

Introducción al desarrollo web para móviles

Índice

1	Introducción.....	3
1.1	Aplicación móvil vs. Aplicación nativa.....	3
1.2	Reglas de usabilidad.....	4
1.3	Buenas prácticas de programación Web para dispositivos móviles.....	6
1.4	Dominio.....	7
1.5	Detección del navegador.....	7
1.6	Inspiración.....	9
2	Instalación de un servidor Web.....	12
2.1	XAMPP para Linux.....	12
2.2	XAMPP para Windows.....	13
2.3	Instalar un servidor Web para Mac.....	16
2.4	Acceso mediante un dispositivo móvil real.....	19
2.5	Instalación del SDK de Android.....	21
2.6	Instalar Xcode.....	22
2.7	Simuladores y Emuladores.....	22
3	HTML.....	22
3.1	Editores HTML.....	23
3.2	Etiquetas.....	23
3.3	Estructura básica de una Web.....	24
3.4	Elementos de la cabecera.....	24
3.5	Etiquetas básicas HTML.....	25
3.6	Listas.....	26
3.7	Tablas.....	26
3.8	Cajas (etiqueta <div>).....	27
3.9	Enlaces.....	28

3.10 Imágenes.....	28
3.11 Formularios.....	28
3.12 Eventos.....	31
3.13 Símbolos HTML.....	32
4 CSS.....	33
4.1 Adjuntar una hoja de estilo.....	34
4.2 Definición de estilos para etiquetas HTML.....	34
4.3 Identificadores y Clases	35
4.4 Estilos CSS básicos.....	36
4.5 Pseudo-clases.....	40
4.6 Capas.....	40
4.7 Más información.....	42

1. Introducción

La Web móvil es la misma Web que la de escritorio, utiliza la misma arquitectura básica y muchas de las mismas tecnologías. Pero existen claras diferencias que impiden que su funcionamiento y manejo sea el mismo, como son: el tamaño de la pantalla, las diferentes formas de manejo (táctil, teclado del móvil, etc.) y el ancho de banda.

Otra diferencia que se debería de tener en cuenta es que es "móvil". Accedemos a Internet desde lugares en los que sería imposible hacerlo con un ordenador de sobremesa o incluso un portátil. Esto influye también en el uso que le damos, se hacen más búsquedas o consultas, además de que la información se debe de disponer de forma más clara y directa.

Debido a todo esto, al programar una Web para móvil debemos de tener en cuenta que ni el contenido, ni la apariencia, ni la estructura de la web va a ser la misma que la que podríamos hacer para un ordenador de escritorio. Hemos de diseñar muy bien este tipo de aplicaciones y orientarlas al uso principal que le va a dar el usuario. Para esto se suele referir a la regla del 20%: el 80% del contenido del sitio web de escritorio no es válido para una web móvil. Por lo que hemos de centrarnos en ese 20% restante, averiguar cuáles es, y optimizar nuestro sitio para este uso.

1.1. Aplicación móvil vs. Aplicación nativa

Los desarrolladores de aplicaciones nativas tienen la ventaja de poder usar funciones no disponibles para la web móvil:

- El uso de interfaces nativas que proveen los propios SDK como iPhone o Android.
- El uso de bases de datos locales. Aunque en HTML 5 se pueden usar un almacenamiento local, hay que reconocer que estas tecnologías están más avanzadas en los sistemas nativos.
- Notificaciones push. A esto se refiere con los avisos centralizados que muestran las aplicaciones, aún cuando están ocultas. Una fuerte razón que no puede ser implementada en una web móvil.
- Geolocalización. Hemos visto algunos ejemplos de geolocalización a través de HTML5, pero que no acaban de alcanzar la misma experiencia de usuario que una aplicación móvil. Podemos interactuar con el mapa o con las funcionalidades asociadas a la localización del usuario, pero no ir mucho más allá.
- Soporte para cámara o vídeo. Las funciones multimedia están perfectamente acopladas a las aplicaciones nativas, pudiendo añadir funcionalidades específicas a nuestra aplicación.

Sin embargo estas diferencias cada vez se van haciendo menores:

- Gracias al uso de HTML5, CSS3 y JavaScript cada vez se pueden hacer más cosas y obtener mejores resultados.

- Cada vez hay mejores frameworks de desarrollo para aplicaciones móviles, como JQuery Mobile o Sencha Touch. Estos nos permiten crear webs con apariencia cercana a las de las aplicaciones nativas, pero a su vez con toda la potencia de la Web. Estos framework nos permiten adaptar el contenido según el dispositivo usado y sus posibilidades técnicas: pantalla táctil, reproducción de vídeos o resolución de pantalla. Por lo que podríamos decir que una Web móvil es mucho más adaptable (además de multiplataforma) que una aplicación móvil.
- La web sigue siendo el negocio principal de muchas empresas de Internet. El desarrollo web no ha muerto por la inclusión de las aplicaciones móviles, sino que se ve afectado por un proceso de cambio hacía la adopción de tecnologías nuevas como HTML5.
- La inclusión de la tecnología *PhoneGap* cada vez está recortando más estas diferencias. *PhoneGap* es la posibilidad de crear una aplicación nativa instalable a partir de una página Web móvil (que se pueda distribuir también en *Android Market* o en la *App Store*). Además estas tecnologías facilitan el uso de funcionalidades del dispositivo móvil directamente a través de código JavaScript, como puede ser el acceso a la cámara, acelerómetro, geolocalización, listado de contactos, comprobar el estado de la conexión, etc.

1.2. Reglas de usabilidad

1. Reducir la cantidad de contenido

Las aplicaciones móviles deben de ser optimizadas dado que el espacio visual es mucho más limitado que en una pantalla de ordenador. Cada píxel cuenta, y no todo lo que es válido para una Web de escritorio lo es para una Web móvil.

Solo debemos de incluir el contenido y las características principales y más importantes. Los contenidos con menor importancia deben de ser eliminados, como contenidos secundarios, normalmente localizados en columnas laterales.

La web móvil debe de estar enfocada a este contenido principal. Facilitar su lectura y navegación, así como mejorar los tiempos de carga reduciendo imágenes y contenidos.

2. Usar una sola columna

Las páginas Web anchas y con varias columnas dificultan la navegación y lectura en un dispositivo móvil. Incluso en los terminales móviles con pantallas más grandes hay que realizar zoom para moverse y ver bien el contenido. Esta es una práctica que debemos evitar, pues tener que ir realizando zoom añade más pasos a la navegación, y en algunos dispositivos no es tan fácil de realizar como en un iPhone.

Lo mejor es tener nuestro contenido en una sola columna que use todo el ancho de la pantalla. Para añadir contenido lo deberemos de hacer hacia abajo (o creando una página nueva), nunca hacia los lados (o creando columnas). Esto nos asegura que el contenido se va a visualizar correctamente, además es mucho más intuitivo realizar *scroll* hacia abajo

para ir leyendo.

3. Muestra los enlaces de navegación de forma diferente

No pongas todos los enlaces de navegación en la parte superior de la pantalla. Si hay muchos desplazarán todo el contenido hacia abajo, y es posible que los tengas que poner muy reducidos.

La página principal debería de contener los enlaces al resto del contenido junto con un buscador (si fuese necesario). El contenido debería de estar en páginas secundarias bien organizado. Por ejemplo, cuando un usuario entra en un sitio de *eCommerce* suelen tener una categoría de producto en mente que quiere consultar, la cual la podrían encontrar usando el buscador o directamente a partir del menú. Es decir, la página principal debe facilitar el acceso rápido a la información más importante de la web.

También hay otras opciones para colocar el menú de navegación, como una lista desplegable o al final de la página. Son muy cómodas las barras de herramientas estáticas que ofrecen las opciones principales (volver a la página inicial, botones principales, etc.).

4. Minimiza la cantidad de datos solicitados

Escribir texto utilizando un terminal móvil es mucho más difícil que hacerlo utilizando el teclado de un ordenador de sobremesa. Además los usuarios suelen escribir mucho más lento y cometer más errores. Por estas razones tenemos que intentar minimizar la cantidad de texto solicitado.

Una forma de conseguir esto es permitir almacenar los datos (usuario, contraseñas, configuración, direcciones, etc.), o aprovechar algunas de las funcionalidades que incorporan los dispositivos móviles (como veremos más adelante).

5. Decide si necesitas más de una Web para móvil

El tamaño de pantalla, la capacidad de procesamiento y de usabilidad varía enormemente de un terminal a otro. Por esta razón a veces debemos de considerar crear varios sitios web con el mismo contenido pero adaptado a diferentes necesidades. Por ejemplo, Facebook tiene m.facebook.com como sitio web principal para móviles, pero además tiene una versión optimizada para pantallas táctiles (touch.facebook.com) y una versión optimizada para conexiones más lentas (0.facebook.com).

6. Diseña para pantallas táctiles, pero también para teléfonos no-táctiles

La forma de navegar por las páginas web es muy diferente según el dispositivo: pantallas táctiles, trackball, joystick, teclado, etc. Estas características también son importantes a la hora de realizar el diseño. Por ejemplo, la principal dificultad está en la selección y pulsado sobre textos o enlaces pequeños. En las pantallas táctiles además se dificulta pulsar sobre elementos que estén muy juntos.

Por esta razón, los enlaces o elementos que puedan ser seleccionados deben de ocupar un mayor espacio en pantalla, incluirlos en botones o cuadros más grandes, que puedan ser

pulsados con facilidad.

7. Aprovecha las funcionalidades que incorporan los móviles

Los teléfonos móviles tienen algunas ventajas sobre los PCs, las cuales pueden facilitar la realización de algunas tareas. Algunas de estas funcionalidades añadidas son:

Realizar llamadas: puede parecer evidente pero es una funcionalidad muy útil que los PCs no pueden realizar tan fácilmente. Por esta razón debemos de aprovecharla para, por ejemplo, llamar directamente al presionar sobre un número de teléfono, facilitar el contacto con un servicio técnico, etc.

Uso de mapas y posición actual: es posible dar la opción al usuario de seleccionar una dirección y que automáticamente se abra en la aplicación de mapas del dispositivo móvil. También es muy interesante el uso de la posición actual para mostrar puntos de interés cercanos, calcular rutas, etc.

Solicitud de información de forma innovadora: como por ejemplo los códigos QR, que se han usado en algunas campañas de publicidad, etc.

1.3. Buenas prácticas de programación Web para dispositivos móviles

Además al programar un sitio Web para dispositivos móviles también se deben de seguir algunas pequeñas pautas. Al cumplirlas, se incrementará el público que puede acceder a los contenidos, creando sitios Web eficaces y haciendo la navegación accesible desde más dispositivos. Las principales pautas que debemos seguir son:

- URL lo más corta posible.
- La barra de navegación debe de estar en la cabecera y ofrecer solo la navegación mínima necesaria.
- Lo más importante, primero. Ofrece al usuario lo que busca. Si sabes que pide muchas noticias de prensa, eso debe ser el primer enlace de la web.
- No recargar automáticamente la página, a menos que se informe claramente al usuario y se ofrezca una forma de poder detener dicha acción. El usuario se podría encontrar con una factura considerable por un descuido.
- Peso limitado. Nuestra Web para móvil no debe de ocupar mucho, para esto se recomienda reducir todo lo posible las imágenes y el contenido multimedia.
- Scroll. Limite el desplazamiento de la página a una dirección.
- Test. Haga pruebas en emuladores y en dispositivos reales.
- Enlaces:
 - Intente que sean mínimos a recursos externos.
 - Intente conseguir un equilibrio entre los enlaces que hay en la web y lo que un usuario tarda en llegar a donde quiere.
 - Identificar de forma clara el destino del enlace. Si no estamos seguros de si el formato del destino es soportado por el dispositivo debemos indicarlo. Por ejemplo, para un pdf o para una web no móvil.

- No usar ventanas emergentes.
- No ofrecer más contenido del solicitado por el usuario y que sea estrictamente necesario para la estructura y navegación de la página.

1.4. Dominio

Existen diferentes alternativas sobre el dominio que podemos usar. En definitiva esta es una decisión personal, pues todas ellas tienen sus ventajas e inconvenientes. La única recomendación que se suele hacer es tener varias opciones disponibles, con la intención de facilitar al máximo el acceso.

Podemos tener un subdominio de nuestro sitio Web especializado para dispositivos móviles. Por ejemplo, si nuestro sitio Web es *www.midominio.com*, el sitio para dispositivos móviles podría ser *m.midominio.com*. Por ejemplo, Facebook tiene disponibles los sitios m.facebook.com (como sitio web para dispositivos móviles) y touch.facebook.com (para dispositivos táctiles).

También podemos usar el dominio principal y diferenciar (desde el cliente o desde el servidor) si se trata de un dispositivo móvil. En este caso el usuario accedería a la misma dirección pero sería redirigido al sitio correspondiente.

Otra opción es comprar un dominio ".mobi" (especial para web móvil) con el mismo nombre que la web principal.

Si optamos por dar diferentes opciones de acceso deberemos crear redirecciones 301 al dominio principal seleccionado para manejar ese contenido, de la forma:

```
<?php
header("HTTP/1.1 301 Moved Permanently");
header("location:http://www.nueva_url.com");
?>
```

1.5. Detección del navegador

Un dilema a la hora de desarrollar contenidos para móviles es cómo diferenciar entre dispositivos móviles y navegadores de escritorio. Esto se puede hacer fácilmente mediante una función de comprobación que nos indique el tipo de navegador que ha solicitado la web. Una vez obtenido el navegador tenemos varias opciones, como se comentaba en la sección anterior: redireccionar al dominio correspondiente, o adaptar el código de nuestra página según el cliente.

A continuación se incluye una función en PHP que nos devuelve un número positivo si detecta que el navegador es un dispositivo móvil, y 0 en caso de ser un navegador de escritorio.

```
public static function mobileBrowser()
```

```

{
    $mobile_browser = '0';

    //$_SERVER['HTTP_USER_AGENT'] -> el agente de usuario que está
    // accediendo a la página.
    if(preg_match('/(up.browser|up.link|mmp|symbian|smartphone|midp|wap|phone)/i',
        strtolower($_SERVER['HTTP_USER_AGENT'])))
    {
        $mobile_browser++;
    }

    //$_SERVER['HTTP_ACCEPT'] -> Indica los tipos MIME que el cliente
    puede recibir.
    if((strpos(strtolower($_SERVER['HTTP_ACCEPT']), 'application/vnd.wap.xhtml+xml')>0)
        or
        ((isset($_SERVER['HTTP_X_WAP_PROFILE']) or
        isset($_SERVER['HTTP_PROFILE'])))
        {
            $mobile_browser++;
        }

        $mobile_ua = strtolower(substr($_SERVER['HTTP_USER_AGENT'],0,4));
        $mobile_agents = array(
            'w3c
', 'acs-', 'alav', 'alca', 'amoi', 'audi', 'avan', 'benq', 'bird', 'blac',
'blaz', 'brew', 'cell', 'cldc', 'cmd-', 'dang', 'doco', 'eric', 'hipt', 'inno',
'ipaq', 'java', 'jigs', 'kddi', 'keji', 'leno', 'lg-c', 'lg-d', 'lg-g', 'lge-',
'maui', 'maxo', 'midp', 'mits', 'mmef', 'mobi', 'mot-', 'moto', 'mwbp', 'nec-',
'newt', 'noki', 'oper', 'palm', 'pana', 'pant', 'phil', 'play', 'port', 'prox',
'qwap', 'sage', 'sams', 'sany', 'sch-', 'sec-', 'send', 'seri', 'sgh-', 'shar',
'sie-', 'siem', 'smal', 'smar', 'sony', 'sph-', 'symb', 't-mo', 'teli', 'tim-',
'tosh', 'tsm-', 'upg1', 'upsi', 'vk-v', 'voda', 'wap-', 'wapa', 'wapi', 'wapp',
'wapr', 'webc', 'winw', 'winw', 'xda', 'xda-');

        //buscar agentes en el array de agentes
        if(in_array($mobile_ua, $mobile_agents)) {
            $mobile_browser++;
        }

        //$_SERVER['ALL_HTTP'] -> Todas las cabeceras HTTP
        if(strpos(strtolower($_SERVER['ALL_HTTP']), 'OperaMini')>0) {
            $mobile_browser++;
        }
        if(strpos(strtolower($_SERVER['HTTP_USER_AGENT']), 'windows')>0) {
            $mobile_browser=0;
        }

        return $mobile_browser;
    }
}

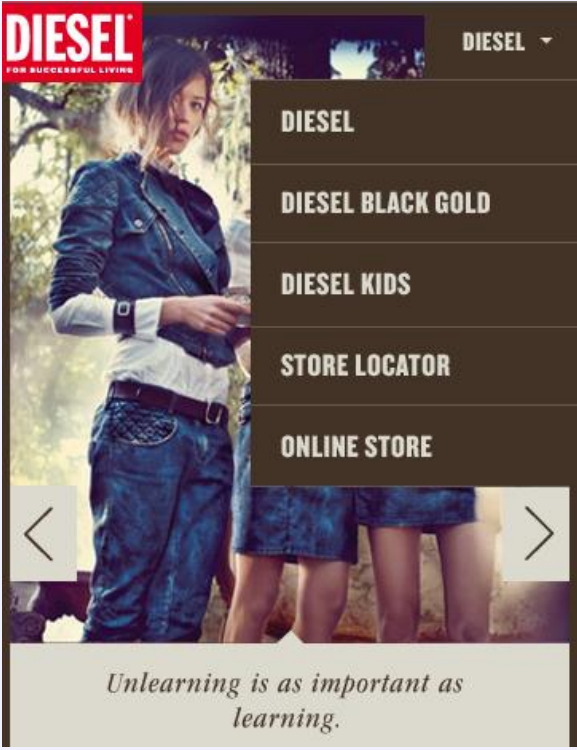

```

También existen librerías un poco más completas para detectar el cliente usando código PHP. En este ejemplo se cubren los casos más básicos, y funcionará en la mayoría de ellos. Pero si queremos una librería más completa podemos consultar: <http://detectmobilebrowsers.mobi/>

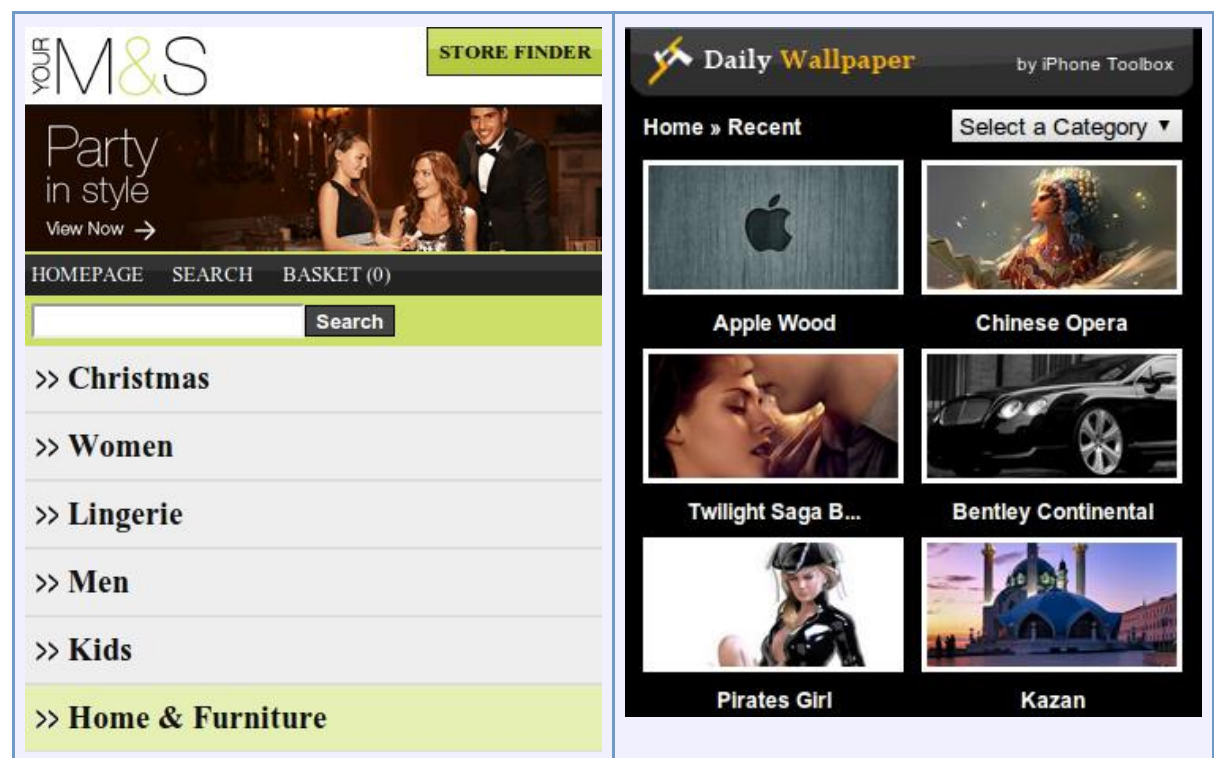
La detección del navegador también la podemos hacer desde el cliente y usando diferentes lenguajes, como por ejemplo JavaScript. En la dirección <http://detectmobilebrowsers.com/> tenemos disponibles librerías que realizan esta función en multitud de lenguajes, como JavaScript, ASP, ASP.NET, C#, Apache, JSP, JQuery, Perl, etc.

1.6. Inspiración

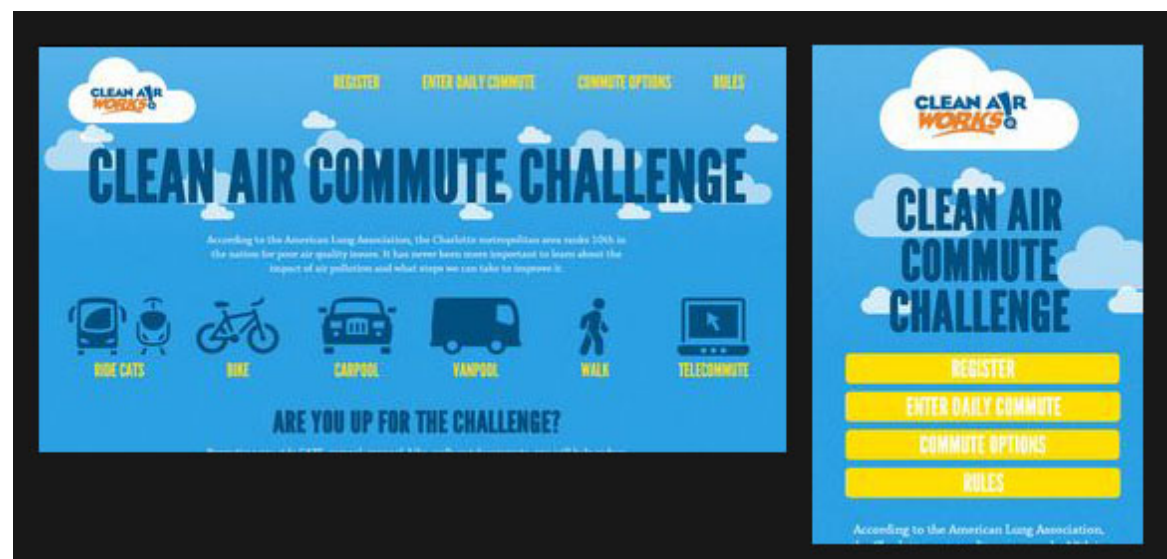
Cuando se empieza en el desarrollo de webs para dispositivos móviles es bueno buscar inspiración, ir a ver que ha hecho gente con más experiencia en el campo. A continuación se incluye una pequeña lista de ejemplos:

http://m.diesel.com 	http://m.flickr.com/ 
http://www.vspink.mobi/	http://mobile.walmart.com/

 <p>PINK VICTORIA'S SECRET</p> <p>PINK NATION Join Now!</p> <p>NEW LOOKS Fall Favorites</p> <p>PINK NFL Team Gear & Freebies</p> <p>GOODIES Mobile Wallpapers</p> <p>FASHION SHOW Pics, Flicks & More</p> <p>SHOP PINK on the Go</p> <p>http://m.marksandspencer.com/</p>	 <p>Walmart</p> <p>What can we help you find?</p> <p>Rollbacks Low prices just got lower.</p> <p>Value of the Day Grab it before it's gone</p> <p>Rollbacks Low prices just got lower</p> <p>Local Ad Find great values at a store nearby</p> <p>Pharmacy Browse \$4 prescriptions, order refills</p> <p>Photo View, share or print your online photos</p> <p>Shopping List Build and manage your shopping lists</p> <p>http://iphonetoolbox.com/dailywallpaper</p>
---	---



En todos de ellos debemos de considerar la adaptación que se ha hecho del contenido entre la web de escritorio y la web para dispositivos móviles. En la siguiente imagen se puede ver una comparación en la que varía la disposición. Pero como hemos dicho, los cambios no son únicamente de disposición, tenemos que recordar la regla del 20%.



2. Instalación de un servidor Web

En la programación Web, una de las herramientas principales que necesitamos es un servidor Web o servidor HTTP. Este es el encargado de compilar el código (según el lenguaje de programación que utilicemos) y enviarlo al cliente utilizando el protocolo de transferencia HTTP.

Dado que instalar un servidor Web completo y configurarlo correctamente es una tarea bastante costosa, para el desarrollo y testeo de aplicaciones en local se suele utilizar un servidor XAMPP. Este es un paquete software de fácil instalación que incluye todo lo necesario para la ejecución de un servidor Web.

Es independiente de plataforma, software libre (licencia GNU), y consiste principalmente en la base de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script: PHP y Perl. El nombre proviene del acrónimo formado por **X** (para cualquiera de los diferentes sistemas operativos), **A**pache, **M**ySQL, **P**HP y **P**erl. Actualmente XAMPP está disponible para Microsoft Windows (WAMPP), GNU/Linux (LAMPP), Solaris y MacOS X (MAMPP).

2.1. XAMPP para Linux

Para su instalación en Linux tendremos que seguir los siguientes pasos:

1. En primer lugar descargamos el software desde:
<http://www.apachefriends.org/en/xampp-linux.html>
2. Abrimos una consola y ejecutamos:

```
sudo tar xvfz xampp-linux-1.7.7.tar.gz -C /opt
```

Ahora ya tenemos instalado el servidor en la ruta '/opt/lampp'.

3. Para inicializar el servidor escribimos:

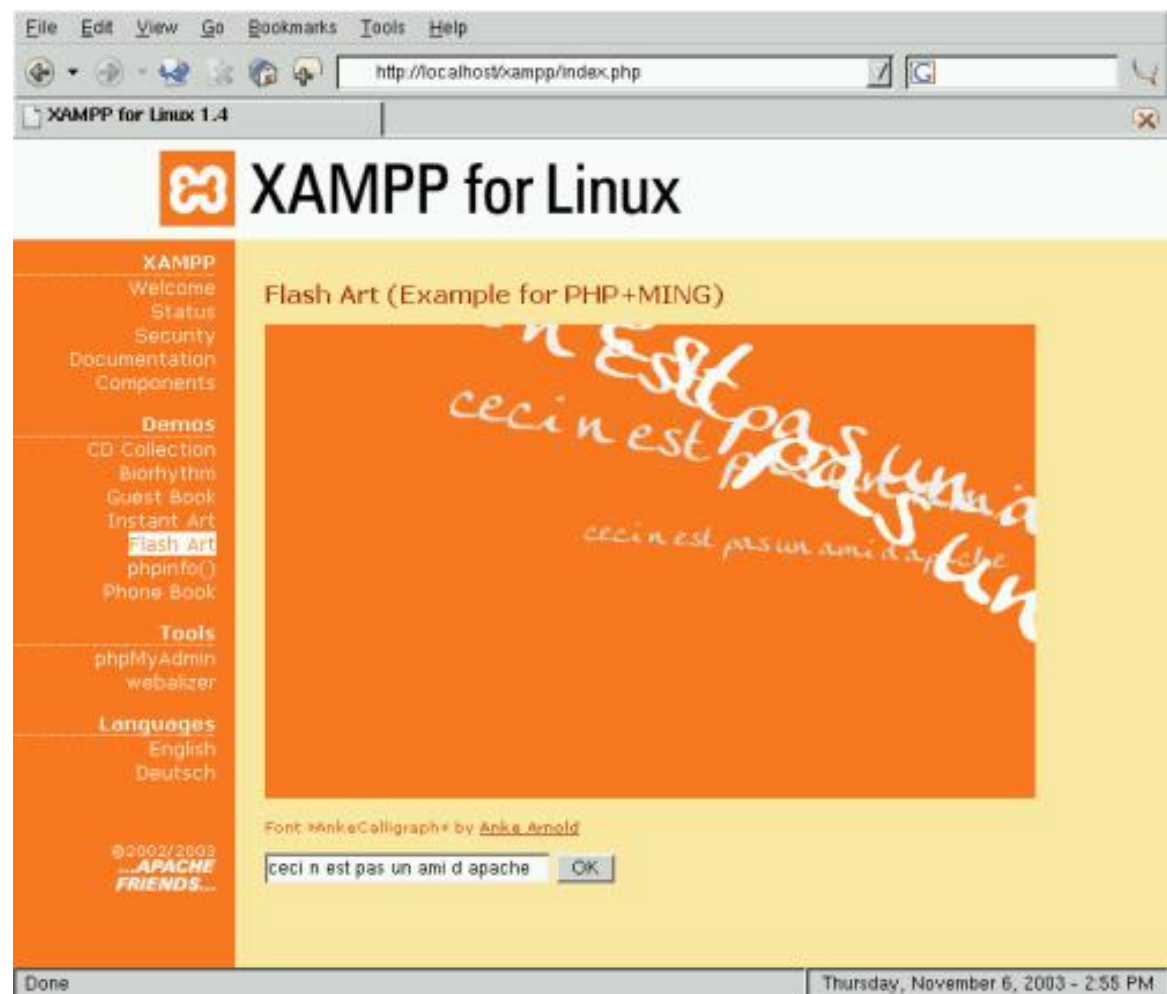
```
sudo /opt/lampp/lampp start
```

Veremos algo como:

```
Starting XAMPP 1.7.7...
LAMP: Starting Apache...
LAMP: Starting MySQL...
LAMP started.
```

Ahora ya tenemos listo nuestro servidor Apache con MySQL.

4. Para comprobar que todo ha ido correctamente abrimos un navegador y escribimos la dirección "<http://localhost>", debería de abrirse una página similar a la siguiente:



El directorio raíz de Apache es “/opt/lampp/htdocs”, que será donde colocaremos nuestras páginas Web.

El servidor viene por defecto sin ninguna opción de seguridad activada (ya que se va a usar en local para pruebas), pero si quisiéramos activarlas tendríamos que escribir:

```
sudo /opt/lampp/lampp security
```

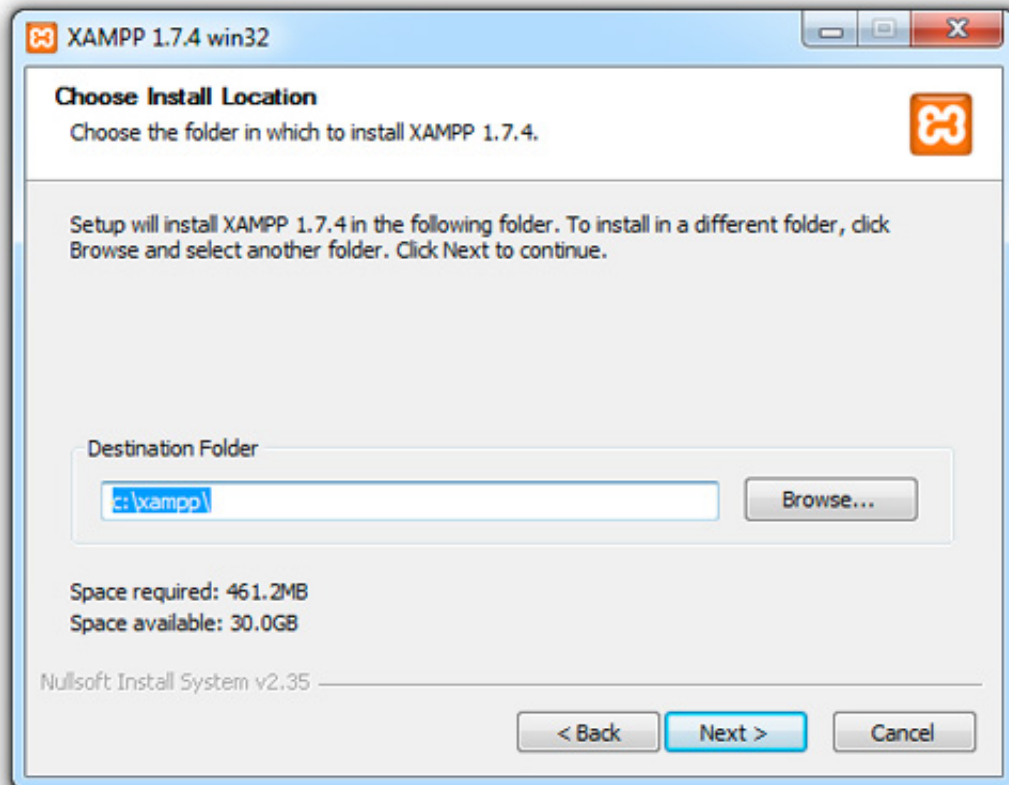
Por último, para detener el servidor Web simplemente ejecutamos en una consola:

```
sudo /opt/lampp/lampp stop
```

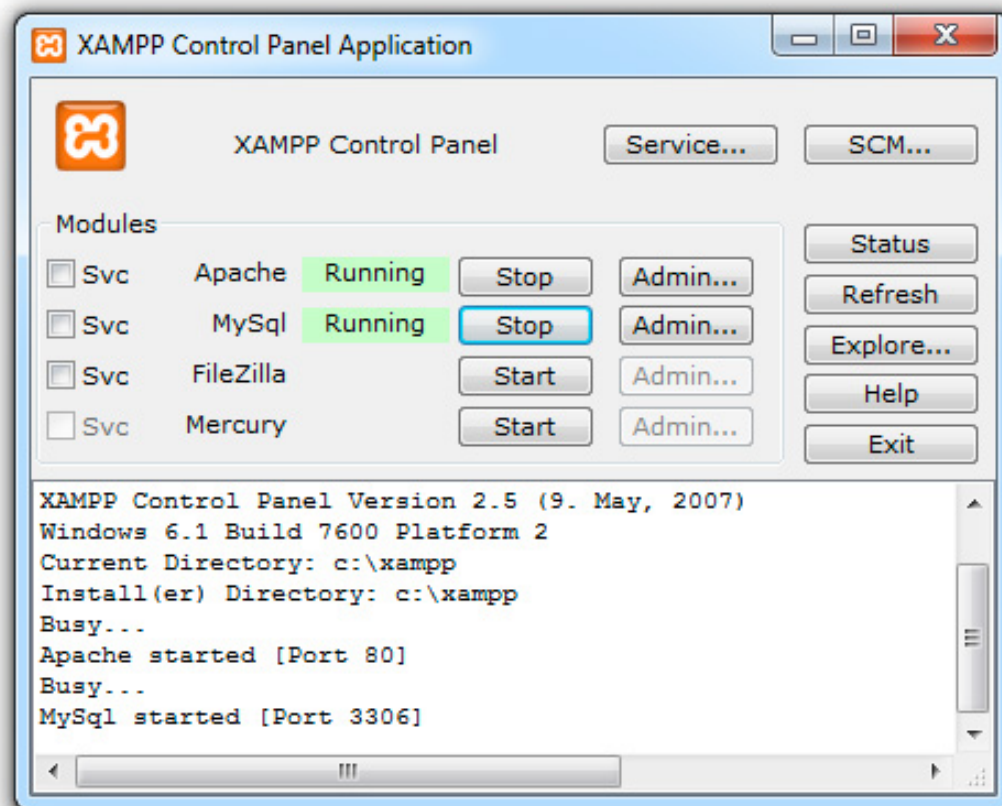
2.2. XAMPP para Windows

Los pasos para instalar XAMPP en Windows son los siguientes:

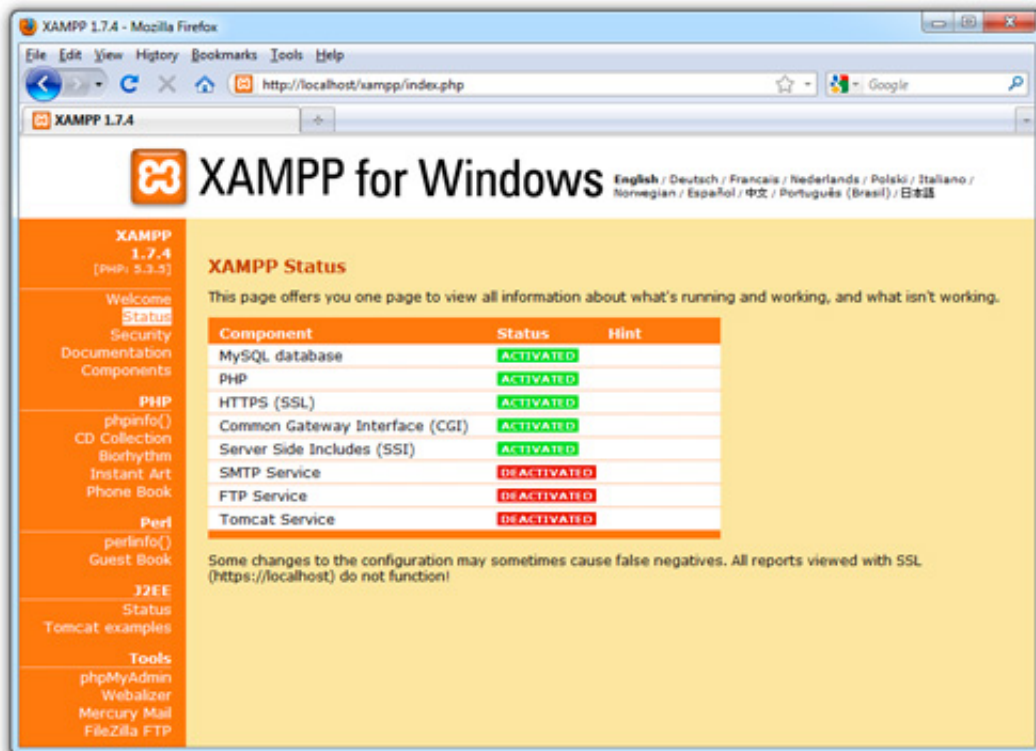
1. Descargamos la última versión del software desde:
<http://www.apachefriends.org/en/xampp-windows.html>
Existen diferentes opciones de descarga, la más sencilla y la que seguiremos aquí es utilizar el “instalador”.
2. Ejecutamos el instalador utilizando los valores por defecto:



3. Después de la instalación abrimos el Panel de Control de XAMPP (se encuentra en el menú de Inicio > Programas > XAMPP). Desde aquí podemos iniciar y detener cada uno de los servicios individualmente.



4. Para comprobar que el servidor está instalado correctamente abrimos un navegador y escribimos la dirección “<http://localhost/>” (o también “<http://127.0.0.1/>”), donde se nos abría una página web como la siguiente:



Si nos avisa el Firewall del sistema tendremos que desbloquear o permitir el acceso a Apache.

En Windows, el directorio raíz de Apache para el contenido Web está en “C:\xampp\htdocs”, que será donde colocaremos nuestras páginas Web.

El servidor viene por defecto sin ninguna opción de seguridad activada (ya que se va a usar en local para pruebas), pero estas opciones se pueden configurar directamente desde un navegador accediendo a la dirección “<http://localhost/security/>”.

2.3. Instalar un servidor Web para Mac

Existen ciertas ventajas del uso de un Mac para el desarrollo de aplicaciones para móviles. Para empezar, el sistema operativo viene con un servidor web Apache instalado. El navegador por defecto, Safari, renderiza correctamente las aplicaciones basadas en WebKit. Y, por su puesto, tiene un excelente simulador para iOS como parte de Xcode.

Si queremos instalar un servidor XAMPP para Mac podemos acceder a la dirección “<http://www.apachefriends.org/en/xampp-macosx.html>” y seguir los pasos de instalación, muy similares a los ya vistos para Linux y Windows.

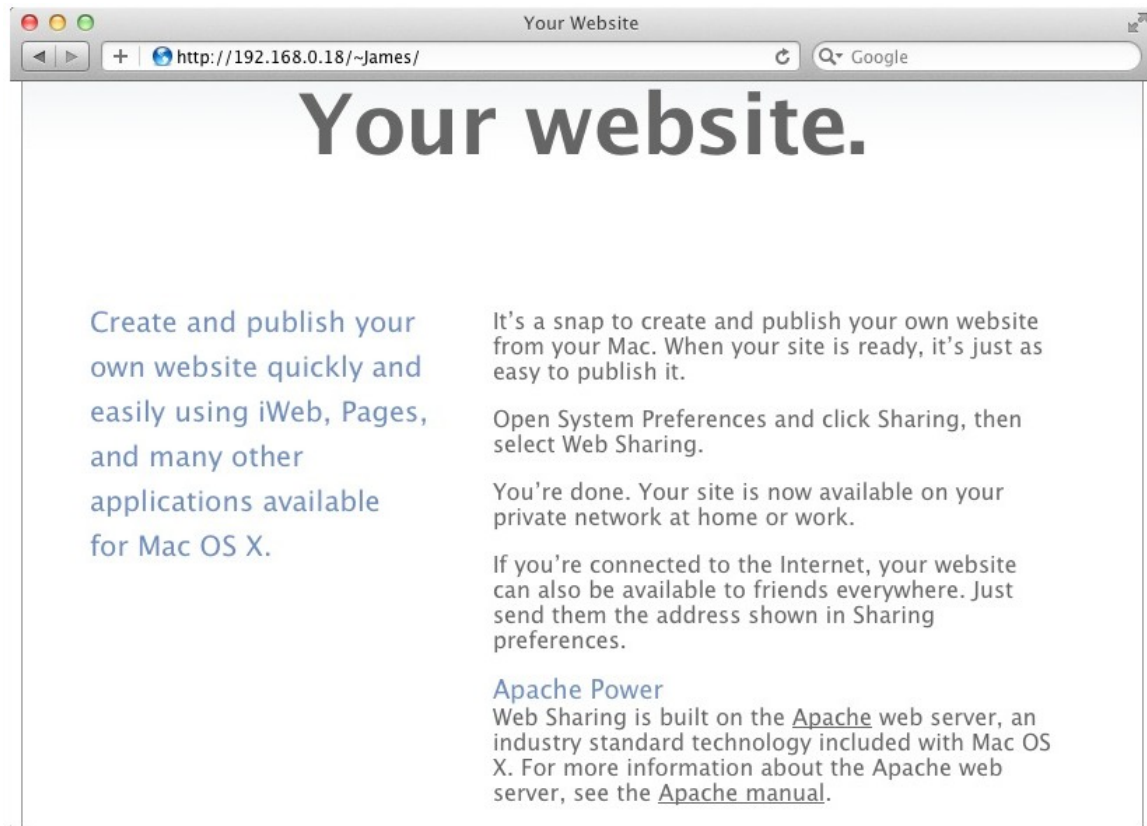
Para este ejemplo vamos a explicar la configuración del servidor Apache que viene con el sistema operativo. En primer lugar abrimos las “Preferencias del Sistema” y vamos al

panel “Sharing”, en el cual deberemos de habilitar la opción “Web Sharing”:



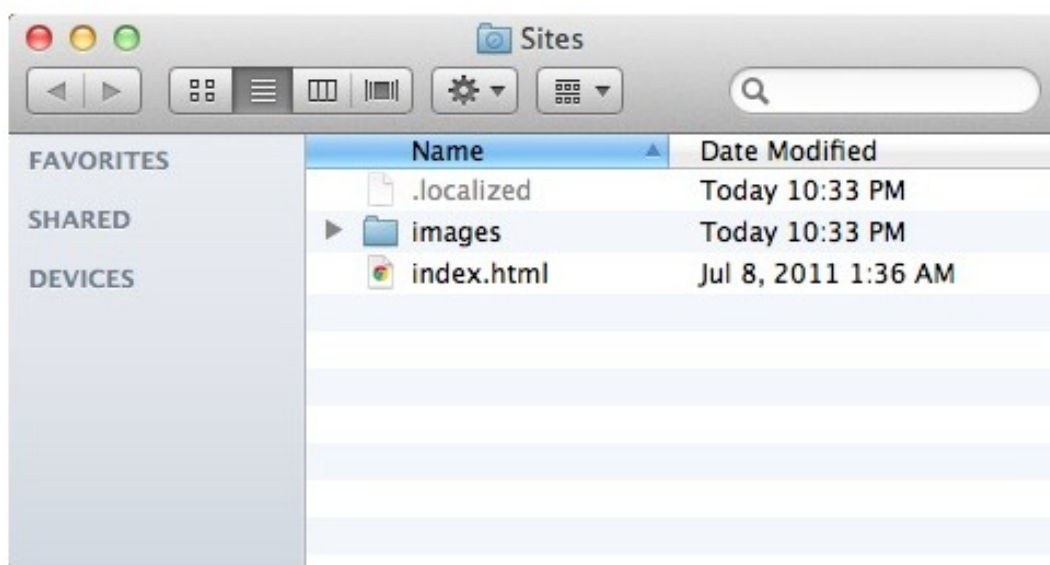
Dependiendo de la versión del sistema operativo que tengamos, aparecerá un botón para abrir la carpeta con el contenido Web y un enlace desde donde podremos comprobar que el servidor se está ejecutando correctamente.

Si lo queremos comprobar nosotros directamente, desde el navegador tendremos que acceder a la dirección IP de nuestra máquina en la red seguida de nuestro nombre de usuario, como podemos ver en la imagen inferior:



Si estuviésemos utilizando un servidor XAMPP desde el navegador podríamos haber accedido directamente a la dirección: “<http://localhost>”.

El directorio para el contenido Web se encuentra normalmente dentro de la carpeta principal con el nombre de “Sites”:

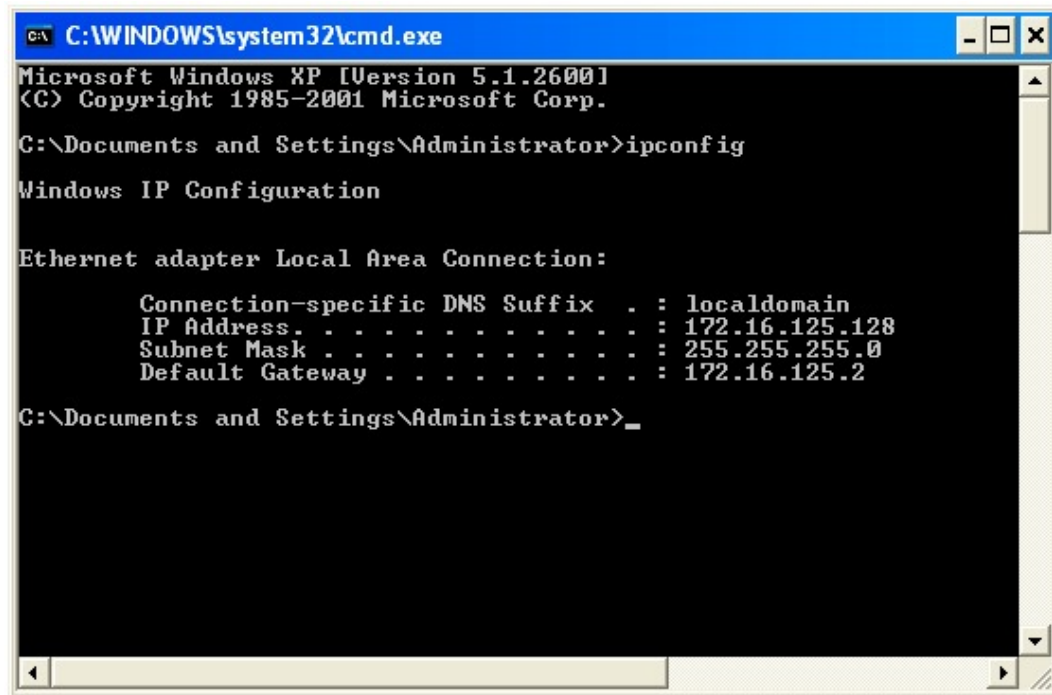


2.4. Acceso mediante un dispositivo móvil real

Si estamos trabajando con el servidor Web disponible en **Mac**, podremos acceder a nuestras páginas Webs de forma externa simplemente conectándonos a la misma red WiFi. Para esto nos tendremos que asegurar que en el ordenador que hace de servidor no se esté ejecutando ningún Firewall que pueda bloquear el acceso desde un cliente remoto. La dirección que tendremos que utilizar será igual que la que usaríamos desde un navegador ejecutado en el mismo ordenador: la dirección IP del servidor en esa red seguida de nuestro nombre de usuario, como podemos ver en la imagen inferior:



Si nuestro servidor está corriendo en una máquina con **Linux o Windows**, entonces tendremos que seguir los siguientes pasos. En primer lugar también nos tendremos que asegurar de que no se esté ejecutando ningún Firewall que pueda bloquear el acceso (si fuera así tendríamos que darle acceso). A continuación obtendremos la dirección IP del servidor. Para esto abrimos un terminal y ejecutaremos el comando “ipconfig” (desde Windows) o “ifconfig” (desde Linux), obteniendo un resultado similar a:



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator>ipconfig

Windows IP Configuration

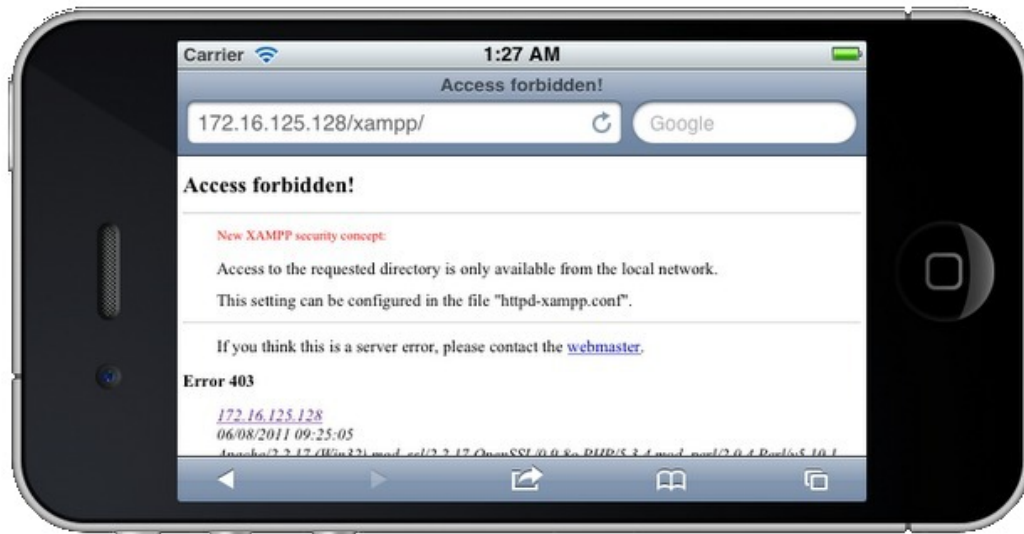
Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . : localdomain
    IP Address. . . . .               : 172.16.125.128
    Subnet Mask . . . . .             : 255.255.255.0
    Default Gateway . . . . .         : 172.16.125.2

C:\Documents and Settings\Administrator>_
```

Utilizando esta dirección IP y estando conectados a la misma red WiFi, ya podremos acceder a nuestro servidor Web desde un dispositivo móvil externo. Simplemente tendremos que escribir la dirección formada como “http://172.16.125.128/mi_web”. Donde en primer lugar colocamos la dirección IP seguida del nombre de la carpeta donde se encuentre nuestro proyecto Web.

Es posible que obtengamos un error de seguridad como el siguiente:



En este caso tendríamos que abrir el fichero “httpd-xampp.conf” que contiene la configuración de nuestro servidor XAMPP. Lo encontraremos en “C:\xampp\apache\conf\extra” en nuestro servidor Windows y en la ruta “/opt/lampp/etc/extra” en Linux. Al final de este fichero podremos ver el siguiente trozo de texto:

```
<LocationMatch
"^(?i:(?:xampp|security|licenses|phpmyadmin|webalizer|server-status|server-info))">
    Order deny,allow
    Deny from all
    Allow from 127.0.0.0/8
    ErrorDocument 403 /error/HTTP_XAMPP_FORBIDDEN.html.var
</LocationMatch>
```

Si no nos importa la seguridad (pues es una red local), podemos abrir el acceso a todos los usuarios cambiando la cuarta línea por “Allow from All”, quedando de la forma:

```
<LocationMatch
"^(?i:(?:xampp|security|licenses|phpmyadmin|webalizer|server-status|server-info))">
    Order deny,allow
    Deny from all
    Allow from all
    ErrorDocument 403 /error/HTTP_XAMPP_FORBIDDEN.html.var
</LocationMatch>
```

Tendremos que reiniciar nuestro servidor Apache para que los cambios tengan efecto (ver en las secciones anteriores) y ya tendremos acceso desde nuestro dispositivo móvil.

2.5. Instalación del SDK de Android

Si queremos utilizar el emulador de Android para testear nuestras Webs tendremos que

instalar el SDK de Android completo. Para esto descargamos el software desde “<http://developer.android.com/sdk/index.html>” y procedemos a su instalación. Este proceso simplemente requiere que descomprimamos el archivo descargado en una carpeta y ejecutemos el SDK Manager (tools/android) para empezar a trabajar.

Para poder realizar simulaciones y visualizar nuestras páginas Web tendremos que crear un dispositivo virtual de Android (Android Virtual Device o AVD).

Si utilizamos Eclipse como entorno de desarrollo, podemos instalar el plugin de Android para Eclipse (ADT Plugin), el cual lo podremos encontrar en la dirección “<http://developer.android.com/sdk/eclipse-adt.html>”. Este plugin integrará el SDK y nuestro simulador en Eclipse.

2.6. Instalar Xcode

Si trabajamos con el sistema operativo de Mac podremos hacer uso del IDE Xcode y de los emuladores de iPhone e iPad que incorpora para testear nuestras aplicaciones en local. Este software lo podemos descargar desde el Apple Developer Center accediendo a la dirección “<http://developer.apple.com/xcode/>”.

2.7. Simuladores y Emuladores

Los simuladores solo tratan de reproducir el comportamiento, para que el resultado se parezca al que obtendríamos con una ejecución real. Los emuladores, por su parte, modelan de forma precisa el dispositivo (hardware y S.O.), de manera que este funcione como si estuviese siendo usado en el aparato original.

Podemos encontrar algunos simuladores online mediante los cuales realizar pruebas rápidas de nuestros proyectos, como:

- [iPhone 4 Simulator](#)
- [Test iPhone](#)
- [iPhone Tester](#)
- [Opera Mini Simulator](#)
- [Emuladores para Nokia N70 y Sony K750](#)

Para poder usar estos simuladores tendremos que tener nuestro proyecto en un servidor Web para poder acceder a través de la dirección de localhost o de la IP.

Para una completa guía de los emuladores disponibles podemos consultar: <http://www.mobilexweb.com/emulators>.

3. HTML

HTML, siglas de *HyperText Markup Language* (“lenguaje de marcado de hipertexto”), es

el lenguaje de marcado predominante para la elaboración 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. El código HTML se escribe en forma de “etiquetas”, mediante las cuales podemos describir la estructura lógica y apariencia del contenido. La apariencia que podemos describir con HTML está bastante limitada, pero el código se puede complementar y mejorar mediante el uso de otros lenguajes como JavaScript o CSS.

3.1. Editores HTML

El lenguaje HTML puede ser creado y editado con cualquier editor de textos básico, como puede ser Gedit en Linux o el Bloc de notas de Windows. Existen además otros editores para la realización de sitios web con características WYSIWYG (What You See Is What You Get, o en español: “lo que ves es lo que obtienes”). Estos editores permiten ver el resultado de lo que se está editando en tiempo real, a medida que se va desarrollando el documento. Ahora bien, esto no significa una manera distinta de realizar sitios web, sino que una forma un tanto más simple ya que estos programas, además de tener la opción de trabajar con la vista preliminar, tiene su propia sección HTML, la cual va generando todo el código a medida que se va trabajando. Algunos ejemplos de editores son Adobe Dreamweaver, KompoZer o Microsoft FrontPage.

Estos editores aceleran o facilitan la creación de código HTML, pero en algunas ocasiones también generan mucho más código del necesario (como es el caso de Microsoft FrontPage). Lo ideal es tener un control total sobre el código que se escribe y utilizar estos editores sólo como una pequeña ayuda. También podemos utilizar otro tipo de editores que simplemente comprueben que el código HTML escrito es correcto (que las etiquetas y atributos son correctos, las etiquetas se cierran correctamente, etc.).

3.2. Etiquetas

Las etiquetas HTML deben de ir encerradas entre corchetes angulares “<>”, y pueden ser de dos tipos:

- Se abren y se cierran, como por ejemplo: negrita o <p>texto</p>.
- Sólo se abren, como
 o <hr/>.

En caso de que no cerremos una etiqueta que deba ser cerrada se producirá un error en la estructura del documento y probablemente también genere errores en la visualización.

Hay etiquetas que además pueden contener atributos, en este caso los atributos se deben de colocar en la etiqueta de inicio, de la forma:

```
<etiqueta atributo1="valor1" atributo2="valor2">...</etiqueta>
```

O para las etiquetas de solo apertura:


```
<etiqueta atributo1="valor1" atributo2="valor2"/>
```

3.3. Estructura básica de una Web

Un documento HTML comienza con la etiqueta **<html>** y termina con **</html>**. Dentro del documento (entre las etiquetas de principio y fin de html) hay dos zonas bien diferenciadas: el encabezamiento, delimitado por **<head>** y **</head>**, que sirve para incluir definiciones iniciales válidas para todo el documento; y el cuerpo, delimitado por **<body>** y **</body>**, donde reside la información del documento.

Las etiquetas básicas o mínimas son:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Ejemplo</title>
</head>
<body>
  ¡Hola mundo!
</body>
</html>
```

La primera línea es el DOCTYPE, o el tipo de documento que viene a continuación. En este caso se usa el estándar de HTML 4.01 (el último estándar adoptado en 1999, ya que HTML5 a fecha de 2011 sigue siendo un borrador). La siguiente etiqueta, **<html>**, define el inicio del documento HTML, e indica que lo que viene a continuación debe ser interpretado como código HTML. Como podemos ver en la última línea, se cierra la etiqueta **</html>**.

3.4. Elementos de la cabecera

La cabecera contiene información sobre el documento que no se muestra directamente al usuario. Como por ejemplo el título de la ventana del navegador, los metadatos o la hoja de estilo utilizada. Dentro de la cabecera **<head></head>** podemos encontrar:

- **<title></title>**: define el título de la página. Por lo general el título aparece en la barra de título encima de la ventana.
- **<link>**: para vincular el sitio con hojas de estilo (ver la sección de CSS para más información):

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

El atributo “rel” es requerido y describe el tipo de documento enlazado (en este caso una hoja de estilo). El atributo “type” es simplemente indicativo del tipo de hoja de estilo enlazada (en este caso CSS).

- **<style></style>**: se utiliza para añadir definición de estilo en línea. No es necesario colocarlo si se va a utilizar una hoja de estilo externa usando la etiqueta **<link>** (que

es lo más habitual y recomendable). El uso correcto sería de la forma:

```
<html>
<head>
  ...
  <style type="text/css">
    Estilos CSS
  </style>
</head>
<body></body>
</html>
```

Para más información ver la sección CSS del manual.

- **<meta/>**: para indicar metadatos como la descripción de la web, los keywords, o el autor:

```
<meta name="description" content="Descripción de la web" />
<meta name="keywords" content="key1,key2,key3" />
<meta name="author" content="Nombre del autor" />
```

- **<script></script>**: permite incluir un script en la Web. El código se puede escribir directamente entre las etiquetas de <script> o cargar desde un fichero externo, indicando mediante el atributo src="url del script" la dirección del fichero. Se recomienda incluir el tipo MIME en el atributo type, que en el caso de código JavaScript sería "text/javascript". Ejemplos:

```
<script src="fichero.js" type="text/javascript"></script>

<script type="text/javascript">
  Código de un script integrado en la página
</script>
```

Cuando usamos el atributo "src" el contenido de estas etiquetas está vacío (no encierra nada), esto es porque lo carga directamente desde el fichero indicado.

3.5. Etiquetas básicas HTML

Las etiquetas HTML que utilizaremos para crear el contenido de nuestra página deben de ir dentro de la sección <body></body>.

Algunas de las etiquetas HTML que más se suelen utilizar son:

- **<h1></h1>** a **<h6></h6>**: encabezados o títulos del documento con diferente relevancia, siendo <h1> la cabecera de mayor nivel.
- **<p></p>**: definición de un párrafo.
- **
**: salto de línea.
- ****: texto en negrita (etiqueta desaprobada. Se recomienda usar la etiqueta ****).
- **<i></i>**: texto en cursiva (etiqueta desaprobada. Se recomienda usar la etiqueta ****).
- **<s></s>**: texto tachado (etiqueta desaprobada. Se recomienda usar la etiqueta ****).

- `<u></u>`: texto subrayado.
- `<center></center>`: texto centrado.
- `<pre></pre>`: texto preformateado, respeta los espacios y saltos de línea.
- ``: Superíndice
- ``: Subíndice
- `<blockquote></blockquote>`: Indica una cita textual, se representa como un párrafo indexado con respecto al margen.
- `<hr/>`: Línea horizontal, usada, por ejemplo, para separar diferentes secciones.
- `<!-- comentario -->`: Comentarios en HTML. El texto del comentario no será visible en el navegador.
- ``: Esta etiqueta no aplica ningún formato por sí misma, sino que provee una forma de definir un estilo o formato a un trozo de texto. Se utiliza junto con una hoja de estilo. Por ejemplo, lo podemos utilizar para marcar palabras en algún color o con algún formato especial.

3.6. Listas

Para definir una lista utilizamos las siguientes etiquetas:

- ``: Lista ordenada (con numeración).
- ``: Lista con puntos (o viñetas).
- ``: Elemento de una lista (tanto numerada como no numerada). Esta etiqueta debe de estar entre las etiquetas `` o ``.

Ejemplo de lista:

```
<ol>
  <li>Elemento 1</li>
  <li>Elemento 2</li>
</ol>
```

3.7. Tablas

Las tablas se definen básicamente mediante tres etiquetas:

- `<table></table>`: define una tabla.
- `<tr></tr>`: fila de una tabla, debe de estar dentro de las etiquetas de una tabla.
- `<td></td>`: celda de una tabla, debe estar dentro de una fila.

Ejemplo de una tabla:

```
<table>
  <tr>
    <td>Fila 1 izquierda</td>
    <td>Fila 1 derecha</td>
  </tr>
  <tr>
    <td>Fila 2 izquierda</td>
```

```
<td>Fila 2 derecha</td>
</tr>
</table>
```

Además también podemos utilizar la etiqueta `<th>` en lugar de `<td>` para indicar una celda de “cabecera”, de esta forma el contenido será resaltado en negrita y en un tamaño ligeramente superior al normal.

En la etiqueta de apertura `<table>` podemos utilizar los siguientes atributos:

- **border=“num”**: Ancho del borde de la tabla en puntos. Si indicamos `border=“0”` tendremos una tabla cuyas divisiones no serán visibles, esta propiedad se suele utilizar para distribuir los elementos en una página Web.
- **cellspacing=“num”**: Espacio en puntos que separa las celdas que están dentro de la tabla.
- **cellpadding=“num”**: Espacio en puntos que separa el borde de cada celda y el contenido de esta.
- **width=“num”**: Indica la anchura de la tabla en puntos o en porcentaje en función del ancho de la ventana. Si no se indica este parámetro, el ancho dependerá de los contenidos de las celdas.
- **height=“num”**: Indica la altura de la tabla en puntos o en porcentaje en función del alto de la ventana. Si no se indica este parámetro, la altura dependerá de los contenidos de las celdas.
Este atributo también se puede utilizar en las etiquetas `<tr>` para indicar la altura de cada fila de forma individual.

En las etiquetas de apertura de celda (`<td>` o `<th>`) podemos utilizar los siguientes atributos:

- **align=“pos”**: Indica como se debe alinear el contenido de la celda, a la izquierda (left), a la derecha (right), centrado (center) o justificado (justify).
- **valign=“pos”**: Indica la alineación vertical del contenido de la celda, en la parte superior (top), en la inferior (bottom), o en el centro (middle).
- **rowspan=“num”**: Indica el número de filas que ocupará la celda. Por defecto ocupa una sola fila. Este atributo se utiliza para crear celdas “multifila”, es decir, una celda que por ejemplo ocupe 3 filas. Tendremos que tener en cuenta que esa celda no se deberá de definir en las siguientes 2 filas (para esas filas se definirá una celda menos).
- **colspan=“num”**: Indica el número de columnas que ocupará la celda. Por defecto ocupa una sola columna. Este atributo se utiliza para crear celdas “multicolumna”, es decir, una celda que por ejemplo ocupe 3 columnas. Tendremos que tener en cuenta que en esa fila tendremos que definir 2 celdas menos.
- **width=“num”**: Indica la anchura de la columna en puntos o en porcentaje en función del ancho de la ventana. Si no se indica este parámetro, el ancho dependerá del tamaño de los contenidos.

3.8. Cajas (etiqueta `<div>`)

La etiqueta `<div></div>` permite crear cajas contenedoras. Estas cajas se utilizan para organizar la disposición de los elementos en la página. Es muy sencillo indicar su posición de forma absoluta o relativa en la página y crear divisiones del espacio para distribuir los elementos. Estas cajas pueden contener cualquier tipo de elemento (texto, imágenes, etc.) u otras etiquetas `<div>` para crear subdivisiones. Se recomienda su uso junto con CSS, en vez de la etiqueta `<table>`, cuando se desea alinear contenido. Para más información consultar la sección sobre CSS.

3.9. Enlaces

Los enlaces permiten vincular partes del documento con otros documentos o con otras partes del mismo documento. Por ejemplo, que al pulsar con el ratón sobre un texto o sobre una imagen se nos redirija a una nueva Web con un contenido diferente.

Para crear un enlace se utiliza la etiqueta `` cuyo atributo href establece la dirección URL a la que apunta el enlace. Por ejemplo, un enlace a la Wikipedia sería de la forma:

```
<a href="es.wikipedia.org">Wikipedia</a>
```

También se pueden crear enlaces sobre otros objetos, tales como imágenes, de la forma:

```
<a href="dirección_URL"></a>
```

La etiqueta de enlace `<a>` también permite el atributo `target="_blank"`, mediante el cual indicamos que el enlace se tiene que abrir en una nueva ventana o en una pestaña nueva del navegador.

3.10. Imágenes

Para incluir una imagen se utiliza la etiqueta ``, la cual requiere el atributo `src` con la ruta en la que se encuentra la imagen. Es conveniente poner siempre el atributo `alt="texto alternativo"`, el cual indica el texto a mostrar en caso de no poder cargar la imagen y también se utiliza para opciones de accesibilidad.

Por ejemplo, para cargar una imagen de cabecera:

```

```

Además existen otros atributos interesantes como `width` y `height` para redefinir el ancho y la altura de la imagen.

3.11. Formularios

Los formularios permiten solicitar información al visitante de una página Web. Están

compuestos por campos de diferente tipo, cuya información se enviará a una dirección URL (indicada en el código) al pulsar el botón de envío.

La declaración de formulario queda recogida por las etiquetas **<form>****</form>**, los cuales deben encerrar la definición de todos los campos del formulario. En la etiqueta de apertura **<form>** tenemos que indicar los atributos básicos:

- **action=“”**: Entre comillas se indica la acción a realizar al enviar el formulario. En general se indicará el nombre de un fichero alojado en el servidor, el cual se encargará de procesar la información. Aunque también se le puede indicar una dirección de correo para que envíe directamente todo el contenido, de la forma: **“mailto:direccion_de_correo”**.
- **method=“”** (post o get): Indica el método de transferencia de las variables. El método **“post”** envía los datos de forma no visible, mientras que el método **“get”** los adjunta a la URL a la que se redirige.
- **enctype= “”**: Especifica el tipo de codificación de la información enviada. Con **method= “get”** no se realiza codificación, solo se cambian caracteres especiales como el espacio, por lo que no es necesario indicar **enctype**. Cuando el valor del atributo **“method”** es **“post”**, podemos utilizar los siguientes valores:
 - **application/x-www-form-urlencoded**: Es el valor predeterminado. Codifica todos los caracteres antes de enviarlos.
 - **multipart/form-data**: Es requerido al enviar archivos mediante un formulario. No codifica la información.
 - **text/plain**: No codifica la información, solo cambia los espacios por el símbolo **“+”**.

Tipos de campos básicos

Para añadir campos al formulario se utiliza la etiqueta **<input/>**, esta etiqueta debe de tener siempre dos atributos:

- **name=“”**: Indica el nombre que se asigna a un determinado campo. Este nombre no aparece visible en la Web, pues se utiliza para poder distinguir cada campo al enviar la información al servidor o por correo. Es como si fuera el nombre de la variable a la que se asigna el valor del campo.
- **type=“”**: Indica el tipo de campo a utilizar. Puede ser de muchos tipos: text, password, checkbox, radio, file, hidden, submit, reset.

A continuación se describen más detalladamente los diferentes tipos de campos **<input/>** según su valor *type*:

- **type=“text”**: campo de tipo texto de una línea. Sus atributos son:
 - **maxlength= “”**: Seguido de un valor que limitará el número máximo de caracteres.
 - **size= “”**: Seguido de un valor que limitará el número de caracteres a mostrar en pantalla. A diferencia de **maxlength** este atributo no limita la longitud del texto que se puede introducir, sino que modifica el tamaño visible del campo.
 - **value= “”**: Indica el valor inicial del campo.

- **type="password"**: Este campo funciona exactamente igual que el de tipo "text", pero ocultará el texto introducido cambiando las letras por asteriscos o puntos. Sus atributos son los mismos que para "text".
- **type="checkbox"**: Este campo mostrará una casilla cuadrada que nos permitirá marcar opciones de una lista (podremos marcar varias opciones a la vez). Para indicar que varias casillas pertenecen al mismo grupo se les debe de dar el mismo nombre para el atributo "name". El texto que queramos que aparezca a continuación de la casilla del "checkbox" se tendrá que escribir después de cerrar la etiqueta <input/>. Sus atributos son:
 - *value=""*: Define el valor que será enviado si la casilla está marcada.
 - *checked*: Este atributo es opcional, y hace que la casilla aparezca marcada por defecto. No necesita indicarle ningún valor.

Ejemplo:

```
<input type="checkbox" name="option1" value="leche"/> Leche<br/>
<input type="checkbox" name="option1" value="pan" checked/> Pan<br/>
<input type="checkbox" name="option1" value="queso"/> Queso<br/>
```

- **type="radio"**: El campo se elegirá marcando de entre varias opciones una casilla circular. Al marcar una casilla el resto de casillas de ese grupo de desmarcarán automáticamente. Para indicar que varias casillas pertenecen al mismo grupo se les debe de dar el mismo nombre para el atributo "name" (ver ejemplo). Además debemos de indicar:
 - *value=""*: Define el valor que será enviado si la casilla está marcada.
 - *checked*: Este atributo es opcional, y hace que la casilla aparezca marcada por defecto. Solo se podrá usar para una casilla. No necesita indicarle ningún valor.

Ejemplo:

```
<input type="radio" name="group1" value="leche"/> Leche<br/>
<input type="radio" name="group1" value="pan" checked/> Pan<br/>
<input type="radio" name="group1" value="queso"/> Queso<br/>
```

- **type="file"**: El atributo file permite al usuario subir archivos. Necesitaremos un programa que gestione estos archivos en el servidor mediante un lenguaje diferente al HTML. El único atributo opcional que podemos utilizar es *size=""* mediante el cual podremos indicar la anchura visual de este campo. Ejemplo:

```
<input type="file" name="datafile" size="40"/>
```

- **type="hidden"**: Este valor no puede ser modificado, pues permanece oculto. Se suele utilizar para enviar al método encargado de procesar el formulario algún dato adicional necesario para su procesamiento. Para indicar el valor de este campo utilizamos el atributo: *value = "valor"*.
- **type="submit"**: Representa el botón de "Enviar". Al pulsar este botón la información de todos los campos se enviará realizando la acción indicada en <form>. Mediante el atributo:
 - *value="texto"*: podemos indicar el texto que aparecerá en el botón.

- **type="reset"**: Al pulsar este botón se borra el contenido de todos los campos del formulario. Mediante el atributo:
 - **value="texto"**: podemos indicar el texto que aparecerá en el botón.

Campos de Selección

Mediante la etiqueta `<select></select>` podemos crear listas de opciones, que nos permitirán seleccionar entre una o varias de ellas. Sus atributos son:

- **name=""**: Nombre del campo.
- **size=""**: Número de opciones visibles a la vez. Si se indica 1 se presentará como un menú desplegable, si se indica mas de uno aparecerá como una lista con barra de desplazamiento.
- **multiple**: Permite seleccionar mas de un valor para el campo.

Las diferentes opciones de la lista se indicarán mediante la etiqueta `<option></option>`. El nombre que se visualizará debe de indicarse dentro de estas etiquetas. Mediante el atributo **value=""** podemos indicar el valor que se enviará con el formulario. También podemos utilizar el atributo **selected** para indicar la opción seleccionada por defecto. Si no lo especificamos, siempre aparecerá como seleccionado el primer elemento de la lista.

```
<select name="Colores" multiple>
  <option value="r">Rojo</option>
  <option value="g">Verde</option>
  <option value="b">Azul</option>
</select>

<select name="Colores" SIZE="1">
  <option value="r">Rojo</option>
  <option value="g" selected>Verde</option>
  <option value="b">Azul</option>
</select>
```

Áreas de texto

Mediante las etiquetas `<textarea></textarea>` podemos crear un campo de texto de múltiples líneas. Los atributos que podemos utilizar son:

- **name=""**: Nombre del campo.
- **cols="num"**: Número de columnas de texto visibles.
- **rows="num"**: Número de filas de texto visibles.

3.12. Eventos

Los eventos permiten ejecutar acciones cuando sucede un determinado evento o se realiza una determinada acción. La forma de definirlos es similar a los atributos (evento="acción"), la acción hará referencia a una función o método en lenguaje JavaScript. Algunos de los eventos que podemos utilizar son:

- **onload**: se activa cuando el navegador termina de cargar todos los elementos de la

página.

- **onunload**: se activa al cerrar una página.
- **onclick**: cuando se presiona el botón del ratón sobre un elemento.
- **ondblclick**: se activa al hacer doble clic sobre un elemento.
- **onmousedown**: se activa al presionar el botón del ratón (mientras que está presionado).
- **onmouseup**: cuando el botón del ratón es liberado.
- **onmouseover**: se dispara cuando el cursor del ratón pasa sobre un elemento.
- **onmousemove**: cuando se mueve el cursor del ratón mientras está sobre un elemento.
- **onmouseout**: se activa cuando el cursor del ratón sale fuera de un elemento (sobre el que estaba).
- **onfocus**: ocurre cuando un elemento recibe el enfoque (el cursor de escritura), ya sea con el puntero o con mediante la tecla tabulador.
- **onblur**: se dispara cuando un elemento pierde el enfoque (ya sea por hacer clic fuera o por presionar la tecla tabulador).
- **onkeypress**: ocurre cuando se presiona una tecla (dentro de un elemento, por ejemplo un campo de escritura).
- **onkeydown**: se dispara cuando una tecla es presionada (dentro de un elemento)
- **onkeyup**: cuando una tecla es soltada.
- **onsubmit**: se activa cuando un formulario es enviado.
- **onreset**: ocurre cuando un formulario es reseteado.
- **onselect**: cuando el usuario selecciona un texto en un campo de texto.
- **onchange**: ocurre cuando un control pierde el enfoque y su valor ha sido modificado desde que recibió el enfoque.

Ejemplo de uso:

```
<script type="text/javascript">
  function saveText() {
    // acciones JavaScript
  }
</script>

<textarea id="myarea" cols="80" rows="15" onkeyup="saveText()"></textarea>
```

3.13. Símbolos HTML

Los caracteres especiales como signo de puntuación, letras con tilde o diéresis, o símbolos del lenguaje; se deben convertir en entidades HTML para que se muestren correctamente en un navegador. La siguiente es una lista de caracteres españoles junto con algunos símbolos especiales y su correspondiente entidad HTML:

á	á		Á	Á
é	é		É	É
í	í		Í	Í

ó	ó		Ó	Ó
ú	ú		Ú	Ú
ü	ü		Ü	Ü
ñ	ñ		Ñ	Ñ
espacio en blanco	 		€	€
< (Menor que)	<		> (Mayor que)	>
&	&		° (grados)	°

Algunos servidores realizan esta conversión automáticamente, pero en general es aconsejable escribir los símbolos directamente. Para obtener una lista mucho más completa de símbolos podemos buscar en Google: “HTML symbols” o visitar la siguiente dirección “<http://www.ascii.cl/htmlcodes.htm>”.

4. CSS

El nombre hojas de estilo en cascada viene del inglés *Cascading Style Sheets*, del que toma sus siglas. CSS es un lenguaje usado para definir la presentación o la apariencia de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). CSS se creó para separar el contenido de la forma, a la vez que permite a los diseñadores mantener un control mucho más preciso sobre la apariencia de las páginas. El W3C (*World Wide Web Consortium*) es el encargado de formular la especificación de las hojas de estilo que sirven de estándar para los navegadores.

En versiones antiguas de HTML se debía de añadir el formato dentro de las propias etiquetas, para indicar por ejemplo su color o tamaño. Esto obligaba a tener que especificar el mismo formato en todas las etiquetas para tener un diseño consistente, además, al cambiar un formato también había que cambiarlo para todas las etiquetas.

Cuando se utiliza CSS, las etiquetas HTML no deberían proporcionar información sobre cómo serán visualizadas. La información de la hoja de estilo será la que especifique cómo se han de mostrar: color, fuente, alineación del texto, tamaño, etc.

Las ventajas de utilizar CSS (u otro lenguaje de estilo) son:

- Control centralizado de la apariencia de un sitio web completo, con lo que se agiliza de forma considerable la actualización del mismo.
- Los navegadores permiten a los usuarios especificar su propia hoja de estilo local, que será aplicada a un sitio web, con lo que aumenta considerablemente la accesibilidad. Por ejemplo, personas con deficiencias visuales pueden configurar su propia hoja de estilo para aumentar el tamaño del texto o remarcar más los enlaces.
- Una página puede disponer de diferentes hojas de estilo según el dispositivo que la muestre o, incluso, a elección del usuario. Por ejemplo, para ser impresa o mostrada

en un dispositivo móvil.

- El documento HTML en si mismo es más claro de entender y se consigue reducir considerablemente su tamaño.

4.1. Adjuntar una hoja de estilo

La información de estilo puede ser adjuntada de tres formas diferentes:

- **Hoja de estilo externa:** es una hoja de estilo que está almacenada en un archivo diferente al archivo HTML (por ejemplo llamado “estilo.css”). Esta es la manera de programar más potente y la que deberemos de utilizar por defecto, porque separa completamente las reglas de formateo para la página HTML. La hoja de estilo debe de ser enlazada con el código HTML de la forma:

```
<html>
<head>
  <link rel="stylesheet" type="text/css" href="estilo.css"/>
  ...
</head>
...
```

- **Hoja de estilo interna:** es una hoja de estilo que está incrustada dentro del documento HTML. En general, la única vez que se usa una hoja de estilo interna, es cuando se quiere diferenciar con algún estilo uno de los ficheros HTML de nuestra Web. Este código debe de estar incluido en la sección de cabecera y entre las etiquetas <style>. Las etiquetas de comentario “<!-- ... -->” sirven para que los navegadores antiguos, que no soportan CSS, no incluyan ese texto en el cuerpo de la página. La forma de incluir este código sería de la forma:

```
<html>
<head>
  <STYLE type="text/css">
    <!--
      H1 {color:blue; text-align:center}
    // -->
  </STYLE>
</head>
...
```

- **Estilo en línea (inline):** es un método para insertar el lenguaje de CSS directamente dentro de una etiqueta HTML. Esta manera de proceder no es totalmente adecuada. El incrustar la descripción del formateo dentro del documento de la página Web, a nivel de código, se convierte en una manera larga, tediosa y poco elegante de resolver el problema de la programación de la página. Este modo de trabajo se podría usar de manera ocasional si se pretende aplicar un formateo con prisa, al vuelo. La forma de incluir un estilo inline sería:

```
<h1 style="color:blue; text-align:center">...</h1>
```

4.2. Definición de estilos para etiquetas HTML

Si lo que queremos es dar formato o redefinir una etiqueta HTML existente, usaríamos la sintaxis:

```
etiqueta {  
  estilo CSS 1;  
  estilo CSS 2;  
  ...  
}
```

En "etiqueta" pondríamos el nombre de la etiqueta (por ejemplo "h1", "p", etc. pero sin los signos < >) y los estilos que definirían esa etiqueta irían encerrados entre las llaves "{...}".

También podemos redefinir varias etiquetas a la vez, separándolas por comas, de la forma:

```
etiqueta1, etiqueta2, etiqueta3 {  
  estilos CSS  
}
```

O redefinir etiquetas "dentro" de otras etiquetas. En este caso el estilo CSS solo se aplicará cuando la etiqueta redefinida se encuentre dentro de la etiqueta contenedora:

```
contenedor etiqueta {  
  estilos CSS  
}
```

Por ejemplo, una etiqueta dentro de una sección <p>:

```
p span {  
  estilos CSS  
}
```

4.3. Identificadores y Clases

A veces tenemos varias etiquetas del mismo tipo pero queremos aplicar diferentes estilos según donde estén. Para esto usamos los identificadores y las clases.

La principal diferencia entre ellos es que los identificadores tienen que ser únicos en todo el documento HTML mientras que las clases pueden repetirse todas las veces que queramos. Los identificadores se suelen usar con etiquetas "neutras" como <div> o para marcar partes de un documento y después aplicar diferentes estilos a cada una (como por ejemplo identificar la cabecera, un logotipo, el menú principal, etc.).

En el siguiente ejemplo podemos ver como podemos indicar el identificador o la clase de una etiqueta HTML. Esto se hace con los parámetros "id" y "class" respectivamente, y se pueden aplicar a cualquier etiqueta:

```
<div id="capitulo2">
  <p>...</p>
  <p class="parrafogris">...</p>
  <p>...</p>
  <p class="parrafogris">...</p>
</div>
```

En este ejemplo “capitulo2” sería una sección única marcada en el documento en la cual podemos aplicar un estilo concreto. El estilo de la clase “parrafogris” se aplicaría sobre las etiquetas “p” indicadas.

En nuestra hoja de estilos, para indicar los estilos que definen un identificador tenemos que escribir el nombre del identificador precedido por una almohadilla “#”, de la forma:

```
#identificador {
  estilos CSS
}
```

Esta definición de estilos se puede combinar con lo que hemos visto en la sección anterior. Por ejemplo, para aplicar un estilo en concreto a la etiqueta “etiqueta1” que esté dentro del “identificador1”:

```
#identificador1 etiqueta1 {
  estilos CSS
}
```

Para aplicar estilos a clases es parecido pero precediendo el nombre de la clase con un punto “.” en vez de una almohadilla. Por ejemplo:

```
.clase {
  estilos CSS
}
```

La definición de una clase también la podemos combinar con lo que hemos visto en la sección anterior. Además también podemos aplicar los estilos de la clase sólo a una determinada etiqueta:

```
etiqueta1.clase1 {
  estilos CSS
}
```

En este caso sólo se aplicaría el estilo a las etiquetas “etiqueta1” que se marque que son de la clase “clase1”, por ejemplo: <etiqueta1 class=“clase1”>...</etiqueta1>. Si intentáramos aplicar esta clase a una etiqueta diferente no funcionaría.

4.4. Estilos CSS básicos

La sintaxis básica para definir un estilo es:

```
atributo: valor;
```

Los diferentes estilos siempre se separan con punto y coma, y después del nombre se pone dos puntos (y no un igual “=”, que es un error típico al confundirse con el HTML).

Muchos de los valores que podemos aplicar a un atributo de estilo son **unidades de medida**, por ejemplo, el valor del tamaño de un margen o el tamaño de la fuente. Las unidades de medida que podemos utilizar son: pixels (px), puntos (pt), centímetros (cm) y pulgadas (in).

A continuación se incluye un resumen de los principales estilos CSS y los valores que se les pueden aplicar:

FUENTES:

- **color:** valor RGB o nombre de color
Ejemplos: color: #009900; color: red;
Sirve para indicar el color del texto. Lo admiten casi todas las etiquetas de HTML. No todos los nombres de colores son admitidos en el estándar, es aconsejable entonces utilizar el valor RGB. Algunos de los principales nombres de colores son: white, black, gray, blue, red, green o yellow, para más nombres podemos consultar la dirección “http://www.w3schools.com/cssref/css_colornames.asp”.
- **font-size:** xx-small | x-small | small | medium | large | x-large | xx-large | Unidades de CSS
Ejemplos: font-size: 12pt; font-size: x-large;
Sirve para indicar el tamaño de las fuentes de manera más rígida y con mayor exactitud.
- **font-family:** serif | sans-serif | cursive | fantasy | monospace | Todas las fuentes habituales
Ejemplos: font-family: arial, helvetica; font-family: fantasy;
Con este atributo indicamos la familia de tipografía del texto. Los primeros valores son genéricos (serif, sans-serif, etc.), es decir, los navegadores las comprenden y utilizan las fuentes que el usuario tenga en su sistema.
También se pueden definir con tipografías normales. Si el nombre de una fuente tiene espacios se utilizan comillas para que se entienda bien.
- **font-weight:** normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900
Ejemplos: font-weight: bold; font-weight: 200;
Sirve para definir la anchura de los caracteres, o dicho de otra manera, para poner negrita con total libertad.
Normal y 400 son el mismo valor, así como bold y 700.
- **font-style:** normal | italic | oblique
Ejemplos: font-style: normal; font-style: italic;
Es el estilo de la fuente, que puede ser normal, itálica u oblicua. El estilo "oblique" es similar al "italic".

PÁRRAFOS:

- **line-height:** normal | unidades CSS
Ejemplos: line-height: 12px; line-height: normal;
El alto de una línea, y por tanto, el espaciado entre líneas. Es una de esas características que no se podían modificar utilizando HTML.
- **text-decoration:** none | underline | overline | line-through
Ejemplos: text-decoration: none; text-decoration: underline;
Establece la decoración de un texto, si está subrayado, sobre-rayado o tachado.
- **text-align:** left | right | center | justify
Ejemplos: text-align: right; text-align: center;
Sirve para indicar la alineación del texto. Es interesante destacar que las hojas de estilo permiten el justificado de texto, aunque recuerda que no tiene por que funcionar en todos los sistemas.
- **text-indent:** Unidades CSS
Ejemplos: text-indent: 10px; text-indent: 2in;
Un atributo que sirve para hacer sangrado o márgenes en las páginas.
- **text-transform:** capitalize | uppercase | lowercase | none
Ejemplos: text-transform: none; text-transform: capitalize;
Nos permite transformar el texto, para que tenga la primera letra en mayúsculas de todas las palabras, o todo en mayúsculas o minúsculas.

FONDO:

- **background-color:** Un color, con su nombre o su valor RGB
Ejemplos: background-color: green; background-color: #000055;
Sirve para indicar el color de fondo de un elemento de la página.
- **background-image:** El nombre de la imagen con su camino relativo o absoluto
Ejemplos: background-image: url(mármol.gif); background-image: url(http://www.url.com/fondo.gif)
Permite colocar una imagen de fondo en cualquier elemento de la página.

CAJAS (<div> o <table>):

- **width:** Unidades CSS | Porcentaje
height: Unidades CSS | Porcentaje
Ejemplos: width: 50px; width: 100%; height: 15px;
Permiten indicar el ancho y altura de un elemento. Se pueden aplicar sobre muchos elementos, como tablas, etiquetas div, imágenes, párrafos “p”, etc. Con algunas etiquetas no funciona, tampoco sirve para indicar espaciado (padding), bordes o márgenes.
- **margin-left:** Unidades CSS
Ejemplos: margin-left: 1cm; margin-left: 0,5in;
Indica el tamaño del margen izquierdo.
- **margin-right:** Unidades CSS
Ejemplos: margin-right: 5%; margin-right: 1in;
Define el tamaño del margen derecho.
- **margin-top:** Unidades CSS

Ejemplos: margin-top: 0px; margin-top: 10px;

Indica el tamaño del margen superior.

- **margin-bottom:** Unidades CSS

Ejemplos: margin-bottom: 0pt; margin-top: 1px;

Indica el tamaño del margen inferior.

- **margin:** <arriba> <derecha> <abajo> <izquierda> | <arriba> <derecha> <abajo> | <arriba-abajo> <izquierda-derecha> | <los 4 márgenes>

Ejemplos: margin: 4px 2px 1px 2px; margin: 4px;

También podemos utilizar el estilo “margin” para indicar todos los márgenes a la vez, esta etiqueta nos permite indicarle desde 4 valores (para cada uno de los márgenes), hasta 1 valor (para aplicarlo sobre todos los márgenes).

- **padding-left:** Unidades CSS

Ejemplos: padding-left: 0.5in; padding-left: 1px;

Indica el espacio insertado, por la izquierda, entre el borde del elemento-continente y el contenido de este. Es parecido a el atributo cellpadding de las tablas. El espacio insertado tiene el mismo fondo que el fondo del elemento-continente.

- **padding-right:** Unidades CSS

Ejemplos: padding-right: 0.5cm; padding-right: 1pt;

Indica el espacio insertado, en este caso por la derecha, entre el borde del elemento-continente y el contenido de este. Es parecido a el atributo cellpadding de las tablas. El espacio insertado tiene el mismo fondo que el fondo del elemento-continente.

- **padding-top:** Unidades CSS

Ejemplos: padding-top: 10pt; padding-top: 5px;

Indica el espacio insertado, por arriba, entre el borde del elemento-continente y el contenido de este.

- **padding-bottom:** Unidades CSS

Ejemplos: padding-bottom: 0.5cm; padding-bottom: 1pt;

Indica el espacio insertado, en este caso por abajo, entre el borde del elemento-continente y el contenido de este.

- **padding:** <arriba> <derecha> <abajo> <izquierda> | <arriba> <derecha> <abajo> | <arriba-abajo> <izquierda-derecha> | <los 4 márgenes>

Ejemplos: padding: 4px 2px 1px 2px; padding: 4px;

Al igual que para “margin”, esta etiqueta nos permite indicarle desde 4 valores (espaciado hasta cada uno de los bordes por separado), hasta 1 valor (para indicar el mismo espaciado hasta todos los bordes).

- **border-color:** color RGB o nombre de color

Ejemplos: border-color: red; border-color: #ffccff;

Para indicar el color del borde del elemento de la página al que se lo aplicamos. Se puede poner colores por separado con los atributos border-top-color, border-right-color, border-bottom-color, border-left-color.

- **border-style:** none | dotted | solid | double | groove | ridge | inset | outset

Ejemplos: border-style: solid; border-style: double;

El estilo del borde, los valores significan: none=ningún borde, dotted=punteado,

solid=sólido, double=doble borde, desde groove hasta outset son bordes con varios efectos 3D.

- **border-width:** Unidades CSS
Ejemplos: border-width: 10px; border-width: 0.5in;
El tamaño del borde del elemento al que lo aplicamos.
- **border:** <grosor> <tipo> <color>
Ejemplo: border: 2px solid red;
De esta forma podemos indicar las tres propiedades del borde a la vez. También podemos utilizar border-top, border-right, border-bottom y border-left para indicar estas tres propiedades para un borde en concreto.
- **float:** none | left | right
Ejemplo: float: right;
Sirve para alinear un elemento a la izquierda o la derecha haciendo que el texto se agrupe alrededor de dicho elemento.
- **clear:** none | right | left
Ejemplo: clear: right;
Indica que no se permiten elementos por ese lado del objeto. Por ejemplo, si tenemos varias cajas una a continuación de otra, al poner "clear:left" en la última caja, esta pasaría a la siguiente línea.

4.5. Pseudo-clases

Una pseudo-clase te permite tener en cuenta diferentes condiciones o eventos al definir una propiedad para una etiqueta HTML, por ejemplo si un enlace ha sido visitado o si el cursor del ratón está sobre un elemento. Algunas de las pseudo-clases que podemos utilizar son:

- *a:link* - enlace que no ha sido explorado por el usuario.
- *a:visited* - se refiere a los enlaces ya visitados.
- *a:active* - enlace seleccionado con el ratón.
- *a:hover* - enlace con el puntero del ratón encima, pero no seleccionado.
- *a:focus* - enlace con el foco del sistema. También puede ser usado para un input.
- *p:first-letter* - primera letra de un párrafo.
- *p:first-line* - primera línea de un párrafo.

Utilizando estos elementos podemos configurar por ejemplo:

```
a:hover { color: blue; }
a:visited { color: darkgreen; }
p:first-letter {color: green; font-size: x-large;}
```

4.6. Capas

Normalmente la posición de los elementos de una página es **relativa**, es decir, depende de los demás elementos de la página. Por ejemplo, un párrafo estará más abajo si antes de él

hay más párrafos o elementos. Debido a esto, normalmente cuando se quería colocar elementos en un sitio concreto, se recurría al uso de tablas (invisibles, solo para estructurar).

Con CSS podemos colocar los elementos en posición **absoluta**, es decir, indicando el tamaño y coordenadas exactas en las que queremos que se coloque. Para organizar la disposición en una Web con CSS se suele usar el elemento `<div>`. Además se le suele dar un identificador único a cada uno, mediante el cual, desde la hoja de estilo, podemos configurar su disposición. También podemos colocar estos elementos con posición relativa a otro elemento que lo contenga, por ejemplo, un `<div>` dentro de otro.

Es común en el diseño Web crear contenedores `<div>` generales en una posición absoluta o centrados en la página, con un tamaño definido, los cuales se utilizarán para contener y disponer el resto de elementos de nuestra Web. Estos otros elementos se pueden alinear de forma sencilla con una alineación “relativa” a sus contenedores. Por ejemplo un contenedor para la cabecera que contenga un par de contenedores para la disposición de logotipo y el texto de cabecera.

Distribución

Para indicar el tipo de distribución o disposición de un elemento lo hacemos mediante el atributo “**position: valor**”. El cual puede tomar los valores:

- *absolute*: La posición del elemento no depende de ninguna otra etiqueta. Esta posición se calcula desde la esquina superior izquierda de la página.
- *fixed*: Al igual que el anterior la posición es absoluta, pero el elemento se queda fijo en el sitio al hacer “scroll”.
- *relative*: Posición relativa a su elemento contenedor. Es la propiedad predeterminada.
- *static*: Al igual que el anterior la posición es relativa, pero no podemos redimensionar (por ejemplo) el objeto.

Posición

Para indicar la posición concreta de una capa utilizamos los atributos: *top*, *bottom*, *left* y *right*, de la forma:

```
top: <posición>;  
left: <posición>;
```

Normalmente sólo se utilizan un par de ellos, como *top* y *left*, o *bottom* y *right*. La posición se especifica mediante unidades de CSS, como por ejemplo en “px”, aunque también admite porcentajes.

Un ejemplo de la definición de una capa sería:

```
#micapa {  
  position: absolute;  
  top: 200px;  
  left: 150px;  
}
```

```
width: 175px;
height: 175px;
border: solid 1px blue;
text-align: center;
color: gray;
}
```

En nuestro documento HTML tendremos un elemento definido de la forma: `<div id="micapa"> ... </div>`, dentro del cual colocaremos texto u otros elementos.

Orden

A veces tenemos varias capas, unas por encima de otras, y queremos especificar un orden, para poder controlar las ocultaciones entre capas. Para esto usamos el z-index, de la forma:

```
z-index: <índice>;
```

Las capas con un índice de Z-index mayor aparecerán por encima de las capas con un z-index menor.

4.7. Más información

Para obtener una referencia mucho más completa sobre las hojas de estilo podemos consultar alguna de las siguientes Webs:

- Referencia detallada de todos los estilos: <http://www.w3schools.com/cssref/default.asp>
- Especificaciones: <http://www.w3.org/Style/CSS/>
- Tutoriales: <http://www.desarrolloweb.com/manuales/manual-css-hojas-de-estilo.html>

