1 Introducción a las aplicaciones web

OBJETIVOS DEL CAPÍTULO

- ✓ Comprender el esquema de funcionamiento de un servicio web.
- ✓ Conocer los lenguajes de marcas y el más utilizado de todos, el lenguaje HTML.
- ✓ Conocer los lenguajes de *script* de cliente y una introducción al más popular JavaScript.
- ✓ Presentar las limitaciones en cuanto a formato de presentación de HTML e introducir las hojas de estilo en cascada (CSS).
- ✓ Introducir los lenguajes de *script* de servidor, sus características y tipos.
- ✓ Ver algunas de las herramientas para el desarrollo web.
- ✓ Conocer la relación entre las páginas web y las bases de datos.

1.1 ESQUEMA DE FUNCIONAMIENTO DE UN SERVICIO WEB

Existen múltiples definiciones sobre lo que son los servicios web, lo que muestra su complejidad a la hora de dar una adecuada definición que englobe todo lo que son e implican. Podemos considerar los servicios web como un conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la Web. Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer unos servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web.

Estos servicios proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones, que interactúan entre sí para presentar información dinámica al usuario. Para proporcionar interoperabilidad y extensibilidad entre estas aplicaciones, y que al mismo tiempo sea posible su combinación para realizar operaciones complejas, es necesaria una arquitectura de referencia estándar.

Un servidor web es un programa que sirve para atender y responder a las diferentes peticiones de los navegadores, proporcionando los recursos que soliciten usando el protocolo HTTP o el protocolo HTTPS (la versión cifrada y autenticada). Un servidor web básico cuenta con un esquema de funcionamiento muy simple, basado en ejecutar infinitamente el siguiente bucle:

1. Espera peticiones en el puerto TCP.
2. Recibe una petición.
3. Busca el recurso.
4. Envía el recurso utilizando la misma conexión por la que recibió la petición.
5. Vuelve al primer punto.

Un servidor web que siga el esquema anterior cumplirá todos los requisitos básicos de los servidores HTTP, aunque solo podrá servir ficheros estáticos.

A partir del anterior esquema se han diseñado y desarrollado todos los servidores de HTTP que existen, variando solo el tipo de peticiones (páginas estáticas, CGI, *Servlets*, etc.) que pueden atender, en función de que sean o no sean multiproceso o multihilados, etc.

1.1.1 SERVICIO DE FICHEROS ESTÁTICOS

Todos los servidores web deben incluir, al menos, la capacidad para servir los ficheros estáticos que se hallen en alguna parte del disco. Un requisito básico es la capacidad de especificar qué parte del disco se servirá. No resulta recomendable que el programa servidor obligue a usar un directorio concreto, aunque sí puede tener uno por defecto.

La mayoría de servidores web permiten añadir otros directorios o subdirectorios para servir, especificando en qué punto del "sistema de ficheros" virtual del servidor se localizarán los recursos.

Algunos servidores web permiten también especificar directivas de seguridad (quién puede acceder a los recursos), mientras que otros hacen posible la especificación de los ficheros que se deben considerar como índice del directorio.

1.1.2 CONTENIDO DINÁMICO

Uno de los aspectos fundamentales del servidor web elegido es el nivel de soporte que ofrece para servir contenido dinámico. Puesto que la mayor parte del contenido web que se sirve no viene de páginas estáticas, sino que se genera de forma dinámica, y esta tendencia se mueve claramente al alza, el soporte para contenido de tipo dinámico que ofrece un servidor web es uno de los puntos críticos en la elección.

La mayor parte de los servidores web ofrecen soporte para CGI (se debe recordar que los CGI son el método más antiguo y sencillo para generar contenido dinámico). Otros muchos ofrecen soporte para algunos lenguajes de programación (normalmente lenguajes interpretados) como PHP, JSP, ASP, etc. Es muy recomendable que el servidor web que vayamos a utilizar proporcione soporte para algunos de estos lenguajes, especialmente PHP, sin tener en cuenta JSP, que normalmente requerirá un software externo para funcionar (como un contenedor de *Servlets*). La oferta es muy amplia, pero antes de elegir un lenguaje de programación de servidor se debe plantear si se desea un lenguaje muy estándar para que la aplicación no dependa de un servidor web o una arquitectura concreta o si, al contrario, la portabilidad no es prioritaria y sí lo es alguna otra prestación concreta que pueda ofrecer algún lenguaje de programación concreto.

1.1.3 SERVIDORES VIRTUALES

Una prestación que gana aceptación y usuarios rápidamente, muy especialmente entre los proveedores de servicios de Internet y las empresas de alojamiento de dominios, es la capacidad de algunos servidores web de facilitar múltiples dominios con una única dirección IP, discriminando entre los diferentes dominios alojados en función del nombre de dominio enviado en la cabecera HTTP. Esta prestación permite la administración racional y ahorradora de un bien escaso, las direcciones IP. Si se necesitan muchos nombres de servidor (porque proporcionamos alojamiento o por cualquier otro motivo) debemos asegurarnos de que el servidor web elegido ofrezca esta facilidad y que el soporte que ofrece para servidores virtuales permita una configuración distinta para cada servidor. Sería perfecto que cada servidor se comportara como si fuese un ordenador diferente.

1.1.4 PRESTACIONES EXTRA

Son muchas las prestaciones que ofrecen los diferentes servidores web para diferenciarse de la competencia. Algunas son realmente útiles y pueden decidir la elección de servidor. Hay que ser conscientes, sin embargo, de que si utilizamos algunas de estas características, o si éstas devienen imprescindibles, ello nos puede ligar a un determinado servidor web e imposibilitar una migración posterior.

1.2 LENGUAJES DE MARCAS

Un "lenguaje de marcado" o "lenguaje de marcas" se puede definir como una forma de codificar un documento donde, junto con el texto, se incorporan etiquetas, marcas o anotaciones con información adicional relativa a la estructura del texto, su presentación. El lenguaje de marcas más conocido en la actualidad es el HTML, que se utiliza en las páginas web.

Las marcas también están formadas de texto, pero que es interpretado cuando se muestra el documento, y suelen llamarse también etiquetas. Existen tres clases de lenguajes de marcas y pueden presentarse todas en un mismo documento.

- **Marcas de presentación**: estas marcas indican el formato-marco del texto. Su uso comienza a reducirse dado que es poco flexible, especialmente en grandes proyectos.
- **Marcas de procedimientos**: estas marcas se utilizan para la presentación del texto, interpretándose cada una en el orden en que aparecen. Por ejemplo, la marca que se agrega inmediatamente antes de un texto para que se vea en negrita. Luego debe existir la marca correspondiente que termine o cierre la negrita. Otras marcas de procedimientos pueden ser centrar texto, cambio de tamaño de fuente, cambios de estilos, etc. Algunos lenguajes de marcas de procedimiento son *nroff*, *troff*, *TeX*, *PostScript*, *HTML*, etc.

- **Marcas descriptivas**: también llamadas *marcado descriptivo* o *semántico*. Aquí se utilizan las marcas para describir fragmentos de texto sin especificar cómo deben representarse. Algunos lenguajes diseñados para esto son el *SGML* y el *XML*.

En los lenguajes de marcas descriptivas el formato está separado del contenido, permitiendo flexibilidad a la hora de reformatear un texto.

Algunos ejemplos de lenguajes de marcas son: *DocBook*, *Extensible HyperText Markup Language* (XHTML), *Extensible Markup Language* (XML), *Generalized Markup Language* (GML), *HyperText Markup Language* (HTML), *Lilypond* (sistema para notación musical), *Rich Text Format* (RTF), *TeX*, *LaTeX* (utilizado generalmente en matemáticas y publicaciones académicas), etc.

1.3 PÁGINAS WEB SENCILLAS CON HTML

HTML son las siglas de *HyperText Markup Language* (Lenguaje de Marcas de Hipertexto), es el lenguaje de marcado predominante para la construcción de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de "etiquetas" o "marcas", rodeadas por corchetes angulares (< >).

HTML fue desarrollado originalmente por Tim Berners-Lee mientras estaba en el CERN, y popularizado por el navegador Mosaic desarrollado en NCSA. Durante el transcurso de la década de 1990 proliferó con el crecimiento explosivo de la Web. Durante este tiempo, se añadieron etiquetas al lenguaje HTML. La web depende de los autores de páginas web y de que las compañías compartan las mismas convenciones de HTML. Esto ha motivado el trabajo conjunto sobre las especificaciones de HTML.

A las instrucciones que forman el lenguaje HTML se les llama elementos o etiquetas y siempre van encerrados entre < y >.

1.3.1 ESTRUCTURA DE UN DOCUMENTO HTML

Toda página HTML debe incluir las etiquetas <HTML> y </HTML>, estas etiquetas nos están indicando que el código contenido entre ellas va a ser HTML.

Los documentos escritos en HTML están estructurados en dos partes diferenciadas: la HEAD (cabecera) y el BODY (cuerpo).

HEAD. La parte <HEAD> es la primera de las dos partes en que se estructura un documento HTML.

En la zona del <HEAD> y </HEAD> reside información acerca del documento, y generalmente no se ve cuando se navega por él. En la zona <HEAD> y </HEAD> se pone el elemento <TITLE> </TITLE> que es una breve descripción que identifica el documento. Es lo que veremos como título de la ventana en los navegadores que lo permitan. Es como se conocerá nuestra página en algunos buscadores y en la agenda de direcciones (*bookmarks*) de los usuarios. Por tanto, parece importante pensarnos bien cómo llamarla. Este título aparecerá en la barra superior del navegador junto con el nombre de dicho navegador.

```
<HEAD>
<TITLE>
Head de un documento
</TITLE>
</HEAD>
```

Figura 1.1. *Head*

ACTIVIDADES 1.1

- Haga un título que sea personalizado para todas sus webs futuras, en dicho título debe aparecer su nombre y apellidos, además del curso. No olvide para todas las prácticas de este tema incorporar este título.

BODY. El cuerpo <BODY> </BODY> es la segunda y última de las dos partes en que se estructura un documento HTML. Esta parte al contrario que <HEAD> es obligatoria, ya que es aquí donde reside el verdadero contenido de la página. A continuación se presenta la primera página web en HTML.

```
<HTML>
<HEAD>
<TITLE> Mi primera pagina</TITLE>
</HEAD>
<BODY>
<CENTER>
<H1>
Mi Primera pagina
</H1>
</CENTER>
<P> Bienvenido a mi primera pagina Web
</BODY>
</HTML>
```

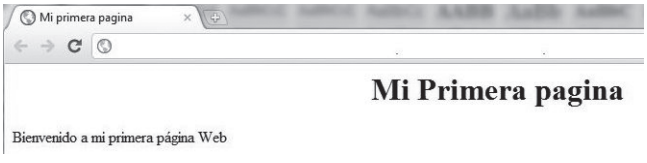

Figura 1.2. *Mi primera web*

Con las etiquetas <CENTER> y </CENTER> indicamos que el texto que está comprendido entre estas etiquetas irá centrado en la pantalla del navegador.

Las etiquetas <H1> y </H1> indican el tamaño del texto, en este caso cabecera 1 (el más grande de todos), que nos permitirá verlo más grande que el resto de elementos de la página web.

Por último la etiqueta <P> nos indica que el texto que viene a continuación es un párrafo normal y corriente, el típico texto que nos vamos a encontrar en cualquier página web.

ACTIVIDADES 1.2

- Haga su primera página web, en esta página aparecerá un texto centrado de tipo *cabecera1* con su nombre y apellidos, después pondrá un texto de tipo párrafo con el contenido "esta es mi tarea número 2". No olvide poner el título personalizado que realizó en la Actividad 1.1.

1.3.2 TAMAÑOS Y TIPOS DE LETRA EN HTML

Para definir distintos tamaños de letra, en HTML se utiliza el elemento <Hx> </Hx> donde *x* es un número que puede variar entre 1 y 6, siendo 1 el tamaño mayor. Podemos probar esto con el siguiente ejemplo:

```
<HTML>
<HEAD>
<TITLE> Pagina Web </TITLE>
</HEAD>
<BODY>
<H1>Encabezado H1</H1>
<H2>Encabezado H2</H2>
<H3>Encabezado H3</H3>
<H4>Encabezado H4</H4>
<H5>Encabezado H5</H5>
<H6>Encabezado H6</H6>
<P>Este es una parrafo normal para comparar con los encabezados
</BODY>
</HTML>
```

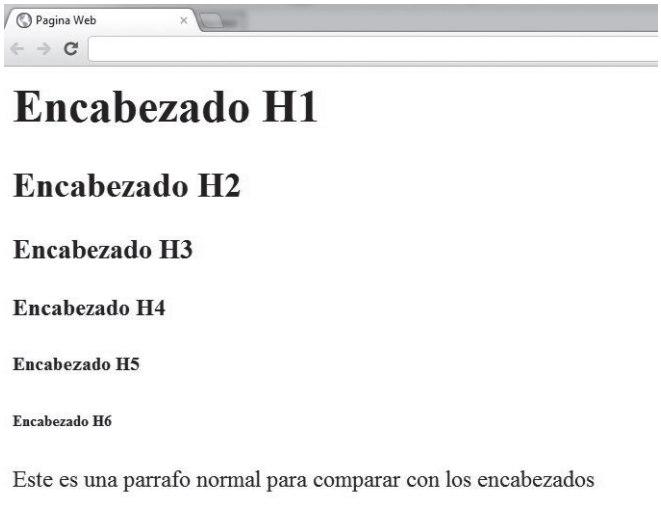

Figura 1.3. *Encabezados*

ACTIVIDADES 1.3

- Haga una página web donde aparezcan los encabezados vistos anteriormente pero en orden inverso, es decir, primero empezando por el encabezado 6 y terminando por el 1. El contenido del texto de los encabezados será su nombre y apellidos. La página terminará con un texto con formato párrafo que ponga "Aquí se acaba la página, hasta otra".

1.3.3 CAMBIOS DE PÁRRAFO Y DE LÍNEA. LÍNEA HORIZONTAL. DIVISIONES DE TEXTO

Para realizar cambios de párrafo (Intro) debemos utilizar
. Hay que tener en cuenta que cuando se cambia de un estilo a otro, se produce un salto de línea de manera automática, sin tener que especificar
.

Si queremos introducir líneas para separar unos contenidos de otros debemos utilizar <HR>, el funcionamiento de <HR> es similar al de
 (Intro) y funciona de manera parecida, ya que
 introduce un salto de línea y <HR> lo que hace es dar un salto de línea pero además incluyendo una línea separatoria, además, al igual que en el caso de
, tampoco existe </HR>. En el ejemplo siguiente se puede apreciar claramente las diferencias existentes entre
 y <HR>.

```
<HTML>
<HEAD>
<TITLE>web</TITLE>
</HEAD>
<BODY>
<H2> Encabezado 1 </H2>
<P>Primer parrafo debajo del encabezado.
<HR>
<H2> Encabezado 2 </H2>
<P>Segundo parrafo debajo del encabezado.
<BR>aqui seguira el resto de la pagina
</BODY>
</HTML>
```

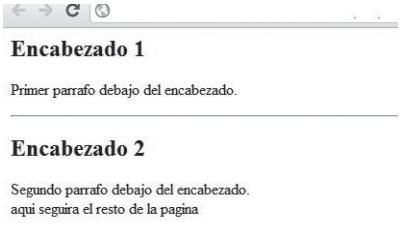

Figura 1.4. *
 y <HR>*

ACTIVIDADES 1.4

- Partiendo de la Actividad 1.3, añada una línea de separación entre cada cabecera distinta. Entre el contenido de las cabeceras y el del párrafo normal deben quedar 3 saltos (Intros).

1.3.4 IMÁGENES

Podemos insertar imágenes en una página web, que nos ayudarán a mejorar el aspecto visual de una página web. Para insertar una imagen es necesario insertar la etiqueta <IMG src='foto'>. Dicha etiqueta no necesita etiqueta de cierre, como sucede con
 y <HR>. Para insertar una imagen en una página web habrá que escribir:

```
<IMG SRC="imagen.gif">
```

Para trabajar de una forma más sencilla y ordenada, es recomendable que todas las imágenes se encuentren en un mismo directorio.

```
<HTML>
<HEAD>
<TITLE>  Web </TITLE>
</HEAD>
<BODY>
<H1>Web con fotos</H1>
<BR> <BR>
<IMG SRC="ordenador1.jpg" WIDTH="180">
<BR>
<IMG SRC="ordenador1.jpg" WIDTH="480">
</HTML>
```

Figura 1.5. *Fotos*

ACTIVIDADES 1.5

- Hacer una web que contenga 3 fotos. Las fotos pueden ser las que quiera el alumno/a.

1.3.5 ENLACES, VÍNCULOS O LINKS

Un enlace es una zona de texto o gráficos que si son cliqueados nos trasladan a otra página o a otra posición dentro de la página actual.

Para incorporar un enlace hay que utilizar la etiqueta <A HREF>. Todo lo que encerremos entre <A HREF> y </A>, ya sea texto o imágenes, será considerado como enlace y se visualizará de manera distinta en el navegador. El texto aparecerá subrayado y de un color distinto al habitual, y las imágenes estarán rodeadas por un borde del mismo color que el del texto del enlace. Al pulsar sobre el enlace, seremos enviados a la página que apuntaba el enlace. Para que el enlace sirva para algo debemos especificarle una dirección. Lo haremos de la siguiente manera:

```
<A HREF="dirección">pulsame </A>
```

Podemos referenciar a páginas que están dentro de mi sitio web o páginas que son externas a mi sitio web.

```
<HTML>
<HEAD>
<TITLE> Web </TITLE>
</HEAD>
<BODY>
<H1> Links </H1>
<BR>
<A HREF="http://www.google.es"> haz clic aqui para ir a google</A>
<BR> <BR>
<A HREF="o.html"><img src="uno.jpg"></A>
</BODY>
</HTML>
```

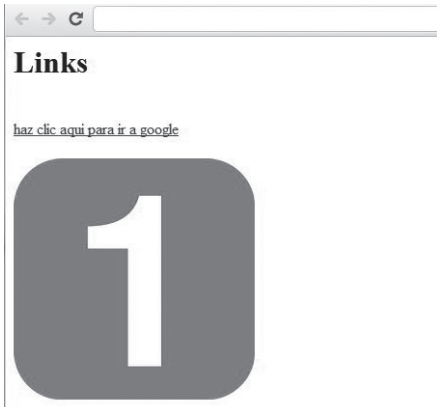

Figura 1.6. *Links*

ACTIVIDADES 1.6

- Hacer una web que enlace a 3 páginas distintas. El primer enlace será a *http://www.ra-ma.es*, el segundo será a *http://www.bing.es* y el tercero será un enlace a la Actividad 1.2 con una imagen que lleve el número 2.

1.3.6 TABLAS

Las tablas nos permiten representar y ordenar cualquier elemento de nuestra presentación en diferentes filas y columnas de modo que podamos resumir grandes cantidades de información de una manera que puede representarse rápida y fácilmente. Cuando hablamos de filas nos referimos a las casillas que están en horizontal y las columnas son las que están de manera vertical.

Una tabla va siempre delimitada por las etiquetas <table> y </table>. Dentro de una tabla vamos a distinguir dos elementos: filas y columnas:

- ✓ **Filas**: se identifican mediante las etiquetas <tr> y </tr>.
- ✓ **Columnas**: se identifican mediante las etiquetas <td> y </td>.

Para completar una tabla vamos rellenando casillas de izquierda a derecha, partiendo de arriba hacia abajo.

Vamos a ver un ejemplo de una tabla que tiene 3 filas y 2 columnas, la primera fila contendrá el título de los campos de la tabla.

```
<HTML>
<HEAD>
<TITLE>web</TITLE>
</HEAD>
<BODY>
<H1>Tablas</H1>
<TABLE BORDER="1">
<TR>
   <TD>casilla 1</TD>
   <TD>casilla 2</TD>
</TR>
<TR>
   <TD>casilla 3</TD>
   <TD>casilla 4</TD>
</TR>
   <TR>
   <TD>casilla 5</TD>
   <TD>casilla 6</TD>
   </TR>

</TABLE>
</BODY>
</HTML>
```

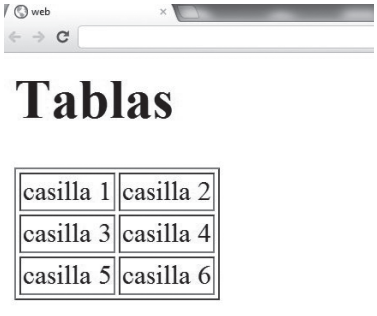

Figura 1.7. *Tablas*

ACTIVIDADES 1.7

- Hacer una página web con una tabla que tenga 2 filas y 5 columnas, en cada una de las casillas irá un numero: 1, 2, 3… hasta el 10.

1.4 LENGUAJES DE SCRIPT DE CLIENTE

Un lenguaje de *script* es un pequeño lenguaje de programación cuyo código se inserta dentro del documento HTML. Este código se ejecuta en el navegador del usuario al cargar la página, o cuando sucede algo especial como puede ser el pulsar sobre un enlace.

Estos lenguajes permiten variar dinámicamente el contenido del documento, modificar el comportamiento normal del navegador, validar formularios, realizar pequeños trucos visuales, etc. Sin embargo, conviene recordar que se ejecutan en el navegador del usuario y no en la máquina donde estén alojados, por lo que no podrán realizar cosas como manejar bases de datos.

Los lenguajes de *script* de cliente más conocidos son: HTML, CSS, JavaScript, Flash, VBScript y los *applets* de Java.

1.5 PÁGINAS WEB SENCILLAS CON JAVASCRIPT

El primer lenguaje de *script* que vio la luz fue el JavaScript de Netscape. Nacido con la versión 2.0 de este navegador en el año 1995 y basado en la sintaxis de Java, su utilidad y el casi absoluto monopolio que entonces ejercía Netscape en el mercado de navegadores permitieron que se popularizara y extendiera su uso.

¿Por qué el JavaScript y no otro lenguaje de programación? Porque:

- ✓ Es moderno (tiene pocos años).
- ✓ Es sencillo (su hermano mayor, el Java, es bastante más complejo).
- ✓ Es útil (el desarrollo de Internet ha sido muy rápido en los últimos años).
- ✓ Es potente: permite la POO (Programación Orientada a Objetos).
- ✓ Es barato: solo necesitamos un editor de textos y un navegador. Permite la "programación visual" (ventanas, botones, colores, formularios, etc.).

HTML dispone de unas etiquetas para incluir código de *script* en una página. Esas etiquetas pueden situarse en la sección HEAD o en el BODY de la página (eso depende de lo que vaya a hacer el *script*). Las sentencias escritas en JavaScript se encapsulan entre las etiquetas <SCRIPT LANGUAGE="JavaScript" TYPE="text/JavaScript"> y </script> o simplemente entre <script> y </script>.

A continuación un ejemplo representado de la primera forma:

```
<HTML>
<BODY>
<SCRIPT LANGUAGE="JavaScript" TYPE="text/JavaScript">
alert ('Esto es un ejemplo con JavaScript');
</SCRIPT>
</BODY>
</HTML>
```

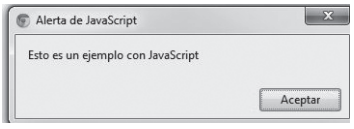

Figura 1.8. *Primer ejemplo*

Un segundo ejemplo representado de la manera "corta":

```
<html><body>
<script>
alert("Bienvenido a mi sitio web. Gracias...");
</SCRIPT>
</BODY>
</HTML>
```

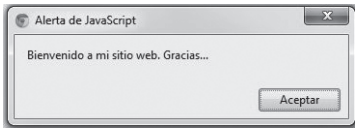

Figura 1.9. *Segundo ejemplo*

ACTIVIDADES 1.8

- Hacer una página web que incluya un mensaje por pantalla en JavaScript que muestre el mensaje "*Tarea 8 de nombre y apellidos del alumno/a*".

ACTIVIDADES 1.9

- Hacer una página web que muestre varios mensajes por pantalla en JavaScript, el primero será "*nombre y apellidos del alumno/a*", el segundo "estás en mi tarea 9" y, el tercero, "Gracias por tu visita".

1.6 HOJAS DE ESTILO

El lenguaje HTML está limitado a la hora de aplicarle formato a un documento. Esto es así porque fue concebido para otros usos (científicos sobre todo), distintos a los actuales, mucho más amplios. Para solucionar estos problemas los diseñadores han utilizado técnicas tales como la utilización de tablas con imágenes transparentes para ajustarlas, utilización de etiquetas que no son estándares del HTML y otras. Estas "trampas" han causado a menudo problemas en las páginas a la hora de su visualización en distintas plataformas.

Además, los diseñadores se han visto frustrados por la dificultad con la que, aun utilizando estos trucos, se encontraban a la hora de maquetar las páginas, ya que muchos de ellos venían maquetando páginas sobre el papel, donde el control sobre la forma del documento es absoluto. Para subsanar estar carencias aparecen las hojas de estilo en cascada CSS.

1.7 PÁGINAS WEB SENCILLAS CON CSS

Para dar valor a una etiqueta con CSS utilizaremos el nombre de la etiqueta separada de ":" (dos puntos), a continuación vendrá el valor de dicha etiqueta. Podemos aplicar estilos CSS a:

- Un sitio web, de modo que se puede definir la forma de todo el sitio de una sola vez. Para ello utilizaremos un fichero externo con extensión *css*.
- Una página individual, podemos dar formato a toda la página. Para ello introduciremos el código CSS entre las etiquetas <style type="text/css"> y </style> en la zona <head> de nuestra web.
- Una porción del documento, aplicando estilos visibles en un trozo de la página, con las etiquetas y .
- Una etiqueta en concreto, llegando incluso a poder definir varios estilos diferentes para una sola etiqueta. Esto es muy importante ya que ofrece potencia en nuestra programación.

1.7.1 ALGUNAS ETIQUETAS CSS

Antes de ver dónde podemos aplicar CSS, vamos a ver algunas etiquetas CSS sencillas, que nos servirán para dar un formato más espectacular a nuestras webs. Si queremos utilizar varias etiquetas, separaremos unas de otras con ";" (punto y coma).

- **Color**: podemos indicar el color del texto, el color vendrá en inglés (*white*, *black*, *blue*, etc.) o utilizar # junto con el valor RGB (#009900).
- **Font-size**: aquí indicamos el tamaño de la letra. Su valor puede expresarse en puntos (pt) como lo haríamos en un editor de texto o mediante unos valores predefinidos (*small*, *medium*, *large*, etc.).
- **Font-family**: en este caso establecemos el tipo de fuente que se representará por pantalla. No utilizar tipos raros ya que si el cliente no tiene esa fuente en su sistema, la página podría no mostrarse bien. Algunos ejemplos: Arial, Times, Courier, etc.

- **Font-weight**: grosor de los caracteres. Podemos expresarlo con números del 100, 200, ..., 900 o valores predefinidos (*normal*, *bold*, *bolder*, *light*, etc.).
- **Font-style**: curvado de los caracteres. Hay 3 valores predefinidos para este estilo: *normal*, *italic* y *oblique*.
- **Text-align**: alineación del texto con respecto a la página. Puede ser izquierda (*left*), derecha (*right*), centrado (*center*) o justificado (*justify*).

1.7.2 CSS EN PEQUEÑAS PARTES DE LA PÁGINA

Para definir estilos en secciones reducidas de una página se utiliza la etiqueta <SPAN STYLE>. Con su atributo style indicamos en sintaxis CSS las características de estilos.

```
<span style="color:blue; font-weight:bolder;">
Esto va en azul y grosor en negrita </span>
<br>
<span style="color:green;font-size:16pt;">
Esta parte en verde y fuente de 16 puntos</span>
<br>
<span style="color:red;font-style:italic;font-weight:normal;">
Aqui mezclo 3: color rojo, italic(cursiva) y grosor normal</span>
```

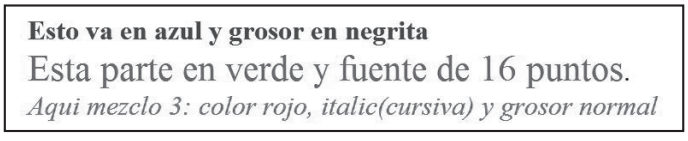

Figura 1.10. *CSS*

ACTIVIDADES 1.10

- Hacer una página que muestre lo siguiente: en la primera línea vendrá el nombre completo del alumno/a en color verde (*green*), grosor del texto *bolder*, la fuente utilizada será *Arial* y el tipo de texto *oblique*. En la segunda línea aparecerá el curso y el ciclo formativo al que pertenece (SMR), en este caso el texto será de color azul (*blue*), la fuente tendrá un tamaño de 18 puntos, el tipo de letra será *Courier* y el texto quedará alineado a la parte izquierda de la página.

1.7.3 ESTILO DEFINIDO PARA UNA ETIQUETA

De este modo podemos hacer que toda una etiqueta muestre un estilo determinado. En el siguiente ejemplo vemos la diferencia entre utilizar CSS con etiquetas a utilizarlas sin CSS.

```
<h1 style="color:blue; font-weight:bolder;">
H1 en azul y grosor en negrita </h1>
<h1>H1 en su modo normal de presentación </h1>
<p style="color:green; font-size:16pt;">
P en verde y fuente de 16 puntos</p>
<p>P en su modo normal de presentacion</p>
```

Figura 1.11. *CSS*

ACTIVIDADES 1.11

- Hacer una página que muestre lo siguiente: en la primera línea vendrá el nombre completo del alumno/a con formato de cabecera 3 (h3) en color rojo (*red*), grosor del texto normal, la fuente utilizada será *Courier* y el tipo de texto será *italic*. En la segunda línea aparecerá el curso y el ciclo formativo al que pertenece con formato cabecera 5 (h5), en este caso el texto será de color azul marino (*navy*), el texto lo más grueso posible (*bolder*), el tipo de letra será *Arial* y el texto aparecerá alineado a la derecha.

1.7.4 ESTILO DEFINIDO PARA TODA UNA PÁGINA

Podemos definir, en la cabecera del documento, estilos para que sean aplicados a toda la página. Esta definición se hace dentro del <head> utilizando las etiquetas <style type="text/css"> y </style>. Es una manera muy cómoda de darle forma al documento y muy potente, ya que estos estilos serán seguidos en toda la página y nos ahorraremos así muchas etiquetas HTML que apliquen forma al documento. Además, si deseamos cambiar los estilos de la página lo haremos de una sola vez. Como puede observarse, cada definición de etiquetas de estilo va encerrada entre **{ }** y si hubiese más de un parámetro, separados por **;**.

```
<html>
<head>
<STYLE type="text/css">
h1 {
font-family:arial;
color: blue;}

p {
font-Family:times;
color: green;
font-weight:bold;}
</STYLE>
</head>
<body>
<h1>Estilos CSS</h1>
Bienvenidos...
<p>Aqui se ve el estilo aplicado</p>
</body>
</html>
```

Figura 1.12. *CSS*

ACTIVIDADES 1.12

- Hacer una página que asigne estilo a las etiquetas siguientes:
  a. Cabecera 2 (h2) en color rojo (*red*), grosor del texto *bold*, la fuente será *Times* y el tipo de texto será *italic*.
  b. Cabecera 5 (h5), el texto será de color azul claro (cian), el texto lo más grueso posible (*bolder*) y el tipo de letra será Arial, además el texto aparecerá alineado en el centro de la página.
- Hacer un texto de prueba dentro de nuestra web donde aparezcan los estilos con su nuevo formato.

1.7.5 ESTILO DEFINIDO PARA TODO UN SITIO WEB

Sería algo parecido al caso anterior, solo que ahora en lugar de tener el estilo contenido en la propia página, lo tenemos en un archivo externo *css*. Así podemos tener varias páginas con el mismo estilo sin tener que copiar en todas ellas la definición. Por lo tanto, si queremos variar algún parámetro en la definición de estilos solo tendremos que cambiarla en el fichero *css* y todas las páginas quedarán automáticamente actualizadas.

Como ya hemos indicado tendremos por un lado el fichero de definiciones de estilos y, por otro, nuestra página web. Para hacer referencia a ese fichero de estilo desde la cabecera de nuestra web (*head*) utilizaremos la etiqueta:

```
<link rel="stylesheet" type="text/css" href="estilos.css">
```

Donde *estilos.css* es el nombre del fichero que tiene definidos los estilos. Vamos a ver el mismo ejemplo anterior utilizando dos ficheros.

Fichero estilo.css

```
h1 {
 font-family:arial;
 color: blue;}

p {
font-Family:times;
color: green;
font-weight:bold;}
```

Nuestra página web

```
<html>
<head>
<link rel="stylesheet" type="text/css" href="estilo.css">
</head>
<body>
<h1>Estilos CSS</h1>
Bienvenidos...
<p>Aqui se ve el estilo aplicado</p>
</body>
</html>
```

El resultado obtenido es el mismo que en la Figura 1.12, ya que hemos aplicado los mismos estilos que entonces.

ACTIVIDADES 1.13

- Hacer una página externa que proporcione estilo a las etiquetas siguientes:
  a. Cabecera 1 (h1) en color amarillo (*yellow*), grosor del texto *bold*, la fuente será *Arial* y el tipo de texto será *italic*.
  b. Cabecera 3 (h3), el texto será de color verde (*green*), el texto lo más grueso posible (*bolder*) y el tipo de letra será *Courier*. El texto aparecerá alineado en la parte derecha de la página.
- Hacer otra página web (*html*) que incorpore dicho estilo (*link rel*). Dentro de nuestra web deben aparecer los estilos con su nuevo formato.

1.8 LENGUAJES DE SCRIPT DE SERVIDOR

Los lenguajes de lado servidor son aquellos lenguajes que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él (HTML).

1.8.1 CARACTERÍSTICAS

Existe una multitud de lenguajes. Cada uno de ellos explota más a fondo ciertas características que los hacen más o menos útiles para desarrollar distintas aplicaciones. La versatilidad de un lenguaje está íntimamente relacionada con su complejidad. Un lenguaje complicado en su aprendizaje permite en general realizar un espectro de tareas más amplio y más profundamente. Es por ello que a la hora de elegir el lenguaje que queremos utilizar tenemos que saber claramente qué es lo que queremos hacer y si el lenguaje en cuestión nos lo permite o no.

1.8.2 TIPOS

A continuación se presentan los principales lenguajes de *script* de servidor, junto con sus ventajas y desventajas.

PHP: es un lenguaje de programación utilizado para la creación de sitios web. PHP es un acrónimo recursivo que significa *PHP Hypertext Pre-processor* (inicialmente se llamó *Personal Home Page*). Surgió en 1995, desarrollado por PHP Group. PHP es un lenguaje de *script* interpretado en el lado del servidor utilizado para la generación de páginas web dinámicas, embebidas en páginas HTML y ejecutadas en el servidor. PHP no necesita ser compilado para ejecutarse. Para su funcionamiento necesita tener instalado Apache o IIS con las librerías de PHP. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas. Los archivos cuentan con la extensión *.php*.

Ventajas:

- Muy fácil de aprender.
- Se caracteriza por ser un lenguaje muy rápido.
- Soporta en cierta medida la orientación a objetos. Clases y herencia.
- Es un lenguaje multiplataforma: Linux, Windows, entre otros.
- Capacidad de conexión con la mayoría de los manejadores de base de datos: MySQL, PostgreSQL, Oracle, MS SQL Server, entre otros.
- Capacidad de expandir su potencial utilizando módulos.
- Posee documentación en su página oficial, la cual incluye descripción y ejemplos de cada una de sus funciones.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Incluye gran cantidad de funciones.
- No requiere definición de tipos de variables ni manejo detallado del bajo nivel.

Desventajas:

- Se necesita instalar un servidor web.
- Todo el trabajo lo realiza el servidor y no delega al cliente, por tanto, puede ser más ineficiente a medida que las solicitudes aumenten de número.
- La legibilidad del código puede verse afectada al mezclar sentencias HTML y PHP.
- La programación orientada a objetos es aún muy deficiente para aplicaciones grandes.
- Dificulta la modularización.
- Dificulta la organización por capas de la aplicación.

ASP: es una tecnología del lado del servidor desarrollada por Microsoft para el desarrollo de sitios web dinámicos. ASP, en inglés *Active Server Pages*, fue liberado por Microsoft en 1996. Las páginas web desarrolladas bajo este lenguaje necesitan tener instalado *Internet Information Server* (IIS). ASP no necesita ser compilado para ejecutarse. Existen varios lenguajes que se pueden utilizar para crear páginas ASP. El más utilizado es VBScript, nativo de Microsoft. ASP se puede programar también en Perl y Jscript (no JavaScript). El código ASP puede ser insertado junto con el código HTML.

Ventajas:

- Usa Visual Basic Script, siendo fácil para los usuarios.
- Comunicación óptima con SQL Server.
- Soporta el lenguaje JScript (JavaScript de Microsoft).

Desventajas:

- Código desorganizado.
- Se necesita escribir mucho código para realizar funciones sencillas.
- Tecnología propietaria.
- Hospedaje de sitios web costosos.

ASP.NET: este es un lenguaje comercializado por Microsoft y usado por programadores para desarrollar entre otras funciones, sitios web. ASP.NET es el sucesor de la tecnología ASP y fue lanzado al mercado mediante una estrategia de mercado denominada .NET. ASP.NET fue desarrollado para resolver las limitaciones que brindaba su antecesor ASP. Creado para desarrollar web sencillas o grandes aplicaciones. Para el desarrollo de ASP.NET se puede utilizar C#, VB.NET o J#. Los archivos cuentan con la extensión (*aspx*). Para el funcionamiento de las páginas se necesita tener instalado IIS con el *Framework* .Net. Microsoft Windows 2003 incluye este *framework*, solo se necesitará instalarlo en versiones anteriores.

Ventajas:

- Completamente orientado a objetos.
- Controles de usuario y personalizados.
- División entre la capa de aplicación o diseño y el código.
- Facilita el mantenimiento de grandes aplicaciones.
- Incremento de velocidad de respuesta del servidor.
- Mayor velocidad.
- Mayor seguridad.

Desventajas:

- Mayor consumo de recursos.

JSP: es un lenguaje para la creación de sitios web dinámicos, acrónimo de *Java Server Pages*. Está orientado a desarrollar páginas web en Java. JSP es un lenguaje multiplataforma. Creado para ejecutarse del lado del servidor. JSP fue desarrollado por Sun Microsystems. Comparte ventajas similares a las de ASP.NET. Fue desarrollado para la creación de aplicaciones web potentes. Posee un motor de páginas basado en los *servlets* de Java. Para su funcionamiento se necesita tener instalado un servidor Tomcat.

Ventajas:

- Ejecución rápida del *servlets*.
- Crear páginas del lado del servidor.
- Multiplataforma.
- Código bien estructurado.
- Integridad con los módulos de Java.
- La parte dinámica está escrita en Java.
- Permite la utilización se *servlets*.

Desventajas:

- Complejidad de aprendizaje.

PYTHON: es un lenguaje de programación creado en el año 1990 por Guido van Rossum y es el sucesor del lenguaje de programación ABC. Python es comparado habitualmente con Perl. Los usuarios lo consideran como un lenguaje más limpio para programar. Permite la creación de todo tipo de programas incluyendo los sitios web. Su código no necesita ser compilado, por lo que se dice que el código es interpretado. Es un lenguaje de programación multiparadigma, lo cual fuerza a que los programadores adopten un estilo de programación particular: programación orientada a objetos, programación estructurada, programación funcional y programación orientada a aspectos.

Ventajas:

- Libre y de código abierto.
- Lenguaje de propósito general.
- Gran cantidad de funciones y librerías.
- Sencillo y rápido de programar.
- Multiplataforma.
- Orientado a objetos.
- Portable.

Desventajas:

- Lentitud por ser un lenguaje interpretado.

RUBY: es un lenguaje interpretado de muy alto nivel y orientado a objetos. Desarrollado en 1993 por el programador japonés Yukihiro "Matz" (Matsumoto). Su sintaxis está inspirada en Phyton, Perl. Es distribuido bajo licencia de software libre (*Opensource*). Ruby es un lenguaje dinámico para una programación orientada a objetos rápida y sencilla.

Ventajas:

- Permite desarrollar soluciones a bajo coste.
- Software libre.
- Multiplataforma.

Desventajas:

- Código desordenado si no se establece convención de sintaxis entre programadores.

1.9 HERRAMIENTAS PARA EL DISEÑO WEB

El código para la programación web puede ser generado con cualquier editor de texto, es muy recomendable no utilizar editores de texto avanzados (Microsoft Word, OpenOffice Writer, etc.) ya que añaden al texto caracteres especiales que no serían "comprensibles" por el navegador. Por lo tanto, si vamos a utilizar editores de texto no específicos sería recomendable utilizar el Bloc de Notas de Windows o el Gnome Editor o Kde Editor de Linux. Existen también herramientas creadas expresamente para diseñar código, hay algunas que son gratuitas y otras de pago, que se van a presentar a continuación de manera breve para que el alumno/a decida la que se adapte a su gusto y pueda profundizar en el manejo de dicha herramienta.

1.9.1 NOTEPAD++

Notepad++ es un editor de texto y de código fuente libre con soporte para varios lenguajes de programación. Solo funciona en Microsoft Windows. Se parece al Bloc de notas en cuanto al hecho de que puede editar texto sin formato y de forma simple. No obstante, incluye opciones más avanzadas que pueden ser útiles para usuarios avanzados como desarrolladores y programadores.

Se distribuye bajo los términos de la Licencia Pública General de GNU. Tiene multitud de lenguajes de programación soportados, incluidos todos los que hemos visto en el capítulo. Además, permite al usuario definir su propio lenguaje: no solo las palabras clave para la sintaxis coloreada, sino también las palabras clave para la envoltura de sintaxis, los comentarios clave y los operadores.

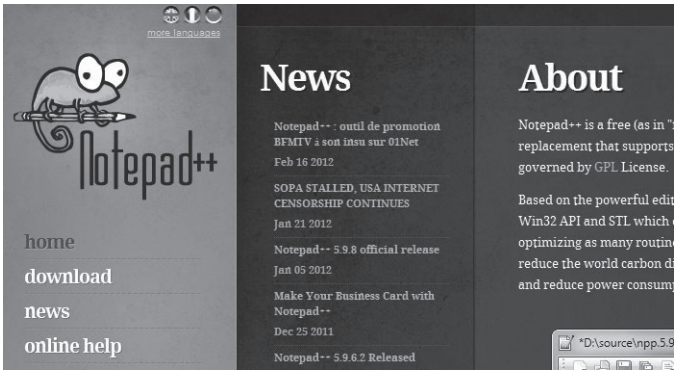

Figura 1.13. *Notepad++*

ACTIVIDADES 1.14

- Descargue e instale Notepad++. Haga capturas de todo el proceso, tanto en la descarga como en la instalación.

1.9.2 DREAMWEAVER

Dreamweaver es un software desarrollado por Adobe, de pago y con versiones para Windows y Mac OS. Es fácil de usar y permite crear páginas web profesionales. Las funciones de edición visual de Dreamweaver permiten agregar rápidamente diseño y funcionalidad a las páginas, sin la necesidad de programar manualmente el código HTML.

Se pueden crear tablas, editar marcos, trabajar con capas, insertar comportamientos JavaScript, etc., de una forma muy sencilla y visual. Además incluye un software de cliente FTP completo, permitiendo, entre otras cosas, trabajar con mapas visuales de los sitios web, actualizando el sitio web en el servidor sin salir del programa.

Figura 1.14. *Dreamweaver*

ACTIVIDADES 1.15

- Descargue e instale Dreamweaver y haga capturas de todo el proceso, tanto en la descarga como en la instalación.

1.9.3 APTANA STUDIO

Aptana Studio es un entorno de desarrollo integrado gratuito basado en Eclipse y desarrollado por Aptana, Inc., que puede funcionar bajo Windows, Mac y Linux y provee soporte para lenguajes como: PHP, Python, Ruby, CSS, Ajax, HTML y Adobe AIR.

Tiene la posibilidad de incluir complementos para nuevos lenguajes y funcionalidades. Aptana Studio es un entorno de software libre que posee la GNU General Public License o la Aptana Public License.

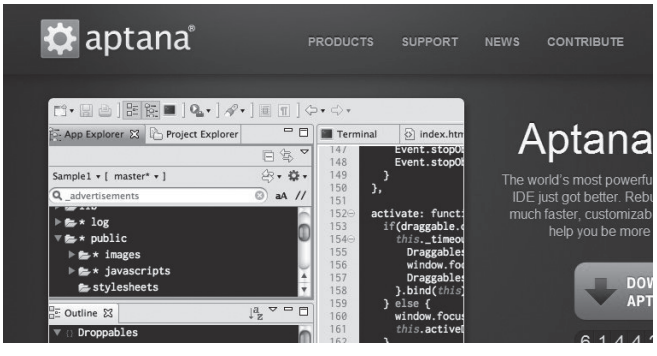

Figura 1.15. *Aptana Studio*

ACTIVIDADES 1.16

- Descargue e instale Aptana Studio. Haga capturas de todo el proceso, tanto en la descarga como en la instalación.

1.10 RELACIÓN ENTRE PÁGINAS WEB Y BASES DE DATOS

Antes de ver la relación entre las páginas y las bases de datos vamos a ver la diferencia entre las páginas web estáticas y dinámicas.

- Las páginas web estáticas siempre se ven igual y el contenido nunca cambia a menos que se cargue una nueva página, o cambiemos el aspecto de la página modificando la página original.
- Las páginas web dinámicas hacen lo contrario, ya que pueden cambiar cada vez que se cargan (sin que tengamos que hacer esos cambios) y pueden cambiar su contenido, basándose en lo que los usuarios hagan, como hacer clic sobre un texto o una imagen.

Uno de los tipos más comunes de páginas web dinámicas son las vinculadas a bases de datos. Esto significa que tenemos una página web que recibe la información de una base de datos (la página web está conectada a la base de datos a través de la programación) e inserta la información en la página web cada vez que esta se carga. Si la información almacenada en la base de datos cambia, la página web conectada a la base de datos también cambiará en consecuencia sin intervención del usuario.

Los lenguajes de programación web orientados al servidor son los que van a permitir la comunicación con las bases de datos; estos lenguajes se han visto en el apartado 8 de este capítulo. Dicha relación se producirá entre el lenguaje de programación orientado al servidor y el sistema gestor de bases de datos. Los sistemas gestores de bases de datos más conocidos y utilizados son los siguientes:

- **MySQL**: es un sistema de gestión de bases de datos relacional, multihilo y multiusuario, desde enero de 2008 una subsidiaria de Sun Microsystems desarrolla MySQL como software libre en un esquema de licenciamiento dual. Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso.

Figura 1.16. *MySQL*

- **Oracle**: es un sistema de gestión de bases de datos objeto-relacional. Se considera a Oracle como uno de los sistemas de bases de datos más completos, destacando su soporte de transacciones, estabilidad, escalabilidad y soporte multiplataforma. Su dominio en el mercado de servidores empresariales ha sido casi total hasta hace relativamente poco. Recientemente sufre la competencia del Microsoft SQL Server de Microsoft y de la oferta de otros como PostgreSQL, MySQL o Firebird.

Figura 1.17. *Oracle*

- **PostgreSQL**: es un sistema de gestión de bases de datos relacional orientado a objetos y libre, publicado bajo la licencia BSD. Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una empresa y/o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre y/o apoyados por organizaciones comerciales.

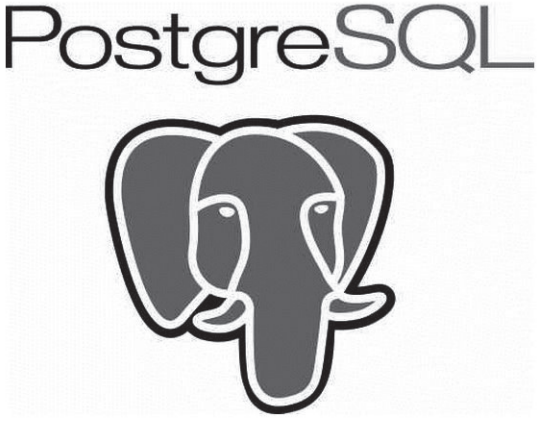

Figura 1.18. *PostgreSQL*

- **Microsoft SQL Server**: es un sistema para la gestión de bases de datos producido por Microsoft basado en el modelo relacional. Sus lenguajes para consultas son T-SQL y ANSI SQL. Microsoft SQL Server constituye la alternativa de Microsoft a otros potentes sistemas gestores de bases de datos como son Oracle, PostgreSQL o MySQL.

Figura 1.19. *SQL Server*

RESUMEN DEL CAPÍTULO

En este primer capítulo se ha visto en primer lugar un pequeño esquema del funcionamiento de un servicio web. Aparecen los lenguajes de marcas, se les llama así porque para su uso nos servimos de una serie de marcas o etiquetas. Lenguajes de este tipo son: XHTML, XML, GML, RTF, Text y, por supuesto, HTML. Para este último se han elaborado páginas sencillas con este lenguaje.

Hemos visto, también, de forma introductoria las hojas de estilo en cascada CSS, que nos van a servir para dar un aspecto más bonito y llamativo a nuestras páginas elaboradas con HTML.

Para crear nuestras páginas web, nos podremos ayudar de herramientas de diseño web, que nos facilitarán mucho la labor de creación. Herramientas tales como Dreamweaver, Aptana, Notepad++, etc.

También se han introducido los lenguajes de *script* de cliente o navegador, viendo las diferencias con los lenguajes de *script* de servidor, estos últimos tan potentes que son capaces de interactuar con sistemas gestores de bases de datos.

EJERCICIOS PROPUESTOS

- **1.** Defina con sus palabras qué es HTML.
- **2.** ¿Qué limitaciones tiene HTML con respecto a presentar la información de forma vistosa?
- **3.** Defina con sus palabras qué es CSS.
- **4.** ¿Qué cree que es anterior, HMTL o CSS? ¿Por qué?
- **5.** Busque en Internet información sobre HTML5, sus principales novedades y características.
- **6.** Busque en Internet información sobre CSS3, sus principales novedades y características.

TEST DE CONOCIMIENTOS

1 La etiqueta que nos permite hacer una tabla es:
- **a)** </TABLE>.
- **b)** <TABLE>.
- **c)** <TD>.
- **d)** <TR>.
- **e)** Todas son ciertas.
- **f)** Todas son falsas.

2 Si queremos que el texto que nos aparece en pantalla aparezca en negrita, debemos utilizar la etiqueta:
- **a)** *Font-family*.
- **b)** *Font-size*.
- **c)** *Font-weight*.
- **d)** *Font-style*.
- **e)** *Color*.
- **f)** *Text-align*.

3 Si queremos que el texto que se nos presenta en pantalla aparezca en cursiva, debemos utilizar la etiqueta:
- **a)** *Font-family*.
- **b)** *Font-size*.
- **c)** *Font-weight*.
- **d)** *Font-style*.
- **e)** *Color*.
- **f)** *Text-align*.

4 Un lenguaje orientado a servidor es:
- **a)** JavaScript.
- **b)** CSS.
- **c)** HMTL.
- **d)** Phyton.
- **e)** Flash.
- **f)** Todas son ciertas.
- **g)** Todas son falsas.

5 Es obligatorio utilizar una herramienta de edición de páginas web para hacer páginas en HTML.
- **a)** Sí, ya que hay que compilarlas con dicho editor.
- **b)** Siempre.
- **c)** No, ya que no existen herramientas de edición de páginas web.
- **d)** No, aunque puede ayudarnos en la elaboración de la página.
- **e)** Todas son falsas.