado asimétrico se basa en utilizar dos claves para establecer la comunicación una de ellas es privada y la otra es pública. El dueño de ambas da a conocer la clave pública, que es utilizada por los interlocutores para cifrar los mensajes que desean enviar al poseedor de la clave privada. Estos mensajes sólo podrán ser decodificados a través de la clave privada. Para establecer una comunicación segura entre n personas se necesitarían $2n$ claves (n privadas y n públicas) y esto representa una gran ventaja frente al número necesario en el cifrado simétrico. El cifrado asimétrico posee otra importante ventaja: si el poseedor de las dos claves cifra un mensaje con su clave privada, dicho mensaje podrá ser descifrado por todos aquellos que dispongan de la clave pública y esto será señal inequívoca de que el mensaje fue enviado por el dueño de la clave privada; este es el fundamento de la firma digital. El gran inconveniente del cifrado asimétrico es que el coste computacional[76] de cifrado, descifrado y transmisión de datos es significante mayor que el producido con un cifrado simétrico que proporcione el mismo nivel de seguridad.

Para la transmisión segura de información a través de la web se utiliza el protocolo **https** (Hypertext Transfer Protocol Secure) que a su vez se basa en SSL (Secure Socket Layers). Por su parte SSL utiliza criptografía híbrida, es decir usa tanto el cifrado simétrico como el asimétrico, lo que permite tomar las ventajas de los dos métodos. En particular el cifrado asimétrico es utilizado para enviar la clave de cifrado simétrico[77] y así la transmisión se información se hace a mayor velocidad.

4. LA AUTORIDAD CERTIFICADORA (CA)/label{sec:ca}

El protocolo **https** se emplea sobre todo en aquellas aplicaciones web donde aparecen transacciones económicas: bancos, tiendas online, ... Cuando se establece la comunicación segura es,

76 Desde el punto de vista del usuario el coste computacional se traduce en una lenta transmisión de la información.

77 En cada sesión establecida entre interlocutores se utiliza una clave simétrica nueva.

seguramente, porque al cliente se le están solicitando datos confidenciales, como por ejemplo el número de una tarjeta de crédito. Cuando la empresa online hace esta solicitud, el cliente debe confiar en dicha empresa para facilitar los datos, pero ¿cómo sabe el cliente que la empresa es realmente quien dice ser?

Aquí es donde aparece en juego una autoridad certificadora o certification authority (CA). Una CA firma con su clave privada el certificado de seguridad que la empresa online proporciona al cliente. Este certificado de seguridad es descifrado a través de la clave pública de la CA y así el cliente tiene la seguridad[78] de que la empresa online es realmente quien dice ser.

El procedimiento de solicitud de un certificado es: **\label{pasoscertificar}**

1. La empresa online que solicita la firma de un certificado de seguridad proporciona a la CA información relativa a sí misma, incluyendo por supuesto la dirección (URL) de su servidor.
2. Si no se proporcionan claves, la CA creará una pareja de claves pública y privada.
3. Con la información anterior se crea un archivo de solicitud de firmado de certificado o CSR (Certificate Signing Request), que contiene la clave pública. Este archivo es el que realmente necesita la CA (por tanto estos tres pasos pueden haber sido dados por la propia empresa solicitante en caso de tener los conocimientos necesarios para llevarlos a cabo).
4. La CA verifica que la información mandada por la empresa solicitante es correcta. Esto lo puede hacer por sí misma o utilizando una Autoridad de Registro o RA.
5. Tras la comprobación y el cobro de una tarifa, la CA firma el certificado de seguridad y se lo devuelve al solicitante.
6. Por último, y utilizando la misma herramienta que fue usada para la creación del archivo CSR, la empresa solicitante puede instalar en el servidor web seguro el certificado.

Este será el certificado que la empresa mande al cliente para que sea aceptado y así poder establecer la comunicación segura cifrando la información que el cliente mande a través de la clave pública proporcionada por la empresa. Para establecer la confianza del usuario con una CA es posible instalar en el navegador un CA raíz. Este certificado suele estar autofirmado por la CA y normalmente es un archivo con extensión **.crt**. Actualmente, los navegadores tienen preinstalados CA raíz por defecto de forma que la admisión de un certificado de seguridad de una empresa, firmado a su vez por una CA en la que se confía (es decir de la que se tiene instalado el CA raíz en el navegador) se producirá de forma automática y completamente transparente para el usuario. En la figura 11.1 se muestra los CA raíz pregrabados en Mozilla (Preferencias —> Avanzado —> Cifrado —> Ver certificados —> Autoridades).

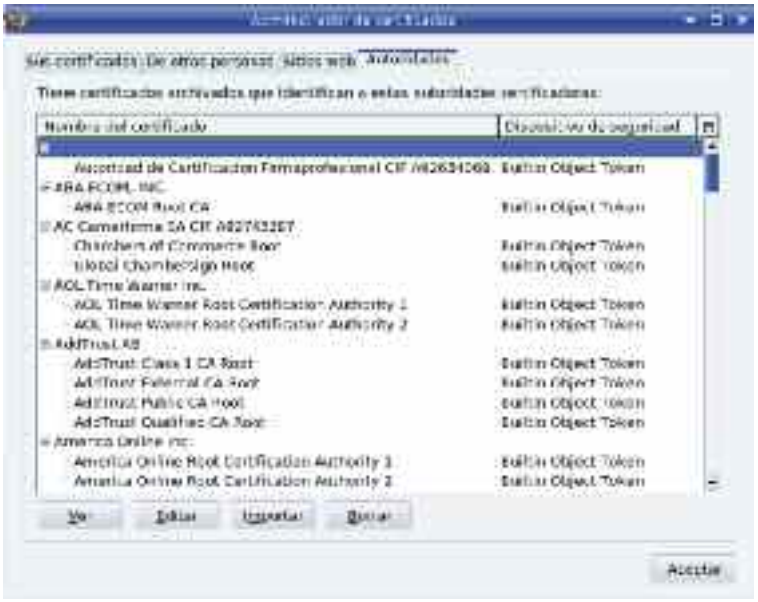


Figura 11.1. Algunos de los CA raíz preinstalados en Mozilla

78 Esto implica que debe existir una confianza en la CA, ya que no existe ningún procedimiento que demuestre realmente que debemos confiar en ella.

5. NUESTRA PROPIA ENTIDAD CERTIFICADORA

En este capítulo se desea poner en marcha un servidor Apache que dará acceso a un sitio web seguro. Para ello, no solo es necesario configurar el servidor adecuadamente y construir el sitio web. Además alguien tendrá que decir al cliente web que puede confiar en el sitio al que está accediendo, y que si se le pide información delicada no debe temer que caiga en manos inadecuadas.

Ese alguien es una entidad certificadora (CA) y aunque podríamos trabajar con alguna de las existentes[79] vamos a crear nuestra propia CA para autofirmar nuestros certificados.

En primer lugar, nuestra CA debe disponer de una pareja de claves pública y privada. Para obtener dicha pareja utilizaremos **openssl**. Utilizando **openssl** la clave privada contiene a su vez la clave pública y por tanto no es necesario generarlas por separado. Dependiendo del algoritmo criptográfico utilizado se obtendrá un tipo de clave u otro. En la práctica los más comunes son RSA (iniciales de los creadores del algoritmo: Ron Rivest, Adi Shamir y Len Adleman) y DSA (Digital Signature Algorithm). RSA se puede utilizar tanto para encriptar información como para firmar documentos, sin embargo sólo se usa para firmar. Por esta razón elegimos el algoritmo RSA. En primer lugar creamos un directorio al que vamos a llamar **CA**, en el cual se almacenará la información referente a la CA que vamos crear. Utilizando la opción **genrsa** a través de **openssl**:

```
[user@CA]$ openssl genrsa -des3 -out privkey.pem 2048
```

Al utilizar la opción **-des3**, **openssl** solicitará una contraseña para generar la clave (en realidad las dos claves). Esta contraseña será solicitada por el servidor cada vez que necesite acceder a la clave; dependiendo de las personas que tengan acceso al servidor y en función de la seguridad, puede ser más cómodo generar la clave sin la opción **-des3**, es decir, sin contraseña:

```
[user@CA]$ openssl genrsa -out privkey.pem 2048
```

La clave es almacenada en el fichero **privkey.pem**[80]. El número 2048 indica el tamaño en bits de la contraseña; por defecto es 512 pero actualmente para claves RSA se recomienda un tamaño de 2048 o mayor.

El siguiente paso consiste en crear un certificado firmado con la clave anteriormente calculada. Los datos que se introducen en el certificado se toman del archivo **openssl.cnf'**. La ubicación de este archivo dependerá de la distribución a utilizar, aunque siempre de **/etc/**. Es posible utilizar el comando **find** para encontrarlo:

```
[root@CA]# find /etc/ -name openssl*
/etc/pki/tls/openssl.cnf
```

Una vez abierto, lo que nos interesa es rellenar la parte que hace referencia a los datos de la CA y que a continuación se muestra un ejemplo de como rellenarlos:

---

79 Trabajar con alguna entidad certificadora implica en la mayor parte de los casos un coste económico. Es posible utilizar CA gratuitas como Cacert.org, sin embargo como la mayoría de los navegadores no incluyen su certificado, se comportan hacia el cliente como si de un certificado autofirmado se tratara.

80 La extensión **pem** son la iniciales de Privacy-enhanced Electronic Mail, y es la que normalmente se utiliza para almacenar las claves generadas.

```
...
[ req_distinguished_name ]
countryName                     = Country Name (2 letter code)
countryName_default             = ES
countryName_min                         = 2
countryName_max                         = 2

stateOrProvinceName             = State or Province Name (full name)
stateOrProvinceName_default     = La Rioja

localityName                    = Locality Name (eg, city)
localityName_default            = Haro

0.organizationName              = Organization Name (eg, company)
0.organizationName_default      = IES Cossio

# we can do this but it is not needed normally :-)
#1.organizationName             = Second Organization Name (eg, company)
#1.organizationName_default     = World Wide Web Pty Ltd

organizationalUnitName          = Departamento electronica
#organizationalUnitName_default =

commonName                      = www.cossio.com
commonName_max                          = 64

emailAddress                    = juanjo@cossio.net
emailAddress_max                = 64
...
```

A continuación es posible generar el certificado, que incluirá nuestros datos a través del siguiente comando. Observa que se pide información acerca de la empresa a certificar, que en este caso somos nosotros mismos tanto como solicitante, como CA. Si pulsamos **intro** en cada pregunta se tomarán los valores por defecto (los que aparecen entre corchetes).

```
[user@CA]$ openssl req -new -x509 -key privkey.pem -out cacert.pem -days 1095
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [ES]:
State or Province Name (full name) [La Rioja]:
Locality Name (eg, city) [Haro]:
Organization Name (eg, company) [IES Cossio]:
Departamento electronica []:
www.cossio.com []:
juanjo@cossio.net []:
```

La opción **req** se utiliza para crear y procesar solicitudes de certificado en formato[81] PKCS#10. En este caso la estamos utilizando para autofirmar un certificado que será usado como CA raíz.

81 El formato PKCS#10 es el estandar de solicitud de certificación.

La opción **-new** crea una nueva solicitud de certificación preguntando al usuario la información relevante que sea necesaria. En este caso se ha empleado la opción **-key** para indicar el fichero con las claves (pública y privada), ya que el certificado contendrá la clave pública que permitirá comprobar la firma; en el caso de no haberse utilizado esta opción **openssl** hubiera generado automáticamente una clave RSA.

La opción **-x509** se usa para proporcionar como salida del comando un certificado firmado, en lugar de una solicitud de certificación. Por tanto esta opción sólo tiene sentido utilizarla si deseamos crear una autoridad de certificación (CA) raíz autofirmada.

La opción **-out** indica el nombre del fichero que contiene el certificado autofirmado de la CA.

Por último la opción **-days** indica el número de días de validez que tendrá el certificado creado. Por defecto el número de días es 30.

Cuando hagamos una conexión a un sitio web seguro que utilice este certificado, el navegador nos mostrará un mensaje pidiéndonos que lo aceptemos. Esto es así porque dicho certificado no está incluido en los certificados que el navegador acepta sin preguntar. Dichos certificados son los mostrados en la figura 11.1. Para cargar nuestro certificado en el navegador web y evitar que nos pregunte si lo aceptamos es posible cargarlo.

Por ejemplo en Mozilla deberíamos ir a **preferencias —> Avanzado —> Cifrado —> Ver certificados —> Atoridades** y pulsamos a continuación **importar**. Seleccionamos el archivo **cacert.pem** creado anteriormente y aceptamos la carga. En el siguiente acceso a una página web que utilice este certificado no se nos pedirá que lo aceptemos.

Figura 11.2. Pasos en la inclusión del certificado creado en el navegador Mozilla/Firefox

Se debe señalar que existen varios formatos de los CA raíz. En general el formato PEM (extensión **.pem**) es utilizado en el mundo Unix y el formato PKCS#12 (extensión **.p12**) en el mundo Microsoft. **Openssl** permite transformar un tipo en otro a través del siguiente comando:

```
[user@CA]$ openssl pkcs12 -export -in ./cacert.pem -inkey ./privkey.pem -out cacert.p12
Enter Export Password: iescossio
Verifying - Enter Export Password:
```

La opción **pkcs12** indica que se desea construir un archivo en formato **.p12**. La opción **-export** se utiliza para señalar que se va a realizar una transformación entre formatos. A través de **-in** se proporciona el certificado autofirmado con extensión **.pem** y a través de **-inkey** el archivo que define la clave privada con la que fue firmado el certificado. La opción **-out** se utiliza para indicar el nombre del archivo **.p12** que se va a crear.

Una vez creado puede ser incluido en los certificados de tu navegador preferido, si este requiere este tipo de formato.

En este punto de la práctica que estamos realizando deberías estar en posesión de tres archivos: **cacert.p12**, **cacert.pem** y **privkey.pem**.

6. SOLICITUD Y OBTENCIÓN DE UN CERTIFICADO FIRMADO

Imagina que posees una empresa que desea ofrecer un servicio web seguro. Tal como se muestra en la página **\pageref{pasoscertificar}**, y en el caso de que no tuvieras los conocimientos necesarios, en primer lugar deberás proporcionar a la CA los datos de tu empresa y servidor para que esta pueda crear el certificado firmado que tu empresa solicita. Como se supone que tienes los conocimientos, los tres primeros pasos de esa lista los realizará tu propia empresa. Creamos otro directorio al que llamamos **server** dentro del directorio **CA** en el que se han generado los anteriores archivos de claves y certificados. A continuación creamos las claves pública y privada que necesita nuestra empresa para tener un servicio web seguro:

```
[user@CA/server]$ openssl genrsa -out serverprivkey.pem 2048
Generating RSA private key, 2048 bit long modulus
.....................................................+++
.....................+++
e is 65537 (0x10001)
```

Dichas claves han sido almacenadas en el archivo **serverprivkey.pem**. A continuación creamos un archivo de solicitud de firma de certificado (CSR) que contendrá todos nuestros datos. Para ello utilizamos de nuevo **openssl** y la información contenida en el fichero **openssl.cnf**. En esta ocasión y para mostrar la introducción de datos por teclado no vamos a cambiar dicho fichero para adaptarlo a los nuevos datos, aunque sería posible (y recomendable) hacerlo.

```
[user@CA/server]$ openssl req -new -key serverprivkey.pem -out servercert.p10
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [ES]:
State or Province Name (full name) [La Rioja]:
Locality Name (eg, city) [Haro]:Logroño
Organization Name (eg, company) [IES Cossio]:Electrónica e Informática S.A.
Departamento electronica []:
www.cossio.com []:www.ingemartin.es
juanjo@cossio.net []:juanjo@ingemartin.es
```

```
Please enter the following 'extra' attributes to be sent with your certificate request
A challenge password []:ingemartin09
An optional company name []:
[user@CA/server]$
```

Se vuelve a hacer uso de **openssl** con la opción **req**. Esta última opción está entendida inicialmente para crear y procesar solicitudes en formato PKCS#10 (que es el estándar de solicitud de certificación, basado en el RFC 2986). Como se ha visto antes, la opción **req** puede utilizarse para crear certificados autorfirmados CA raíz, pero no es su función por defecto. El resto de opciones indicadas en el comando ya han sido explicadas anteriormente,por lo que se intuye que el resultado será un fichero **servercert.p10** que se debería enviar a la CA que deseemos lo firme.

Obsérvese que se ha cambiado parte de la información que contenía el fichero **openssl.cnf** para que parezca información personal de una empresa diferente a la CA. Tras este paso ya tenemos el archivo **servercrt.p10**, que podrá ser enviado a una CA para que lo firme tras la comprobación de los datos proporcionados y el pago de la correspondiente tarifa. En nuestro caso, como nos hemos convertido en una CA podremos firmarlo sin recurrir a un servicio externo.

```
[user@CA/server]$ openssl x509 -req -in servercert.p10 -out servercert.pem -CA
../cacert.pem -CAkey ../privkey.pem -CAcreateserial -days 1135
Signature ok
subject=/C=ES/ST=La Rioja/L=Logro\xC3\x83\xC2\xB1o/O=Electr\xC3\x83\xC2\xB3nica e
Inform\xC3\x83\xC2\xA1tica S.A./CN=www.ingemartin.es/emailAddress=juanjo@ingemar-
tin.es
Getting CA Private Key
[juanjo@CA/server]$
```

Utilizamos el comando suplementario **x509** utilizado para firmar certificados. La opción **-req** es utilizada para indicar que el fichero de entrada va a ser una solicitud de certificado; en nuestro caso **servercert.p10**, indicado a través de la opción **-in**. La salida, opción **-out**, será el archivo de certificado firmado **servercert.pem**. Mediante la opción **-CA** se indica el fichero de certificación raíz de la autoridad certificadora. Con la opción **-CAkey** se indica la ubicación de la clave privada de la CA. La opción **Cacreateserial** es necesario para que se cree un número de serie de certificación, en caso de no utilizar esta opción obtendremos un error. Por último mediante **-days** indicamos el periodo de validez de la firma.

7. CONFIGURACIÓN DE APACHE PARA UN SITIO SEGURO

Apache es un servidor web capaz de servir páginas seguras y no seguras a la vez. El servicio web "normal" es proporcionado a través del puerto 80 y el servicio web seguro a través del puerto 443. Posteriormente, a través de un ejercicio, se mostrará como configurar Apache para dar los dos servicios a la vez, pero en esta sección el trabajo se centrará en configurar el sitio web seguro. Para ello es necesario conocer, en primer lugar, varias directivas Apache. Algunas ya han sido comentadas en otros capítulos, pero se repiten aquí por claridad.

```
NameVirtualHost dirección ip:443
```

Un servidor web no sirve únicamente un sitio, sino que puede albergar varios de ellos. En Apache esto se consigue a través de los denominados **host virtuales**. Para poder definirlos es

necesario utilizar esta directiva indicando la dirección IP del equipo servidor y el puerto. En este caso el puerto 443 indica que los host virtuales a definir usarán el protocolo **https**.

```
<VirtualHost dirección ip:443> </VirtualHost>
```

Este elemento contendrá los parámetros y directivas de configuración de un host virtual. Debe contener la dirección IP del servidor y el puerto a utilizar. Dentro de este elemento se encontrarán las directivas definidas a continuación.

```
DocumentRoot ruta entre comillas dobles
```

Indica la ruta absoluta, con respecto al sistema de archivos del servidor, en la que se encuentra el sitio web a servir. Para el servidor este directorio será el raíz del sitio web.

```
ServerName nombre de dominio cualificado:443
```

Con esta directiva se indica el nombre con el que será identificado el sitio web a servir y el puerto por el que se da el servicio. Por ejemplo **www.websegura.com:443**.

```
ServerAdmin dirección de correo
```

Indica la dirección del correo del administrador del sitio. Esta dirección es la mostrada en la pantalla del navegador cuando se produce algún mensaje de error.

```
SSLEngine on
```

Directiva utilizada para habilitar el trabajo del servidor con SSL/TLS. Es necesario activarla a **on**, ya que su valor por defecto es **off**.

```
SSLCipherSuite tipo de cifrado
```

Con esta directiva se establecen los métodos de cifrado aceptados para dar el servicio web seguro. Por ejemplo `SSLCipherSuite RSA:+HIGH:+MEDIUM'. En **http://httpd.apache.org/docs/2.2/mod/mod_ssl.html#table1** puedes encontrar los diferentes tipos de cifrado posibles.

```
SSLProtocol tipo
```

Indica los tipos (versiones) de protocolos SSL que se permiten usar en el servicio. Estos protocolos pueden ser SSLv2, SSLv3 y TLSv1 (listados aquí desde el más antiguo al más moderno). Si utilizas **all**, se permitirán todos ellos.

```
SSLCertificateFile archivo
```

Directiva utilizada para indicar el certificado del servidor.

```
SSLCertificateKeyFile archivo
```

Indica el fichero que contiene la clave privada del servidor.

```
SSLCertificateChainFile archivo
```

Directiva utilizada para indicar el archivo donde se encuentran encadenados (uno tras otro) los certificados de las autoridades de certificación (CA). Debe estar definido bajo la codificación **pem**.

```
#   Certificate Authority (CA):
SSLCACertificateFile archivo
```

Archivo donde se encuentra el certificado de la autoridad de certificación con la que trabajas. Debe tener una codificación **pem**.

```
<Directory Ruta del DocumentRoot entre comillas dobles> </Directory>
```

Este elemento es utilizado para dar los adecuados permisos dentro del directorio raíz del sitio o para configurar el funcionamiento bajo unas determinadas condiciones.

Ejercicio práctico n.º 1

Configura un sitio web seguro como un host virtual de Apache e identificado a través del nombre de dominio cualificado (FQDN- Fully Qualified Domain Name) **www.websegura.com**. Para ello utiliza las claves y los certificados creados en la sección \ref{sec:ca}. El contenido del sitio puede ser cualquiera y basta con que de un mensaje de bienvenida.

Solución ejercicio n.º 1

Puesto que Apache debe responder ante la llamada a **www.websegura.com** en primer lugar necesitamos que un servidor de nombres (DNS) este configurado para resolver este nombre y convertirlo en la dirección IP del servidor. La configuración de un DNS excede el propósito de este libro y es de suponer que el lector conoce[82] el procedimiento.

Para comprobar que el nombre de dominio es resuelto adecuadamente puede utilizarse el comando **nslookup**, válido tanto para Windows como para Linux. En el presente ejercicio la misma máquina con dirección 192.168.10.150 se utiliza como servidor web y como servidor de nombres. En tu caso la dirección IP será aquella que tengas configurada en tu interfaz de red.

82 El libro "Servicios de Internet" (ISBN 84-8465-202-5), de los mismos autores, cubre ampliamente ese aspecto y puede utilizarse como referencia para subsanar esta parte del ejercicio.

```
[root@directorio]# nslookup www.websegura.com
Server:         127.0.0.1
Address:        127.0.0.1#53

Name:   www.websegura.com
Address: 192.168.10.150
```

Tras tener configurado el DNS, creamos el fichero **index.html** que se utilizará para dar la bienvenida. Puede ser algo tan sencillo como lo siguiente:

```
<HTML>
 <P ALIGN=CENTER>
  <FONT COLOR="#ff0000" SIZE=6>
   <SPAN STYLE="background: #cccc00">
   Bienvenido a una web segura
   </SPAN>
  </FONT>
 </P>
</HTML>
```

En el que se ha centrado la frase *Bienvenido a una web segura*, se le ha dado un color de fondo y un color a la fuente. Este archivo se guardará como **index.html** en el directorio **/var/www/SSL** destinado a ser nuestro DocumentRoot (este directorio puede ser otro dependiendo del sistema operativo que estés utilizando para realizar el ejercicio).

El próximo paso consistirá en modificar correctamente el fichero **httpd.conf** (fichero de configuración de Apache). Sabiendo que los archivos con los certificados y claves están en las rutas **/etc/httpd/conf/server/** y **/etc/httpd/conf/CA/** (esto dependerá del proceso de creación de los mismos y del sistema operativo que se esté utilizando), incluiremos las siguientes líneas:

```
Listen 443
    ...
#Otras líneas de configuración del archivo
    ...
ServerName www.websegura.com:443
    ...
#Otras líneas de configuración del archivo
    ...
NameVirtualHost 192.168.10.150:443

 <VirtualHost 192.168.10.150:443>
   DocumentRoot "/var/www/SSL"
   ServerName www.websegura.com:443
   ServerAdmin webmaster@cossio.com
   SSLEngine on
   SSLCipherSuite RSA:+HIGH:+MEDIUM
   SSLProtocol all -SSLv2
   SSLCertificateFile /etc/httpd/conf/server/servercert.pem
   SSLCertificateKeyFile /etc/httpd/conf/server/serverprivkey.pem
   SSLCACertificateFile /etc/httpd/conf/CA/cacert.pem
```

```
        <Directory “/var/www/SSL”>
            Allow from all
        </Directory>
    </VirtualHost>
```

La directiva Listen permite indicar a Apache que debe escuchar las peticiones que le lleguen por el puerto 443, destinado a **https**.

Como se van a emplear un host virtual para definir el sitio, la directiva **ServerName www.websegura.com:443** no sería necesario utilizarla, pero siempre es conveniente tener un **ServerName** en la configuración general (fuera de los host virtuales), aunque sólo sea para evitar que Apache te de un aviso (warning).

El resto de las directivas empleadas han sido explicadas en la sección anterior. Comentar únicamente que la directiva **SSLProtocol all -SSLv2** está indicando que de los tres protocolos definidos anteriormente se utilizan todos menos la versión 2 de SSL. De ahí que aparezca el signo menos delante de SSLv2. Además se ha indicado a través de <Directory></Directory> que todo el mundo tiene acceso al sitio web. Por defecto esto no es así y por tanto esta indicación es necesaria.

Una vez hecho esto basta con reiniciar el servicio:

```
[root@directorio]# apachectl restart
```

y a continuación ir a un navegador para probar el funcionamiento. Observa que el navegador nos pedirá que confiemos en el sitio web al que vamos a acceder indicando por quien está firmado el certificado. Evidentemente tendremos que confiar ya que la entidad de certificación somos nosotros mismos. El resultado será algo similar a lo mostrado en la figura 11.3.

Figura 11.3. Resultado tras la invocación del sitio creado

Ejercicio práctico n.º 2

Configuración de Apache para que sirva sitios web seguros y no seguros. Crea otro host virtual en el servidor que de servicio al sitio **www.webnormal.cossio.com**. Este sitio utilizará protocolo **http** y será servido junto al sitio **www.websegura.com** configurado en el ejercicio anterior.

Solución ejercicio n.º 2

Al igual que en el ejercicio anterior debe configurarse un DNS para que resuelva el nombre **www.webnormal.com**.

A continuación creamos un nuevo **index.html** similar al anterior que muestre por pantalla, por ejemplo, el texto *Funcionamiento de la webnormal*.

Posteriormente modificaremos el archivo **httpd.conf** para que tenga el siguiente aspecto:

```
Listen 443
    ...
#Otras líneas de configuración del archivo
    ...
ServerName www.websegura.com:443
    ...
#Otras líneas de configuración del archivo
    ...
NameVirtualHost 192.168.10.150:80

Listen 80

<VirtualHost 192.168.10.150:80>
 DocumentRoot "/var/www/html/webnormal"
 ServerName www.webnormal.com
 </VirtualHost>

NameVirtualHost 192.168.10.150:443

 <VirtualHost 192.168.10.150:443>
    ...
#Anteriores líneas de configuración del sitio seguro
    ...
</VirtualHost>
```

Una vez reiniciado el servicio se pueden invocar ambos sitios web obteniéndose resultados similares a los mostrados en las figuras 11.3 y 11.4.

Figura 11.4. Resultado tras la llamada a ***www.webnormal.com***

Ejercicio práctico n.º 3

Configura un sitio web seguro adicional en tu servidor. Este sitio debe ser creado para realizar una inscripción segura (transmisión de datos seguros) al mayor evento informático que se da en La Rioja: Rioj@party (visita la página **www.riojaparty.com**).

Debe crearse un formulario donde cada participante de la Party pueda introducir su nombre, dni, email, sexo, edad y comunidad autónoma de procedencia. Estos datos son obligatorios porque así la organización puede realizar estadísticas de participación.

La organización del evento dispensa carnés con fotografía y el apodo (nick) del participante, será necesario que los participantes puedan indicar este dato, así como subir una foto durante el proceso de inscripción de forma opcional. Para evitar problemas de almacenamiento o colapsado del sistema el tamaño máximo de la fotografía será 100kbytes.

Estos datos serán enviados al servidor y se procesarán a través de un programa escrito en Python. El procesamiento consistirá en guardar los datos en una base de datos MySQL.

¡¡Advertencia!! Para realizar el ejercicio define un nombre de dominio cualificado **inscripcion.riojaparty.es** (de esta forma evitaremos problemas de resolución de nombres, ya que la página verdadera tiene extensión **.com**)

Solución ejercicio n.º 3

Puesto que es Python quien procesa los datos del formulario será necesario tener instalado Python en nuestro sistema. De la misma forma, como los datos deben ser almacenados en una base de datos MySQL, este servicio también deberá estar instalado. Si utilizas Linux seguramente tu distribución llevará ambas cosas incluidas y no tendrás que hacer nada. Si utilizas Windows, la mejor opción es instalar Xampp (bajo el modo instalación; en ningún caso utilizar la opción de descompresión en un directorio sin instalación).

Para poder almacenar los datos de inscripción es necesaria una base de datos. Vamos a suponer que la base de datos se llama **inscrip_riojaparty** que posee una tabla llamada **participantes**, accesible por el usuario **control** con contraseña **riojaparty**. Esto lo puedes hacer desde la línea de comandos o a través de phpMyAdmin, tal y como se ha explicado en otros capítulos. La estructura puede ser la mostrada en la figura 11.5.

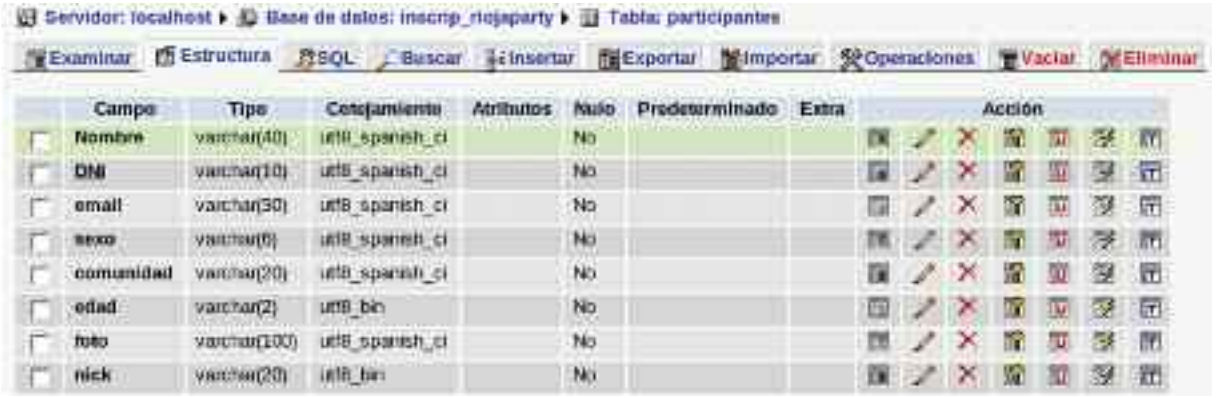


*Figura 11.5. Estructura de la tabla **participantes** perteneciente a la base de datos **inscrip_riojaparty***

Una vez que tengamos Apache será necesario **mod_python** para poder utilizar programas Python junto al servidor web. La instalación de este módulo ya fue explicada en el capítulo dedicado al mismo, por lo que no se volverá a explicar.

Para que Python pueda trabajar con las bases de datos de MySQL es necesario instalar el paquete MySQLdb. Si utilizas Linux, tu distribución contendrá un .rpm, .deb o similar para ser instalado a través de **rpm**, **apt**, **aptitude**, **synaptic**, etc. Por ejemplo en Mandriva basta con escribir:

```
[root@directorio]# urpmi python-mysql
```

(En debian y ubuntu el paquete se llama python-mysqldb.)

Si utilizas Windows debes ir a la página **http://sourceforge.net/projects/mysql-python** y pulsar en la pestaña download, la cual te conducirá a otra página de la que te podrás descargar los **.exe** para realizar la instalación. Evidentemente debes tener instalado antes Python (que puede ser descargado de **www.python.org**).

Suponiendo que todo está instalado correctamente el primer paso que daremos será crear el formulario que mande los datos a un programa Python a través de **mod_python**. Como se vio en el capítulo -X-, una opción podría ser utilizar PSP para tratar el contenido dinámico, de forma similar a como puede hacerlo PHP. El contenido básico del archivo **formulario.psp** podría ser:

```
<html>
  <h2> <font color="#800000">Inscripción a Rioj@party </font></h2>
  <form method="post" action="programa python" enctype="multipart/form-data">
    <fieldset><legend>Datos a rellenar obligatoriamente</legend>
    <center><table ><tr>
      <td align="right"><font color="#800000"> <b>Nombre:</b></font></td>
      <td align="right"> <input type=text name="nombre"></td>
      <td align="right"><font color="#800000"> <b>DNI (sin letra):</b></font></td>
      <td align="right"> <input type=text name="dni"></td></tr><tr>
      <td align="right"><font color="#800000"> <b>Email:</b></font></td>
      <td align="right"><input type=text name="correo"></td>
      <td align="right"><font color="#800000"> <b>Sexo:</b></font></td>
        <td align="center">
        hombre <input type="radio" name="sexo" value="hombre"/>
        mujer<input type="radio" name="sexo" value="mujer"/></td></tr><tr>
                    <td align="right"><font color="#800000"> <b>Comunidad de
procedencia:</b></font></td>
      <td align="center"><select name="comunidad">
                        <option>—selecciona—</option> <option>Andalucía</option>
<option>Aragón</option>
        <option>Asturias</option><option>Baleares</option> <option>Canarias</option>
        <option>Cantabria</option> <option>Castilla y León</option>
        <option>Castilla La Mancha</option> <option>Catuluña</option>
       <option>Extremadura</option> <option>Galicia</option> <option>La Rioja</option>
        <option>Madrid</option> <option>Murcia</option> <option>Navarra</option>
        <option>País Vasco</option> <option>Comunidad Valenciana</option>
      </select></td>
      <td align="right"><font color="#800000"> <b>Año de nacimiento:</b></font></td>
      <td align="center"> <select name="nacimiento">
        <option>—selecciona—</option>
        <%
```

```
        import datetime
        year=datetime.date.today().timetuple()[0]
        opt="<option>%d</option>" %(year-1)
        for i in range(2,65):
            opt=opt+("<option>%d</option>" %(year-i))
        #fin del código
        %>
        <%=opt%>
    </select> </td> </tr> </table> </center> </fieldset>
<fieldset class="imagen">
<legend>Datos de incripción opcionales: Nick y Fotografía (máx. 100kB)  </legend>
    <center>
   <font color="#800000"><b>Escribe tu Nick:</b></font><input type=text name="nick">
    <font color="#800000"> <b>Busca la imagen y súbela:</b></font>
    <input type="file" id="foto" name="imagen" />
    </center>
</fieldset><center>
<input type=submit value="Inscribirme">
</center></form></html>
```

Observa que el nombre del programa que responderá a la solicitud enviada a través del formulario no está definido todavía y si pulsas el botón **Inscribirme** del formulario obtendrás un error. En azul también se ha indicado el código Python embebido. Recuerda que entre **<% %>** se coloca el codigo a ejecutar, que no tendrá ninguna repercusión sobre la salida, y entre **<%= %>** el código que provocará un vuelco sobre la pantalla del navegador. El código de este ejemplo únicamente hace las siguientes operaciones:

— Importa el paquete **datetime** para poder calcular el año en el que nos encontramos.

— En la variable **year** se almacena el año. **datetime.date.today()** devuelve la fecha actual y **timetuple()** una tupla con los valores de (año, mes, día, hora, minuto, segundo, día de la semana, día del año); por esa razón tomamos el primer elemento de la tupla **[0]**, para tomar el año.

— **opt** es una cadena de caracteres que se incrementa en tamaño, utilizando un bucle **for**, al sumarle continuamente otras cadenas de la forma **"<option>*año*</option>"**. Al finalizar el bucle la variable **opt** contendrá 65 porciones similares a la anterior. Cuando veas el formulario a través del servidor web, visualiza el código fuente para comprobar el texto que Python genera y que permite que el cliente web disponga desde el año actual hasta 65 años anteriores la posibilidad de elegir año de nacimiento (es de suponer que a la party no vayan personas mayores de 65 años).

— El comentario precedido del # (sostenido o sharp) indica a PSP cuando acaba el contenido del bucle, ya que en HTML no tienen sentido las indentaciones que Python necesita para funcionar.

— Finalmente **<%=opt%>** introduce dentro del código HTML la cadena **opt** antes calculada.

Suponiendo que la dirección IP de nuestro servidor es la 192.168.10.150 y los certificados creados anteriormente, para configurar una web sergura, el archivo **httpd.conf** debería contener el siguiente host virtual:

```
<VirtualHost 192.168.10.150:443>
DocumentRoot "/var/www/html/inscripcion/"
ServerName www.inscripcion.riojaparty.es:443
```

```
ServerAdmin webmaster@riojaparty.com
SSLEngine on
SSLCipherSuite RSA:+HIGH:+MEDIUM
SSLProtocol all -SSLv2
SSLCertificateFile /etc/httpd/conf/server/servercert.pem
SSLCertificateKeyFile /etc/httpd/conf/server/serverprivkey.pem
<Directory "/var/www/html/inscripcion/">
   AddHandler mod_python   .psp
   PythonHandler mod_python.psp
   PythonDebug on
   Allow from all
</Directory>
</VirtualHost>
```

¡¡Importante!! Recuerda que la directiva `PythonDebug on' es útil mientras se hacen las comprobaciones de diseño, pero que debe eliminarse cuando se está dando una utilidad real a la configuración. En otro caso podría trasladar al cliente web información clave para la seguridad del sitio.

En cualquier caso el uso de PSP como Python embebido en HTML ya fue explicado en el capítulo -X- y este puede ser un buen momento para explicar otra forma de utilizar la potencia de `mod_python' y PSP. Una de las mayores críticas que tienen PHP, ASP, JSP, ... es la mezcla de hipertexto con instrucciones de programación. Esto produce un código poco legible, que muchos diseñadores web tachan de una mala práctica de programación, como por ejemplo Gregory Trubetskoy, el creador de **mod_python**. Sin embargo, Trubetskoy también apunta que a pesar de ser una mala práctica, millones de usuarios PHP muestran una clara demanda por este estilo de programación, obteniendo sitios exitosos implementados bajo este estilo. Si utilizamos PSP de la misma forma estamos cayendo bajo la misma crítica, por esta razón PSP ofrece algo más: Debido a que PSP puede operar sin ser llamado directamente en la URL del navegador, es posible separar el código Python del código HTML. Veamos un ejemplo utilizando el código anterior.

En primer lugar, y puesto que no se van a hacer llamadas a ficheros **.psp** desde el navegador, la configuración de Apache debe cambiarse. En **handler** que utilizaremos será el **Publisher**. Con respecto a la anterior configuración, bastará con cambiar la información relativa a **<Directory "/var/www/html/inscripcion/">**:

```
<Directory "/var/www/html/inscripcion/">
   SetHandler mod_python
   PythonHandler mod_python.publisher
   PythonDebug on
   Allow from all
</Directory>
```

Utilizamos la directiva **SetHandler** en lugar de **AddHandler**, indicando que será **mod_python** el que manejará las direcciones indicadas en la URL, independientemente de la extensión, dentro de ese directorio.

Esto significa que la URL del navegador deberá hacer referencia a un programa Python, tal y como se vio en el capítulo -X-. Ese programa Python lo vamos a llamar por ejemplo **incripcion.py**. A su vez, **incripcion.py** llamará a un archivo plantilla, al que llamaremos por ejem-

plo **inscripcion.tmpl**[83]. El programa **.py** invocará, a través de PSP, al archivo **.tmpl** para construir el sitio web dinámico que se mostrará por pantalla. El sitio web se construirá combinando los datos calculados en el **.py** y la información HTML contenida en el **.tmpl**.

En la figura 11.6 se muestra un esquema de lo que ocurre cuando utilizamos **mod_python** de esta forma.

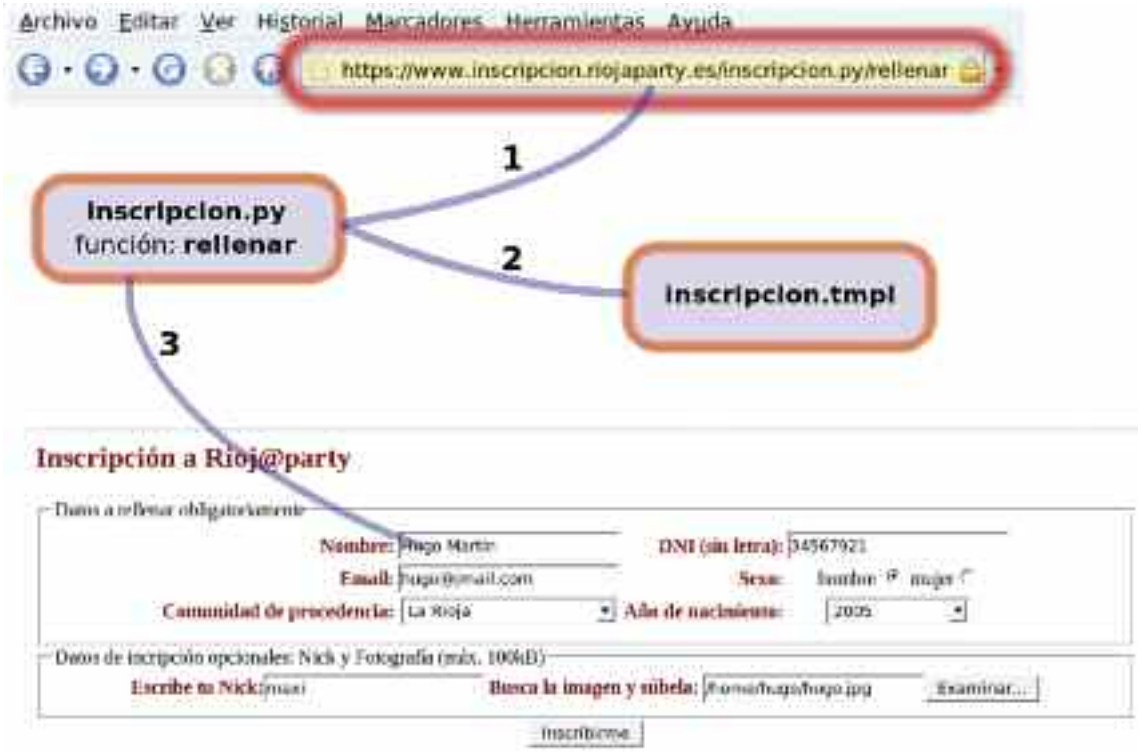


Figura 11.6. Secuencia de operaciones que se sucede para presentar el contenido web al cliente

En primer lugar escribimos la URL que apunta al método o función del programa **.py** que tiene la capacidad de mandar el contenido HTML al navegador. En el ejemplo mostrado en la figura esa función es rellenar[84]. La primera parte del contenido de **inscripcion.py** es:

```
#coding: UTF8
from mod_python import psp
from datetime import date
import os
import MySQLdb

rutafotos='var/www/html/inscripcion/fotos/'
year=date.today().timetuple()[0]

def rellenar(req):
    opt="<option>%d</option>" %(year-1)
```

83 La extensión tmpl se ha elegido arbitrariamente, como una simplificación de template (plantilla en inglés). Tú puedes escribir la extensión que desees.

84 Observa en la URL que el nombre de dominio es **www.inscripcion.riojaparty.es**. Dentro de ese sitio hay un programa llamado **inscripcion.py**. A su vez ese programa, como veremos más tarde, contiene un método o función denominada **rellenar**, que será la encargada de construir adecuadamente el contenido HTML que será mandado al cliente web.

```
    for i in range(2,65):
        opt=opt+("<option>%d</option>" %(year-i))

    tmpl=psp.PSP(req,filename='inscripcion.tmpl')
    tmpl.run(vars={'options':opt})
```

*Listado 10.1. Primera parte del programa **inscripcion.py** que construye, trata y responde al formulario de inscripción a Rioj@party*

La primera línea indica que se utiliza la codificación UTF8, esto es necesario para que las tildes y otros símbolos se impriman adecuadamente. Las siguientes líneas son para importar las librerías que va a necesitar nuestro programa. Evidentemente, se necesita la librería **psp** de **mod_python**. El programa debe calcular el año actual, tal y como se ha explicado anteriormente, por esa razón necesita la función **date** de la librería **datetime**. También se la librería **os** porque el programa utilizará la función **os.path.join** para concatenar la ruta del archivo que contiene la fotografía que subirá el cliente web que desea inscribirse en la party. En lugar de importar toda la librería (**import os**) se podría haber importado únicamente esa función escribiendo **from os.path import join**. Por último es necesario importar la librería **MySQLdb**, que será nuestro nexo de unión entre Python y MySQL.

Posteriormente se definen dos variables globales: **rutafotos** que apunta al directorio[85] en el que se almacenarán las fotos que vayan a subir aquellos que se inscriban y **year** que almacena el año actual y que es utilizada por varias funciones (métodos).

A continuación comienza la definición de la función **rellenar**. Esta se encarga de calcular los datos que necesita el formulario, para después procesarlos junto al código HTML y enviarlo al cliente web. Al ser invocada en el navegador lo primero que hace es escribir dentro de la variable **opt** una cadena de texto como por ejemplo la siguiente: **<option>2012</option> <option>2011</option> ... <option>1949</option>**, donde el primer número (2012 en este ejemplo) es el año actual menos 1. Por tanto si obtenemos la anterior cadena de texto es porque estamos en el año 2013. El código es el mismo que se presentó al describir el archivo **formulario.psp** al comienzo del ejercicio 3, salvo que ahora no está embebido en el código HTML y por tanto no necesita de los delimitadores **<% %>**. Lo interesante aquí es que a continuación, y a través de **psp**, se llama al fichero **inscripcion.tmpl** convirtiéndolo en el objeto **tmpl**. Este archivo necesita de una variable llamada **options** para funcionar correctamente como documento HTML, y el contenido de esa variable no es sino la cadena de texto **opt** calculada anteriormente. A través del método **run** y el diccionario **vars** confeccionado con los datos necesarios para que la plantilla **inscripcion.tmpl** quede completamente construida se procesa la información y se convierte en código HTML que se envía al navegador.

El contenido de **inscripcion.tmpl** es:

```
<html>
  <meta http-equiv="content-type" content="text/html;charset=utf-8" />
  <h2> <font color="#800000">Inscripción a Rioj@party </font></h2>
  <form method="post" action="respuesta" enctype="multipart/form-data">
    <fieldset><legend>Datos a rellenar obligatoriamente</legend>
    <center><table ><tr>
      <td align="right"><font color="#800000"> <b>Nombre:</b></font></td>
```

85 Observa que ese directorio ya debe de estar creado. El programa lo utiliza para dejar archivos, pero en ningún caso lo creará.

```
<td align="right"> <input type=text name="nombre"></td>
<td align="right"><font color="#800000"> <b>DNI (sin letra):</b></font></td>
<td align="right"> <input type=text name="dni"></td></tr><tr>
<td align="right"><font color="#800000"> <b>Email:</b></font></td>
<td align="right"><input type=text name="correo"></td>
<td align="right"><font color="#800000"> <b>Sexo:</b></font></td>
<td align="center">
hombre <input type="radio" name="sexo" value="hombre"/>
mujer<input type="radio" name="sexo" value="mujer"/></td></tr><tr>
<td align="right"><font color="#800000"> <b>Comunidad de procedencia:</b></font></td>
<td align="center"><select name="comunidad">
<option>—selecciona—</option> <option>Andalucía</option> <option>Aragón</option>
<option>Asturias</option><option>Baleares</option> <option>Canarias</option>
<option>Cantabria</option> <option>Castilla y León</option>
<option>Castilla La Mancha</option> <option>Catuluña</option>
<option>Extremadura</option> <option>Galicia</option> <option>La Rioja</option>
<option>Madrid</option> <option>Murcia</option> <option>Navarra</option>
<option>País Vasco</option> <option>Comunidad Valenciana</option>
</select></td>
<td align="right"><font color="#800000"> <b>Año de nacimiento:</b></font></td>
<td align="center"> <select name="nacimiento">
<option>—selecciona—</option>
<%=options%>
</select> </td> </tr> </table> </center> </fieldset>
<fieldset class="imagen">
<legend>Datos de incripción opcionales: Nick y Fotografía (máx. 100kB) </legend>
<center>
<font color="#800000"><b>Escribe tu Nick:</b></font><input type=text name="nick">
<font color="#800000"> <b>Busca la imagen y súbela:</b></font>
<input type="file" id="foto" name="imagen" />
</center>
</fieldset><center>
<input type=submit value="Inscribirme">
</center></form></html>
```

Listado 10.2. Contenido de la plantilla procesada con PSP

Observa que prácticamente todo el código es HTML. Ciertamente la única necesidad PSP que tiene es <%=options%>, y que acabamos de ver como se calcula. Otro aspecto importante a señalar es la función encargada de procesar el formulario: **action="respuesta"**. Puesto que la página web es llamada a través de **inscripcion.py**, la función (o método) **respuesta** pertenece a ese programa, tal y como se muestra en la segunda parte del listado de este programa:

```
def respuesta(req):
    if not (req.form['nombre'] and req.form['correo'] and req.form['dni'] and req.form['sexo'] and
req.form['comunidad'] and req.form['nacimiento']):
        return "Debes completar todos los campos"

    edad=year-int(req.form['nacimiento'])

    if req.form['imagen'].value!='':
        foto=req.form['imagen'].file.read(100000)
        nombrefichero=os.path.join(rutafotos,req.form['imagen'].filename)
        ficherofoto=open(nombrefichero,'w')
```

```
        ficherofoto.write(foto)
        ficherofoto.close()
    else:
        nombrefichero=''

  conn=MySQLdb.connect(host='localhost',user='control',passwd='riojaparty',db='inscrip_rioja-
party')
    cursor=conn.cursor()
    cursor.execute("SET NAMES 'utf8'")
    cursor.execute("""INSERT INTO participantes (Nombre, DNI, email, sexo, comunidad, edad,
foto, nick) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)""",(req.form['nombre'], req.form['dni'],
req.form['correo'], req.form['sexo'], req.form['comunidad'], edad,
nombrefichero,req.form['nick']))
    cursor.close()
    conn.close()

    tmpl=psp.PSP(req,filename='feedback.tmpl')
    tmpl.run(vars={'inscrito':req.form['nombre']})
```

*Listado 10.3. Segunda parte del programa **inscripcion.py** que construye, trata y responde al formulario de inscripción a Rioj@party*

La única entrada de la función **respuesta** es el objeto **req**. Él es el único dato/objeto que llega del formulario. Podría haber sido definida como:

```
def respuesta(req, nombre, correo, dni, sexo, comunidad, nacimiento, foto, nick) :
```

para recibir en cada una de las variables el contenido de cada uno de los **<input type=... name=...>** (contenido asociado a `name') mandados al servidor, tal y como se mostró en el ejemplo del capítulo -X-. Sin embargo, en este programa se va a mostrar el uso de la variable global **form**. Esta variable queda definida en forma de diccionario[86] en tiempo de ejecución de PSP y contiene todos los datos transmitidos por el formulario.

Así la primera línea de código de dicho método consiste en comprobar si el cliente que desea inscribirse a completado los campos obligatorios del formulario: nombre, correo, dni, sexo, comunidad y nacimiento. A estas variables se accede a través de la estructura req.form['...'], así el **if** comprueba si **req.form['nombre']**, **req.form['correo']**, **req.form['dni']**, **req.form['sexo']**, **req.form['comunidad']** y **req.form['nacimiento'])** contienen información y en caso de que alguno no la tenga devuelve un mensaje por pantalla indicando que todos esos campos deben completarse.

Todos los campos contienen la información requerida para ser almacenada, salvo los correspondientes a año de nacimiento y fotografía subida al servidor. En el primero de los casos lo que deseamos es la edad, que puede se calcula fácilmente a través de la expresión **edad=year-int(req.form['nacimiento'])**. En el segundo de los casos la operación es un poco más compleja.

86 Un diccionario Python es una especie de array al que accedemos a través de nombres. A un array normal accedemos a los datos almacenados en distintas posiciones utilizando números, por ejemplo **A[2]** o **A[5]**. Si **A** fuera el diccionario **A={'edad':7, 'nombre':'Ariadna'}**, accederíamos a los datos escribiendo **A['edad']** lo que devolvería 7 o **A['nombre']** que devolvería Ariadna.

En la base de datos almacenaremos la dirección y nombre del fichero (foto) subido al servidor por la persona que desea inscribirse, no el contenido del archivo. El archivo es almacenado en el directorio **var/www/html/inscripcion/fotos/**[87], este directorio fue definido como la variable global **ruta** al principio del programa.

El condicional **if req.form['imagen'].value != '':** comprueba si el formulario ha enviado un fichero. En el caso de que el contenido de **req.form['imagen'].value** no este vacío ejecutará las instrucciones contenidas en él. En primer lugar la instrucción:

```
foto=req.form['imagen'].file.read(100000)
```

almacena en la variable **foto** los primeros 100000 bytes del archivo mandado por el formulario, ya que este era el máximo impuesto en el ejercicio 3. Si el archivo mandado tuviera un tamaño menor, evidentemente sólo se almacena la cantidad de bytes que posea.

La siguiente instrucción guarda en la variable **nombrefichero** la ruta y nombre completo del lugar donde se almacena. El comando **os.path.join** se encarga de concatenar la ruta con el nombre del fichero. Por ejemplo si el nombre del fichero subido es **carol.jpg** (que es almacenado en el campo **filename** del objeto **eq.form['imagen']** enviado por el formulario) la variable **nombrefichero** sería: **var/www/html/inscripcion/fotos/carol.jpg**.

A continuación se abre un fichero nuevo llamado **ficherofoto** con posibilidad de escritura, opción **w**. Sobre él se guarda el contenido del fichero leído del formulario y finalmente se cierra. Esta es la labor que desarrollan las tres líneas codigo:

```
ficherofoto=open(nombrefichero,'w')
ficherofoto.write(foto)
ficherofoto.close()
```

El siguiente paso consiste en almacenar en la base de datos. El primer paso es abrir una conexión a la base de datos **inscrip_riojaparty**. Posteriormente, invocando al método **cursor()**, se crea un cursor a través del cual introducir las órdenes y procesar los estamentos SQL:

```
conn=MySQLdb.connect(host='localhost',user='control',passwd='riojaparty',db='inscrip_riojaparty')
cursor=conn.cursor()
```

Ahora ya es posible trabajar con la base de datos a través del método **execute()** del objeto **cursor** que hemos creado. Con el fin de que las tildes y símbolos se almacenen correctamente en la base de datos activamos el almacenado de texto con codificación UTF8.

```
cursor.execute("SET NAMES 'utf8'")
cursor.execute(""INSERT INTO participantes (Nombre, DNI, email, sexo, comunidad, edad, foto, nick) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)"",(req.form['nombre'], req.form['dni'], req.form['correo'], req.form['sexo'], req.form['comunidad'], edad, nombrefichero,req.form['nick']))
cursor.close()
conn.close()
```

87 Si utilizas Windows aquí habrás tenido que definir una ruta compatible con dicho sistema operativo.

A continuación almacenamos en la base de datos los datos proporcionados por el formulario y los calculados en el programa (por ejemplo la edad). Observa que todos los datos se almacenan en la base de datos como cadenas de texto, **%s**. La forma de hacerlo es:

```
cursor.execute(“”“INSERT INTO /it{tabla} (campo-1, campo-2, ..., campo-n) VALUES (%s,
%s, ...,  %s)”“”,(valor-campo-1, valor-campo-2, ..., valor-campo-n))
```

Finalmente se cierra el cursor y la conexión a la base de datos.

Al igual que en la función **rellenar**, a continuación con los datos obtenidos de la programación podemos configurar un página web procesada a través de PSP. En este caso se toma la plantilla **feedback.tmpl** que deseará imprimir en el conjunto HTML la variable **inscrito**. Esto es hecho a través de las instrucciones:

```
tmpl=psp.PSP(req,filename='feedback.tmpl')
tmpl.run(vars={'inscrito':req.form['nombre']})
```

El contenido de feedback.tmpl es:

```
<html>
<meta http-equiv=”content-type” content=”text/html;charset=utf-8” />
<h2> <font color=”#800000”>Inscripción a Rioj@party </font></h2>
<h3> <center> Enhorabuena <%=inscrito%>! Te has inscrito correctamente en Rioj@party
2009.</center></h3>
Nos vemos los días 19, 20 y 21 de Junio en Haro.
</html>
```

El proceso de inscripción presenta finalmente dos partes. La primera aquella en la que alguien se inscribe y la segunda es cuando el sistema responde diciendo que la inscripción es correcta. Cuando pruebes el funcionamiento de la solución propuesta deberías obtener algo similar a lo mostrado en la figura 11.7.

Figura 11.7. Inscripción y feedback del sistema tras la misma

Notas finales

El ejercicio 3 estaría resuelto, pero para la utilización de una forma real todavía habría que resolver algunos problemas que se quedan en manos del lector:

— Puedes añadir una página intermedia entre el formulario y el feedback que muestre a usuario que acaba de inscribirse que los datos recogidos por el formulario. Sería como una comprobación última antes de almacenarlos en la base de datos.

— Sería atractivo introducir animaciones de algún tipo en la página, que la hicieran más atractiva.

— Sería interesante añadir una función, como la presentada en el capítulo -X-, para mandar un correo electrónico a cada participante correctamente inscrito indicando los datos que ha introducido.

— Cuando se introduce una dirección de correo electrónico incorrecta (que no contiene un nombre cualificado de dominio) el sistema devuelve un error que debería tratarse a través de excepciones Python para producir un comportamiento hacia el cliente web mucho mejor.

— Cuando se introduce un DNI que ya está almacenado en la base datos, el sistema devuelve un error, ya que este campo no puede estar repetido (es un campo primario). Esto también podría solucionarse manejando excepciones.

— Tras las inscripciones sería interesante hacer un programa Python que accediera al contenido de la base de datos para generar estadísticas.

Capítulo 12

PUBLICACIÓN DE CONTENIDOS EN INTERNET

Una vez creada toda la estructura del sitio Web, el último paso es publicar los contenidos en Internet. En los capítulos anteriores hemos visto cómo diseñar nuestro sitio Web y publicarlo, tanto localmente, 127.0.0.1, como dentro de nuestra Intranet (por ejemplo, 192.168.1.0/24), pero no cómo hacerlo para que sea accesible desde cualquier ordenador que tenga una conexión a Internet. Para ello tenemos dos posibilidades:

Figura 12.1. Esquema simplificado de conexión entre clientes y servidor en Internet

1. **Contratar los servicios de un Hosting público**. Es la solución más cómoda y adoptada por usuarios anónimos, y la mayor parte de las pequeñas y medianas empresas (PyMEs). Lo único que tenemos que hacer es subir los documentos Web que componen el sitio Web que haya sido diseñado previamente, normalmente vía FTP, al equipo servidor del Hosting, y llevar a cabo su gestión y actualización. De esta forma, nos desentendemos de aspectos tales como el de contratar los servicios de un proveedor de servicios de Internet para la obtención de una dirección IP pública, dar de alta el nombre de dominio a asignar al sitio Web (por ejemplo, www.escarbapedal.com), la configuración del equipo servidor con todos sus servicios (por ejemplo, HTTP, HTTPS, FTP, SMTP, POP3, etc.), su mantenimiento y la gestión de su seguridad, o el pago de la factura eléctrica correspondiente a tener el equipo servidor 24 horas al día encendido. Los hay gratuitos y de pago, caracterizándose los primeros por sus bajas prestaciones en relación a la capacidad de almacenaje disponible y el reducido ancho de banda, además de la habitual introducción de propaganda (banners) en las páginas de nuestro sitio Web, aspectos que pueden solventarse pagando una cuota en los segundos.

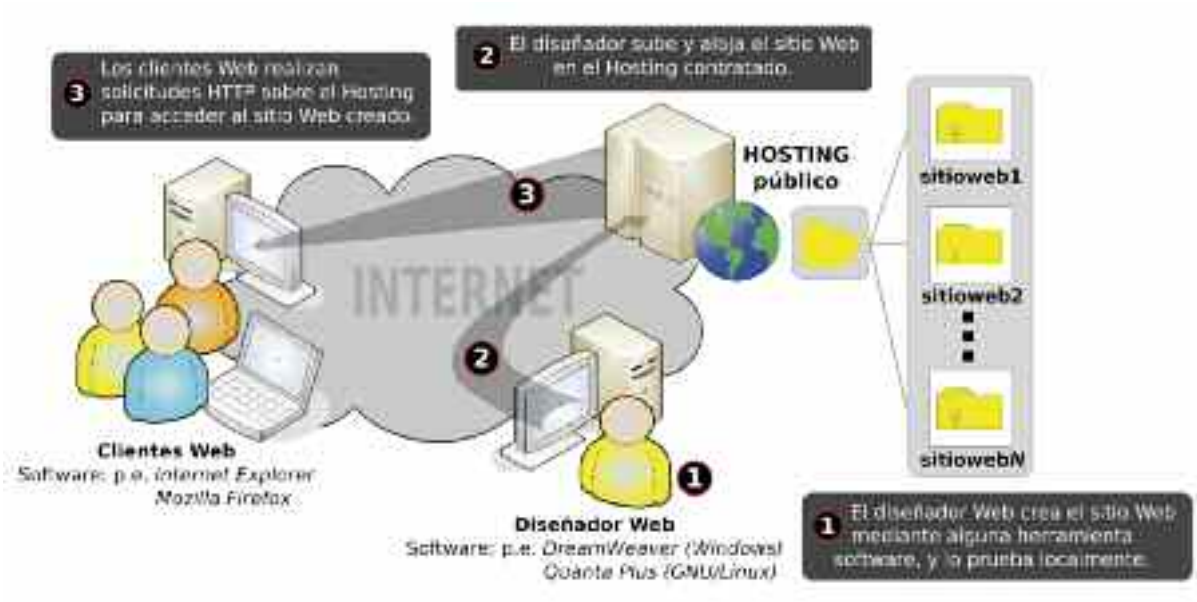


Figura 12.2. Publicación de contenidos a través de un Hosting público

2. Dar servicio Web desde nuestra Intranet. Se recomienda esta opción cuando queramos tener control total sobre los servicios que ofrecemos sin tener que depender de la calidad de servicio de una entidad externa.

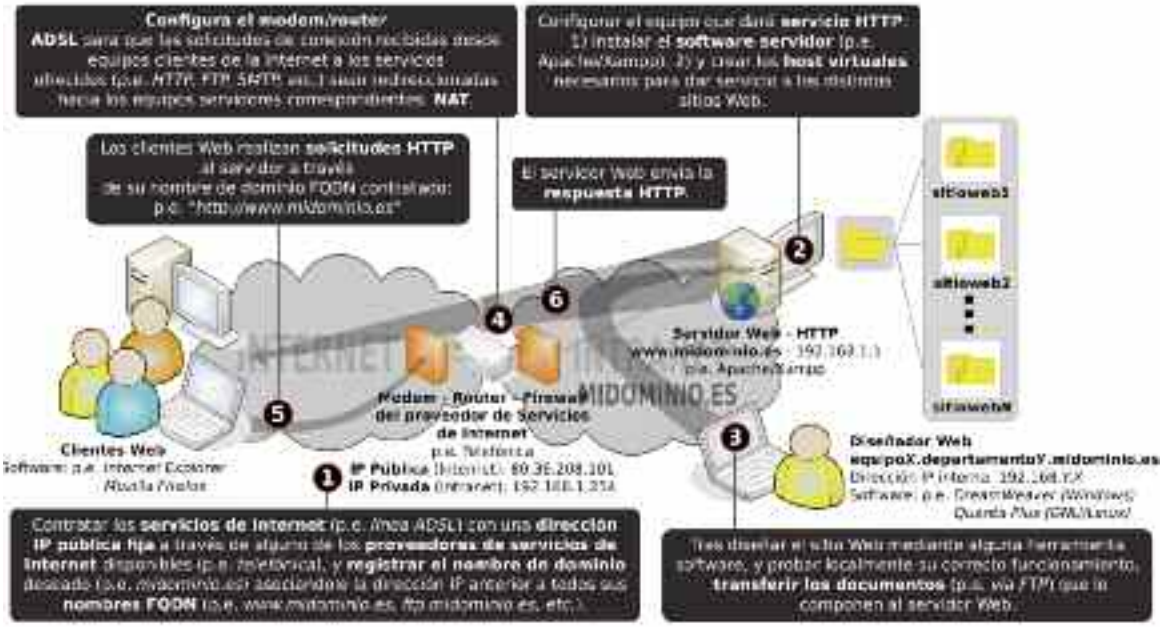


Figura 12.3. Publicación de contenidos Web desde nuestra propia Intranet

Como puede observarse en la figura X, el ejemplo más básico de esta situación se corresponde con el servicio HTTP que puede ser ofrecido por un usuario de su Intranet tras contratar una linea ADSL y registrar un nombre de dominio. Los pasos para su implementación podrían resumirse de la siguiente forma:

Paso n.º 1. El usuario o entidad interesada en ofrecer servicio HTTP de manera autónoma se pone en contacto con alguno de los proveedores de servicios de Internet existentes (por ejemplo, *Telefónica, ONO, Jazztel, etc.*) para contratar una nueva línea ADSL con una dirección IP pública fija (por ejemplo,

80.36.208.101), y registrar un nombre de dominio (por ejemplo, *midominio.com*)[88]. Frente a una dirección IP dinámica, la adquisición de una dirección fija facilita su asociación a los nombres de dominio cualificados (*FQDN, Full Qualified Domain Name*) que se definan: *www.midominio.es*, *ftp.midominio.es*, etc.

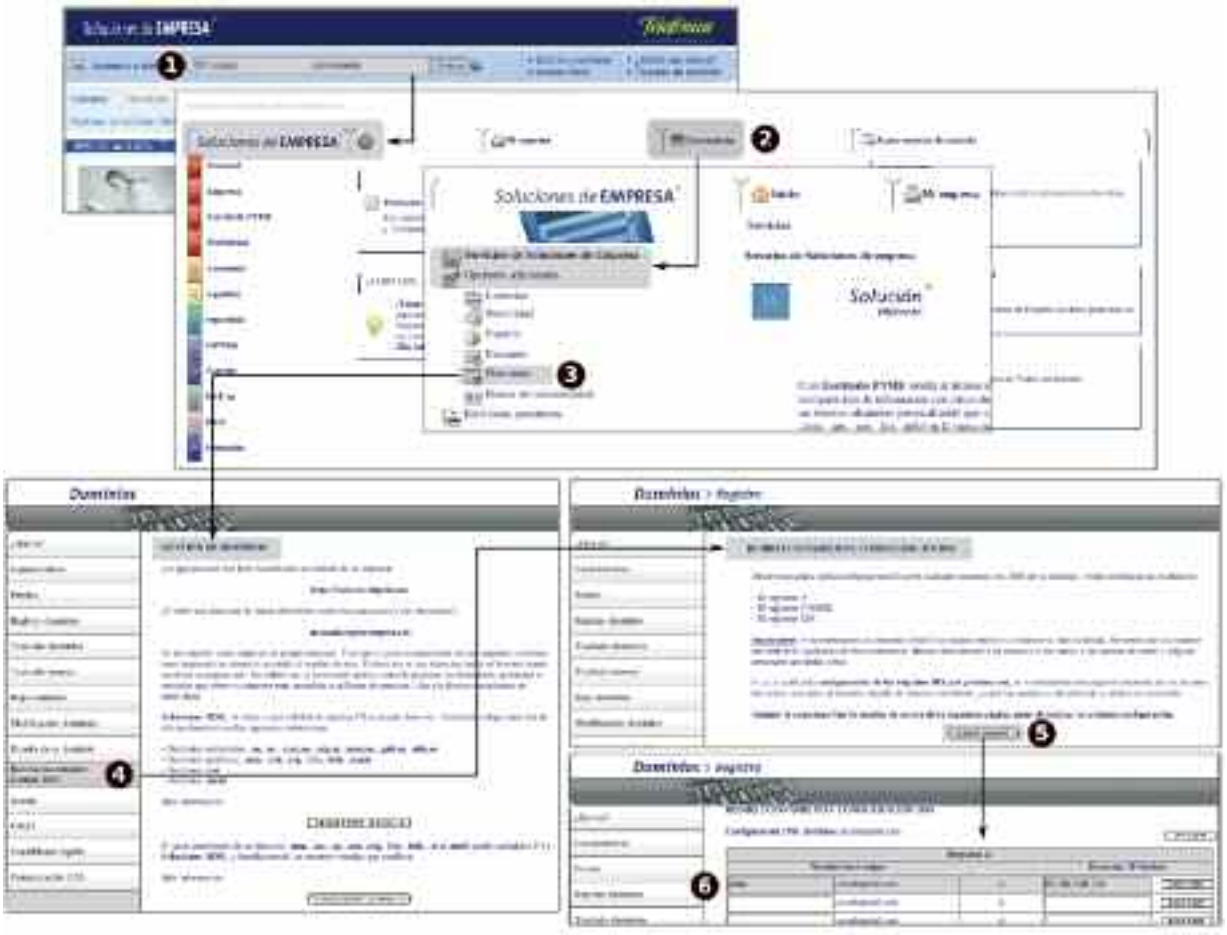


Figura 12.4. Gestión de dominios a través de "Soluciones de Empresa" de Telefónica: 1) Tras adquirir una línea ADSL con Telefónica ingresamos en su portal Soluciones de Empresa, 2) accedemos a su sección de contratación, 3) opción dominios, 4) desde donde podremos registrar un nuevo nombre de dominio, 5) gestionar su configuración, y 6) crear los nombres FQDN necesarios

En la figura 12.4 se muestra como gestionar un servicio de resolución de nombres de dominio (*DNS, Domain Name Server*) contratado a través de alguna de las empresas del sector (por ejemplo, *Soluciones de Empresa de Telefónica*), haciendo uso de los registros de dirección **A** (*Address*, por ejemplo, *www.midominio.es A 80.36.208.101*), registros de asociación de alias de nombres cualificados **CNAME** (*Canonical Name*, por ejemplo, *web.midominio.es CNAME www.midominio.es*), y registros asociados a cuentas de email del dominio **MX** (*Mail Exchanger*, por ejemplo, *mail.midominio.es MX 10 80.36.208.101*)[89].

Paso n.º 2. El usuario o entidad configura el equipo dentro de la Intranet que se encargará de dar el el servicio Web HTTP: 1) por ejemplo, instalación del software de sistemas GNU/Linux Mandriva, 2) instalación del software servidor Web Apache, 3) y configuración de los hosts virtuales encargados de dar servicio a los distintos sitios Web que se desee alojar en el servidor.

---

88 La mayor parte de los proveedores de servicios de Internet ofrecen al cliente la posibilidad de registrar un nombre de dominio. En caso contrario será necesario ponerse en contacto con un registrador competente; a través de Internet podemos encontrar multitud de ellas.

89 Para poder profundizar más en el servicio DNS, se remite al lector al libro de *Instalación y mantenimiento de Servicios de Internet* escrito por los mismos autores.

Paso n.º 3. El diseñador del sitio Web transfiere al equipo servidor los distintos documentos Web que lo componen (HTML, CSS, JS, PHP, etc.), por ejemplo, vía FTP, alojándo todo este contenido dentro de la carpeta raíz indicada en la configuración de Apache a través de su directiva *DocumentRoot*.

Paso n.º 4. Siguiendo el manual de instrucciones del modem/router ADSL suministrado por la empresa proveedora de servicios de Internet, se configura para redireccionar hacia el equipo servidor Web las solicitudes de conexión recibidas por el puerto HTTP/80[90] {nota pie de página: Mediante la directiva de configuración de Apache *Listen* indicamos las direcciones IP y puertos de comunicaciones del servidor a través del cual se escucharán solicitudes HTTP. Las peticiones recibidas por el modem/router por alguno de estos puertos deberán redireccionarse a la dirección IP del servidor.}. Este redireccionamiento se conoce como traducción de direcciones de red, NAT (*Network Address Translation*).

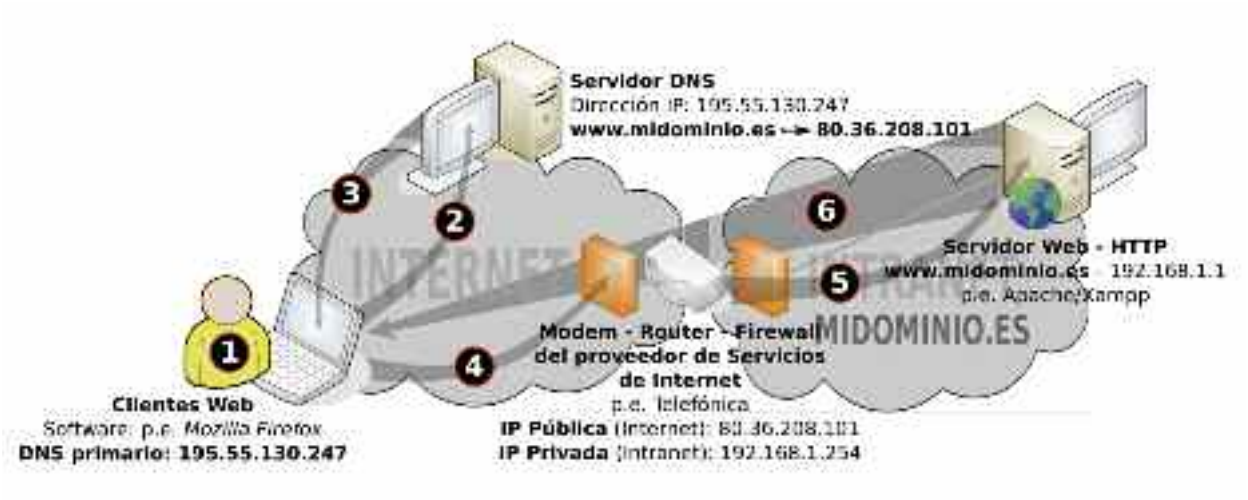


Figura 12.5. Comunicación HTTP cliente-servidor:
1) Un usuario realiza una solicitud HTTP a través de una aplicación cliente Web (por ejemplo, Mozilla Firefox), "http://www.midominio.es"; 2) el equipo cliente se dirige al servidor DNS primario que tenga configurado para conocer la dirección IP pública asociado al FQDN "www.dominio.es"; 3) el servidor DNS resuelve el FQDN devolviendo al equipo cliente la dirección IP asociada, "80.36.208.101"; 4) el equipo cliente realiza la petición HTTP al equipo con dirección IP "80.36.208.101"; 5) La petición es recogida por el modem/router y es redireccionada (NAT) al equipo servidor de la Intranet "192.168.1.1"; 6) Tras aceptar la solicitud de conexión el servidor Web, emite una respuesta HTTP, enviando la página de inicio del sitio Web.

Para comprender la importancia de la NAT, en la figura X se muestra de una forma detallada los pasos que se dan a partir del momento en que un usuario con conexión a Internet realiza una solicitud de conexión al servicio Web HTTP que estamos configurando[91].

A modo de ejemplo, en la figura 12.6 se muestra cómo configurar la NAT en un modem/router de la marca SMC, mediante la interfaz Web disponible para su gestión, *http://direcciónIP-router*.

90 Mediante la directiva de configuración de Apache Listen indicamos las direcciones IP y puertos de comunicaciones del servidor a través del cual se escucharán solicitudes HTTP. Las peticiones recibidas por el modem/router por alguno de estos puertos deberán redireccionarse a la dirección IP del servidor.

91 Para poder profundizar más en la traducción de direcciones IP, NAT, se remite al lector al libro de *Instalación y mantenimiento de Servicios de Internet* escrito por los mismos autores.

Figura 12.6. Configuración de la NAT:
1) Desde un equipo cliente de la Intranet hacemos una solicitud de conexión HTTP al modem-router, http://direcciónIP-router*; 2) Tras autenticarnos configuramos la NAT para redireccionar las solicitudes recibidas por el modem/router hacia equipos de la Intranet,* LAN IP Address.

Paso n.º 5. El usuario realiza una solicitud de conexión HTTP al servicio ofrecido, "*http://www.midominio.es*". Esta petición recibida por el modem/router es redireccionada hacia el servidor correspondiente.

Paso n.º 6. El servidor acepta la petición de conexión HTTP, y responde entregando el documento Web solicitado: por defecto, *index.html*, *index.php*... o el indicado a través de la directiva de configuración de Apache *DirectoryIndex*.

Si generalizáramos la situación anterior a una empresa u organización con una Intranet más compleja dividida en zonas privadas protegidas y desprotegidas (*demilitarizadas, DMZ*), los pasos a seguir para la publicación de contenidos en Internet de manera autónoma serían equivalentes, tal como se puede observar en la figura 12.7.

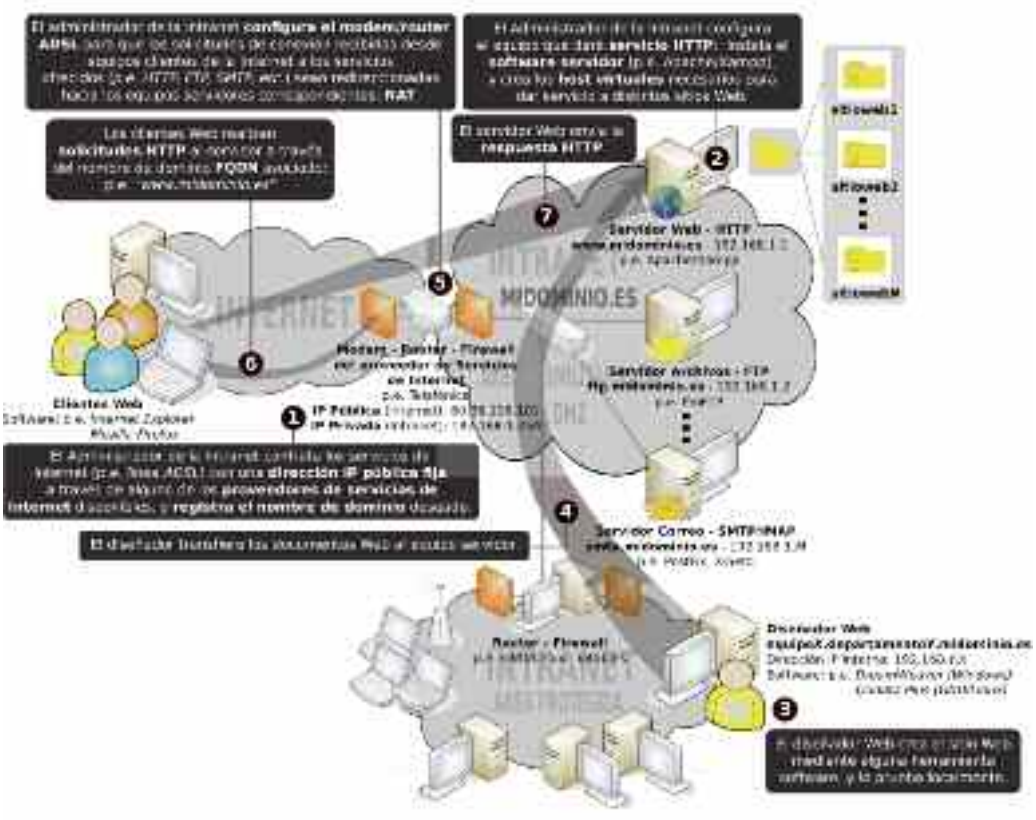


Figura 12.7. Publicación de contenidos en Internet desde la DMZ de una Intranet