

DAW
Desarrollo de Aplicaciones Web
2º Curso

DWES
Desarrollo Web Entorno Servidor

UD 1. Introducción a los lenguajes de servidor

UD 1.1 Tecnologías para aplicaciones web

IES BALMIS
Dpto Informática
Curso 2025-2026
Versión 1 (05/2025)

UD1.1 – Tecnologías para aplicaciones web

ÍNDICE

1. Introducción
2. Estándares Web
3. Navegadores
4. Estructura de las páginas web
5. Tipos de AW
6. Funcionamiento, arquitectura y evolución de las AW
7. Sintaxis de las URL y lenguajes de aplicaciones web
8. Comparativa de las AW y las aplicaciones de escritorio
9. Desarrollo de AW
10. Sistemas Gestores de Contenidos (CMS)
11. El alojamiento o hosting de sitios web
12. Glosario de términos y estándares

1. Introducción

Internet: red de redes de ordenadores que pueden conectarse entre sí, independientemente de la plataforma, gracias a un protocolo estándar de comunicación denominado TCP/IP.

Podéis consultar un vídeo sobre la historia de Internet en:

https://www.youtube.com/watch?src_vid=PMAdfSHRids&v=i4RE6dBAjH4

Internet, también llamada WWW (world wide web) o simplemente web, ha pasado de ser una inmensa “biblioteca” de páginas estáticas a convertirse en un servicio que permite acceder a multitud de prestaciones y funciones, así como a infinidad de servicios.

Internet se basa en dos pilares fundamentales:

- **La forma de acceso:** a estos contenidos se accede a través del protocolo HTTP.
- **El lenguaje utilizado:** los contenidos se muestran analizando las instrucciones o comandos del lenguaje HTML

El protocolo **HTTP** (Hypertext Transfer Protocol) es un protocolo simple que permite una implementación sencilla de un sistema de comunicaciones al enviar cualquier fichero de forma fácil, simplificando el funcionamiento del servidor y posibilitando que servidores poco potentes atiendan cientos o miles de peticiones y reduzcan de este modo los costes de despliegue.

El lenguaje **HTML** (HyperText Markup Language) es un lenguaje de marcas que proporciona un mecanismo de páginas enlazadas, altamente eficiente y de uso muy simple.

Inicialmente la web era simplemente una colección de páginas estáticas, documentos, etc., que podían consultarse o descargarse.

El siguiente paso fue la inclusión de un método para confeccionar páginas dinámicas conocido como **CGI** (Common Gateway Interface).

Posteriormente, se empiezan a desarrollar alternativas a los CGI para solucionar el problema de rendimiento que presentan, y surgen sistemas de ejecución de módulos más integrados en el servidor, con lenguajes de programación interpretados como **Java, PHP o ASP** que permiten incluir código en las páginas HTML.




Estas tecnologías dieron paso a las **aplicaciones web** instaladas en un servidor y a las que se accede mediante un navegador.

Una **Aplicación Web** son las herramientas que los usuarios pueden utilizar accediendo a un servidor web a través de Internet, o de una intranet, mediante un navegador. En otras palabras, son una aplicaciones software que se ejecutan en un servidor web y se codifican en un lenguaje soportado por los navegadores web.

En la actualidad, el desarrollo de las tecnologías de Internet, junto con su capacidad de almacenamiento y ejecución de programas, ha dado lugar a lo que se conoce como **computación en la nube (cloud computing)**. Un ejemplo de ello es Google Apps.

2. Estándares Web



Las tecnologías principales para la construcción de páginas web son el **HTML (Hypertext Markup Language)** y **CSS (Cascading Style Sheets)**.

	<ul style="list-style-type: none"> • HTML proporciona la información estructurada en secciones, párrafos, título, imágenes, etc. Su versión actual es HTML5.
	<ul style="list-style-type: none"> • CSS es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML, XML, SVG. Proporciona la distribución de los elementos y su estilo (colores, tipos de letra, fondos, efectos...). Su versión actual es CSS3.
	<ul style="list-style-type: none"> • Bootstrap es un framework de código abierto para desarrollar sitios y aplicaciones web responsive (adaptables a diferentes dispositivos). • Dispone de clases predefinidas con CSS que nos proporcionan funcionalidad de manera rápida y sencilla. • Características: <ul style="list-style-type: none"> ◦ Ahorra tiempo: No tienes que diseñar todo desde cero. ◦ Responsive por defecto: Se adapta a móviles, tablets y PCs. ◦ Consistencia visual: Mantiene un diseño uniforme. ◦ Personalizable: Puedes modificar colores, tamaños, etc. ◦ Gran comunidad: Muchos recursos, temas y plantillas disponibles.

JavaScript es un lenguaje de programación versátil que se ejecuta en el navegador .
Mediante programación en el cliente (navegador) podemos:

- **Manipular el DOM:** Cambiar contenido HTML, estilos CSS y estructura de la página dinámicamente.
- **Eventos:** Responder a clicks, teclas, movimientos del mouse, etc.
- **Animaciones:** Crear efectos visuales con CSS + JS.
- **Validación de formularios:** Verificar datos antes de enviarlos al servidor.
- **Peticiones HTTP (AJAX/Fetch):** Consumir APIs REST (como Twitter, Google Maps, etc.).

Aunque algunos elementos de la sintaxis recuerden a Java, no tiene nada que ver.

	<ul style="list-style-type: none"> • Las páginas web pueden programarse con diversos lenguajes de script, aunque prácticamente sólo se usa JavaScript • Con JavaScript se puede modificar la página y ejecutar código cuando se interactúa con ella a través del modelo de objetos del documento (DOM – Document Object Model). El DOM es una librería de programación (API) para manipular el documento HTML cargado en el navegador • Se hacen peticiones al servidor web en segundo plano y se actualiza el contenido de la web utilizando AJAX (Asynchronous JavaScript And XML), tecnología que utiliza Javascript para intercambiar contenido en XML.
	<ul style="list-style-type: none"> • jQuery es una de las librerías más utilizadas para generar páginas dinámicas en el navegador. Es un recubrimiento de la API DOM que aporta facilidad de uso, potencia y compatibilidad entre navegadores. Se usa para gestionar el interfaz (la página) y para peticiones AJAX. • Simplifica mucho la programación en Javascript.

3. Navegadores

En nuestro modelo Cliente/Servidor, el Navegador es el que ejerce el papel de cliente. Siguiendo la definición de Wikipedia, un navegador o navegador Web (del inglés, web browser) es un programa que permite ver la información que contiene una página Web (ya se encuentre ésta alojada en un servidor dentro de la World Wide Web o en un servidor local, incluso si se trata de ficheros almacenados en el propio equipo cliente).

El navegador interpreta el código, HTML generalmente, en el que está escrita la página Web y lo presenta en pantalla permitiendo al usuario interactuar con su contenido y navegar hacia otros lugares de la red mediante enlaces o hipervínculos.

La funcionalidad básica de un navegador Web es permitir la visualización de documentos de texto, posiblemente con recursos multimedia incrustados. Los documentos pueden estar ubicados en la computadora en donde está el usuario, pero también pueden estar en cualquier otro dispositivo que esté conectado a la computadora del usuario o a través de Internet, y que tenga los recursos necesarios para la transmisión de los documentos (un software servidor Web).

El seguimiento de enlaces de una página a otra, ubicada en cualquier computadora conectada a la Internet, se llama navegación, de donde se origina el nombre navegador (aplicado tanto para el programa como para la persona que lo utiliza, a la cual también se le llama cibernauta).

En el mercado podemos encontrar cierta variedad de navegadores. Actualmente, los más extendidos son Mozilla Firefox, Microsoft Edge y Google Chrome. Podemos seguir un poco de la historia de aparición y evolución de los navegadores en Wikipedia (<http://es.wikipedia.org/wiki/Navegadores>) y en otras webs.



4. Estructura de las páginas web

Una **página web** es un documento electrónico adaptado para la Web, y que normalmente forma parte de un sitio Web. Su principal característica son los hipervínculos que enlazan con otra página, siendo esto el fundamento de la Web.

Una página web está compuesta principalmente por información (texto, módulos multimedia o ambos) así como por hiperenlaces; además puede contener o asociar datos de estilo para especificar cómo debe visualizarse, y también aplicaciones embebidas para hacerla interactiva.

Para entender la estructura de una página web, primeramente deberemos distinguir entre páginas web estáticas y páginas web dinámicas. En las primeras, la información aparece siempre tal cual fue escrita y diseñada; en las segundas, existe código de programación que adecua el contenido de la página a las solicitudes de los clientes, normalmente, accediendo a una base de datos.

Páginas web estáticas

Están basadas en código HTML y tienen una estructura limitada por etiquetas de dicho lenguaje.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8"/>
  <TITLE>Título de la Página Web </TITLE>
  ...
  Otras etiquetas HTML propias de la cabecera, código JavaScript o
  definición de estilos
  ...
</head>
<body>
  ...
  Otras etiquetas HTML propias que maquetan el contenido visual de la
  Página Web, llamadas a funciones JavaScript definidas en la cabecera,
  ...
</body>
</html>
```

Un ejemplo sería:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8"/>
  <title>Un pequeño documento</title>
</head>
<body>
  <h1>Titulo principal de mi documento</h1>
  <!-- Comentarios que no se muestran en la página -->
  <p>Ya he empezado a programar con <abbr title="Hyper Text Markup
Language">HTML</abbr>.</p>
</body>
</html>
```

Páginas web dinámicas

Además de código HTML, tendrán código en algún lenguaje de programación como PHP, ASP, SQL embebido, etc.; que permita la variación del contenido de la página, ya sea mediante consultas a una base de datos o por interacción con el usuario.

Con estas “páginas programadas”, lo que se visualiza por el navegador puede ser distinto cada vez. Parte del código programado se ejecutará en el servidor y así se construirá cada vez una página distinta que es la que se descargará en el cliente.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8"/>
  <TITLE>Título de la Página Web </TITLE>
  ...
  Otras etiquetas HTML propias de la cabecera, código JavaScript o
  definición de estilos
  ...
</head>
<body>
  ...
  Otras etiquetas HTML propias que maquetan el contenido visual de la
  Página Web, llamadas a funciones JavaScript definidas en la cabecera,
  ...
  <?php
  Código PHP con sentencias de flujo, sentencias SQL embebidas,...
  ?>
  ...
  Otras etiquetas HTML propias que maquetan el contenido visual de la
  Página Web, llamadas a funciones JavaScript definidas en la cabecera,
</body>
</html>
```

En las páginas web dinámicas, el código HTML y el de otros lenguajes de programación se puede ir alternando en diferentes ocasiones, siempre y cuando se advierta del principio y final de un bloque de código con las etiquetas correspondientes.

Para que el servidor sepa que deba traducir otro lenguaje, se le avisa mediante la extensión del fichero. Por ejemplo, para incluir PHP, el fichero, en lugar de tener la extensión “.html”, ahora deberá ser “.php”.

5. Tipos de AW

Cuando estamos utilizando un ordenador, una tablet o un teléfono móvil podemos estar utilizando básicamente dos tipos de aplicaciones:

- **Aplicaciones de escritorio:** Aquellas aplicaciones que no necesitan ninguna conexión a Internet o a una red de ordenadores para funcionar. Este tipo de aplicaciones suelen llamarse aplicaciones de escritorio, y podemos encontrar ejemplos muy variados: un procesador de textos, un lector de libros electrónicos, un reproductor de música o vídeo, e incluso videojuegos que tengamos instalados.
- **Aplicaciones Web:** Aquellas aplicaciones que sí necesitan conexión, bien sea a Internet o a un ordenador de su red local. En este otro grupo también tenemos ejemplos variados de aplicaciones. Por ejemplo, si compartimos un documento de texto en Google Drive, o si abrimos el navegador para acceder a una plataforma de un curso online, o incluso si jugamos a videojuegos junto con otras personas de otros lugares.

Una **aplicación web** es una aplicación informática distribuida cuya interfaz de usuario es accesible desde un cliente web, normalmente un navegador web.

Atendiendo a la forma de conectarse los distintos dispositivos tendríamos dos grandes grupos de aplicaciones:

- **Las aplicaciones P2P (peer-to-peer)**, donde todos los elementos conectados a la red tienen el mismo "rango", por así decirlo, y comparten información entre ellos. Es el mecanismo en el que se basan varios programas de descarga, como los de descarga de torrents.
- **Las aplicaciones cliente-servidor**, llamadas así porque consisten en que un conjunto de ordenadores (llamados clientes) se conectan a uno central (llamado servidor) que es el que les proporciona la información y los servicios que solicitan. En el caso de videojuegos donde nos conectamos a otro lugar para jugar con otras personas, estamos utilizando aplicaciones cliente-servidor, donde nuestro ordenador (uno de los clientes), tiene instalada una parte de la aplicación y el servidor al que se conecta le proporciona la información de los escenarios y del resto de jugadores y personajes.

Dentro del tipo de aplicaciones cliente-servidor, las aplicaciones web son un subtipo, quizá el más numeroso. Se llaman así porque accedemos a una página web para poderlas ver y usar.

Nos vamos a centrar en este último tipo de aplicaciones, las aplicaciones web. Con la expansión de Internet y de las redes de ordenadores, son aplicaciones en auge, y no dejan de aparecer nuevas herramientas y tecnologías para desarrollar aplicaciones web cada vez más vistosas, versátiles y adaptadas a gran variedad de dispositivos.

En esta unidad veremos qué ventajas e inconvenientes pueden tener las aplicaciones web con otros tipos de aplicaciones (en concreto con las de escritorio), y también veremos qué arquitectura tienen, es decir, qué mecanismos son los que las hacen funcionar como funcionan. También daremos alguna pincelada sobre qué patrones de arquitectura software podemos utilizar para desarrollar aplicaciones web.

6. Funcionamiento, arquitectura y evolución de las AW

En una aplicación web podemos distinguir en primer lugar dos grandes bloques:

- **el cliente**, donde está el usuario, que utiliza un navegador web (Google Chrome, Mozilla Firefox, Internet Explorer, etc.) para acceder a la aplicación
- **el servidor**, donde está ubicada la aplicación (el foro, la red social, el blog, el curso online, etc.), y que se encarga de atender las peticiones de los clientes y proporcionarles la información que solicitan

Las características habituales de las AW para navegadores son:

- Los clientes Web se conectan usando conexiones HTTP a través de TCP/IP
- Existen distintos tipos de usuarios.
- El procesamiento principal se realiza en el servidor.
- La aplicación accede a bases de datos.

Funcionamiento

Como hemos comentado, las aplicaciones web son un tipo de aplicaciones cliente/servidor. Este tipo de arquitecturas distribuyen las tareas entre quienes prestan los recursos y servicios (los servidores) y quienes los solicitan (los clientes).

En general, los pasos que se siguen en la comunicación cliente-servidor son:

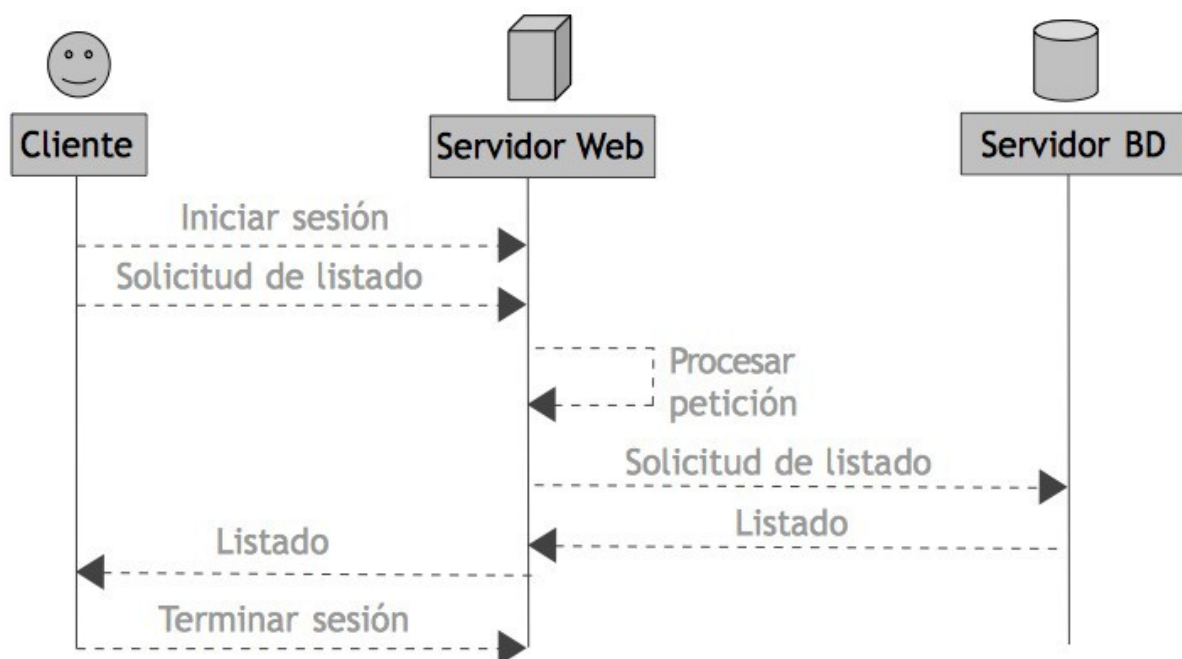
- 1) El cliente inicia sesión en el servidor
- 2) El cliente solicita al servidor el recurso o servicio que quiere utilizar (una página web, un documento, subir información, etc.)
- 3) El servidor recibe la respuesta del cliente, la procesa y decide qué programa debe darle servicio, enviando la petición a dicho programa.
- 4) El programa responsable procesa la petición, prepara la respuesta y la entrega al servidor.
- 5) El servidor envía la respuesta al cliente
- 6) El cliente puede volver al paso 2 y realizar una nueva petición, o bien
- 7) El cliente termina la sesión en el servidor.

En general, el servidor no tiene por qué ejecutarse solo, sino que podemos tener diferentes aplicaciones en diferentes equipos (o en el mismo), lo que se conoce como **arquitectura multicapa o multinivel**. Por ejemplo, un servidor de bases de datos en una máquina, un servidor web en otra (o en la misma que el de bases de datos), un servidor de correo electrónico... y así distribuir los procesos y el trabajo a realizar, e incluso configurar opciones de seguridad y rendimiento separadas para cada servidor.

Arquitectura Cliente/Servidor

La arquitectura de una aplicación define como se organizan los distintos módulos que la componen.

Por ejemplo, si el cliente conectara con el servidor para pedir un listado de noticias almacenadas en una base de datos, expresado como un diagrama de secuencia, el funcionamiento básico de esta petición (y de la arquitectura cliente-servidor en general) puede verse como algo así:



En este ejemplo, el servidor web y el servidor de bases de datos podrían estar instalados en la misma máquina o en máquinas separadas, cada una con su hardware específico y control de acceso de usuarios específico. En cualquier caso, estamos hablando de una arquitectura de tres niveles (cliente, servidor web y servidor de base de datos), que es algo bastante habitual en las aplicaciones web, pues casi todas cuentan con una base de datos con información que consultar y modificar.

Sin el servidor de base de datos, estaríamos ante una arquitectura de dos niveles, donde el servidor es polivalente, y puede responder directamente a las peticiones de los clientes sin consultar con otros servidores o aplicaciones. Esta opción es menos flexible, menos segura y puede ofrecer peor rendimiento en sistemas congestionados, al no poder dividir el trabajo entre distintos tipos de servidores.

Un servicio web ofrece una interfaz de programación de una determinada funcionalidad (servicio) accesible a través de Internet y basada en estándares W3C (<http://www.w3c.es/>).

Componentes de un Servidor (WAMP, LAMP)

El servidor será un ordenador al que los clientes puedan acceder a través de la red, ya sea Internet o una Intranet de una empresa o local.

Para ser en realidad un servidor deberá tener instalado el software necesario. Hay servidores de ficheros o datos, servidores de impresión, servidores de correo,... pero lo que en este curso nos interesan son los Servidores Web.

Para que una aplicación web funcione necesitamos una serie de componentes:

<i>Sistemas Operativos</i>	<i>Servidor Web (HTTP, traductor de HTML)</i>	<i>Gestor de BD</i>	<i>Lenguajes Script de Servidor</i>
Windows	Apache	MySQL/MariaDB	PHP
Linux	Nginx	PostgreSQL	JSP
	IIS (Internet Information Server)	SQL Server	Javascript
		Oracle	TypeScript
		...	

Una combinación muy extendida en Internet es un servidor con Apache, MySQL y PHP. Para las prácticas de los desarrolladores suelen instalarse unos paquetes de software que instalan estos componentes de manera local en cualquier ordenador:

Las siglas corresponden a:

- **X:** Sistema operativo. Puede ser L=Linux, W=Windows, M=MAC OS o X=Multiplataforma
- **A:** Servidor web de Apache;
- **M:** MySQL/MariaDB, el gestor de bases de datos;
- **P:** Lenguaje Script de programación del Servidor Web. Generalmente son Perl, PHP, o Python, los lenguajes de programación.

Así tenemos:

- WAMP: Apache + MySQL + PHP, para instalar en Windows.
- LAMP: Apache + MySQL + PHP, para instalar en Linux.
- MAMP: Apache + MySQL + PHP, para instalar en MAC OS.
- XAMP: Apache + MySQL + PHP, para instalar en multiplataforma.

Podríamos resumir que gracias a Apache, se pueden gestionar las páginas web codificadas en HTML; gracias al MySQL se puede gestionar información en bases de datos; y gracias al traductor de PHP se pueden definir páginas dinámicamente, programándolas y permitiendo la inclusión de sentencias SQL que actuarán sobre MySQL.

Evolución de la arquitectura de las aplicaciones web

Arquitectura con Cliente estático y Servidor estático

- La web está formada por HTML, CSS, Imágenes, PDF, etc... (pero no incluye JavaScript).
- A este tipo de web no se le suele llamar aplicación web porque nada es dinámico. Se denomina simplemente página web.

Arquitectura Cliente estático y Servidor dinámico

- Cuando se ejecuta código, normalmente se hacen consultas a una base de datos para recuperar la información
- Lo más habitual es que se genere la página HTML de forma dinámica
- También se pueden generar recursos de otro tipo (imágenes, PDFs...)
- Si el usuario pulsa un link, se recarga la página al completo
- Es la arquitectura de las primeras “aplicaciones web” y todavía sigue habiendo muchas web con esta arquitectura
- El contenido es dinámico, porque se ejecuta código en el servidor para generar dicho contenido

Arquitectura Cliente dinámico y Servidor estático

- El contenido de la página web está alojado en el disco duro del servidor (estático)
- El cliente es dinámico porque las páginas incluyen código JavaScript que se ejecuta en el navegador
- Este JavaScript se usa para incluir efectos gráficos para mostrar u ocultar información en función de los elementos que se seleccionan (para documentos largos), mostrar menús desplegables o crear páginas adaptables para móviles (responsive)

Arquitectura Cliente dinámico y Servidor dinámico

- La mayoría de las aplicaciones web actuales son dinámicas tanto en cliente como en servidor
- Se usa JavaScript para realizar efectos gráficos
- Se usa JavaScript para refrescar contenido parcial de la página en segundo plano (AJAX - Asynchronous JavaScript And XML). Aunque existen muchas aplicaciones web que no usan AJAX, la mayoría de las aplicaciones que se han desarrollado en los últimos años sí lo usan en algunas de sus páginas por la mejora en la experiencia del usuario

7. Sintaxis de las URL y lenguajes de aplicaciones web

URL

Hemos visto que, en el esquema de funcionamiento de una aplicación web, el cliente solicita recursos al servidor. La forma en que los solicita es mediante URLs. Una URL (Uniform Resource Locator) es una manera de identificar y localizar cada recurso de una web. Por ejemplo, cuando escribimos en un navegador una dirección como `http://www.miweb.com/paginas/pagina.html`, estamos introduciendo una URL para localizar un recurso (en este caso, una página HTML).

Una URL se compone de:

- **El protocolo**, que indica las reglas que se van a seguir para comunicarse cliente y servidor. Veremos más adelante algunos ejemplos de protocolos, pero para lo que nos importa, en una URL el protocolo va al principio, hasta los dos puntos y el delimitador `//`. En nuestro ejemplo, el protocolo sería `http://`
- **El nombre de dominio**, que identifica al servidor y la empresa/web a la que vamos a conectar. Va justo detrás del protocolo, hasta la siguiente barra. Normalmente termina en `.com`, `.es`, `.net`, etc. En nuestro ejemplo sería `www.miweb.com`
- **La ruta hacia el recurso**, que comprende todas las carpetas y subcarpetas (si las hay) y el nombre de archivo que queremos obtener. En nuestro ejemplo, la ruta sería `/paginas/pagina.html`

El navegador obtiene la URL que ha escrito el usuario, transforma el nombre de dominio en una dirección IP (según su servidor DNS asignado) y envía al servidor indicado la petición.

Lenguajes

Al hablar de aplicaciones web en los siguientes temas, hablaremos de algunos conceptos que nos deben empezar a sonar ya, como los ya comentados de cliente, o servidor. Otro de estos conceptos es el de lenguaje. Para las personas, un lenguaje es un conjunto de palabras y signos que nos permiten comunicarnos. Para un ordenador viene a ser algo parecido: un conjunto de palabras y signos que permiten que el usuario le dé una serie de órdenes, y que el ordenador sepa interpretar.

En el ámbito de las aplicaciones web, distinguimos dos tipos de lenguajes:

- **Lenguajes en el entorno cliente o lenguajes cliente**: son los que permiten que el cliente interactúe con la aplicación web. Para ello, el cliente debe poder ver la aplicación web en el navegador, e interactuar con ella (pinchar en enlaces, rellenar formularios, etc.). En este lado, normalmente se tienen dos lenguajes para que todo esto se pueda llevar a cabo: el lenguaje HTML para poder crear y visualizar páginas web, y el lenguaje Javascript para poder facilitar la

interacción entre el usuario y el navegador. Los veremos con más detalle en temas posteriores.

- **Lenguajes en el entorno servidor o lenguajes servidor:** son los que permiten que el servidor realice ciertas tareas cuando le llegan las peticiones de los clientes, como por ejemplo consultar una base de datos, guardar información en un fichero, o cargar una foto que el usuario está subiendo al servidor. En este otro lado, existen varias familias de lenguajes que podemos elegir, dependiendo del servidor web que queramos utilizar. Por ejemplo, podemos utilizar lenguaje ASP .NET (para servidores de Microsoft y entornos Windows), o el lenguaje JSP (lenguaje Java para aplicaciones web), o el lenguaje PHP, entre otros.

Algunos ejemplos de lenguajes cliente son:

- **JavaScript**
- TypeScript

Algunos ejemplos de lenguajes servidor son:

- Java EE, ASP.NET, **PHP**, JSP, ColdFusion, Ruby on Rails, Python Django, Groovy, Node.js, Scala Play, ...
- CGI (C, Perl, ...), aunque estos cada vez se usan menos

En nuestro caso, usaremos al principio Javascript para el cliente y PHP para el servidor.

8. Comparativa de las AW y las aplicaciones de escritorio

Una vez hemos visto qué arquitectura tienen las aplicaciones web, ¿qué ventajas e inconvenientes podemos ver que tienen frente al otro gran grupo de aplicaciones, las de escritorio?

Las ventajas de las aplicaciones web son:

- Una de las principales ventajas que tienen es que **el cliente no tiene que instalar** nada en su equipo, o casi nada. Basta con tener instalado un navegador web (Firefox, Opera, Chrome, Edge, etc.). El servidor es el que necesita tener instalado el software para hacer funcionar la aplicación (servidor web, base de datos, archivos necesarios, etc.). Debido a la arquitectura de las aplicaciones web, el navegador suele quedar relegado a mostrar la interfaz de usuario (menús, opciones, formularios, etc.), mientras que toda la compleja lógica de la aplicación se lleva en el lado del servidor.
- Otra ventaja es que **el cliente apenas tiene carga de trabajo**. El servidor lleva gran parte del peso del procesamiento (almacenar datos, enviar archivos, etc.). Esto en ocasiones puede suponer una desventaja, si el servidor se satura.

Veremos que hay mecanismos para que el cliente haga parte del trabajo y libere así al servidor, con lenguajes como Javascript, y tecnologías como AJAX.

- **Bajo coste en actualizar los equipos con una nueva versión.** Cuando existen varios usuarios en equipos diferentes, **no necesitamos instalar la misma aplicación en todos los equipos para utilizarla**, ya que todos se conectarán con el navegador al servidor web.

Por otra parte, si hacemos un cambio en la aplicación (por ejemplo, cambiamos los colores de los menús, o añadimos alguna funcionalidad más), se hará en el servidor y automáticamente lo verán actualizado todos los usuarios.

Esto hace que **las aplicaciones web sean más fáciles de mantener y actualizar.**

- **Acceso a la última y mejor versión.** Como consecuencia del punto anterior, se evita que pueda existir algún equipo que ejecute una versión diferente y desactualizada. Si existen ordenadores con distintas versiones del programa se pueden originar problemas de consistencia en la información o pérdida de funcionalidad.
- **Información centralizada.** En una aplicación web, no solamente la lógica de la aplicación está centralizada en el servidor, sino también los datos que se ubican en una base de datos centralizada (en ese servidor u otro destinado a tal fin).

La centralización tiene la ventaja de facilitar el acceso a la misma.

- **La compatibilidad** es otro factor importante. Muchas aplicaciones de escritorio sólo funcionan para ciertos sistemas operativos, o funcionan diferente dependiendo del sistema. Una aplicación web, en general, funciona de la misma forma independientemente de la plataforma. Eso sí, para dispositivos móviles suelen tener una versión adaptada, sin tanto texto y con funcionalidades más reducidas o localizadas, para poderse manejar mejor en una pantalla pequeña.
- **El control de usuarios y la seguridad** también están centralizados en el servidor, de forma que desde él podemos ver y dar permisos a quien intenta acceder. En una aplicación de escritorio es más difícil controlar que no se entre de forma ilegal en alguno de los equipos donde esté instalada.
Es más, al no ubicarse los datos en los ordenadores de los usuarios, en caso de robo o incendio de los puestos de trabajo, la empresa no ha perdido información y puede instalar rápidamente un nuevo puesto de trabajo (PC con un navegador web).
- **La movilidad**, ya que al estar la aplicación instalada en un servidor remoto, si disponemos de conexión a Internet (fija o móvil) podemos utilizar la aplicación desde cualquier lugar.
- **La escalabilidad del sistema**, ya que la mayor parte del trabajo recae en el servidor, se pueden siempre mejorar sus prestaciones y ampliar la capacidad de número de clientes mejorando el hardware del servidor, añadiendo más servidores, etc.

El equipo cliente queda relegado a mostrar los resultados y formularios, para lo cual no es necesario un hardware potente en los puestos de trabajo, lo que se

traduce en reducción de costes y una mayor longevidad en el uso de los mismos (no hay que cambiar el hardware de los puestos porque ahora se requieran operaciones más complejas).

Los inconvenientes de las aplicaciones web son:

- Uno de los inconvenientes que podemos encontrar en una aplicación web es su **riqueza gráfica**, si la comparamos con aplicaciones de escritorio. Existen algunos tipos de efectos (animaciones, 3D, etc.) que aún son difíciles o costosas de conseguir en una aplicación web, aunque poco a poco van apareciendo librerías que mejoran las posibilidades (generalmente realizados en Javascript).
- Otro inconveniente evidente es la **necesidad de conexión a la red** para poder utilizar la aplicación, aunque algunas aplicaciones como Dropbox permiten trabajar con un documento sin conexión y actualizar los cambios en el servidor cuando exista conexión.
- **El tráfico generado en la red** para acceder al servidor también puede ser un factor importante a considerar. Si hay varios clientes accediendo a la aplicación y solicitando datos, la cantidad de información que se envía por Internet o que se solicita al servidor puede llegar a colapsarlo, y que deje de dar servicio a todos los clientes hasta que se recupere.
 - Esto no es un problema en otro tipo de arquitecturas, como por ejemplo las redes P2P, donde el número de clientes mejora el rendimiento del sistema, al haber más lugares desde donde compartir y cargar/descargar los recursos.
- Otro inconveniente es el **tiempo de respuesta**. Al ser aplicaciones cliente/servidor, puede pasar un tiempo considerable desde que, por ejemplo, enviamos un formulario hasta que el servidor nos avise de que los datos se han guardado correctamente. Estos tiempos son mucho menores en aplicaciones de escritorio, al no necesitar comunicación entre equipos.
- También podemos citar entre los inconvenientes la apariencia. A pesar de que las aplicaciones web son más compatibles, su **apariencia final depende del navegador** que utilice el cliente para visualizarlas. Así, la misma aplicación puede verse de forma diferente según si estamos utilizando Chrome, Internet Explorer, Firefox, etc.

9. Desarrollo de AW

A la hora de comunicar clientes y servidores, es necesario establecer un protocolo de comunicación, es decir, una serie de reglas que indiquen qué tipo de mensajes se van a intercambiar, en qué orden y qué contenido va a tener cada tipo de mensaje, de forma que los dos extremos de la comunicación (cliente y servidor) puedan entenderse.

Al trabajar con aplicaciones web, los **protocolos de comunicación** más empleados son:

- **HTTP** (HyperText Transfer Protocol), un protocolo existente desde 1990 y que permite la transferencia de archivos en general, aunque principalmente de archivos HTML (es decir, documentos web). Se sigue un esquema de peticiones y respuestas entre cliente y servidor como el visto anteriormente.
- **HTTPS**, versión segura del protocolo anterior, donde los datos de las peticiones y las respuestas se envían encriptados, para que nadie que intercepte la comunicación pueda descifrar el contenido de la misma. Este tipo de protocolos se suele utilizar en sistemas bancarios, plataformas de pago (Paypal, por ejemplo), y otras aplicaciones que manejen información delicada (DNIs, números de tarjetas de crédito, etc.).

Normalmente, los navegadores web cambian automáticamente del protocolo “normal” HTTP a HTTPS al conectar con páginas que necesitan ser más seguras (login, datos de pago, etc.). Se puede comprobar el cambio mirando la barra de dirección del navegador: al acceder al protocolo seguro se mostrará el protocolo https en la barra, o bien el icono de un candado. Sin embargo, sí deberemos configurar nuestro servidor web para aceptar comunicaciones HTTPS, si fuese el caso.

- Existen otros protocolos menos utilizados a la hora de trabajar con aplicaciones web, pero que se utilizan igualmente en otras aplicaciones que requieran de Internet. Por ejemplo:
 - Para el envío y recepción de correo electrónico se emplean los protocolos **SMTP** o **POP3/IMAP**, respectivamente.
 - Para enviar archivos a un servidor remoto se puede emplear (además del propio protocolo HTTP) el protocolo **FTP** o su versión segura con comunicación cifrada **SFTP**.

La evolución ha tenido como resultado que hay una gran cantidad de tecnologías, librerías, herramientas y estilos arquitectónicos para desarrollar una aplicación web.

Existen dos enfoques en el desarrollo de aplicaciones web:

- Creación de webs con tecnologías de desarrollo
- Creación de webs con sistemas gestores de contenido

Creación de webs con tecnologías de desarrollo

- **Arquitecturas de aplicaciones web:** Una aplicación web puede tener diferentes arquitecturas. Esto determina cómo se usan las diferentes tecnologías existentes
- **Tecnologías de cliente:** Tecnologías que permiten crear interfaces de usuario atractivos y permiten la comunicación con el servidor. Basadas en HTML, CSS y JavaScript.
- **Tecnologías de servidor:** Tecnologías que permiten implementar el comportamiento de la aplicación web en el servidor: funcionalidad de la aplicación, generación de informes, compartir información entre usuarios, envío de correos, etc...
- **Bases de datos:** La gran mayoría de las webs necesitan guardar información. Las bases de datos son una parte esencial del desarrollo web.

Creación de webs con sistemas gestores de contenido

- Existen aplicaciones web cuya principal funcionalidad es la **publicación de contenido**: blogs, páginas de empresas, organismos públicos, etc.
- Todas estas webs tienen **mucho en común**, prácticamente sólo se diferencian en el contenido y en el aspecto gráfico
- Para desarrollar este tipo de webs, en vez de desarrollar la web con técnicas de desarrollo, se utiliza **un software ya desarrollado** y se personaliza y adapta a las necesidades
- A las aplicaciones de este tipo se las denomina **Sistemas Gestores de Contenido (CMS - Content Management System)**.

Internet y las aplicaciones web han hecho evolucionar la administración de sistemas en muchos aspectos

- Para que una aplicación web funcione necesita que el sistema donde se instale disponga de un **servidor web** y habitualmente una **base de datos**
- Como la web tiene que estar disponible para los usuarios de Internet, habitualmente se instala en sistemas que se alquilan a terceros: **alojamiento en la nube (cloud)**
- Como las aplicaciones web pueden tener un número muy grande de usuarios y tienen que estar **siempre disponibles**, se utilizan técnicas de **escalabilidad** (posibilidad de crecer y aumentar fácilmente las características) y **tolerancia a fallos** (el servidor no debe pararse ante un fallo).

10. Sistemas Gestores de Contenidos (CMS)

Un Sistema Gestor de Contenidos (CMS - Content Management System) es una Aplicación Web genérica que permite la creación y administración de contenidos vía web.

Las características principales son:

- El sistema permite manejar de manera independiente el contenido y el diseño, permite el cambio de diseño (con templates o themes)
- Los CMS han evolucionado para convertirse en un nuevo modelo de desarrollo de aplicaciones web configurando y adaptando módulos con un interfaz web
- Existen multitud de CMS con enfoques y objetivos diferentes
 - Ejemplos: Drupal (PHP), Joomla (PHP), Wordpress (PHP), Plone (JavaScript), Moodle (PHP), Liferay (Java)
- Características generales
 - **Multiplataforma:** Suelen funcionar en distos sistemas operativos, ya que están implentados en lenguajes script de servidor.
 - **Múltiples idiomas:** Están pensados para una audiencia internacional y proporciona opciones para crear un portal multilingüe
 - **Administración y Análisis Administración via Web:** La administración y configuración del sistema se puede realizar enteramente con un navegador

Los CMS se estructuran en zonas accesibles según los permisos del usuario que los utiliza. Las zonas existentes más comunes son:

- **Zona Pública (FrontEnd):** Es la parte visible de forma pública que no necesita identificación de usuario y por tanto se puede consultar de forma anónima.
- **Zona de Administración (BackEnd):** Es la parte que usan los administradores para instalar y configurar el CMS.
- **Zona Privada:** Es la parte accesible por un usuario que se ha identificado. Según el nivel de permisos podrá ver más o menos contenido. En muchas ocasiones se le permite crear contenido en el CMS. Por ejemplo: cuando en un blog un usuario identificado crea una entrada de contenido.

11. El alojamiento o hosting de sitios web

Cuando una empresa desea hacerse visible en Internet, debe crear su página web y hacerla pública. Podrá subcontratar este trabajo o hacerlo con su propio personal, si dispone del adecuado.

Para conseguir una página publicada en Internet, deberá contratar un servicio de alojamiento en el servidor de una empresa dedicada a ello, lo que se conoce como “hosting”.

Si la dirección que deba escribir el posible visitante en su navegador web, queremos que tenga relación con el nombre de la empresa, también deberemos “comprar un dominio”. Se dice comprar, aunque realmente se trata de un alquiler, ya que si dejamos de pagar perderemos el dominio.

Dominio

El primer paso a dar será comprar el dominio, aunque normalmente, la primera vez se contratan el dominio y el hosting simultáneamente.

El nombre de dominio es un nombre único en Internet, de modo que cuando se escribe en el navegador nos enseñará la página web correspondiente y no otra. Los nombres de dominio van asociados a una terminación concreta como .com, .es, .org, .net, ...

Normalmente, se ofrece un contrato anual por un nombre de dominio, aunque se suelen encontrar ofertas por más años o, incluso, el dominio gratis si se contrata a la vez algún servicio de hosting.

Para evitar la duplicidad de dominios, son asignados por la ICANN (Internet Corporation for Assigned Names and Numbers), una organización internacional sin ánimo de lucro, responsable de asignar espacio de direcciones numéricas del Protocolo de Internet (IP) y de la gestión del sistema de nombres de primer nivel genéricos (gTLD) y de códigos de países (ccTLD).

En el proceso de registro de un nombre de dominio, deberemos acudir a través de Internet a un proveedor de servicios de Internet. Éste comprobará si el nombre de dominio está disponible (no registrado por nadie más) y nos planteará su oferta para formalizar el contrato.

En el caso de que un dominio esté ocupado o en uso, existe la posibilidad de “transferir el dominio”. Esto puede ser porque lo tengamos registrado nosotros en otro servidor y deseamos cambiarlo o porque sea propiedad de otro usuario y le solicitemos que nos lo ceda. Esto último entraría dentro de lo que se conoce como compraventa de dominios.

Ejercicio 1.1 - Comprobar la disponibilidad de un dominio

Existen multitud de portales que ofrecen este servicio.

Puedes usar <https://www.ionos.es/dominios/consulta-de-dominios> para realizar tu consulta

Sitio web o Espacio web (Hosting)

Los proveedores de espacio web suelen dar servicio a muchos clientes (miles), de modo que se suelen encargar del registro de sus dominios y del hosting correspondiente a cada uno.

Existe la opción de que el dominio lo administre un proveedor y “redireccione” a los navegadores a otro proveedor donde está el hosting.

El espacio web consiste en realidad en espacio de disco duro ofrecido para almacenar las páginas y datos de cada cliente. Si el proveedor es medianamente importante deberá contar con varios ordenadores con potencia y capacidad de almacenamiento suficientes. Así se reparte la carga de tráfico de red y de espacio de almacenamiento reservado.

Existen otros conceptos relacionados, como servidor compartido o servidor dedicado, IP fija, etc.

Al igual que en el caso de los dominios, se suelen hacer contratos anuales, aunque habrá ofertas por un número mayor de años o incrementos por pagar cada mes sin compromiso de permanencia.

Servicios del proveedor

Cuando se contrata un hosting es fundamental elegir un paquete adecuado de servicios por el que pagar. Lo normal es que nos ofrezcan grupos de servicios y características diferentes de cada uno, según vamos pasando de una oferta para un individuo hacia una oferta para una empresa con grandes necesidades.

A continuación, enumeramos algunos servicios y aspectos importantes a tener en cuenta a la hora de elegir un hosting:

- **Más dominios:** si se nos ofrece contratar sucesivos dominios sin incremento de precio.
- **Subdominios:** es la posibilidad de subdividir nuestro dominio en secciones, por ejemplo, tienda.midominio.com, fundacion.didominio.com, ...
- **Tráfico mensual:** es el límite de MB o GB de transferencia de información que permitirán a nuestro dominio cada mes.
- **Espacio web:** espacio de disco duro que nos ofrecen (MB, GB, ilimitado).
- **Correo electrónico:**
 - cuántas cuentas de correo nos ofrecen
 - si es de tipo POP, IMAP o webmail
 - características como crear listas de correo, alias, reenvío, etc.
- **FTP o SFTP:**
 - si nos ofrece servicio FTP para la transferencia de ficheros entre nuestro equipo y el servidor, o si se ofrece WebFTP.
 - Número de cuentas FTP que se podrán crear.
- **Bases de datos:**
 - si nos ofrecen la posibilidad de crear y gestionar bases de datos y de qué tipo (MySQL, Server SQL, ...)
 - número de bases de datos y capacidad máxima de cada una.
- **Programación:** los lenguajes de programación que soporte el servidor, como PHP, ASP, Perl, Python,...
- **Soporte técnico:** si nos ofrecen atención telefónica, consultas vía correo electrónico, atención online, FAQ, manuales, tutoriales,...
- **Otros servicios:** tecnologías multimedia (streaming de video, podcast de audio,...), estadísticas, logs, acceso seguro, comercio electrónico, ...

Ejercicio 1.2 - Comparación de proveedores de hosting para PHP

Realiza una actividad de búsqueda y comparación de proveedores de hosting

12. Glosario de términos y estándares

Sistemas operativos	Windows Linux MAC
Navegadores	Chrome Firefox Opera Safari Microsoft Edge
Web	URL Web Responsive
Servidores web	Apache Nginx IIS – Internet Information Server (Microsoft)
Estándares de archivos web	HTML / HTML5 / XHTML CSS / CSS3 XML
Estándares de contenido dinámico en cliente	Javascript Jquery Bootstrap
Estándares de lenguaje script en el Servidor	PHP Javascript ASP.NET Phyton Java
Bases de Datos	MySql Oracle SQL Server (Microsoft) Firebird PostgreSQL