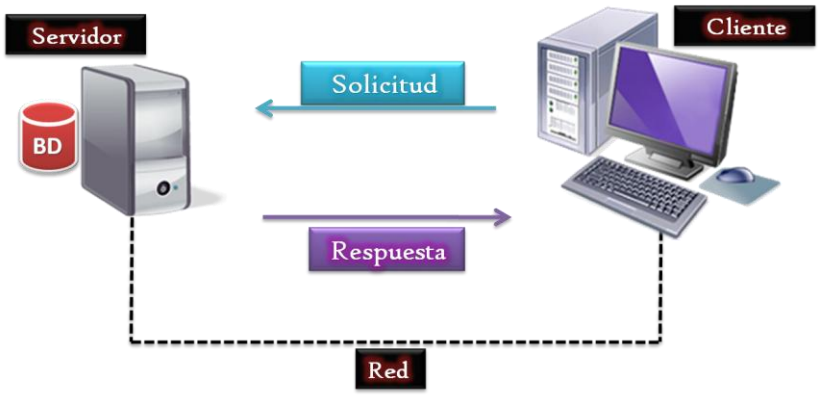
UD 1. IMPLANTACIÓN DE ARQUITECTURAS WEB

1. Introducción. Algunos conceptos previos

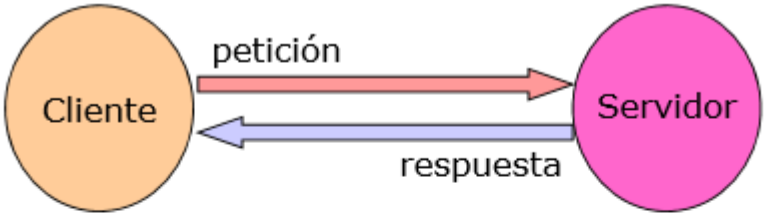
Internet es una red de redes a escala mundial de millones de computadoras interconectadas con el conjunto de protocolos propio (TCP/IP).

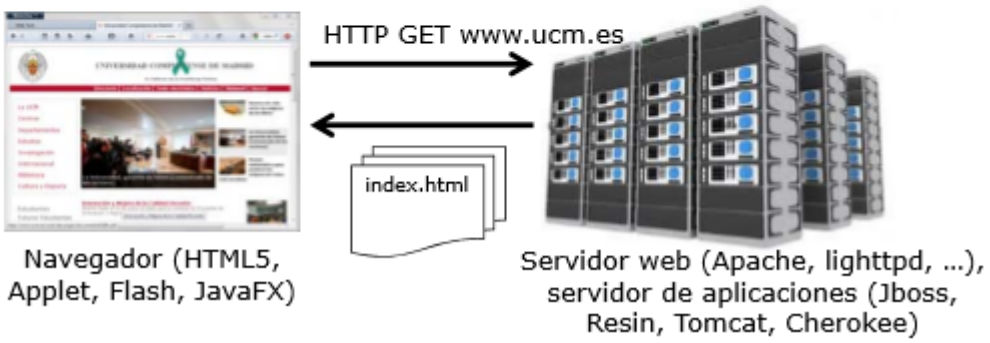
La World Wide Web, la Web o WWW, es un sistema de hipertexto que funciona sobre Internet. Desde el punto de vista de la arquitectura de la www se distinguen 3 componentes.

- **Servidor**: que es donde residen realmente los datos, reglas y lógica de la aplicación. Se le conoce con el término back-end.
- **Cliente**: Donde se encuentra el usuario final utilizando la aplicación por medio de un navegador/browser. Se le conoce con el término front-end.
- **Conexión de red:** La comunicación entre servidor y cliente se establece a través de la estructura de comunicación, valiéndose de los protocolos de comunicación propios de la web (en concreto a través del HTTP o Protocolo de transferencia de hipertexto)

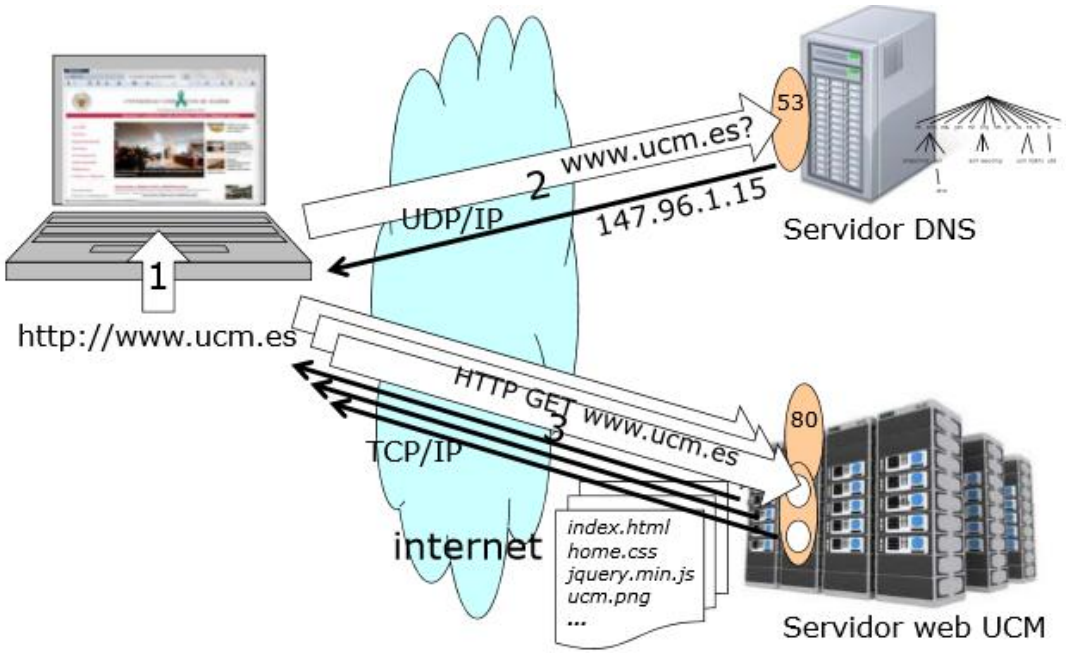


De esta forma la **WWW** sigue el modelo **cliente-servidor.** Un cliente demanda servicios o recursos a un servidor a través de una interfaz, usando un protocolo.





Su funcionamiento es el siguiente:



Los **estándares WWW** especifican los mecanismos necesarios para su funcionamiento. Los más importantes son:

- **Modelo estándar de nombres**: todos los servidores, así como el contenido de la WWW se denominan según un Localizador Uniforme de Recursos (Uniform Resource Locator: URL).

  Las URL indican cómo localizar en Internet un determinado recurso. Su formato es:

protocolo://maquina:puerto/camino/fichero

- El protocolo habitualmente es http (HyperText Transport Protocol) o https (HyperText Transport Protocol Secure), aunque pueden emplearse otros protocolos como ftp (File Transfer Protocol).
- La máquina es el nombre o la IP del servidor al cual nos queremos conectar. Habitualmente, se emplea un nombre (como Google.com) que es traducido a una IP por el servicio de DNS.
- Después de la máquina, separado por ":" puede indicarse el puerto al cual nos queremos conectar. Por defecto, cada protocolo tiene un puerto asignado; por ejemplo http tiene asignado el puerto 80 y https tiene asignado el puerto 443. Al publicar una página web en Internet, lo habitual es emplear el puerto por defecto; sin embargo cuando estamos construyendo esa página web en nuestro equipo local a menudo se emplean puertos diferentes para depuración y testeo de dicha página.
- Camino es la ruta en el sistema de archivos de la máquina remota donde se encuentra el recurso al cual queremos acceder. Dicha ruta es relativa al directorio raíz de la web.
- Fichero es el recurso concreto al que queremos acceder dentro de un directorio de la máquina.

Por ejemplo:

http://java.org:4040/ejemplo/inicio.html

Accedería empleando el protocolo http al puerto 4040 de la máquina java.org. Dentro de esa máquina debería haber un servidor web sirviendo el contenido de algún directorio de la máquina. Partiendo de ese directorio, el servidor web iría al directorio "ejemplo" y buscaría el recurso con nombre inicio.html, que serviría al cliente que lo ha solicitado.

- **Contenido**: a todos los contenidos en la WWW se les especifica un determinado tipo permitiendo de esta forma que los browsers (navegadores) los interpreten correctamente.

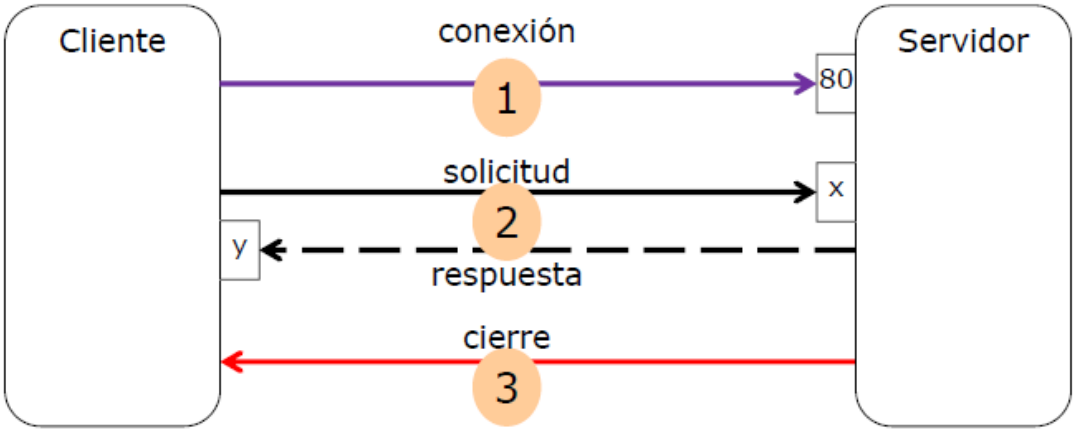
- **Formatos de contenidos estándar:** todos los navegadores soportan un conjunto de formatos estándar, por ejemplo HTML, JavaScript, etc.
- **Protocolos estándar**: éstos permiten que cualquier navegador pueda comunicarse con cualquier servidor web. El más comúnmente usado en WWW es HTTP (Protocolo de Transporte de HiperTexto), que opera sobre el conjunto de protocolos TCP/IP.

Esta infraestructura permite a los usuarios acceder a una gran cantidad de aplicaciones y servicios de terceros.

2. Protocolo HTTP

Protocolo de aplicación para transferencia de hipertexto. Funciona sobre TCP/IP. Permite a un navegador (cliente, user agent) solicitar una página a un servidor y que éste la envíe. Basado en el envío de comandos y respuestas en texto ASCII.

Ejemplo de funcionamiento del protocolo



Para recuperar el fichero en el URL http://www.dominio.com/path/fichero.html

- El cliente solicita una conexión en el puerto 80 del host www.dominio.com
- El cliente envía a través del nuevo socket el siguiente texto:

```
GET /path/fichero.html HTTP/1.0
From: usuario@ucm.es
User-Agent: HTTPTool/1.0
[línea en blanco CRLF]
```

- La respuesta del servidor llegará por el mismo socket:

```
HTTP/1.0 200 OK
```

```
Date: Wed, 12 Dec 2012 12:09:34 GMT
Content-Type: text/html
Content-Length: 1354
<html>
<body>
<h1>Bienvenido al curso de la Web</h1>
(resto del contenido)
..
</body>
</html>
```

- Tras enviar la respuesta el servidor cierra el socket

Mensajes HTTP Request (solicitud)

El mensaje http está formado por

- Línea inicial
- 0..n líneas de cabecera
- Línea en blanco (CRLF)
- Cuerpo de mensaje opcional (un fichero, solicitud de datos, datos,resultado de una solicitud)

Es decir las peticiones de http siguen el siguiente formato:

Método SP URL SP Versión Http CRLF (nombre-cabecera: valor-cabecera (, valor-cabecera)*CRLF)* Cuerpo del mensaje

SP significa espacio en blanco. CRLF significa retorno de carro. Todo lo que aparece entre paréntesis es opcional. Cuando después de un paréntesis hay un *, significa que el contenido del paréntesis puede aparecer repetido cero o más veces.

La primera parte de la petición es el método que se emplea para realizar la petición (Habitualmente GET o POST). Existen diferentes métodos

- GET. Devuelve el recurso identificado en la URL pedida.
- HEAD. Funciona como el GET, pero sin que el servidor devuelva el cuerpo del mensaje. Es decir, sólo se devuelve la información de cabecera.
- POST. Indica al servidor que se prepare para recibir información del cliente. Suele usarse para enviar información desde formularios.
- PUT. Envía el recurso identificado en la URL desde el cliente hacia el servidor.

- OPTIONS. Pide información sobre las características de comunicación proporcionadas por el servidor. Le permite al cliente negociar los parámetros de comunicación.
- TRACE. Inicia un ciclo de mensajes de petición. Se usa para depuración y permite al cliente ver lo que el servidor recibe en el otro lado.
- DELETE. Solicita al servidor que borre el recurso identificado con el URL.
- CONNECT. Este método se reserva para uso con proxys. Permitirá que un proxy pueda dinámicamente convertirse en un túnel. Por ejemplo para comunicaciones con SSL.

A continuación, separada mediante un espacio en blanco, se indica la URL del recurso que se está pidiendo. Después, separada por un nuevo espacio en blanco, se indica la versión del protocolo http que se está empleando 1.0 o 1.1. A continuación se pueden encontrar una o varias cabeceras (HTTP 1.1 define 46 y requiere al menos una Host). Las cabeceras tienen asociado un nombre y después llevan ":". Después de los ":" debe ir un valor para la cabecera. Es posible que una misma cabecera tenga asociados varios valores. En este caso, los valores se separan por ","; los espacios en blanco dentro del valor de una cabecera se considera que forman parte del valor, y no que son separadores. En el mensaje puede haber varias cabeceras separadas por retorno de carro. Después de las cabeceras, viene el cuerpo del mensaje, que puede ser vacío (es el caso del método GET) o no (es el caso del método POST).

Por ejemplo, una petición http podría ser:

```
GET /en/html/dummy?name=MyName&married=not+single&male=yes HTTP/1.1
Host: www.explainth.at
User-Agent: Mozilla/5.0 (Windows;en-GB; rv:1.8.0.11) Gecko/20070312
Firefox/1.5.0.11
Accept: text/xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-gb,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

En este caso, el cuerpo está vacío. Un ejemplo de petición http con cuerpo no vacío podría ser:

```
POST /en/html/dummy HTTP/1.1
Host: www.explainth.at
User-Agent: Mozilla/5.0 (Windows;en-GB; rv:1.8.0.11) Gecko/20070312
Firefox/1.5.0.11
Accept: text/xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-gb,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://www.explainth.at/en/misc/httpreq.shtml
Content-Type: application/x-www-form-urlencoded
Content-Length: 39
name=MyName&married=not+single&male=yes
```

Este último caso podría ser el resultado del envío de un formulario del cliente al servidor. Esta petición tipo post sería completamente equivalente a la anterior petición get en lo que a envío de datos de un formulario al servidor se refiere. En este caso, el contenido del mensaje es sólo la última línea. En el formulario había tres campos, uno con nombre "name", otro con nombre "married", y el último con nombre "male". En la respuesta, cada uno de los tres campos está separado por el símbolo "&". Cada campo va seguido del valor que el usuario ha introducido en el formulario para dicho campo, y separando el nombre del campo y su valor va el signo "=".

Mensajes HTTP Response (respuesta)

Una respuesta del servidor en el protocolo http sigue la siguiente estructura:

```
Versión-http SP código-estado SP frase-explicación CRLF
(nombre-cabecera: valor-cabecera ("," valor-cabecera)* CRLF)*
Cuerpo del mensaje
```

El **código de estado** es un código que indica si la petición ha tenido éxito o habido algún error con ella. La frase de explicación suele proporcionar alguna explicación del error.

Las cabeceras siguen la misma estructura que en las peticiones. Después de las peticiones, podemos encontrar la respuesta del servidor. Un ejemplo de una respuesta del servidor podría ser:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
<html>
```

```
<head> <title> Título de nuestra primera página </title> </head>
<body>
¡Hola mundo!
</body>
</html>
```

Los códigos de estado del protocolo http son números de tres dígitos que forman parte de las respuestas http. Estos códigos explican qué ha sucedido al intentar llevar a cabo una petición. Estos códigos son:

- Códigos 1xx : Mensajes
  - 100-111 Conexión rechazada
- Códigos 2xx: Operación realizada con éxito
  - 200 OK
  - 201-203 Información no oficial
  - 204 Sin Contenido
  - 205 Contenido para recargar
  - 206 Contenido parcial
- Códigos 3xx: Redireción
  - 301 Mudado permanentemente
  - 302 Encontrado
  - 303 Vea otros
  - 304 No modificado
  - 305 Utilice un proxy
  - 307 Redirección temporal
- Códigos 4xx: Error por parte del cliente
  - 400 Solicitud incorrecta
  - 402 Pago requerido
  - 403 Prohibido
  - 404 No encontrado
  - 409 Conflicto
  - 410 Ya no disponible
  - 412 Falló precondición
- Códigos 5xx: Error del servidor
  - 500 Error interno
  - 501 No implementado
  - 502 Pasarela incorrecta
  - 503 Servicio nodisponible
  - 504 Tiempo de espera de la pasarela agotado
  - 505 Versión de HTTP no soportada

3. Aplicaciones web

En la ingeniería de software se denomina aplicación web a aquellas 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, es una aplicación software

que se codifica en un lenguaje soportado por los navegadores web en la que se confía la ejecución al navegador.

Las aplicaciones web son populares debido a lo práctico del navegador web como cliente ligero, a la independencia del sistema operativo, así como a la facilidad para actualizar y mantener aplicaciones web sin distribuir e instalar software a miles de usuarios potenciales.

¿Por qué las aplicaciones web han tomado tanta relevancia?

La esencia del concepto es no dejar que el cliente realice demasiadas tareas, sino solo lo necesario para que lleve a cabo su trabajo y dejar que en el lado del servidor se realicen las operaciones importantes: almacenamiento de datos, transacciones, reglas del negocio y la lógica del programa.

El concepto de aplicación web ha tomado una mayor relevancia con el auge de las redes locales y la popularidad de las Intranets e Internet, ofreciendo la oportunidad de acceso a dichas aplicaciones a través de computadores y otros dispositivos móviles. Internet ha elevado y extendido aún más el concepto de aplicación web para servir a usuarios ubicados en cualquier sitio donde se tenga acceso a Internet.

VENTAJAS DEL SOFTWARE WEB

Si bien, aunque los puntos que se exponen a continuación dependerán del tipo de aplicación y de la solución concreta sobre para la que se haga el software, a grandes rasgos éstas son las ventajas que presentan las aplicaciones web con respecto a las de escritorio:

- **No requiere instalar software especial en los clientes**. En esencia, para acceder a un software web solo necesitamos disponer de un navegador de páginas web (Internet Explorer, Firefox, Opera, Chrome, 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 negocio se lleva en el lado del servidor.
- **Bajo coste en actualizar los equipos con una nueva versión**. Los navegadores web visualizan las páginas web que son servidas por el servidor web dinámicamente. En ese

sentido, es el servidor quien ejecuta la mayor parte del código de la aplicación y suministra de forma centralizada las vistas (las páginas) a los navegadores conectados. En consecuencia, no hay que instalar nada en los puestos de trabajo, ya que la actualización se realiza en el servidor y automáticamente la ven todos los usuarios.

- Acceso a la última versión. Como consecuencia del punto anterior, se evita que pueda existir algún equipo que ejecute una versión diferente y desactualizada. Si en los distintos ordenadores hay diferentes 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 negocio 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.

- Seguridad y copias de seguridad. Como consecuencia del punto anterior, al disponer de los datos centralizados, es más fácil establecer y llevar el control de una política de copias de seguridad centralizada. Es más, al no ubicarse los datos en el puesto de trabajo, en caso de fallo en los ordenadores que no sean el servidor, la empresa no ha perdido información y puede desplegar rápidamente un nuevo puesto de trabajo (PC con un navegador web).

- Movilidad. Este es un concepto relativo y dependiente de la implantación concreta. Si el software está ubicado en un servidor web en Internet o bien disponemos de una intranet externalizada (extranet), cualquier usuario con un portátil y una conexión a Internet móvil podría acceder a la aplicación.

- Reducción de costes en los puestos cliente (**mayor longevidad**). Debido a que las páginas se ofrecen desde el servidor web (donde se suelen ejecutar la mayoría de los procesos y la lógica de negocio), 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.

- **Multiplataforma.** Una ventaja significativa es que las aplicaciones web deberían funcionar igual independientemente de la versión del sistema operativo instalado en el cliente.

En vez de crear clientes para Windows, Mac OS X, GNU/Linux y otros sistemas operativos, la aplicación web se escribe una vez y se ejecuta igual en todas partes.

- **No requieren instalación en el cliente**, con la reducción de tiempo y la reducción de espacio en el disco duro que esto conlleva.

- Portables. Es independiente del ordenador donde se utilice (un PC de sobremesa, un portátil) porque se accede a través de una página web (sólo es necesario disponer de acceso a Internet).

La reciente tendencia al acceso a las aplicaciones web a través de teléfonos móviles requiere sin embargo un diseño específico de los ficheros CSS para no dificultar el acceso de estos usuarios.

Sin embargo no todo son ventajas. Debemos recordar que en el mundo real de los requisitos y restricciones no existe la solución perfecta, si no la más o menos adecuada al caso planteado. Por ello, una solución web también tiene sus inconvenientes, unos derivados del modelo web y otros como consecuencia de cómo se implante.

INCONVENIENTES DEL SOFTWARE WEB

- Habitualmente ofrecen menos funcionalidades que las aplicaciones de escritorio. Se debe a que las funcionalidades que se pueden realizar desde un navegador son más limitadas que las que se pueden realizar desde el sistema operativo. Pero cada vez los navegadores están más preparados para mejorar en este aspecto.

La aparición de HTML 5 representa un hito en este sentido.

- La disponibilidad depende de un tercero, el proveedor de la conexión a Internet o el que provee el enlace entre el servidor de la aplicación y el cliente.

- Si falla alguno de los servidores (web, de aplicaciones o de base de datos) no se podrá trabajar en ninguno de los clientes.

- Inconsistentes escritas con HTML, CSS, DOM y otras especificaciones estándar entre los distintos navegadores que pueden causar problemas en el desarrollo y soporte de estas aplicaciones, principalmente debido a la falta de adhesión de los navegadores a dichos estándares web (ahora bastante menos que antes).

- Posibilidad de los usuarios de personalizar muchas de las características de la interfaz (tamaño y color de fuentes, tipos de fuentes, inhabilitar Javascript) puede interferir con la consistencia de la aplicación web.

4. Tecnologías asociadas a las aplicaciones web.

¿Qué se lleva actualmente? → https://w3techs.com/ y http://gs.statcounter.com/

- En el cliente:
  - Navegadores: Safari (mac), Chrome, Firefox, Opera, IE
  - Lenguajes de programación: En una página HTML se pueden incrustar elementos computacionales y scripts. Javascript (librerías como JQuery, Bootstrap…) Flash, applets java, Ajax
- En el servidor:
  - Servidores: Procesan mensajes HTTP de clientes y devuelven mensajes con la información solicitad. Apache, IIS, nginx…
  - Lenguajes de programación: CGI, JEE, PHP, ASP,
  - Gestores de contenidos: Frameworks para la creación y administración de contenidos de sitios web Permiten la edición de los contenidos por varias personas con distintos roles (administrador, editor, participante, etc.). Facilita el control de los contenidos en un sitio colaborativo. Blogs, foros, wikis, plataformas e-learning…Joomla,Drupal, Wordpress…
- Estándares en la web. Desde 1995 el W3C (World Wide Web Consortium) define los estándares de la Web
  - Protocolos y Lenguajes. XML y derivados (css, xsl, xquery, xschema…), HTML, JSON
  - Accesibilidad

¿Qué lenguajes son de servidor y cuáles de cliente?

5. Diseño lógico de Arquitecturas Web. Modelos.

Una aplicación web necesita de una estructura que permita su acceso desde diferentes lugares (máquinas). La gran mayoría de las arquitecturas web en la actualidad se basan en un modelo **cliente/servidor**: una comunicación en la que uno de los extremos ofrece uno o más servicios y el otro hace uso de él.

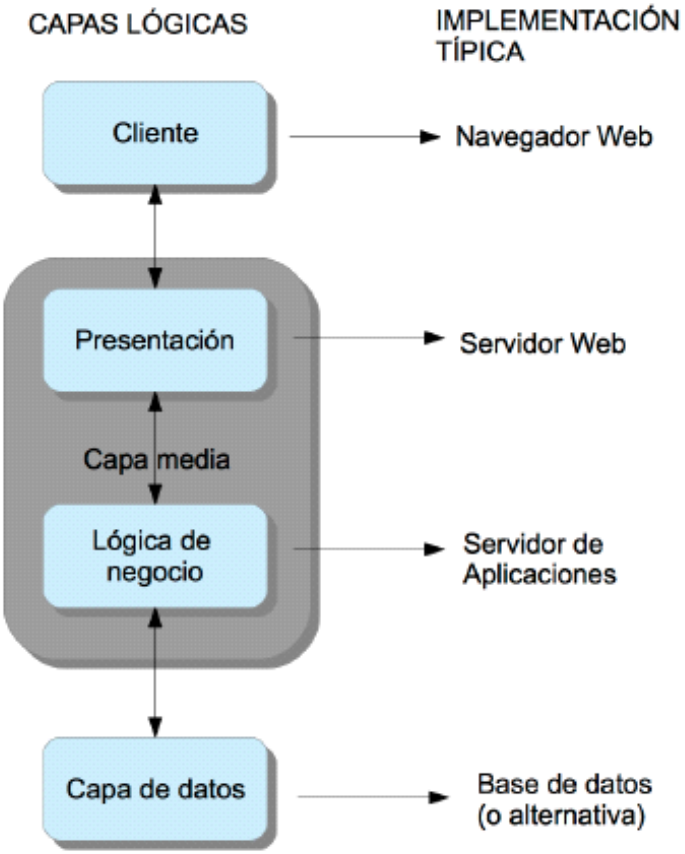
La estructura de una Arquitectura Web, normalmente, sigue el **modelo de capas**. El caso más común es el modelo de 3 capas que se explica a continuación.

1. Capa cliente.

Es generalmente el navegador Web ejecutándose en la máquina del usuario final. Interpreta las peticiones del usuario y presenta los resultados al usuario.

2. Capa de negocio.

Consta de un servidor Web y de Aplicaciones.



En esta capa es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso.

Aquí la lógica puede ser mucho más compleja que en la capa anterior. Existen muchas tecnologías que pueden usarse en este nivel, PHP, ASP, JSP, PERL, etc.

La capa de negocio contiene la lógica principal de procesamiento de datos dentro de nuestra aplicación Web. Se comunica con la capa de cliente, 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.

Como se comenta en el párrafo anterior, la capa de negocio puede estar a su vez dividida en dos: capa de presentación y una capa de lógica de negocio.

- La **capa de presentación** es la encargada de la navegabilidad, validación de los datos de entrada, formateo de los datos de salida, presentación de la web, etc. integrando la parte dinámica en la estática. Además también procesa las páginas que envía el cliente (por ejemplo datos en formularios). Algunas soluciones para esta subcapa son los ASP de Microsoft o los JSP de Java. Esta parte la realiza generalmente un servidor web. Se trata de la capa que se presenta al usuario.

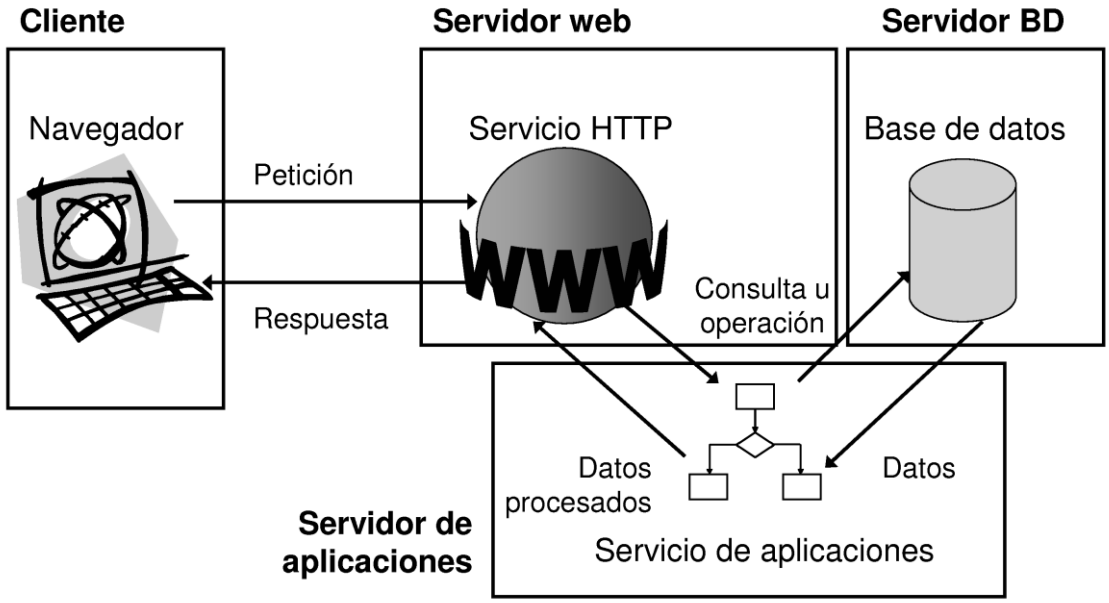
- La **capa de lógica de negocio** lleva a cabo operaciones más complejas. Se corresponde con la implantación de un servidor de aplicaciones. Realiza muchos tipos de operaciones, como por ejemplo realizar todas las operaciones, gestionar el flujo de trabajo y gestión de las operaciones de accesos a datos desde la capa de presentación.

3. **Capa de datos.**

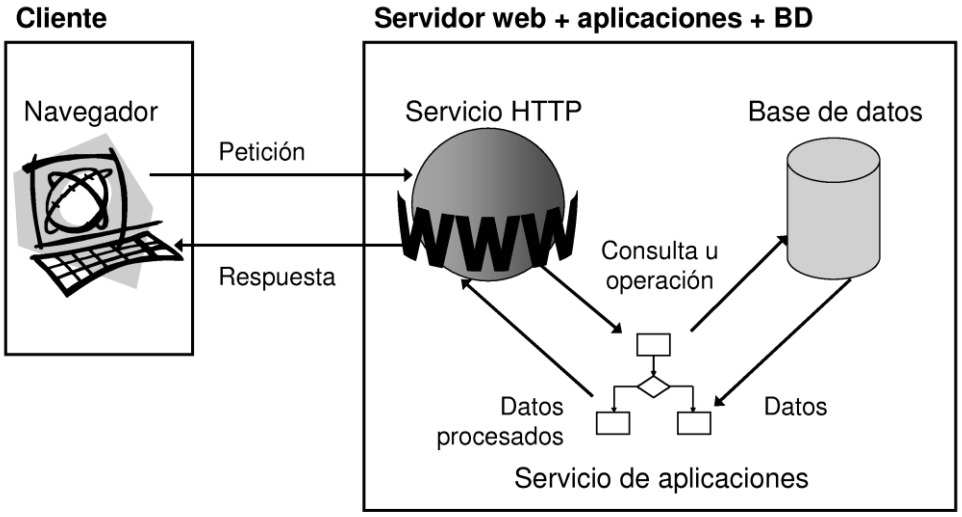
Hace referencia a los servidores de datos, es decir al sistema de almacenamiento y acceso a datos que se utilizan para confeccionar la página Web con dichos datos. Generalmente es un gestor de bases de datos (MySQL, PostgreSQL, Oracle…) pero pueden ser ficheros de texto plano, ficheros XML, etc.

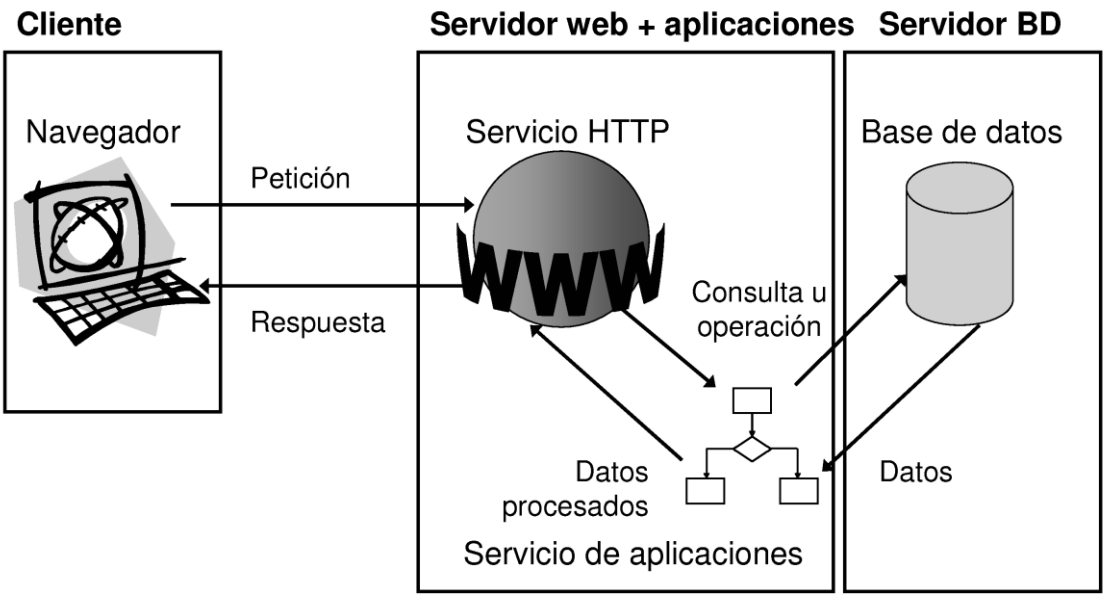
Una opción cada vez más usada es la creación de ficheros XML a partir de datos almacenados en una base de datos.

Un ejemplo del modelo completo estaría compuesto por un navegador Chrome con un servidor Apache y un Tomcat que se conecte a un servidor con una base de datos Oracle.



¿Qué crees que es mejor, separar las distintas funcionalidades en distintos servidores o utilizar un único servidor que incluya estas funcionalidades?





¿Cuál crees que es el objetivo de separar las distintas funcionalidades en distintos servidores?

El objetivo de separar las distintas funcionalidades (servicio de HTTP, lógica de negocio y lógica de datos) en distintos servidores es aumentar la **escalabilidad** del sistema de cara a obtener un mayor rendimiento.

Al separar las distintas funcionales en distintos servidores, cada uno de ellos se puede configurar (dimensionar) de forma adecuada a los requisitos que presenta cada uno de ellos. Por ejemplo, para ofrecer el servicio de HTTP hace falta un ordenador con una buena conexión a Internet, rápido pero sin grandes necesidades de almacenamiento. Sin embargo, para el servidor de bases de datos hace falta un ordenador con mucha memoria y con un disco duro de alta capacidad de almacenamiento y rápido para mantener todos los datos.

6. Plataformas web libres y propietarias.

Una plataforma web es el entorno de desarrollo de software empleado para diseñar y ejecutar un sitio web. En términos generales, una plataforma web consta de cuatro componentes básicos:

1. El **sistema operativo,** bajo el cual opera el equipo donde se hospedan las páginas web.
2. El **servidor web y servidor de aplicaciones,** es el software que maneja las peticiones desde equipos remotos a través de Internet. En el caso de páginas estáticas, el servidor web simplemente provee el archivo solicitado, el cual se muestra en el navegador. En el caso de sitios dinámicos, el servidor web se encarga de pasar las solicitudes a otros programas que puedan gestionarlas adecuadamente.
3. El **sistema gestor de bases de datos** se encarga de almacenar sistemáticamente un conjunto de registros de datos relacionados para ser usados posteriormente.
4. Un **lenguaje de programación** que controla las aplicaciones de software que corren en el sitio web.

Diferentes combinaciones de los cuatro componentes señalados, basadas en las distintas opciones de software disponibles en el mercado, dan lugar a numerosas plataformas web o más comúnmente denominadas "**stack**", aunque, sin duda, hay dos que sobresalen del resto por su popularidad y difusión: LAMP y WISA.

LAMP

Surge de las iniciales de los componentes de software que la integran:

Linux: Sistema operativo.

Apache: Servidor web.

MySQL: Gestor de bases de datos.

PHP: Lenguaje interpretado PHP, aunque a veces se sustituye por Perl o Python.

WISA

Está basada en tecnologías desarrolladas por la compañía Microsoft; se trata, por lo tanto, de **software propietario.** La componen los siguientes elementos:

Windows: Sistema operativo.

Internet Information Services: servidor web.

SQL Server: gestor de bases de datos.

ASP o ASP.NET: como lenguaje para scripting del lado del servidor.

Existen muchas otras plataformas que trabajan con distintos sistemas operativos (Unix, MacOS, Solaris), servidores web (incluyendo algunos que se han cobrado relativa popularidad como Lighttpd y LiteSpeed), bases de datos (Postgre SQL) y lenguajes de programación.

WAMP la configuración Windows-Apache-MySQL-PHP es bastante común pero sólo como plataforma de desarrollo local.

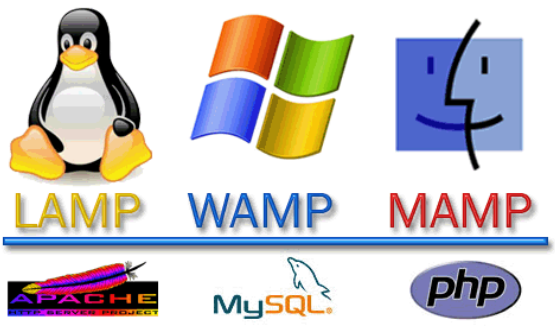
XAMP : Es multiplataforma, por lo que funciona en sistemas Windows, Linux, Mac e incluso hasta Solaris.

MAMP: análogo para el sistema operativo de Apple (*Mac + Apache + MySQL + PHP*).

MEAN (MongoDB + ExpressJS + AngularJS + NodeJS), un sistema que utiliza tecnologías que se han puesto muy de moda últimamente, como bases de datos MongoDB y el entorno de programación NodeJS, junto a frameworks como ExpressJS y AngularJS, basados en JavaScript. Es multiplataforma, y sólo requiere instalar previamente NodeJS y MongoDB para funcionar y comenzar a desarrollar.

WPN-XM Server Otra solución muy interesante es WPN-XM (Windows + PHP + Nginx + XDebug + MariaDB), un servidor que implementa sistemas aún más libres, reemplazando MySQL por MariaDB, o el servidor web Apache por el ligerísimo Nginx.

NMP Server (*Nginx + MySQL + PHP*)



7. ¿Qué se necesita para montar un servidor Web y de Aplicaciones?

Lo primero que se necesita es una máquina con una potencia capaz de atender las peticiones que vaya a procesar. Es muy recomendable quesea una máquina dedicada o que cumpla otras funciones relacionadas con intercambio de información en internet como gestionar correo electrónico o **FTP**.

También es vital que el **sistema operativo** que elijamos sea estable.
Es conveniente que lleve cierta seguridad y control de permisos integrado. Los sistemas más habituales son diferentes versiones de UNIX (por ejemplo Solaris) pero las diferentes distribuciones de Linux están tomando una posición fuerte por su bajo (o inexistente) coste. Windows también es una buena opción en sus versiones servidor, pero es más caro que Linux.

Tendremos que disponer de una **dirección IP estática**. Por supuesto debe ser una dirección de internet a no ser que se esté montando una intranet. Nuestra máquina debe ser accesible desde redes remotas.

Los nombres y direcciones de internet que conocemos se basan en un sistema llamado **DNS** que lo que hace es convertir esas direcciones legibles para nosotros en direcciones IP y viceversa. Si nuestra dirección IP cambia frecuentemente cuando alguien fuera a acceder a nuestra página esta le aparecería como no disponible a pesar de que todo el resto del sistema estuviera trabajando.

Por el mero hecho de estar expuestos al exterior, habrá que configurar nuestra máquina para que sea accesible pero impidiendo la conexión de desconocidos a partes del sistema críticas o que no queremos publicar. En definitiva, hay que mejorar la seguridad.

La **conexión a internet 24 horas** se da por garantizada, pero también implica ciertos problemas como la adquisición de dispositivos de red que aguanten ese horario sin sobrecalentarse o saturarse.

8. Alternativas – Webhosting

Realmente el término Web Hosting incluye el tener un servidor propio, pero en la actualidad se utiliza para denominar el alquilar espacio en un servidor de otra compañía. Generalmente esta compañía está dedicada a ello específicamente. Las ventajas de este caso son las obvias: no tenemos que preocuparnos de adquirir ni mantener ni el hardware ni el software necesario. Además la fiabilidad del servicio de una empresa especializada suele ser muy alta.

→*www.000webhost.com/*

→ *www.1and1.es*

…

Ejercicio → Hacer una comparativa de precio, ancho de banda, cuantos sitios web, cuantas bbdd y cuanto espacio para bbdd, lo mismo para cuentas de correo, cuentas FTP, espacio de disco entre distintos webhostings.

Escalabilidad.

En el entorno en que se ubican las aplicaciones web, uno de los principales factores que puede afectar al rendimiento de las mismas es el número de usuarios, ya que éste puede verse incrementado de forma vertiginosa en un periodo de tiempo relativamente corto.

El éxito o el fracaso de un sitio web orientado al usuario común vendrá determinado, entre otros aspectos, por el dimensionamiento del sistema sobre el que se instala y soporta el software que sustenta dicho sitio. En consecuencia, uno de los requisitos fundamentales de una aplicación web es que sea completamente escalable sin que un

aumento de los recursos dedicados a la misma suponga modificación alguna en su comportamiento o capacidades.

La escalabilidad de un sistema web puede ser:

- Verticalmente: Una máquina por cada capa del sistema, e incluso si esto no fuera suficiente, distribuyendo los elementos de una misma capa entre distintas máquinas servidoras.
- Horizontalmente: consiste en aumentar el número de nodos es decir se trata de clonar el sistema en otra máquina de características similares y balancear la carga de trabajo.
- Cluster: consiste en crear agrupaciones de servidores de forma que su carga de trabajo vaya a ser distribuida entre la granja de servidores que forman el cluster, de modo transparente al usuario y al administrador.