

Unidad 1. Bases de las aplicaciones web

1.1 El servicio web I: web estática

El servicio web se concibió en origen como un sistema para la publicación y consulta instantánea de documentos digitales llamados páginas web a través de una red mundial de ordenadores. Fue creado por **Tim Berners-Lee** y **Robert Cailiau** en el CERN (*Organización Europea para la Investigación Nuclear*).

La idea era simple. Desde un ordenador conectado a una red, usando un programa específico, se solicita un documento o página web indicando su localización. Dicha solicitud llega al ordenador también conectado a la red que tiene el documento. Ese ordenador envía el documento al que lo pidió. Una vez recibido el documento, el programa antes citado muestra el documento al usuario.

Para desarrollar esta idea en este formato simple de su nacimiento se necesitaba:

- Una **red** para interconectar los ordenadores que contienen los documentos y los ordenadores desde los que se consultan. La incipiente red Internet parecía la red perfecta. Dicha red funciona en base a los protocolos TCP/IP. Y éste es el primer requisito del sistema web: opera sobre una red TCP/IP.
- Un programa o aplicación que se ejecute en el ordenador que contiene los documentos (lo llamaremos **servidor web**, tanto al programa como al ordenador). Dicho programa debe estar pendiente de las peticiones que le lleguen, y cuando sea el caso, buscar en sus sistemas de archivos el documento correspondiente y transmitirlo hasta el ordenador que lo pidió.
- Un programa o aplicación que se ejecute en cualquier ordenador que permite pedir a un servidor web un documento concreto de los que ofrece, que reciba el documento procedente del servidor, y que muestre el documento. A este programa lo llamaremos **cliente web** o de forma más popular **navegador web** (web browser).
- Un **protocolo** para que cliente y servidor se puedan comunicar. Esto implica establecer, entre otras cosas, la secuencia de diálogo, el formato de las peticiones, el formato de las respuestas, las formas de comunicar error, etc.. Este protocolo se denomina **HTTP** (*Hyper Text Transfer Protocol*).
- Un formato para los documentos o **páginas web**. En origen se pensó en crear documentos al uso de los de papel, es decir, que contuvieran fundamentalmente texto e imágenes. Pero el formato digital permitía introducir una novedad fundamental. Cuando un documento hiciera referencia a otro, ese referencia debía ofrecer un método para llegar directamente al documento referenciado. Esto es lo que llamamos enlace o link. Este formato se llamó **hipertexto** (*Hyper Text*). Normalmente, las distintas páginas de una misma entidad están enlazadas entre sí, y al conjunto se le llama **sitio web** (*website*) de la entidad. Se creó entonces un lenguaje para componer este tipo de documentos o páginas web llamado **HTML**.

(*HyperText Markup Language*). A partir de ahora, cuando indiquemos página web estaremos hablando de un documento en HTML.

- Como se pretendía que además de los documentos HTML se pudieran transmitir otro tipo de archivos (aunque el navegador o cliente no pudiera interpretarlos), se hizo necesario un sistema para identificar los distintos tipos. Se aprovechó un sistema de tipos creado para el correo electrónico, denominado tipos **MIME**.
- Un sistema de identificación de cada documento. Como no se quería ejercer control alguno ni centralizar los documentos, la identificación de cualquier documento requiere dos partes: la localización en la red del ordenador que lo contiene, y la localización del documento dentro de ese ordenador. Ese tipo de identificador de cada documento se denominará **URL**.

1.1.1 Funcionamiento del servicio web: cliente, servidor y URL

El servicio web funciona siguiendo un modelo cliente-servidor en una red TCP/IP, ya sea local o interconectada con las demás redes a través de Internet. Es decir, la petición de documentos se realiza desde un programa cliente a un programa servidor. La máquina del servidor puede, o no, pertenecer a la misma red que la del cliente. En caso negativo, la del cliente debe poder alcanzar la del servidor mediante los dispositivos y mecanismos necesarios que interconecten su red a la del servidor (cadena de routers).

Desde la **parte del cliente**, es el programa **navegador web (web browser)** con el que el usuario interacciona para solicitar a un servidor web el envío de una página web o archivo.

La petición de una página web o archivo se realiza escribiendo en el navegador web su dirección, o pulsando sobre un enlace que tiene programada su dirección. Básicamente, la estructura de dicha dirección, denominada URI (*Uniform Resource Identifier*) o URL (*Uniform Resource Locator*) consiste en:

- Una primera sección donde se indica el **protocolo** (en caso del servicio web será http o si se trata de una transmisión cifrada será https). Va seguido del carácter dos puntos : y dos barras // .
- La dirección **IP del ordenador del servidor** o su **nombre de dominio DNS** (por ejemplo, www.debian.org).
- El **puerto** donde se realiza la petición, esto es, el puerto donde escucha el servidor (por ejemplo, el puerto 80). Va precedido del carácter dos puntos : .
- Y la **ruta de la página web o archivo**. El servidor dispone de un sistema de archivos virtual que es público, y donde contiene todos los documentos o archivos que puede enviar. Esta ruta indica cuál es el archivo concreto que se desea de todos los que se encuentran públicos en el servidor (por ejemplo, index.html, colocado en el directorio raíz del sistema de archivos virtual)

La dirección resultante de los ejemplos sería:

`http://www.debian.org:80/index.html`

Normalmente el cliente HTTP pueden asumir valores por defecto, de modo que escribiendo en la dirección del navegador web `www.debian.org` se obtendría el mismo resultado. El cliente HTTP asume por defecto el protocolo (http), así como el puerto (80).

Si no se indica ruta de documento, se le está indicando al servidor que envíe la página que tenga configurada como índice para el directorio raíz (normalmente *index.html*, *index.htm*, *index.php*, etcétera). Si se indica sólo la ruta de un directorio sin archivo, se le está indicando al servidor que envíe la página que tenga configurada como índice para ese directorio (normalmente *index.html*, *index.htm*, *index.php*, etcétera).

El navegador envía la correspondiente petición al servidor usando el protocolo HTTP.

Desde el **punto de vista del servidor** su función es, básicamente, atender las peticiones de documentos o archivos procedentes de los navegadores según el protocolo HTTP.

Cuando le llega una petición, el servidor toma la ruta solicitada y busca el archivo que ésta indica en el sistema de archivos virtual público. Si lo encuentra, lo envía al navegador del ordenador que lo pidió mediante una respuesta tipo HTTP. Si no lo encuentra envía un código de error (error 404), acompañado de un mensaje normalmente en formato de página web, todo ello también mediante una respuesta tipo HTTP. En ambos casos al final se libera la conexión. Esto supone que el protocolo HTTP es un **protocolo sin estado**, es decir, no se recuerda ningún suceso de conexiones anteriores a la actual.

Cuando el documento o archivo llega al **navegador**, pueden pasar dos cosas.

- Si el navegador es capaz de trabajar con el tipo de fichero correspondiente, lo trata. Esto puede suceder de dos maneras:
 - El navegador puede trabajar con el tipo de archivo directamente. El ejemplo más evidente es cuando se trata de un archivo HTML. En tal caso, el navegador comienza a dibujar el documento siguiendo las instrucciones indicadas en el código HTML. Para ello, puede que sea necesario pedir elementos incrustados en el documento pero que no van dentro del archivo HTML, como imágenes.
 - El navegador puede trabajar con el tipo de archivo gracias a complementos o extensiones que tiene instalado (plugins). Por ejemplo, en el caso de animaciones interactivas Flash (archivos swf), si el navegador tiene el plugin correspondiente, podrá reproducirlo. Pero si no lo tiene, no podrá hacerlo.
- Si el navegador no puede trabajar con el tipo de archivo recibido debe preguntar al usuario qué hacer con él, si guardarlo donde el usuario indique, o invocar a una aplicación externa que pueda trabajar con el correspondiente tipo de archivo. Por ejemplo, si el tipo de archivo recibido es un mp3 (y el navegador no tiene instalado ningún plugin para reproducir este tipo de archivos), el navegador pregunta si guardar el archivo o si abrirlo con alguna de las aplicaciones instaladas en el ordenador capaz de reproducirlo.

Como se ha comentado anteriormente, siempre se libera la conexión, por lo que ésta sólo tiene la duración correspondiente a la transmisión de la página solicitada. Con esto se consigue una gran optimización de los recursos del servidor. Para cada objeto que se transfiere por la red se realiza una conexión independiente. Por ejemplo, si el cliente HTTP solicita una página que contiene dos imágenes integradas, se realizan tres conexiones: una para el documento HTML y dos para los archivos de imágenes.

Como se observa, se trata de un sistema cliente-servidor, en el que la mayoría parte del trabajo recae en el cliente HTTP y su capacidad para soportar nuevas demandas y

funcionalidades. Esto es debido a que, en general, los clientes web no sólo asumen el papel de clientes del protocolo HTTP, sino que normalmente también actúan como clientes de los protocolos FTP, SMTP/POP, etc..

Una de las tendencias del mercado apunta a una integración en un único sistema cliente de todos los servicios que ofrece Internet. Un buen ejemplo es la integración de clientes de correo electrónico en páginas web, los denominados webmail. Esto supone un aumento de la complejidad de las tecnologías web, tanto del lado del servidor como del cliente.

En la siguiente figura se muestra un esquema del proceso completo de transferencia de una página web, desde que el usuario solicita una página hasta que el cliente HTTP la muestra en el formato apropiado.

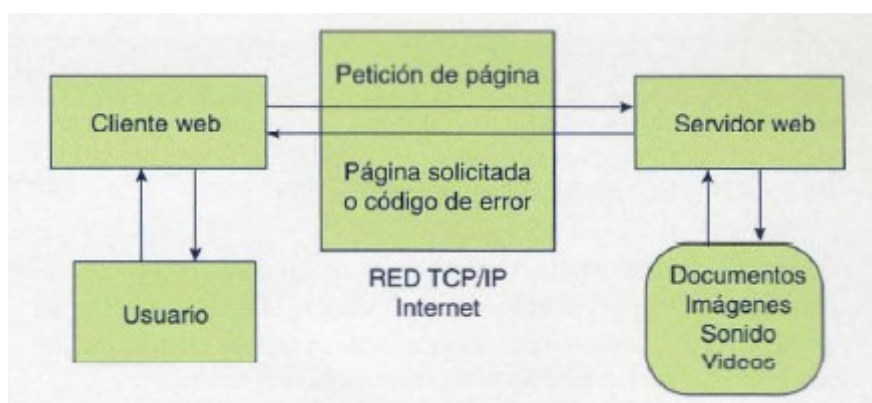


Ilustración 1: Esquema de una conexión HTTP

1. El usuario especifica en el cliente HTTP la dirección de la página que desea consultar o archivo que desea obtener.
2. El cliente HTTP decodifica la información de la URL, segmentando las distintas partes (protocolo de acceso, IP o nombre de dominio del servidor, puerto, etcétera) y añadiendo las que faltan con valores por defecto.
3. El cliente establece una conexión con el servidor web y solicita la página o el objeto indicado en la ruta de la URL. Si no se especifica ninguna, el servidor asumirá que se pide la página índice.
4. El servidor busca el archivo indicado en la ruta de la URL en su sistema de archivos virtual, y envía dicha página o archivo (o, en ausencia de éstos, un código de error) al cliente.
5. El cliente inicia la tarea de tramitación de la respuesta, por ejemplo, interpretando los códigos HTML para mostrar la página web, mostrando el mensaje de error, llamando a la aplicación externa adecuada para tratar el objeto u ofreciendo al usuario guardar el objeto.
6. Se cierra la conexión.

1.1.2 El protocolo HTTP

A continuación se detallan diferentes aspectos de una comunicación HTTP, comenzando por la descripción de la propia comunicación.

La comunicación HTTP

El intercambio de información entre el cliente y el servidor web se concreta mediante mensajes, que son las peticiones que se realiza el cliente al servidor, así como las respuestas en sentido inverso. El contenido de dichos mensajes son líneas de texto, cada una de las cuales contiene las órdenes y parámetros con la sintaxis definida por el protocolo HTTP

Dado que se trata de un protocolo **sin estado**, cada transacción HTTP es una comunicación distinta. Existen dos tipos de mensajes, los de petición o solicitud (*Request*) y los de respuesta (*Response*), cada una con una sintaxis determinada.

El formato de un mensaje genérico HTTP es el siguiente:

- **Línea de comienzo.** Indica el tipo de mensaje:
 - orden HTTP más sus parámetros (métodos de petición o *Request*),
 - o resultado de la solicitud (métodos de respuesta o *Response*).
- **Líneas de encabezado (si son obligatorias) o cero (si son opcionales)**, acabadas con un CR-LF (es decir, un retorno de carro «CR» y a continuación una línea «LF»).
- **Separador**, que no es más que otro CR-LF.
- Por último, el **contenido o cuerpo del mensaje**.

Veamos con un poco más sobre las líneas de comienzo y las líneas de encabezado.

Líneas de comienzo en Métodos de petición (*Request*)

Una petición HTTP, en su formato más básico, tiene la sintaxis «método, espacio en blanco, URI, espacio en blanco y versión».

El método le indica qué debe hacer con el URI. Por último, la versión indica el número de versión del protocolo que el cliente entiende. Por ejemplo, una petición normal utiliza el método GET para demandar del servidor el URI solicitado, y su sintaxis sería:

```
GET /index.html HTTP/1.0
```

La versión 1.0 de HTTP contempla tres métodos, que son **GET** (utilizado para obtener cualquier tipo de información del servidor siendo el método invocado cuando se realiza un clic sobre un enlace), **HEAD** (parecido a **GET**, pero en la respuesta no se devuelve el cuerpo del mensaje, sólo la cabecera), y **POST** (dedicado al envío de información desde el cliente).

La versión posterior del HTTP, la 1.1, amplía estas posibilidades, implementando nuevos métodos, surgidos de las propias necesidades. Algunos de estos métodos son **PUT**, para enviar el recurso objeto de la URI del cliente al servidor, **DELETE**, para borrar el recurso solicitado del servidor, **OPTIONS**, con el que el cliente obtiene información del servidor para poder negociar los valores de los parámetros de la comunicación, etcétera.

Líneas de comienzo en Métodos de respuesta (Response)

La sintaxis de una respuesta HTTP es, en su formato más básico, una línea de estado y tiene una estructura fija, donde se indica versión, código de estado y texto explicativo. Sirva como ejemplo una posible respuesta de error

```
HTTP/1.1 405 Method Not Allowed
```

o si es satisfactoria, por ejemplo:

```
HTTP/1.1 200 OK
```

Los posibles códigos de estado se identifican con números de tres cifras, y se clasifican en cinco grupos, según sean:

- informativos (con códigos 1xx),
- de éxito de la solicitud (códigos 2xx),
- para redireccionar la solicitud (código 3xx),
- error generado por el cliente (códigos 4xx),
- errores generados en el servidor (códigos 5xx).

Son respuestas típicas los códigos 200, para indicar OK de la petición, y el 404, para indicar que el objeto solicitado no está disponible (por ejemplo, porque no existe, caso típico de una URI mal introducida por el usuario o indicada en un enlace).

Todos los campos contemplados en los métodos de respuesta pueden ser consultados en la sección adecuada (*Response Header Field*) del documento de la W3C correspondiente al RFC 2616 (accesible en la web del W3C) o en múltiples páginas en internet ([wikipedia](https://es.wikipedia.org/wiki/HTTP)).

Línea de encabezado HTTP

Los encabezados de los mensajes constituyen un campo fundamental ya que definen en gran parte la información que se intercambia entre clientes y servidores, dándole así flexibilidad al protocolo. Estas líneas permiten que se envíe información descriptiva en la propia transacción, permitiendo, por ejemplo, la autenticación o identificación de usuarios.

La sintaxis de una línea simple de encabezado se compone del nombre de encabezado, seguido de dos puntos, y el valor correspondiente.

Los encabezados pueden clasificarse en grupos, según su función:

- **De ámbito general:** manejan información que puede ser utilizada tanto por clientes como por servidores, y que se aplican a una sesión completa de comunicación.
- **De solicitud:** los utiliza el cliente para enviar información adicional al servidor, en sus peticiones de algún servicio.
- **De respuesta:** utilizados por el servidor para enviar información añadida al cliente.
- **De entidad:** con información relacionada directamente con el recurso que se le va a proporcionar al cliente.

Todos los campos posibles, con la pertinente explicación de cada uno, de las líneas de los encabezados HTTP pueden ser consultados en la sección adecuada (*Header Field*

Definition) del documento de la W3C correspondiente al RFC 2616 (también accesible en la web del W3C).

No obstante, como último apunte, estas propuestas están en constante evolución, ya que existen nuevos documentos para poder estandarizar la forma y formato de las nuevas extensiones al protocolo HTTP.

1.1.3 El sistema de tipos: Tipos MIME

Los **tipos MIME** (**Multipart Internet Mail Extension**) fueron definidos en los RFC 2045, 2046, ..., y, como su nombre indica, fueron en un primer momento utilizados para extender las características del correo electrónico, pero hoy su uso se ha extendido, siendo también denominados **IMT** (**Internet Media Types**).

Los tipos MIME se componen de un tipo y un subtipo. Por ejemplo, un fichero, que es un documento de texto y que ha sido escrito en HTML, tendrá como tipo MIME **text/html**.

El registro de los tipos MIME lo controla la **IANA** (**Internet Assigned Numbers Authority**), según lo especificado en el RFC 2048, y en su sitio web podemos obtener la lista completa y actualizada de los tipos registrados. Es importante el registro de tipos MIME, ya que esto asegura que dos tipos de contenido distintos no acaban con el mismo nombre. El prefijo especial "x-" queda reservado para tipos experimentales (desplegados sin que haya terminado el proceso de registro) o tipos de uso interno de organizaciones. Un ejemplo de este tipo es el **image/x-fwf**. El protocolo HTTP usa tipos MIME en sus encabezados, por ejemplo para las siguientes funciones:

- **Informar al cliente el tipo de datos que esta recibiendo del servidor.** Esto se hace con el encabezado **Content-Type**. Por ejemplo, un navegador típico puede manejar los datos de tres maneras distintas según el tipo MIME indicado en **Content-Type**:
 - visualizar el documento, por ejemplo con tipos **text/html**
 - llamar a una aplicación externa, por ejemplo, con tipos **application/pdf**
 - preguntarle al usuario qué hacer ante un tipo que no se entiende, por ejemplo, **image/x-fwf**.
- **Permitir la negociación de contenido.** El cliente, en su petición, incluye los tipos MIME que acepta. Por ejemplo, si un navegador puede soportar documentos comprimidos de tipo **application/zip**, lo indicará con el encabezado HTTP mediante la expresión **Accept: application/zip**
- **Encapsular uno o más objetos dentro del cuerpo de mensaje**, mediante los **tipos MIME multipart** (definidos en el RFC 2046). Quizás el ejemplo más conocido sea el tipo **multipart/form-data**. El tipo **multipart/form-data** ha sido definido en el RFC 1867, y sirve para enviar los datos de un formulario del cliente al servidor (mediante el método POST).

1.1.4 El formato de las páginas web

HTML

Una página web recibida desde un servidor siempre es un documento en formato HTML¹. HTML es un lenguaje de marcas que permite definir la estructura y el contenido textual de la página web, así como la ubicación en el documento y la URL del resto de contenido (imágenes, elementos Flash, sonidos, vídeos, etc.).

Los documentos HTML se identifican con el tipo MIME text/html, y se almacenan en archivos de texto plano con extensión .htm o .html.

Cuando un navegador recibe un documento HTML (bien recibiendo un tipo text/html de un servidor o bien abriendo un archivo .htm o .html local) va interpretando su código y dibujando la correspondiente página.

Una página web está formada por una serie de elementos. El código de un documento HTML se forma mediante etiquetas, cada una de las cuales corresponde a un elemento. El formato de una etiqueta HTML es el siguiente:

```
<tipo-elemento atributo="valor"> contenido </tipo-elemento>
```

donde:

- **tipo-elemento** es el tipo del elemento que estamos insertando en la página. Por ejemplo, **p** indicaría que es un párrafo, **a** indicaría que es un enlace, **img** indicaría que es una imagen, etc.
- **atributo="valor"**. Cada tipo de elemento permite definir varias características. Por ejemplo, en un párrafo podemos indicar si la alineación es a la izquierda, a la derecha o centrada, en un enlace debemos indicar la URL donde llevará el enlace, en una imagen debemos indicar la URL de la imagen, etc.. Para definir estas características se tiene para cada tipo-elemento un conjunto de atributos que se pueden establecer a distintos valores. Se puede tener en un elemento una lista de atributos con sus correspondientes valores.
- **contenido**. Es el contenido del elemento, normalmente texto y/o más elementos anidados. Por ejemplo, un párrafo tendrá texto (que se escribe directamente), pero además puede tener uno o varios enlaces (que necesitarán las correspondientes etiquetas. El fin del contenido viene marcado por </tipo-elemento>.

La **versión** más extendida de **HTML** en la actualidad es la **4.01**, que data de 1999. Todos los navegadores actuales son capaces de interpretar código HTML 4.01. Se encuentran muy avanzados los trabajos de la versión **HTML5**, existiendo borradores bastante sólidos. HTML5 incorpora novedades muy importantes que permiten entre otras cosas la introducción de audio y vídeo y el manejo interactivo de gráficos de forma directa, sin necesidad de usar otras tecnologías (propietarias o no). De esta manera, una vez que el navegador sea compatible con el estándar HTML5, no necesitará de extensiones para la reproducción de vídeo, audio o el manejo interactivo de gráficos, ni el creador de la web necesitará editores especiales para incorporar estos elementos en sus páginas. Hasta ahora, son compatibles con el borrador actual de HTML5 las versiones actuales de los

¹ Esto no significa que se pida un archivo .htm o .html y se reciba tal archivo; es así en la mayoría de los casos, pero en otros no se pide un archivo .htm o .html, si no que se da una orden al servidor y éste crea un documento HTML que transmite al cliente, sin que ese documento corresponda a un archivo almacenado en el servidor.

navegadores Firefox, Chrome, Safari y Opera.

CSS

Las hojas de estilo en cascada (*Cascade Style Sheets*, **CSS**) se utilizan para definir la presentación de documentos escritos en HTML. Las hojas de estilo definen cómo se va a mostrar el documento en pantalla.

Al crear una página web tenemos que comprobar qué elementos tiene (párrafos, tablas, listas, etc.) y qué hacen esos elementos (son enlaces, botones que tienen una función, etc.). Todo esto queda recogido en el documento HTML. Pero además, debemos tener en cuenta el aspecto de cada elemento (color, tamaño, tipo de letra, etc.), así como la posición de cada uno dentro de la página. Esto último, desde HTML4, se puede hacer por separado en una hoja de estilo CSS.

Con esto conseguimos separar lo que es el contenido de la página con su presentación, de manera que al modificar el contenido no es necesario cambiar la presentación, y viceversa, se puede cambiar la presentación por completo sin tener que cambiar el contenido. Además, una misma presentación se puede usar para distintas páginas con distinto contenido.

Normalmente la hoja de estilo es un documento separado con tipo MIME `text/css` y si es un archivo con extensión `.css` (también se trata de un archivo de texto plano). Desde el código del documento HTML se hará referencia a la URL de la hoja de estilo CSS a utilizar (etiqueta `<link>` en la cabecera del documento). No obstante, también se puede incluir el código de la hoja de estilo directamente dentro del documento HTML, usando la etiqueta `<style>` en la cabecera del documento.

Las ventajas de usar CSS son:

- Sencillez al realizar cambios en la presentación del contenido.
- Facilidad para dividir el contenido de una página.
- Mayor accesibilidad de la web para personas con discapacidades, al separar el contenido y permitir usar por ejemplo, otras hojas de estilo con letras más grandes o elementos software de lectura por voz.

La versión actual es **CSS2**, de 1998. Se está trabajando y existen borradores bastante sólidos de la versión **CSS3**, que al igual que HTML5 incluye muchas e importantes novedades, sobre todo en lo que se refiere a colocación de los elementos en la página.

Al igual que con HTML, es responsabilidad del navegador soportar una versión más o menos avanzada de CSS. En la actualidad, todo los navegadores soportan CSS2 y al menos los que incorporan HTML5 también lo hacen con CSS3, ya que la combinación de ambas técnicas (junto con Java Script) permiten las mismas funcionalidades que Flash, la tecnología propietaria dominante hoy en día en la web.

JavaScript

Un **lenguaje de script** permite crear programas que realicen acciones dentro de una página web, como cambiar dinámicamente el contenido de la página, modificar el comportamiento del navegador, validar formularios, realizar trucos visuales, etc..

Los script son interpretados y ejecutados por el navegador cuando se carga la página o cuando se produce un determinado acontecimiento detectado por el navegador, como pulsar un botón o un enlace, pasar por encima de un elemento, pulsar una tecla, etc.. Al ser ejecutados por el navegador, no pueden contar con los recursos del servidor para su ejecución, por ejemplo, no pueden consultar en bases de datos del servidor.

El lenguaje más usado es **JavaScript**, desarrollado por *Netscape Communications* para su navegador *NetScape Navigator* (precursor de *Mozilla Firefox*) y es soportado por todos los navegadores actuales.

Un script de JavaScript puede estar directamente incorporado dentro del código HTML de la página mediante la siguiente etiqueta:

```
<script language="JavaScript"> Código </script>
```

Sin embargo, también puede estar en un archivo externo y ser incluido en el código HTML de la página mediante la etiqueta:

```
<script src="URL_del_script"> </script>
```

Como hemos indicado anteriormente, todos los navegadores actuales incluyen un motor de JavaScript. De hecho, en la actualidad, debido al uso intensivo del mismo en tecnologías como AJAX o en combinación con HTML5 y CSS3 para producir los mismos efectos que Flash, se puede decir que la principal preocupación de los desarrolladores de navegadores es aumentar el rendimiento de su motor de JavaScript (componente del navegador que permite la ejecución de los scripts).

AJAX

Tomado y modificado de Wikipedia

AJAX (*Asynchronous JavaScript And XML*) es una técnica de desarrollo web para crear aplicaciones interactivas que se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano para obtener información parcial de la página. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, es decir, cargando sólo lo “nuevo”, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones web. Esto apoyado en la gran y creciente versatilidad del navegador para reproducir todo tipo de formatos, permite construir aplicaciones web de una gran riqueza visual e interactiva.

AJAX es una tecnología asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. JavaScript es el lenguaje en el que normalmente se efectúan las funciones de llamada de AJAX mientras que el acceso a los datos se realiza mediante *XMLHttpRequest*, objeto disponible en los navegadores actuales.

AJAX es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores dado que está basado en estándares abiertos como *JavaScript* y *Document Object Model (DOM)*.

1.2 El servicio web II: la web dinámica

Conforme internet iba teniendo más usuarios que accedían con mayor frecuencia, las páginas web estáticas comenzaron a presentar un problema. Los usuarios demandaban cada vez más información más actualizada. Sin embargo, crear nueva información suponía actualizar (rehacer) los correspondientes archivos HTML, tarea tediosa, costosa y que además requería de alguna persona que dominara tal lenguaje.

Se hacía necesario un mecanismo que permitiera la actualización de los contenidos de las páginas de una forma más sencilla y simple, al punto de que pudiera hacerse por simples usuarios. Además, resaltamos que lo importante de actualizar era el contenido, y no el formato de página. De hecho, el usuario debía verla como la misma página con distinto contenido.

Además, la actualización a la vieja usanza suponía casi siempre la eliminación, al menos de la parte pública, de los contenidos anteriores.

Los dos inconvenientes citados hacen evidente la necesidad de separar los contenidos del diseño, y que ambos se mezclen a la hora de mostrarse al usuario.

Por otro lado, también resultaría interesante que el usuario pudiera determinar los contenidos que desea ver e, incluso, el aspecto de la página o la organización de los contenidos en la misma. Esto hace necesario un lugar donde almacenar las preferencias del usuario, así como algún software que dibuje la página en función de sus preferencias de presentación y selección de contenidos, además, de como es evidente, un sistema de identificación.

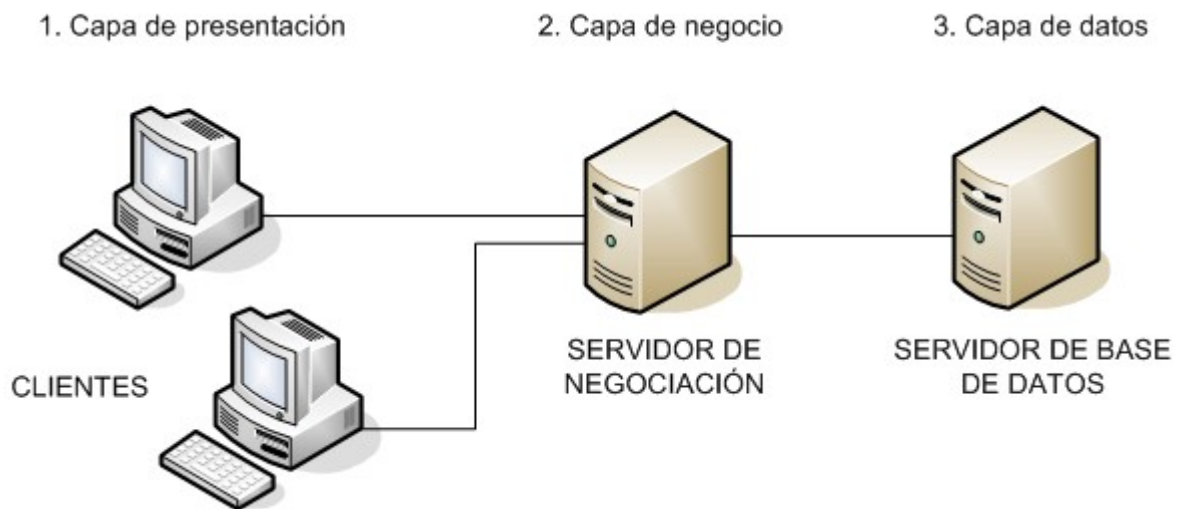
Y aun más, hemos mencionado anteriormente que la generación de contenidos debe ser en cierta forma independiente de los conocimientos técnicos sobre tecnologías web como lenguajes de páginas. Esto supone la necesidad de software que permita de forma sencilla crear y publicar contenidos en la web.

A estas necesidades responde bien un modelo de programación que pasamos a tratar, el modelo de tres capas.

1.2.1 El modelo de tres capas

Extraído de Wikipedia

La programación por capas es un estilo de programación en el que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño; un ejemplo básico de esto consiste en separar la capa de datos de la capa de presentación al usuario.



La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado. Un buen ejemplo de este método de programación sería el modelo de interconexión de sistemas abiertos.

Además, permite distribuir el trabajo de creación de una aplicación por niveles; de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles, de forma que basta con conocer la API que existe entre niveles.

En el diseño de sistemas informáticos actual se suele usar las arquitecturas multinivel o Programación por capas. En dichas arquitecturas a cada nivel se le confía una misión simple, lo que permite el diseño de arquitecturas escalables (que pueden ampliarse con facilidad en caso de que las necesidades aumenten).

El diseño más utilizado actualmente es el diseño en tres niveles (o en tres capas).

1. **Capa de presentación:** es la que ve el usuario (también se la denomina "capa de usuario"), presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" (entendible y fácil de usar) para el usuario. Esta capa se comunica únicamente con la capa de negocio.
2. **Capa de negocio:** es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él. También se consideran aquí los programas de aplicación.
3. **Capa de datos:** es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Todas estas capas pueden residir en un único ordenador, si bien lo más usual es

que haya una multitud de ordenadores en donde reside la capa de presentación (son los clientes de la arquitectura cliente/servidor). Las capas de negocio y de datos pueden residir en el mismo ordenador, y si el crecimiento de las necesidades lo aconseja se pueden separar en dos o más ordenadores. Así, si el tamaño o complejidad de la base de datos aumenta, se puede separar en varios ordenadores los cuales recibirán las peticiones del ordenador en que resida la capa de negocio.

Si, por el contrario, fuese la complejidad en la capa de negocio lo que obligase a la separación, esta capa de negocio podría residir en uno o más ordenadores que realizarían solicitudes a una única base de datos. En sistemas muy complejos se llega a tener una serie de ordenadores sobre los cuales corre la capa de negocio, y otra serie de ordenadores sobre los cuales corre la base de datos.

En una arquitectura de tres niveles, los términos "capas" y "niveles" no significan lo mismo ni son similares.

El término "capa" hace referencia a la forma como una solución es segmentada desde el punto de vista lógico: Presentación / Lógica de Negocio / Datos.

En cambio, **el término "nivel" corresponde a la forma en que las capas lógicas se encuentran distribuidas de forma física**. Por ejemplo: Una solución de tres capas (presentación, lógica del negocio, datos) que residen en un solo ordenador (Presentación+lógica+datos). Se dice que la arquitectura de la solución es de tres capas y un nivel. Una solución de tres capas (presentación, lógica del negocio, datos) que residen en dos ordenadores (presentación+lógica, lógica+datos). Se dice que la arquitectura de la solución es de tres capas y dos niveles.

1.2.2 Aplicaciones web

¿Cómo se usaría el modelo de tres capas en las páginas web para solucionar los problemas que hemos mencionado antes?

Crearemos un modelo de **tres capas** en, normalmente, **dos niveles**.

El **primer nivel** lo constituye el ordenador que ejecuta el **navegador**. Éste tiene la capa de presentación (diseño visual y e interacción con el usuario) y parte de la capa lógica (verificación de formatos de datos en formularios, por ejemplo). El **segundo nivel** lo constituye el **servidor**, que tiene la mayor parte de la capa de lógica y la capa de datos.

La diferencia fundamental está en el comportamiento del servidor. Ahora el servidor no es un simple almacén y expendedor de documentos ya creados. Dispone de una base de datos, que normalmente tendrá, entre otras cosas, el contenido que pueda llevar la página. Y por otro lado, dispone de un intérprete de algún lenguaje de scripts que:

- puede introducir, modificar y recuperar (leer) datos de la base de datos;
- puede crear documentos HTML en función de la petición de usuario y de los datos leídos, y tomando archivos de referencia para crear el diseño.

Entonces, ante una petición del navegador, el servidor **ejecuta un script**. Dicho script, normalmente, **va construyendo un documento HTML** a partir de otros y de los **datos que obtiene de la base de datos**. Concluido ese documento HTML, **no es almacenado** definitivamente en ningún lugar, sino que es **enviado al navegador**. Entonces el navegador recibe un documento HTML como de costumbre y lo **"dibuja"**. Pero en este caso, el documento HTML ha sido construido a medida de la petición

realizada por el usuario.

Este es el ejemplo más sencillo de funcionamiento. Otra posibilidad sería que el script tomara los datos de un formulario enviados desde el navegador y, con o sin trabajo previo sobre ellos, los almacene adecuadamente en la base de datos, para que posteriormente puedan ser recuperados e incorporados a páginas web. Esta forma es la que permite crear contenidos sin necesidad de editar archivos HTML.

Cuando una página web se construye de esta manera, decimos que es una página web dinámica. Un sitio web constituido por páginas dinámicas suele denominarse web dinámica o aplicación web.

1.2.3 Tecnologías usadas en webs dinámicas

En el navegador

Para poder pedir y mostrar páginas web dinámicas el navegador no necesita absolutamente nada distinto a lo que necesita para las estáticas.

Lenguaje de Script en el servidor

Como hemos indicado, la mayor parte de la capa lógica cae en el lado del servidor. Se necesita la posibilidad de poder ejecutar programas que en base a la petición captada por el servidor y tomando información de la base de datos sean capaces de producir páginas web, documentos HTML, que devolver al cliente, para que éste lo muestre.

El primer paso fue utilizar programas escritos en lenguajes ya existentes como **C** o **Perl**. Estos programas eran de alguna manera independientes del servidor, y simplemente se ponían en marcha cuando eran llamados por él y le entregaban su resultado. Esta técnica se denominaba **cgi** (*Common Gateway Interface*). Pero los lenguajes no estaban adaptados a la construcción de páginas web, lo que hacía el código complejo y farragoso.

Nacen entonces una serie de lenguajes interpretados que se comunican directamente con el servidor. Estos lenguajes sí que están enfocados al diseño de páginas web, aunque pueden hacer cualquier otra tarea, como lenguajes de programación que son. En ellos, los correspondientes programas que generan las web son fáciles e intuitivos, al punto de poder incluir directamente código HTML en los programas.

Ejemplos de estos tipos de lenguajes son:

- **PHP** (*Hypertext Preprocessor*). Es el más extendido y es software libre.
- **ASP** (*Active Server Pages*). También es muy usado, sobre todo en el mundo empresarial, ya que es de *Microsoft* y se integra bien con otras tecnologías de esta compañía.
- **JSP** (*JavaServer Pages*). ~~Es menos usado~~. Fue desarrollado por Sun para permitir la construcción de webs dinámicas con un código similar al de Java.

Para que un servidor soporte este tipo de lenguajes, deberán tener instalado el correspondiente **intérprete** y la correspondiente **extensión para ese lenguaje del servidor** que se esté usando. El uso de estos lenguajes es totalmente transparente a los navegadores.

Gestor de Base de datos

Como hemos indicado, las páginas web dinámicas se construyen tanto en contenido como en diseño en base a la información almacenada en una base de datos.

Una **base de datos** es un conjunto de información almacenada formada por datos y sus relaciones. La base de datos permite agregar información, modificar la información existente y sobre todo localizar la información que almacena por múltiples criterios.

Las bases de datos se organizan en **tablas**. Cada columna de una tabla se denomina **campo**, y corresponde con una característica cuyo valor se quiere guardar para cada entidad registrada. Por contra, cada fila corresponde a una instancia o ejemplo de una entidad concreta, con los valores concretos que dicha instancia posee para cada característica, y se denomina **registro**. Entre los campos de distintas tablas se pueden establecer relaciones que permiten realizar búsquedas de información que impliquen el uso de los datos almacenados en varias tablas.

Para poder manejar una base de datos es necesario un gestor de base de datos. Los **gestores de base de datos** son programas que sirven de interfaz entre la base de datos y las aplicaciones que la utilizan. Cuando una aplicación quiere obtener información de una base de datos, no accede a ella directamente, sino que hace una petición al gestor de base de datos indicándole lo que quiere, y es éste el que accede a la base de datos para obtener la información solicitada y devolvérsela a la aplicación que la pidió. Lo mismo sucede cuando se quiere guardar información en la base de datos.

Para las páginas web dinámicas en la base de datos se suele guardar:

- los contenidos de las páginas, por separado y clasificados convenientemente para que se pueda acceder a ellos de forma selectiva;
- la configuración del aspecto de la página;
- los usuarios y sus preferencias;
- etc..

Existen muchos tipos de gestores de base de datos, entre los que destacamos:

- **MySQL**. Es el más usado en Internet para las páginas web dinámicas. La empresa *Sun* (adquirida por *Oracle*) se encarga de su desarrollo.
- **PostgreSQL**. Su principal característica es que es software libre.
- **Firebird**.
- **Microsoft SQL Server**. Es el gestor de base de datos de *Microsoft*.


En principio los gestores de base de datos no tienen nada que ver con el servidor web, ya que no es el servidor el que accede, sino los programas en lenguaje PHP, ASP o JSP mediante el correspondiente intérprete o los llamados mediante CGI los que utilizan datos de la base de datos y por tanto los que interaccionan con el gestor.

1.3 Servidor web y hosting

1.3.1 Servidor web


Un servidor web es una aplicación que usando el protocolo HTTP recibe peticiones de archivo de otras máquinas y los transfiere usando este mismo protocolo. Normalmente, los archivos solicitados son documentos HTML (páginas web), pero pueden ser de cualquier otro tipo. El servidor web debe estar siempre ejecutándose para recibir las peticiones.

Algunos ejemplos de servidores web son los siguientes:

- **Apache.** Se trata de un servidor de código abierto que funciona para múltiples plataformas, como *Windows*, *Linux*, etc.). La versión actual es la 2.2. Es el más utilizado en todos los sistemas. Su configuración se establece mediante archivos de configuración.
- **Internet Information Server (ISS)** . Se trata de un servidor web de *Microsoft* con licencia propietaria, y que se ejecuta sólo sobre sistemas *Windows*. Se incluye por defecto en todos los sistemas operativos de servidor de *Microsoft* (*Windows 2000 Server*, *Windows 2003 Server* y *Windows 2008 Server*). También existe una versión limitada en *Windows XP*. Se configura mediante una aplicación gráfica.

Dos de las tareas más importantes que debe tener un servidor son la **parada** y el **arranque**. La forma de hacer estas tareas difieren de un servidor a otro y de un sistema operativo a otro. Debido a la variedad, no mencionamos aquí el procedimiento, pero indicamos que, una vez decidido el sistema a usar y el servidor a usar, es imprescindible saber hacer estas tareas.

En cuanto a la configuración, todo servidor web tiene muchísimas opciones a configurar, de las cuales nosotros sólo vamos a citar las dos más importantes para nuestro propósito:

- **Directorio público.** En todo servidor web se establece un directorio del sistema de archivos local de manera que todo lo contenido de ese directorio (incluyendo los subdirectorios) es público a través del servicio web (es decir, es accesible para otras máquinas mediante accesos HTTP). Podemos decir que ese directorio local constituye el directorio raíz de sistema de archivos público del servidor web.
 - *Apache.* Para establecer este directorio en Apache debemos ir al archivo general de configuración *httpd.conf* o *apache2.conf* (su ubicación depende del sistema operativo y de la instalación). En él, buscamos la opción **DocumentRoot** y a continuación establecemos el directorio local que queremos sea el raíz del sistema de archivos público. Si lo cambiamos, para que el cambio tenga efecto, debemos parar y arrancar de nuevo el servidor.
-  **ISS.** En *Internet Information Service Manager* buscamos la opción ???
- **Archivo índice de cada directorio.** Cuando a un servidor web llega una petición que hace referencia a un directorio del sistema de archivos público, el servidor debe decidir qué documento de ese directorio debe transferir. Se puede establecer una configuración general para todos los directorios, que posteriormente se puede particularizar para algunos de ellos. Esa configuración consiste en una lista de

archivos a transferir por orden de prioridad. Por ejemplo, si la configuración general está establecida a *index.php index.htm index.html*, cuando al servidor llegue una petición que indica sólo un directorio buscará en ese directorio *index.php*; si lo encuentra lo transfiere, y si no busca *index.htm*; si lo encuentra lo transfiere, y si no busca *index.html*; si lo encuentra lo transfiere, y si no lanza un *error 404*.

- **Apache.** En apache la configuración general del archivo índice se hace en el archivo *httpd.conf* o *apache2.conf* (su ubicación depende del sistema operativo y de la instalación). En él, buscamos la opción **DirectoryIndex** y a continuación establecemos la lista de archivos a servir por orden de prioridad separados por espacio en blanco. La configuración para un directorio concreto se haría de forma similar pero en el archivo *.htaccess* contenido en ese directorio.
- ISS. ???

1.3.2 Intérprete PHP

Para poder utilizar en nuestro servidor páginas dinámicas necesitamos un intérprete de algún lenguaje de scripts de servidor. Hemos elegido PHP como lenguaje, por ser el más usado para webs dinámicas.

Para instalar el intérprete de PHP procedemos de la manera habitual del sistema operativo correspondiente:

- *Linux:* `sudo apt-get install php5`
- *Windows:* descargar el paquete de instalación y seguir las instrucciones.

Pero debemos recordar que el intérprete no es suficiente, también es necesario un módulo para el servidor web. Y este módulo, como es lógico, depende del sistema operativo que estemos usando y del servidor que estemos usando.

- En *Linux* con *Apache*: `sudo apt-get install libapache2-mod-php5`
- En *Windows* con *Apache*: seguir las instrucciones de este [enlace](#)
- En *Windows* con *ISS*: descargamos e instalamos el módulo, y a continuación accedemos al *IIS Manage / Handel Mappings*. Allí, pulsamos *Add Script Map* y completamos los datos para indicar dónde está el archivo *php5isapi.dll* (que será el directorio donde se ha instalado el módulo). Finalmente, configuramos *Request Restrictions* en la pestaña *Mapping* indicando sólo archivos (**File**).

El **archivo de configuración** de php es **php.ini** (su ubicación depende del sistema operativo usado y de la instalación).

1.3.3 Gestores de bases de datos

Los sistemas gestores de bases de datos son aplicaciones que sirven para manejar bases de datos, como las que necesitan nuestras páginas web dinámicas. El gestor más utilizado para páginas web es **MySQL**, que de código abierto con licencia GNU. Existen versiones para todas la plataformas, incluidas por supuesto *Windows* y *Linux*. La instalación, como siempre, depende del sistema base que se use:

- *Linux:* `apt-get install mysql-server mysql-client` . Durante la

instalación se nos pedirá el *password* para el usuario *root*.

- *Windows*: se descarga el correspondiente instalador y se siguen las instrucciones. También habrá que especificar una contraseña para el usuario *root*.

En una web dinámica el gestor de bases de datos es utilizado por el intérprete del lenguaje de script, y no por el servidor web. Por tanto, no es necesario instalar ningún módulo adicional en el servidor ni realizar ninguna configuración.

Toda la **configuración de MySQL** se almacena en un archivo, que en *Windows* es *my.ini* y en *Linux* es *my.cnf*.

1.3.4 Aplicaciones de instalación integrada

La instalación de servidor web, intérprete de PHP, módulo de PHP para el servidor web y gestor de bases de datos, junto con alguna herramienta para facilitar la manipulación de bases de datos, es un trabajo largo y que puede producir errores. Para facilitar el proceso se han creado herramientas de instalación integradas que reúnen todo el conjunto de aplicaciones y que se instalan sin problemas. Algunas de estas aplicaciones son:

- **AppServ**. Es una herramienta de código abierto para *Windows*. Instala *Apache*, *PHP*, *MySQL* y *phpMyAdmin* (interfaz web para manejar las bases de datos de *MySQL*), dejando las aplicaciones configuradas para su funcionamiento inmediato.
- **XAMPP**. Es una herramienta también de código abierto con versiones para cualquier plataforma (*Windows*, *Linux*, *MacOS*, etc.). Instala *Apache*, *PHP*, *MySQL*, *PERL* (lenguaje de programación de scripts) y *phpMyAdmin*, entre otras muchas cosas. A veces, las versiones para *Windows* se denominan **WAMP** y las versiones para *Linux* se denominan **LAMP**.
- **Bitnami**. Es un proyecto de código abierto que desarrolla paquetes de software para aplicaciones web. Aparte de instalar *Apache*, *PHP* y *MySQL*, también instala aplicaciones como sistemas de gestión de contenidos, foros, wikis, etc.. La instalación de los paquetes es muy simple: accedemos a la página web, seleccionamos la aplicación que deseamos (*Joomla*, *Drupal*, *phpBB*, etc.) y el sistema operativo donde lo vamos a instalar (*Windows*, *Linux*, *Mac*), y finalmente instalamos el paquete. Desde Julio de 2009 existe la opción de descargar máquinas virtuales (distribuciones de *Linux* con todos los elementos necesarios para que funcione la aplicación directamente).

1.3.5 Hosting

Las empresas tienen dos opciones a la hora de colocar páginas web dinámicas o estáticas en internet. La primera es utilizar servidores propios que pueden tener en la propia empresa o en empresas donde hay más servidores y una conexión con alta velocidad de subida. La segunda es contratar un servicio de alojamiento en servidores de otras empresas dedicadas específicamente a prestar este servicio, lo que se conoce como **hosting**.

Contratar un servicio de hosting supone disponer de espacio en un servidor web conectado constantemente a internet mediante una conexión con una velocidad de subida elevada. Existen distintos tipos de hosting:

- **Alojamiento gratuito.** El alojamiento gratuito es extremadamente limitado cuando se lo compara con el alojamiento de pago. Estos servicios generalmente agregan publicidad en los sitios y tienen un espacio y tráfico limitado. Se nos asigna un nombre de usuario y contraseña para acceder a nuestro espacio y configurarlo mediante un panel de control on-line sencillo.
- **Alojamiento compartido (shared hosting).** En este tipo de servicio se alojan clientes de varios sitios en un mismo servidor, gracias a la configuración del programa servidor web. Resulta una alternativa muy buena para pequeños y medianos clientes, es un servicio económico debido a la reducción de costos ya que al compartir un servidor con cientos miles o millones de personas o usuarios el costo se reduce dramáticamente para cada uno, y tiene buen rendimiento. ~~Un solo servidor puede alojar hasta 1 millón de proyectos.~~

Entre las desventajas de este tipo de hospedaje web hay que mencionar sobre todo el hecho de que compartir los recursos de hardware de un servidor entre cientos o miles de usuarios disminuye notablemente el desempeño del mismo. Es muy usual también que las fallas ocasionadas por un usuario repercutan en los demás por lo que el administrador del servidor debe tener suma cautela al asignar permisos de ejecución y escritura a los usuarios. En resumen las desventajas son: disminución de los recursos del servidor, de velocidad, de desempeño, de seguridad y de estabilidad.

Se nos asignará un nombre de usuario y contraseña para acceder a nuestros servicios, que podremos configurar gracias a un panel de control on-line más completo, como **cpanel**.

- **Alojamiento revendedor (reseller).** Este servicio de alojamiento está diseñado para grandes usuarios o personas que venden el servicio de Hospedaje a otras personas. Estos paquetes cuentan con gran cantidad de espacio y de dominios disponibles para cada cuenta. Aquí también se nos asigna un nombre de usuario y contraseña, y el panel de control es más completo para poder manipular las nuevas opciones. También suele ser **cpanel**.
- **Servidores virtuales (VPS, Virtual Private Server).** Mediante el uso de una máquina virtual, la empresa ofrece al cliente el control de un ordenador aparentemente no compartido. Sin embargo, varias máquinas virtuales de varios clientes están hospedadas en la misma máquina física. Así se pueden administrar varios dominios de forma fácil y económica, además de elegir los programas que se ejecutan en el servidor, teniendo el control absoluto sobre el mismo. Por ello, es el tipo de producto recomendado para empresas de diseño y programación web.
- **Servidores dedicados.** El término servidor dedicado se refiere a una forma avanzada de alojamiento web en la cual el cliente alquila o compra un ordenador completo, y por tanto tiene el control completo y la responsabilidad de administrarlo. El cuidado físico de la máquina y de la conectividad a Internet es tarea de la empresa de alojamiento, que suele tenerlo en un centro de datos.

Un Servidor dedicado es la contraparte del hospedaje web compartido, ya que en esta modalidad de hospedaje web se utilizan todos los recursos de un servidor para un solo usuario o cliente, mientras que un servidor dedicado puede contener miles de usuarios o sitios web. Sin embargo cuando se desea el máximo desempeño y disponibilidad de recursos para un sistema o empresa la opción de un servidor dedicado es la más óptima, ya que se puede garantizar de una manera

más objetiva el desempeño, la velocidad la estabilidad, del sistema página o información que se tenga publicada mediante este servidor.

La principal desventaja de elegir un servidor dedicado, es el costo del servicio, ya sea que se tenga un servidor local en la empresa o se contrate uno en algún *datacenter*, el costo siempre irá por arriba de el hospedaje web compartido. También hay que tener en cuenta que se requiere de un administrador con amplios conocimientos en el tema del manejo de servidores ya que las configuraciones principales, repercuten el correcto funcionamiento del mismo, por lo que se hace patente la necesidad de tener un buen técnico a cargo o bien una empresa profesional responsable.

- **Colocación (o housing).** Este servicio consiste básicamente en vender o alquilar un espacio físico de un centro de datos para que el cliente coloque ahí su propio ordenador. La empresa le da la corriente y la conexión a Internet, pero el ordenador servidor lo elige completamente el usuario (hasta el hardware).
- **Alojamiento web en la nube (o cloud hosting).** El alojamiento en la Nube se realiza con un gran conjunto de máquinas trabajando como una sola, conectadas a un grupo de sistemas de almacenamiento; todo ello unido mediante virtualización por hardware. Está controlado por un software capaz de mover, ampliar o reducir recursos en tiempo real.

Cuando se contrata un hosting se suele contratar también uno o varios dominios. Un **dominio** es un nombre en internet dentro de alguno de los **dominios de primer nivel** (*top level domain*): .com, .org, .es, etc.. A los dominios incluidos directamente en un dominio de primer nivel se les llama **dominios de segundo nivel**. Dentro de un dominio de segundo nivel adquirido por nosotros podemos crear tantos subdominios como queramos, que serán **dominios de tercer nivel**. A veces, estos dominios de tercer nivel se revenden a un precio más bajo que los de segundo nivel.

Normalmente llamamos **plan de hosting** o **plan de alojamiento** a la contratación de un servicio de hosting más un dominio de segundo nivel durante un periodo de tiempo. Las características que tenemos que observar a la hora de contratar un plan de hosting son:

1. **Tipo de hosting:** de entre los que hemos indicado antes
2. **Tipo de sistema operativos y software en el servidor.** Esto es importante, ya que si nuestra página incorpora técnicas de web dinámica necesitaremos soporte que el servidor tenga el software correspondiente para que esas páginas funcionen. Por ejemplo, en caso de usar páginas ASP será necesario un servidor Windows con IIS. En caso de usar páginas PHP el sistema operativo nos dará igual, pero será necesario comprobar que se soporta la versión de PHP en la que hemos realizado la página.
3. **Espacio disponible.** Cantidad de espacio que disponemos para conectar nuestros archivos.
4. **Transferencia.** Máximo de transferencia mensual que se admite para nuestras páginas.
5. **Número de dominios alternativos.** Cantidad de dominios que pueden apuntar a nuestro espacio.

6. **Número de subdominios.** Cantidad de subdominios que podemos crear en el dominio contratado. Técnicamente podría ser ilimitado, pero puede estar limitado por contrato.
7. **Número de bases de datos.** En alojamientos gratuitos o servidores compartidos se puede limitar el número de bases de datos a crear en el correspondiente gestor de bases de datos.
8. **Número de cuentas de correo.** Puesto que los servidores disponen de software servidor de correo electrónico, se nos ofrece la posibilidad de crear un número determinado de cuentas de correo electrónico que usen el nombre de dominio contratado. El espacio que ocupen los mensajes en esas cuentas es el mismo que ocupan los archivos “públicos”.
9. **Panel de control.** Modelo del panel de control on-line para gestionar todos los servicios contratados. El más habitual es **cPanel**, aunque muchos hosting desarrollan sus propios paneles de control.
10. **Otros:** antivirus, antispam, correo web, copias de seguridad, etc..

El plan de hosting se contrata durante un periodo de tiempo, y se tarifica anual, trimestral, mensualmente o como acuerden el cliente y la empresa de hosting. Es importante indicar que al contratar un dominio en un plan de hosting, el dominio es titularidad del cliente, que tendrá que seguir pagándolo cada año, pero no necesariamente vinculado con la empresa de hosting a través de la cual lo adquirió.

1.4 Web 2.0 y aplicaciones web

1.5 Sistemas Gestores de contenidos

1.5.1 Definición

Quizá los mayores ejemplos de web dinámicas son los sistemas gestores de contenidos (o gestores de contenidos a secas). Los **sistemas gestores de contenidos (CMS, Content Management System)** son aplicaciones que, usando tecnologías de web dinámica, permiten crear, clasificar, gestionar y publicar información en la World Wide Web sin necesidad de conocimientos técnicos.

Para ello separan completamente contenidos y diseño. Todos los contenidos se almacenan en una o varias bases de datos. El diseño se define mediante una mezcla de archivos HTML, CSS y parámetros guardados en las bases de datos. Por último, los archivos de código mezclan diseño y contenido para obtener la página deseada.

Por tanto las páginas se generan dinámicamente consultando los archivos de código que los componen (scripts de servidor), los archivos de diseño y las bases de datos a las que tienen acceso, produciendo como resultado final documentos HTML temporales que

son enviados al navegador solicitante. Además, mediante la comunicación por formularios desde el navegador hasta el servidor, permiten desde la misma aplicación y sólo usando formularios añadir o modificar datos en la base de datos, ya sean contenidos o parámetros de diseño.

1.5.2 Estructura

Generalmente un CMS presenta la siguiente estructura.

- **Base de datos.** Almacena todo el contenido de la web, así como muchos parámetros de configuración, organización, diseño, usuarios y contraseñas. Los sistemas de bases de datos más habituales de los CMS suelen ser *MySQL* o *Postgree*.
- **Programación.** Son archivos de código en algún lenguaje de script de servidor como *PHP* o *ASP*. Cuando a un servidor se le solicita el acceso a uno de estos archivos, éste lo ejecuta a través del módulo correspondiente. El script normalmente irá accediendo a los archivos de diseño y la base de datos para ir construyendo el documento HTML temporal con la información solicitada en la petición. Ese documento HTML será el que se envíe al navegador. En otras ocasiones, el script también capturará la información introducida por el usuario en el formulario (que llegará al servidor como parte de la petición) y la insertará convenientemente en la base de datos, para un posterior uso.
- **Diseño.** Se trata de un conjunto de archivos HTML y CSS junto a información almacenada en la base de datos que definen el diseño de la web, es decir, la maqueta o dibujo sobre el que se insertará el contenido que la programación se encarga de extraer de la base de datos. Los lenguajes usados son los usados para las páginas estáticas, es decir, HTML y CSS, complementados con JavaScript y tecnologías como AJAX. Generalmente constan de un modelo básico preestablecido y disponen de plantillas de diversos diseños que se pueden cargar y usar en lugar de la básica. Una plantilla estará formada por el correspondiente conjunto de archivos HTML, CSS, JavaScript, XML, imágenes, etc..

1.5.3 Características y partes funcionales

Las principales **características** de los gestores de contenidos son:

- Permiten interactuar con los usuarios mediante comentarios, encuestas, votaciones, etc..
- La edición de los contenidos es sencilla, los gestores suelen utilizar editores WYSIWYG (*What You See Is What You Get*), que sirven para dar formato a los contenidos de forma fácil e intuitiva.
- Se puede programar la fecha de publicación de un contenido; cuando éste se escribe, no tiene por qué publicarse inmediatamente.
- Reducción del tamaño de las páginas para descargar en el servidor, el contenido se almacena en la base de datos y al generarse dinámicamente las páginas en el servidor el coste de gestión se reduce considerablemente.
- Creación ilimitada de usuarios y grupos, se les puede asignar permisos a los usuarios y a los grupos para realizar funciones.

- La maquetación solo se realiza en un primer momento, no hay que preocuparse del diseño cada vez que se va a crear un artículo.
- Envío de boletines informativos y correos electrónicos; los gestores poseen mecanismos para enviar correos masivos.
- Soportan el formato RSS para publicar y agregar contenido.
- Control de estadísticas de acceso al gestor.
- Facilidad para actualizar el gestor de contenidos; algunos gestores tienen mecanismos para actualizarse automáticamente.
- Recuperación de información; con los gestores se pueden hacer copias de seguridad y restaurarlas.
- Miles de plantillas de diseño y plugins disponibles en internet.

En cuanto al aspecto funcional, los CMS suelen tener dos partes:

- Parte pública. Parte destinada a ser vista por los usuarios registrados y no registrados en el CMS. Puede llegar a incluir los elementos necesarios para la construcción de contenido, aunque éstos se encuentran habitualmente en la otra parte.
- Parte de administración. Parte destinada a usuarios privilegiados que pueden, entre otras muchas cosas, crear contenidos, configurar el CMS, modificar el aspecto visual, crear nuevos usuarios y asignarles permiso, añadir nuevas funcionalidades al CMS, configurar estas funcionalidades, etc..

1.5.4 Licencias de uso

Una licencia es un contrato que se realiza entre dos personas. Cuando nos referimos a licencias de software es el contrato que se realiza entre el creador del programa y las personas que lo van a utilizar.

Existen muchos tipos de licencias, vamos a ver los dos grupos de licencias más comunes que se utilizan en los gestores de contenidos.

- **Licencia de código abierto.** Se tiene acceso al código fuente y permite que los programadores desarrollen libremente modificaciones. Un ejemplo de licencia de código abierto es GPL GNU (General Public License GNU), una licencia de distribución de software gratuito que permite distribuirlo, modificarlo y utilizarlo.

Algunas premisas de código abierto son:

- Libre redistribución. El software puede ser regalado o vendido libremente.
- Código fuente. Debe estar incluido o poder obtenerse libremente.
- Trabajos derivados. La redistribución de modificaciones debe estar permitida.
- Sin discriminación de personas o grupos. Sin discriminación de áreas de iniciativa, los usos comerciales no pueden ser excluidos.
- Distribución de la licencia. Se aplican los mismos derechos a todo el que reciba el programa. La licencia no debe ser específica de un producto, el programa no

puede licenciarse solo como parte de una distribución mayor.

- La licencia no debe restringir otro software. La licencia no puede obligar a que algún otro software que sea distribuido con el software abierto sea también de código abierto.
- La licencia tiene que ser tecnológicamente neutral. No se debe requerir la aceptación de la licencia por medio de un acceso por clic de ratón o de otra forma específica del medio de soporte del software.
- **Licencia propietaria.** Limita la posibilidad de modificar o redistribuir el código, y el código fuente no está disponible. En ellas, los propietarios establecen los derechos de uso, distribución, redistribución, copia, modificación, cesión y en general, cualquier otra consideración que se estime necesaria.

1.5.5 Tipos

En internet hay cientos de gestores de contenidos y se pueden clasificar dependiendo de la licencia que tienen, el uso que se les da, el lenguaje de programación en que están desarrollados, etc..

En este apartado explicamos los gestores de contenidos que más se utilizan, que precisamente son los que tienen licencias del tipo GPL o parecida. Los gestores se pueden clasificar dependiendo de su uso en:

- **Uso genérico**, para organizar y publicar cualquier tipo de información:
 - Drupal (GPL, usa PHP y MySQL o Postgree).
 - Joomla (GPL, usa PHP y MySQL o Postgree).
 - MojoPortal (CPL – Common Public License, usa ASP.NET en Windows y como sistema de base de datos puede usar SQL Server, MySQL, SQLite, etc.
- **Blogs**, para crear sitios web que se actualizan periódicamente y cuyo contenido se ordena cronológicamente:
 - Wordpress (GPL, usa PHP y MySQL).
 - LifeType (GPL, usa PHP y MySQL).
- **Foros**, para crear sitios donde se publican mensajes.
 - PhpBB (GPL, usa PHP y MySQL, SQL Server, Access o Oracle).
 - SMF – Simple Machines Forum (GPL, usa PHP y MySQL).
- **Wikis**, para crear sitios web que se pueden editar por múltiples usuarios.
 - Dokuwiki (GPL, usa PHP y MySQL).
 - MediaWiki (GPL, usa PHP y MySQL). Es el software que usa Wikipedia.
- **Álbumes de fotos**, para crear galerías fotográficas.
 - Coppermine Photo Gallery (GPL, usa PHP y MySQL).
 - Gallery (GPL, usa PHP y MySQL).
- **Aprendizaje en línea**, para crear sitios web de aprendizaje o aulas virtuales.
 - Moodle (GPL, usa PHP y MySQL).

- **Comercio electrónico**, para crear tiendas on-line.
 - OsCommerce (GPL, usa PHP y MySQL).

1.5.6 Procedimiento general de instalación

El procedimiento estándar de instalación de los sistemas gestores de contenidos suele tener un pasos comunes que detallamos a continuación.

1. Descargar el software comprimido.
2. Subir el software comprimido a la parte pública del servidor web (normalmente vía FTP o un administrador de archivos que ofrezca el panel de control del hosting).
3. Descomprimir el software en el servidor.
4. Crear una base de datos en el sistema gestor de bases de datos para el CMS. Además, se puede crear un usuario específico para esa base de datos y darle todos los privilegios sobre esa base de datos. La base a priori puede estar vacía. Esto se hará mediante herramientas como *phpmyadmin* o la que establezca el panel de control del hosting.
5. Ejecutar desde un navegador el script de instalación. Normalmente sólo hay que acceder a través del navegador al directorio donde se ha descomprimido el software (directorio que será público).
6. Seguir las instrucciones del script de instalación. Habrá una serie de pasos comunes a todos los CMS:
 - a) Verificación de los requisitos técnicos en el servidor.
 - b) Permitir la creación o modificación del archivo de configuración en el servidor, lo que puede suponer cambiar permisos de lectura y escritura en el directorio del servidor donde está la web o directamente en el archivo.
 - c) Dar la información de la base de datos creada para usar con el CMS (habrá que indicar **servidor** donde está el CMS – si es el mismo se indica **localhost**, si no hay que dar su dirección pública – , **nombre de la base de datos**, **usuario** con permisos de modificación de la base de datos y su **contraseña**). Se comprobará la comunicación con la base de datos.
 - d) Escritura o modificación del archivo de configuración con los datos dados.
 - e) Creación de la estructura de la base de datos (tablas y parte de contenido) este proceso es automático.
 - f) Establecimiento de características generales de sitio: nombre, usuario administrador, contraseña, etc..
 - g) Escritura o modificación del archivo de configuración con los datos dados.
 - h) Reestablecimiento de los permisos de lectura y escritura adecuados para proteger el archivo de configuración y la web completa.
 - i) Opcionalmente, borrado o renombrado de los archivos del script de instalación.

Todo el resto de configuración del CMS se hará desde el propio CMS.