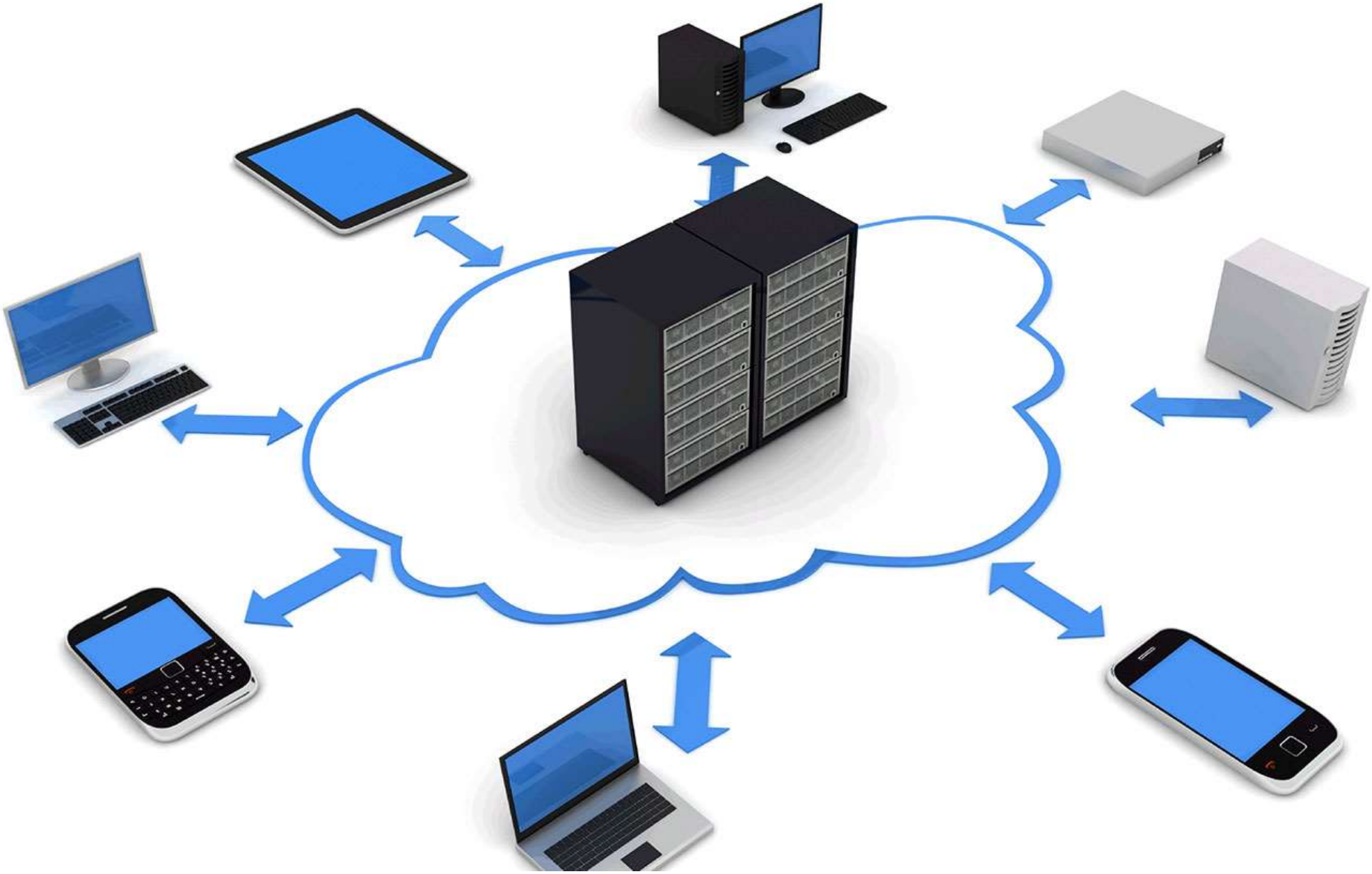
Arquitecturas y herramientas de programación



Desarrollo Web en Entorno Servidor

Contenido

1. Introducción.
2. El concepto de web y su evolución.
3. Elementos de la arquitectura cliente-servidor.
4. Aplicaciones web y aplicaciones de escritorio.
5. Funcionamiento de una aplicación web.
6. Creación de aplicaciones web.
7. Tecnologías para crear aplicaciones web.
8. Preparación del sistema operativo del servidor.
9. Servidor web: instalación y configuración.
10. Módulos y componentes necesarios.
11. Sistema gestor de base de datos: instalación y configuración.
12. Soluciones integradas.

1. Introducción

Internet forma parte de nuestras vidas. Todos utilizamos la Red para nuestras tareas cotidianas, ocio o trabajo, en gran medida a través de la Web.

Desde su nacimiento, al final de los años 80, la evolución de la Web ha sido vertiginosa. Hemos pasado de una Web con contenidos estáticos, a una Web con todo tipo de servicios en la que se comparten también imágenes y vídeo. Actualmente, se utilizan cada vez más las **aplicaciones web**, que ofrecen funcionalidades similares a las aplicaciones de escritorio, sobre todo en áreas como la **ofimática web**.

Por otra parte, el auge de los dispositivos móviles ha propiciado un gran cambio en la forma de crear aplicaciones web, siendo necesaria la adaptabilidad de las páginas web (**Responsive Web**).

Tecnológicamente, en la creación de las aplicaciones se imponen los **patrones MVC** y lenguajes como: **Ruby, Python, JavaScript**…, **HTML5** ha eliminado a Flash.

Otra revolución llega con el **Internet de las Cosas - IoT**, que permite que los dispositivos de la vida diaria (electrodomésticos, coches,…) se conecten a Internet para ampliar sus funciones. Ello hace que se envíen enormes cantidades de información a Internet, con lo que cobran importancia las tecnologías **Big Data**.

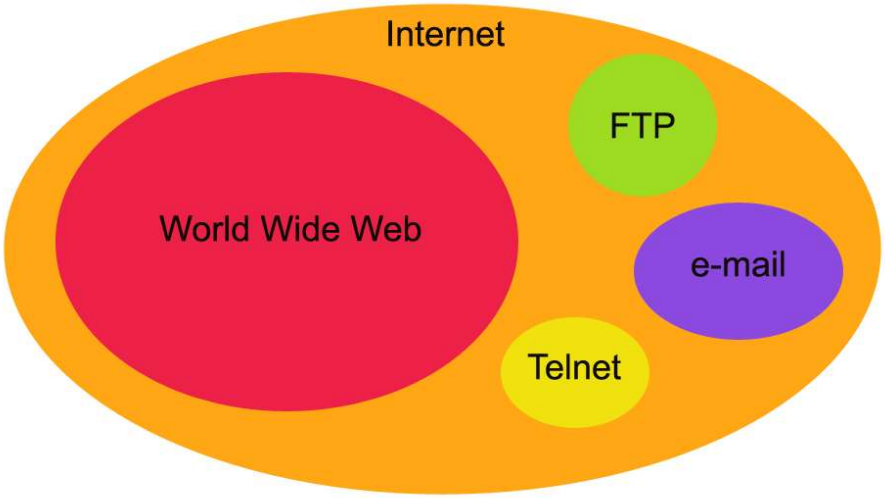
La **virtualización** y la computación en la nube (**cloud computing**) hace que el hardware deje de ser importante para delegar sus funciones a servidores. Aplicaciones que antes se instalaban en el equipo del usuario ahora se accede a ellas como un servicio a través de la Web.

2. El concepto de web y su evolución

- **El concepto de web**

Internet es la red a través de la cual los dispositivos pueden compartir información, interconectando máquinas en todo el mundo a través del protocolo TCP/IP.

La **Web** o **World Wide Web** es una manera de acceder a la información a través de Internet, mediante el protocolo HTTP (protocolo de transferencia de hipertexto). Es una parte de Internet.



El **World Wide Web Consortium** (**W3C**) es una comunidad internacional que desarrolla los estándares que aseguran el crecimiento de la Web a largo plazo.(www.w3c.es)

2. El concepto de web y su evolución

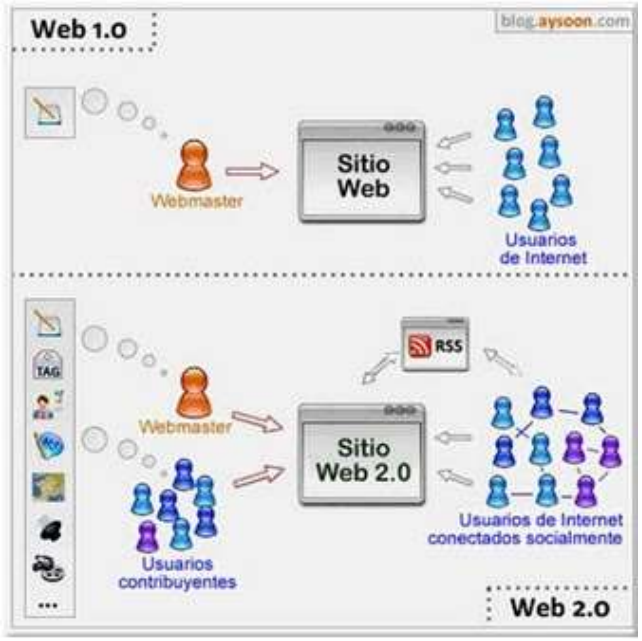
- **Evolución de la web**

Web 1.0

En las etapas iniciales la Web tenía una finalidad principalmente divulgativa y las páginas eran estáticas, por lo que su contenido permanecía invariable.

Web 2.0

En 2004 se emplea este término. También se conoce como Web Social, ya que el usuario cobra un papel relevante, puede crear y compartir información. Aparecen redes sociales, blogs, wikis,… Las páginas web son dinámicas e interactivas.



2. El concepto de web y su evolución

Todas las páginas web que se cataloguen como Web 2.0 reúnen los siguientes requisitos:

- **Aplicaciones ricas en Internet** (RIA, Rich Internet Applications). Son aquellas aplicaciones web que ofrecen resultados que las hacen casi indistinguibles de las aplicaciones de escritorio. Ej: Office 365.
- **Arquitectura orientada al servicio** (SOA, Service Oriented Architecture). Conjunto de tecnologías y técnicas que permiten diseñar aplicaciones como un conjunto de servicios que resuelven peticiones de los usuarios. Así, se pueden crear pequeños elementos software reutilizables de forma independiente al lenguaje con el que fueron creados.
- **Web social**. En las aplicaciones Web 2.0 el usuario pasa a ser el centro de estas. El usuario se conecta con otros usuarios a través de la aplicación y participa generando contenidos. Ej: Twitter, Facebook, Instagram.

Web 3.0

En la actualidad se utiliza el término Web 3.0 o **Web semántica**. El objetivo principal es acercar al usuario al lenguaje natural y adaptar la navegación a sus preferencias. Surgen nuevas formas de búsqueda y almacenamiento. La publicidad selectiva es un ejemplo.

Web 4.0

Sería la tendencia futura. El usuario no solo interactuará con la web en búsqueda de información, sino que esta será capaz de mostrar soluciones completas a sus necesidades.

3. Elementos de la arquitectura cliente-servidor

La arquitectura cliente-servidor es un modelo de aplicación distribuida con dos componentes principales: el servidor y el cliente.

El servidor tiene como tarea proporcionar las respuestas a las peticiones de los clientes.

Ventajas de la arquitectura cliente-servidor:

- **Escalabilidad**. Es posible aumentar la capacidad tanto de servidores como de clientes.
- **Disponibilidad**. Al estar distribuidas las funciones y responsabilidades entre varios ordenadores independientes, es posible reemplazar, reparar, actualizar o incluso trasladar un servidor, sin que sus clientes se vean afectados por ese cambio.
- **Control centralizado,** por parte del servidor.

Desventajas:

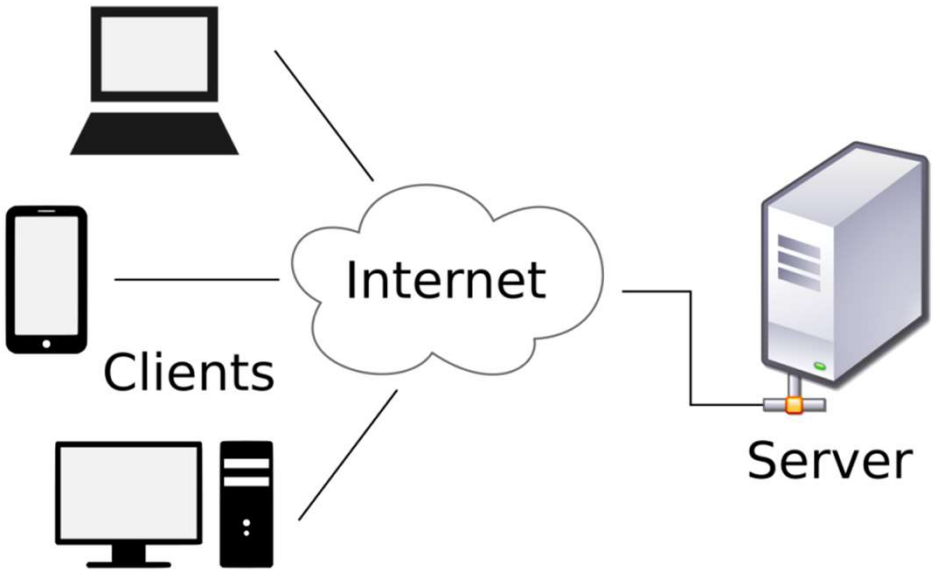
- **Posible congestión del tráfico en la Red**, ya que existen gran cantidad de peticiones simultáneas al mismo servidor. Esto puede ser problemático.
- **Necesidad de contar con potentes recursos hardware en el servidor**, para que sea capaz de procesar de forma óptima las peticiones de los clientes.
- **Mantenimiento complejo**.

3. Elementos de la arquitectura cliente-servidor

Arquitectura de dos niveles

Se refiere a un sistema con dos elementos: **cliente** y **servidor**.

El servidor, polivalente, resuelve las peticiones que realizan los clientes.



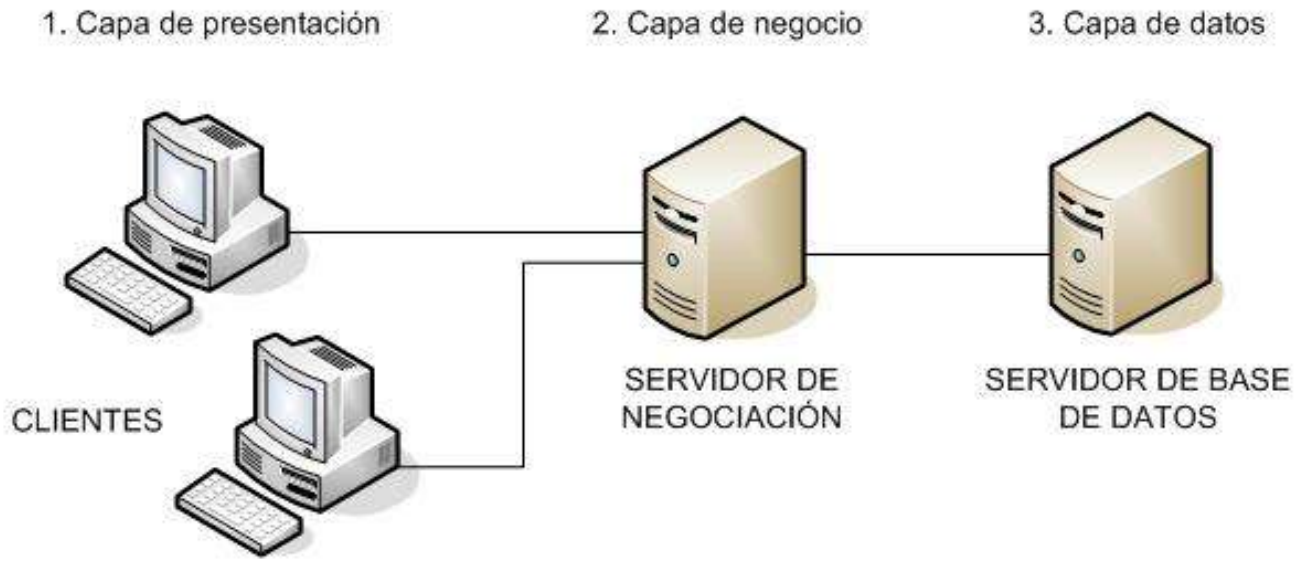
Si hay una gran cantidad de procesos o son muy complejos, este modelo de dos capas no resulta eficaz. Se puede producir congestión en la Red.

3. Elementos de la arquitectura cliente-servidor

Arquitectura de tres niveles

La arquitectura de tres niveles intenta evitar la sobrecarga equilibrando las tareas que tiene que soportar el servidor. Para ello se reparten las funciones en tres niveles:

- **Cliente**: Es el ordenador que, a través de la interfaz de usuario, realiza la petición de recursos al servidor.
- **Servidor de aplicaciones**: El servidor hará uso de otro servidor para poder resolver las peticiones de recursos realizadas por los clientes.
- **Servidor de datos**: el nivel de base de datos permite darle los datos requeridos al servidor de aplicaciones.



4. Aplicaciones web y aplicaciones de escritorio

Aplicaciones de escritorio

Son aquellas aplicaciones que se desarrollan en un sistema operativo específico para ser ejecutadas bajo éste. Son más o menos eficientes dependiendo de la configuración hardware del ordenador donde se instalen. Son las que habitualmente tenemos instaladas en nuestro ordenador personal o de empresa, y pueden ser tanto de pago como libres.

Aplicaciones web

Son todas aquellas aplicaciones accesibles a través de un navegador. Ejemplos son: la Wikipedia, el correo electrónico o un servicio de comercio electrónico.

4. Aplicaciones web y aplicaciones de escritorio

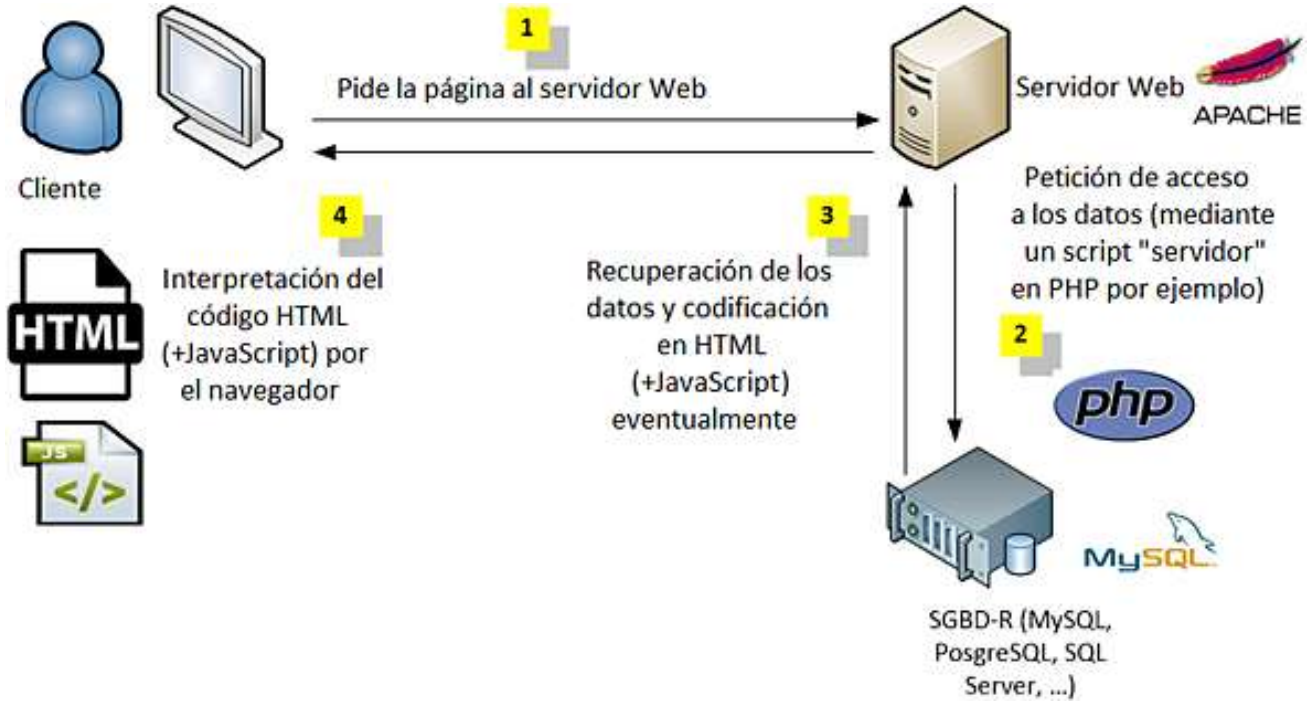
- **Ventajas del software web**:
  - **Ausencia de costes de actualización**: como son los navegadores los que visualizan las páginas web es suficiente con actualizar estos, de manera gratuita.
  - **Datos centralizados**: el servidor no solo aloja las páginas web sino también los datos, estando estos en una base de datos centralizada.
  - **Ausencia de instalación**: no es necesario instalar nada, solo un navegador para acceder al servicio.
  - **Actualización constante**: los equipos siempre acceden a la última versión actualizada de la aplicación.
  - **Movilidad**: acceso desde cualquier ubicación con conexión a Internet.
- **Desventajas del software web**:
  - **Menor potencia**: las aplicaciones de escritorio suelen tener mejores prestaciones y ser más potentes.
  - **Infrautilización del hardware**: las aplicaciones web no supeditan su rendimiento a unas mejores prestaciones del hardware del ordenador.
  - **Conectividad rápida y constante**: es necesario un acceso rápido y fiable a Internet.

5. Funcionamiento de una aplicación web

Existen dos estrategias para la creación de aplicaciones web:

Lenguajes y tecnologías en el lado del cliente: son elementos que se incorporan junto al código HTML de una aplicación web y son interpretados en el navegador del usuario.

Lenguajes y tecnologías en el lado del servidor: son aplicaciones creadas con lenguajes y elementos que se interpretan en el servidor que aloja la aplicación.



5. Funcionamiento de una aplicación web

- **Funcionamiento en el lado del cliente**

En este modo, la página entregada por el servidor web puede contener, además del código **HTML**, elementos pertenecientes a otros lenguajes y tecnologías:

CSS: lenguaje de las hojas de estilos. Permite modificar la apariencia de una página web.

JavaScript: lenguaje sencillo y potente, cuyo código se puede incluir directamente en una página web. Aporta interactividad y un gran número de utilidades a las páginas web.

Flash: componente incrustado mediante una etiqueta HTML que hace referencia a un archivo .swf que es el que contiene el código flash. Requiere de un plugin en el navegador.

Silverlight: componente similar al anterior, propiedad de Microsoft.

Applets de Java: componente incrustado mediante la etiqueta HTML <applet>, con referencia a un archivo Java. Requiere de un plugin JVM en el navegador.

El hecho de requerir plugins instalados, y de procesar aplicaciones creadas en varios lenguajes y tecnologías hace que todo el esfuerzo lo realice el navegador (**tecnología web de cliente pesado**).

Actualmente HTML5 ha desbancado a Flash, y todos los navegadores pueden traducir CSS y JavaScript.

5. Funcionamiento de una aplicación web

5. Funcionamiento de una aplicación web

- **Funcionamiento en el lado del servidor**

Cuando un usuario hace una petición a un recurso web, el servidor se da cuenta que contiene elementos a interpretar en el lado del servidor, y pide al **servidor de aplicaciones** adecuado que traduzca esos elementos antes de enviar el resultado al navegador.

El servidor de aplicaciones, que suele ser un módulo software, enviará el resultado al servidor web en un formato traducible en el lado del cliente, es decir, un documento HTML. Finalmente, el servidor web enviará al cliente este resultado.

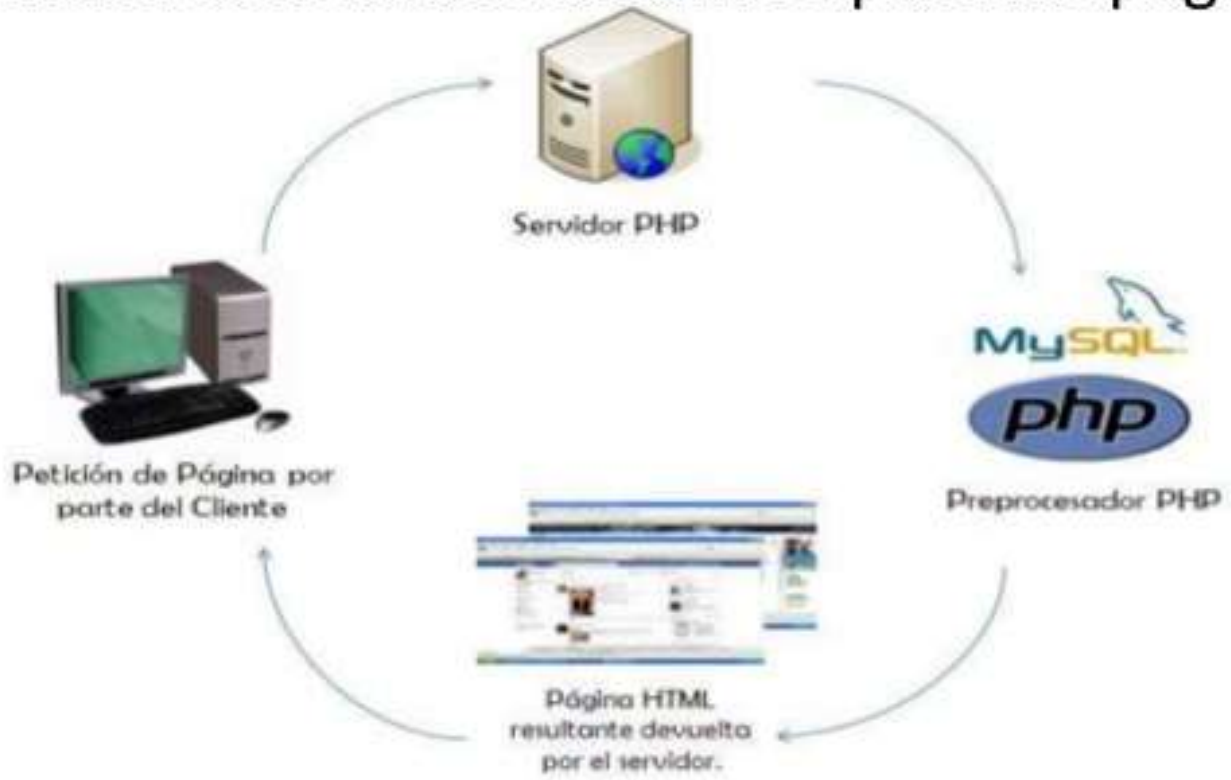
La ventaja de este modo de trabajo es que el navegador puede ser más ligero (**tecnología de cliente ligero**) y la parte pesada se la lleva el servidor web.

Hoy en día, ambos métodos se mezclan, creando aplicaciones que contienen elementos en el lado del servidor (**back-end**) y en el lado del cliente (**front-end)**. Por ejemplo, se pueden crear páginas PHP que incluyen código JavaScript, lo que implica proceso en el lado del servidor (traducir el PHP) y en el del cliente (traducir el JavaScript).

5. Funcionamiento de una aplicación web

Programación del lado del servidor

- Accede a recursos del servidor (base de datos).
- Genera contenidos dinámico para las paginas.



6. Creación de aplicaciones web

- **Servidor web**

Máquina o software capaz de interpretar **peticiones web** realizadas a través del protocolo HTTP o HTTPS y de devolver el resultado, que suele conllevar la entrega de un recurso alojado en el servidor. También se le llama servidor http. Ej: Apache.

La mayoría de peticiones http resultan en la entrega de un documento HTML.

- **Servidor de aplicaciones web**

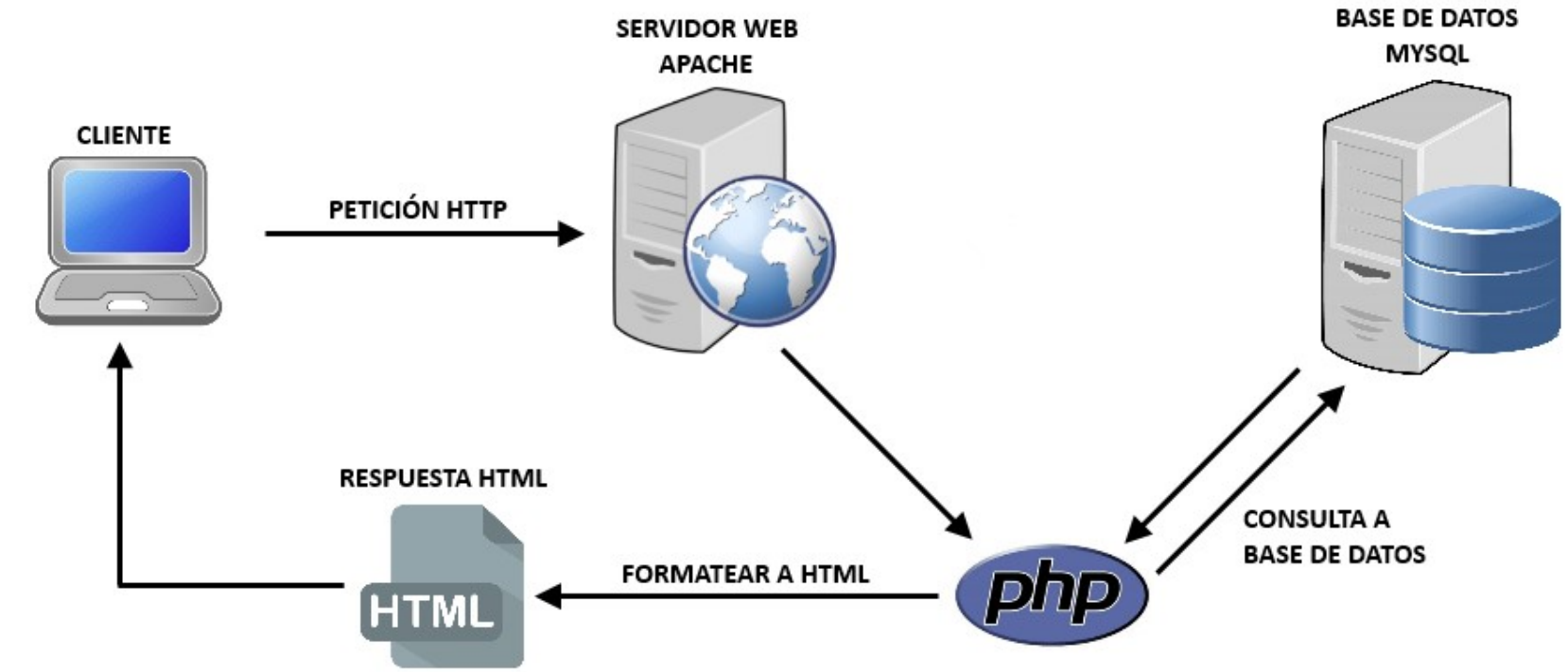
Es el encargado de **traducir instrucciones** hechas en lenguajes del lado del servidor y entregar el resultado de esa traducción al servidor web que pidió dicha traducción. Además, el código del lado del servidor puede implicar el acceso a otros servidores (como acceso a bases de datos).

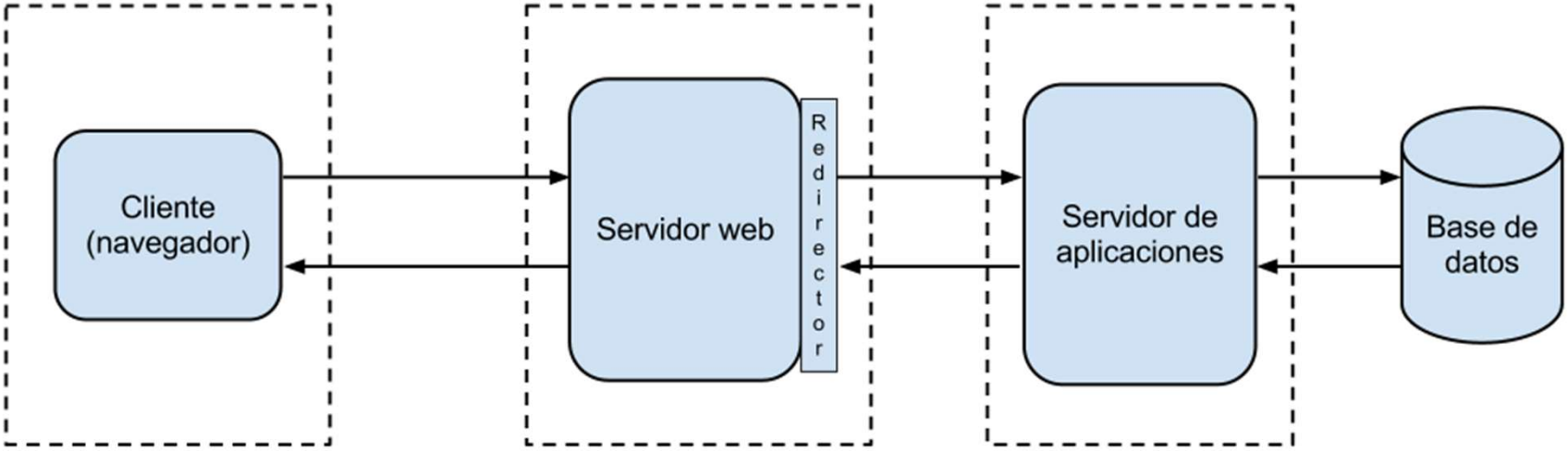
Los servidores de aplicaciones trabajan en conjunto con los servidores web de forma que el proceso es transparente al usuario.

En la práctica, los servidores de aplicaciones son módulos software añadidos al servidor web, de modo que ambos pueden confundirse. Pero no todos los servidores web se comportan como servidores de aplicaciones web. Su labor es diferente: el primero se encarga de las peticiones web y el segundo traduce lenguajes de servidor.

Ej: Servidores PHP, servidores .NET, servidores Java, etc.

6. Creación de aplicaciones web





6. Creación de aplicaciones web

- **Modelo de capas**

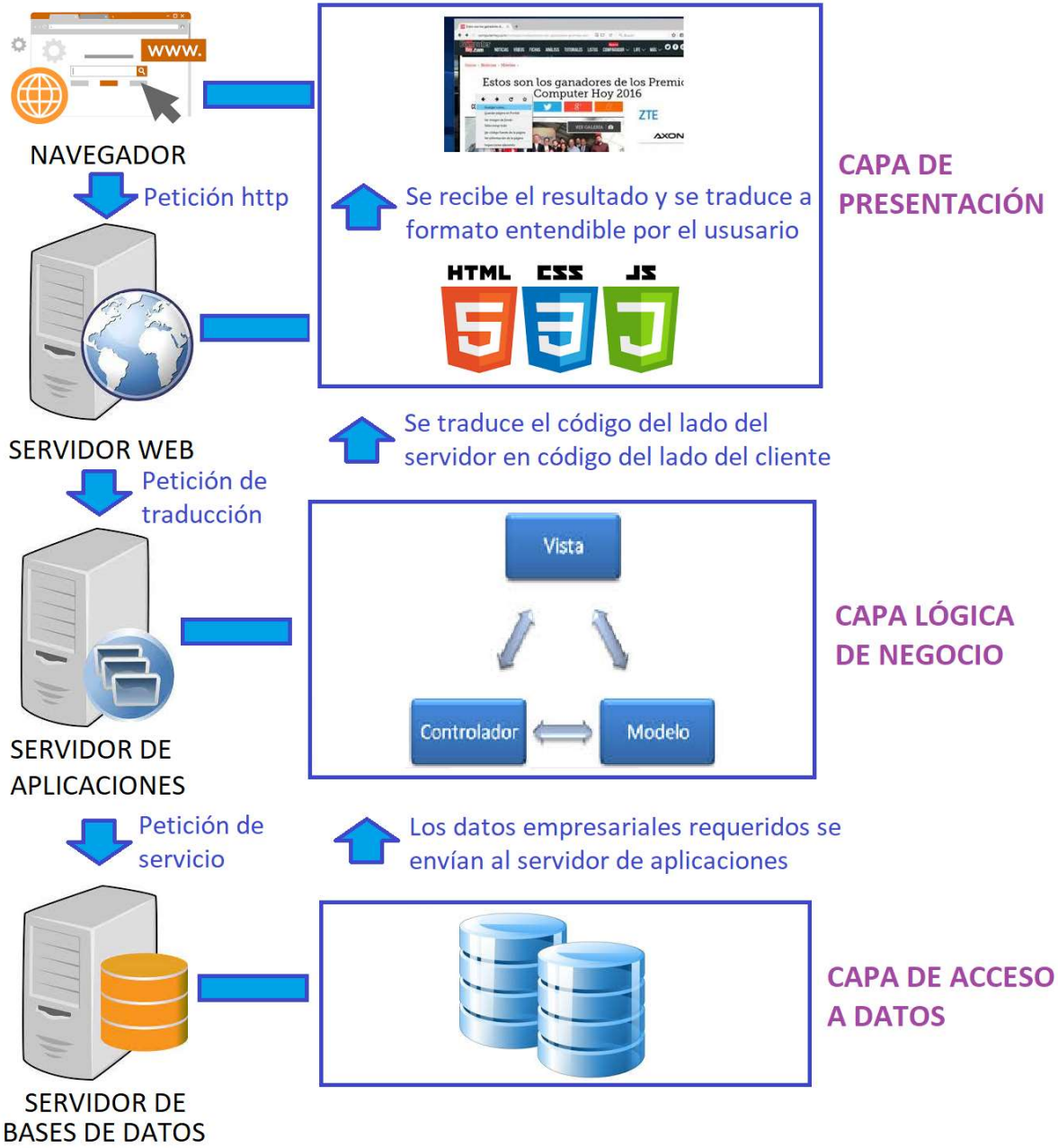
Las aplicaciones web actuales utilizan un modelo basado en tres capas:

Capa de presentación: maneja la parte de la aplicación web que ve el usuario, es decir, presenta la información al usuario. Comprende el código del lado del cliente (HTML, JavaScript, CSS, ..) que le llega al navegador, aunque haya sido generado del lado del servidor.

Capa lógica de negocio: es la encargada de gestionar el funcionamiento de la aplicación. En ella se encuentran los ficheros escritos en lenguaje interpretable del lado del servidor, y cuyo resultado se enviará al servidor web, para que lo envíe al cliente que hizo la petición. Habitualmente, la programación en esta capa se realiza según el **paradigma MVC**.

Capa de acceso a datos: Es la que contiene la información, los datos. En ella se encuentra el sistema gestor de base de datos (SGBD). El servidor de aplicaciones hace peticiones a estos servidores para obtener los recursos requeridos para cumplimentar la petición original. Los servidores de esta capa entregan los recursos solicitados y el servidor de aplicaciones se encargará de situarlos de forma adecuada en el resultado, que viajará hasta el navegador del cliente.

6. Creación de aplicaciones web



6. Creación de aplicaciones web

- **Programación front-end y back-end**

  - **Front-end**: Se refiere a la parte del desarrollo encargado de producir la apariencia final de la aplicación que verá el usuario. En cierto modo es la interfaz del usuario. También forman parte del front-end las personas encargadas del diseño y la programación HTML y CSS de la página, así como del JavaScript, es decir, del funcionamiento de la capa de presentación. Lo que el front-end intenta conseguir es una buena experiencia de usuario.

  - **Back-end**: corresponde a la parte de la aplicación que queda oculta al usuario o a los desarrolladores de esta parte. El funcionamiento de la capa de lógica de negocio y de acceso a datos sería el back-end, donde se usan los lenguajes del lado del servidor y de base de datos. Es la parte de la aplicación encargada de que funcione debidamente la aplicación.

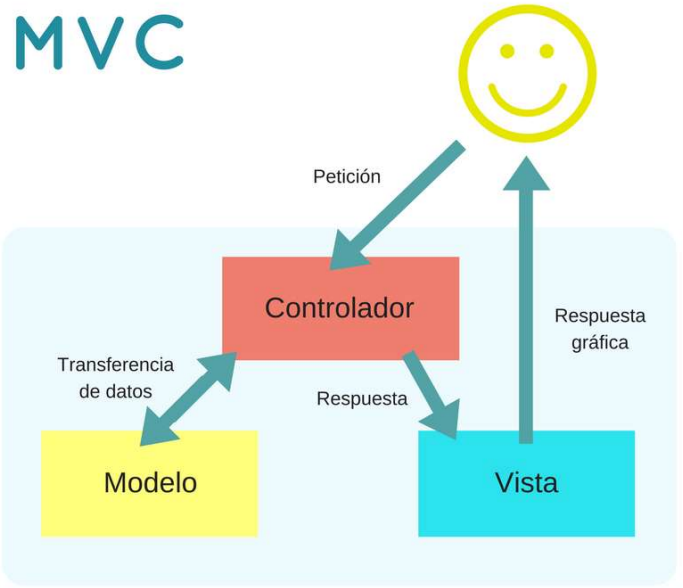
6. Creación de aplicaciones web

- **Paradigma MVC**

La separación de código en aplicaciones web es una cuestión determinante para su mantenimiento y organización de forma sostenible en el futuro. Puesto que la generación del código final reside en la capa lógica, es aquí donde se aplica el paradigma MVC.

Esta técnica separa la programación de la capa lógica en otras tres:

- **Modelo**: contiene el código encargado de recopilar la información procedente de la capa de acceso a datos.
- **Vista**: genera la presentación de cara al usuario. Es la encargada de definir la interfaz de usuario.
- **Controlador**: capa encargada de manejar las peticiones de usuario y de comunicar las capas anteriores para obtener los recursos necesarios para dichas peticiones. Es decir, determina la petición, requiere al modelo los datos necesarios y se los envía a la vista para que genere el resultado final que será enviado al usuario. En definitiva, es un mediador entre el modelo y la vista.



7. Tecnologías para crear aplicaciones web

Se indicarán las principales tecnologías para programar el back-end, el lado del servidor.

- **CGI, Interfaz de Pasarela Común (Common Interface Gateway)**

La idea es crear una aplicación (utilizando cualquier lenguaje de programación) en un servidor de forma que la comunicación entre dicha aplicación y el servidor web se realice mediante la interfaz CGI.

Cuando a un servidor web le llega una petición http que contiene implícita una llamada a una interfaz CGI, el servidor web pide a la aplicación que resuelva dicha petición, que contendrá los parámetros necesarios para su correcta finalización, y le envíe el resultado, que será una respuesta compatible para ser enviada directamente por el servidor web.

- **Lenguajes de programación en el lado del servidor**

Son lenguajes que tienen especial facilidad para crear aplicaciones web, casi siempre ayudados por frameworks. Comparados con los lenguajes de script son más potentes pero más difíciles de aprender.

Perl, **Python** (fácil y potente, muy utilizado actualmente), **Ruby** (fácil y potente), **Java** (ha sido el rey, ahora tiene más competencia), **C#** (ideado por Microsoft), **JavaScript** del lado del servidor(node.js).

7. Tecnologías para crear aplicaciones web

- **Lenguajes de script de servidor**

Se basan en la incrustación de código en el HTML de una página web. La página debe tener una extensión especial, por medio de la cual el servidor web detecta que el archivo requiere de la intervención de un servidor de aplicaciones.

- **PHP (Personal Home Pages)**: Es la tecnología del lado del servidor más utilizada. Basado en C y Perl, es un lenguaje fácil de aprender.
- **ASP (Active Server Pages)**: Tecnología de Microsoft pensada para utilizar en servidores web IIS (Internet Information Server). Actualmente se llama ASP.net y se utiliza en la plataforma .NET de Microsoft.
- **JSP (Java Server Pages)**: Lenguaje de script del lado del servidor basado en Java.
- **ColdFusion**: Propiedad de Adobe, es el más sencillo de todos pero requiere de servidores especiales, y que resultan caros. Está en desuso.

- **Plataformas de desarrollo de servicios web empresariales**

Se trata de soluciones de desarrollo de aplicaciones web completas. Las más populares son:

- **J2EE (Java 2 Enterprise Edition)**: plataforma de creación de aplicaciones web de Java.
- **.NET**: plataforma de Microsoft.

7. Tecnologías para crear aplicaciones web

- **Frameworks MVC**

Se trata de marcos de trabajo que recomiendan una estructura concreta de trabajo que separe los diferentes elementos de la aplicación: no solo el modelo-vista-controlador, sino también los archivos de configuración, elementos estáticos, enrutamiento de peticiones…

Los más populares actualmente son:

- **Ruby on Rails**: muy exitoso por su facilidad y buenos resultados. Basado en Ruby.
- **Apache Struts**: muy famoso para la creación de aplicaciones J2EE.
- **Spring**: otro marco para trabajar en Java J2EE con bastante éxito. También hay versión para aplicaciones .NET.
- **Django**: escrito y pensado para utilizar con Python.
- **Zend**: framework muy popular escrito para PHP.
- **Laravel:** otro popular framework de PHP. Moderno y potente.
- **Angular.js**: framework para JavaScript creado por Google. Muy utilizado para la creación de aplicaciones web SPA (Single Page Applications), que permite que una única página vaya mutando para ir cargando solo los elementos de la misma que se requieren.

8. Preparación del sistema operativo del servidor

La elección e instalación del Sistema Operativo es el primer paso para preparar un entorno local orientado a crear aplicaciones web, bien sea físicamente o en una máquina virtual.

Linux: en los ordenadores personales no es habitual, pero en los servidores web es el sistema operativo dominante. Ello puede ser una razón para elegir Linux. Otra razón puede ser que la mayoría de distribuciones son gratuitas y también es más barato contratar un servidor real en este sistema en Internet. Muchos servidores web (como Apache) se crearon para ser instalados en sistemas Linux, al igual que otras muchas aplicaciones para desarrolladores. Distribuciones Linux son: Debian, Ubuntu, Red Hat Enterprise, Suse Enterprise, CentOS.

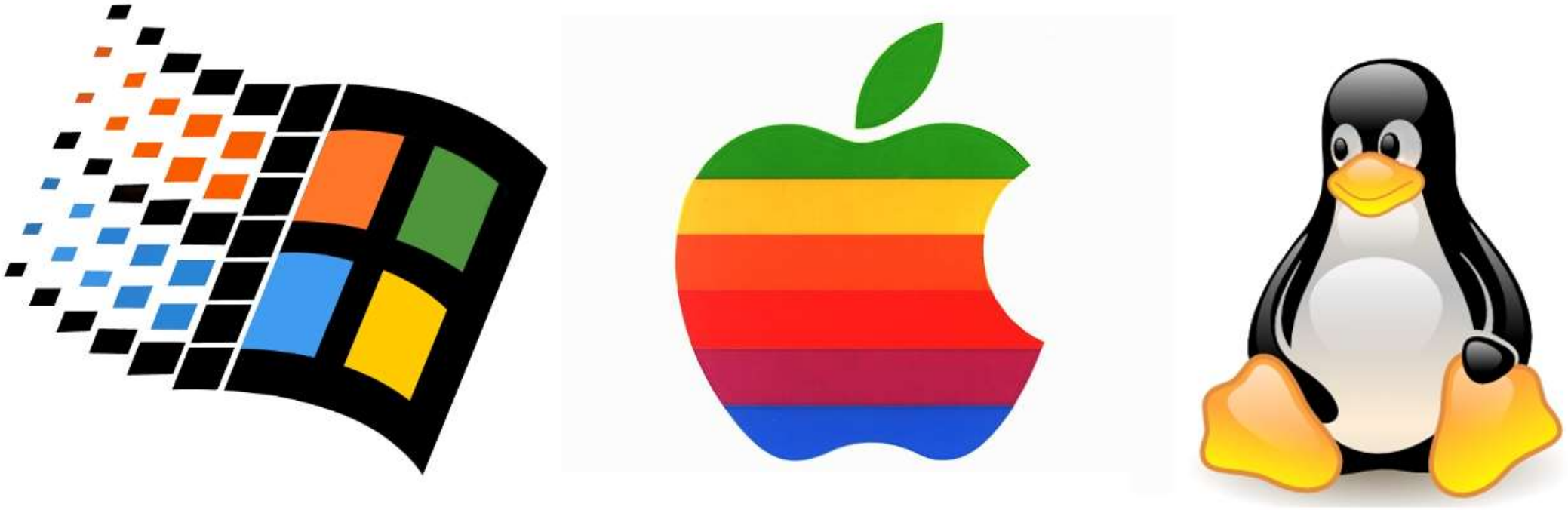
Windows: la mayor parte del software de desarrollo web posee versiones para Windows. Hay servidores web propios de Windows, como IIS de Microsoft. Lo ideal es instalar alguna versión Windows Server, pero se pueden implementar servidores web en cualquier versión. Es posible que al ser nuestro sistema habitual de trabajo, tengamos mayor conocimiento sobre su gestión.

OSX: (sistema operativo de Apple para Macintosh). Se debe disponer de un equipo de la marca. Conlleva un coste más elevado, pero auna lo mejor de Linux y Windows: facilidad de uso y gestión, buen funcionamiento y muchas aplicaciones. Los sistemas Mac están basados en Unix, con lo que la configuración en línea de comandos es parecida a los sistemas Linux.

8. Preparación del sistema operativo del servidor

La elección del sistema operativo también puede estar determinada por el resto de elementos. Con PHP se suele elegir Linux, y con ASP se elige Windows.

Sin embargo, la tendencia actual es que el sistema operativo sea poco determinante gracias a las facilidades de traspaso del entorno de una máquina a otra. Por ello, dicha elección responde más bien a preferencias personales.



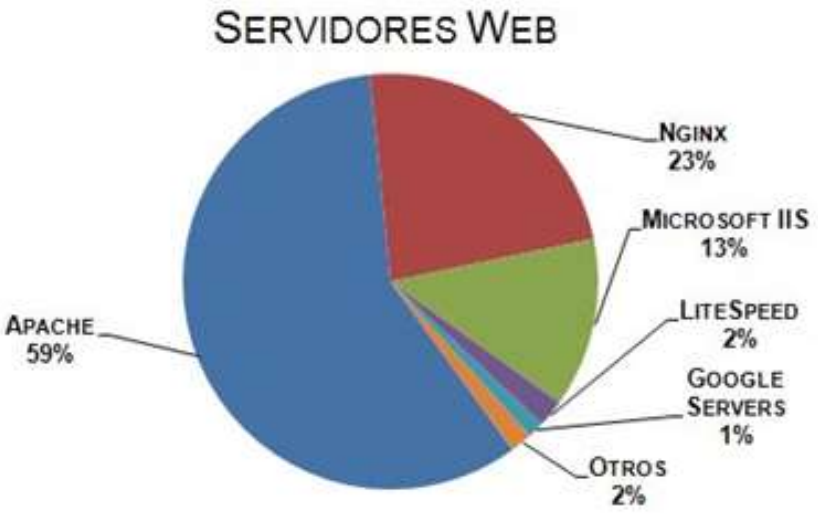
9. Servidor web: instalación y configuración

Para dotar a un ordenador de la capacidad de mostrar páginas web y permitir que otros usuarios accedan a ellas, es necesario instalar un servidor web.

El acceso puede ser a través de una red local o de Internet. Con un servidor web local es posible realizar páginas web y modificarlas sin necesidad de subirlas a Internet.

En el mercado hay varios servidores web disponibles. La elección dependerá de la plataforma sobre la que va a ser instalado el servidor web y del lenguaje de programación que se utilice. Con tecnologías PHP, lo lógico es instalar Apache, si se opta por programar en .NET se utilizará IIS (Internet Information Server).

- **Apache**: servidor web gratuito, multiplataforma, modular y muy robusto. Tiene ciertas limitaciones de velocidad que hacen que en ciertos entornos no sea del todo óptimo.
- **Nginx**: servidor web gratuito, multiplataforma y con grandes prestaciones a la hora de atender multitud de peticiones concurrentes.
- **IIS**: servidor web propietario, orientado a Windows y dirigido a aplicaciones Microsoft.



9. Servidor web: instalación y configuración

Ya que para implementar aplicaciones web utilizaremos PHP, nuestra elección será Apache.

- **Instalación de Apache**

Se puede instalar Apache compilando su código fuente, o descargar código ya compilado y ejecutable. En su página web: https://httpd.apache.org se pueden encontrar las diferentes formas de instalación en función del sistema operativo.

Una vez instalado, es posible realizar diferentes operaciones con el servidor a través de su centro de control: arrancar, parar, reiniciar.

Para saber que está correctamente instalado, escribir en el navegador: http://localhost o http://127.0.0.1

9. Servidor web: instalación y configuración

- **Estructura de directorios de Apache**

Una instalación típica posee los siguientes subdirectorios dentro del directorio de instalación de Apache:

- bin: contiene los archivos ejecutables, los programas para configurar, gestionar, ejecutar o detener el servidor.
- cgi-bin: directorio usado para almacenar programas del lado del servidor.
- conf: contiene los archivos de configuración de Apache.
- error: contiene los archivos con los mensajes de error del servidor.
- htdocs: directorio por defecto para guardar las páginas web que sirve Apache.
- icons: contiene los iconos que usa el servidor para mostrar en algunos de sus mensajes.
- include: archivos de cabecera del código fuente de Apache.
- lib: contiene archivos de librería de Apache.
- logs: archivos de información sobre conexiones y errores acontecidos.
- manual: contiene el manual de Apache.
- modules: módulos y extensiones del servidor.

9. Servidor web: instalación y configuración

- **Iniciar y parar la ejecución del servidor web Apache**

Windows:

Apache se instala como un **servicio** de Windows.

El arranque y parada se puede realizar desde la pantalla de servicios (**services.msc**), o desde el icono **Apache Monitor** (archivo del mismo nombre en el directorio **bin**).

Habiendo configurado correctamente la variable PATH del sistema, también se pueden realizar estas tareas desde el símbolo del sistema (privilegios de administrador) a través del programa que controla el servidor: **httpd**. Se podrá:

- Instalar el servidor como servicio en Windows: **httpd –k install**
- Arrancar el servidor: **httpd –k start** (o: **net start apache2.4**, siendo apache2.4 el nombre del servicio)
- Detener el servidor: **httpd –k stop** (o: **net stop apache2.4**)
- Reiniciar el servidor: **httpd –k restart**
- Desinstalar el servicio: **httpd –k uninstall** (o: **sc delete apache2.4**)

9. Servidor web: instalación y configuración

- **Iniciar y parar la ejecución del servidor web Apache**

Linux:

En Linux no hace falta instalar Apache como servicio.

Los comandos son similares. En algunas distribuciones en lugar de **httpd –k** se utiliza **apachectl** o **apache2** (puede que estos no tengan exactamente las mismas opciones).

- Ejecutar el servidor: **httpd –k start**
- Detener el servidor: **httpd –k stop**
- Reiniciar el servidor: **httpd –k restart**
- Lanzar el servidor: **httpd**

A veces, conviene eliminar el proceso a través del PID, porque con solo detener Apache no basta para realmente pararle.

9. Servidor web: instalación y configuración

- **Iniciar y parar la ejecución del servidor web Apache**

Opciones del comando httpd:

-k start: lanza el servidor Apache.

-k stop: para el servidor Apache.

-k restart: reinicia el servidor Apache.

-D nombre: define un nombre para las directivas <IfDefine name>.

-d directorio: permite indicar un directorio raíz alternativo para Apache.

-f rutaArchivo: permite indicar un archivo alternativo de configuración.

-C “directiva”: procesa la directiva indicada antes de leer la configuración.

-c “directiva”: procesa la directiva indicada después de leer la configuración.

-v: muestra la versión de Apache.

-V: muestra las opciones de compilación.

-h: ayuda para conocer las opciones de httpd.

-l: lista de módulos compilados.

-L: lista de directivas.

-t: ejecuta el analizador de sintaxis para los archivos de configuración de Apache.

-T: igual pero no comprueba la sintaxis.

9. Servidor web: instalación y configuración

- **Configuración del servidor web Apache**

Para modificar el funcionamiento de Apache, se utilizan sus archivos de configuración.

- **httpd.conf**, situado en el directorio **conf** de la instalación de Apache.
- En algunas instalaciones es: **apache2.conf**

Para actualizar los cambios de configuración habrá que reiniciar el servidor Apache.

Los archivos de configuración están compuestos de los siguientes elementos:

- **Directivas**: se trata de una palabra clave a la que le sigue un valor. Ej: Listen 80
  Para mostrar la lista completa de directivas: httpd -L
- **Secciones** o **contenedores**: permiten dividir el documento, y que las directivas solo se apliquen a una parte del documento. Se configuran mediante etiquetas (como en XML).
- **Comentarios**: texto que comienza con el signo **#**. Sirve para documentar el archivo de configuración.

- **Archivos .htaccess**

Son archivos de configuración que ubicados en un directorio solo afectan a ese directorio.

Por defecto, Apache no los tiene en cuenta. Para que sí se tengan en cuenta, se debe utilizar la directiva **AllowOverride** con el valor **all**.

9. Servidor web: instalación y configuración

- **Contenedores o secciones**

Las directivas que se coloquen dentro de un contenedor se aplican solo a los elementos a los que se refiere dicho contenedor.

P.ej.: En un contenedor de directorio, las directivas dentro de ese contenedor modificarán el funcionamiento de dicho directorio.

Sintaxis:

```
<nombreSección argumentos>
        directivas …
</nombreSección>
```

- **Directory**: permite establecer una configuración para un directorio concreto.
- **Files**: establece directivas para archivos con un nombre concreto.
- **DirectoryMatch**: aplica configuración a directorios mediante una expresión regular.
- **FilesMatch**: aplica configuración a varios archivos mediante una expresión regular.
- **Location**: las directivas de esta sección se referirán a una URL.
- **LocationMatch**: igual, pero especifica varias URL mediante expresión regular.
- **VirtualHost**: permite configurar un servidor virtual.
- **IfDefine**: directivas a ejecutar si se pasa al servidor un comando durante su ejecución.
- **IFModule**: directivas a ejecutar si Apache ha cargado un módulo concreto.

9. Servidor web: instalación y configuración

- **Principales directivas de Apache**

Se pueden poner en una sección o fuera de cualquier sección (serán globales).

- **ServerName**: nombre del servidor (para servidor local se utiliza **localhost**)
- **ServerAlias**: permite establecer nombres alternativos al servidor virtual. Solo se utiliza dentro de contenedores VirtualHost.
- **ServerRoot**: indica la raíz de la instalación de Apache. (La raíz debe contener **conf** y **log**).
- **Listen**: puerto de escucha del servidor. Se puede escuchar por más de un puerto.
- **DocumentRoot**: indica la ruta raíz de las páginas web. Generalmente es **htdocs**.
- **DirectoryIndex**: nombre del archivo índice del directorio. Normalmente es **index.html**, pero se puede cambiar o se puede indicar más de uno.
- **ErrorDocument**: permite indicar qué mostrar cuando ocurra un error. A la directiva le sigue: nº de error y texto o documento a mostrar.
- **ServerAdmin**: almacena el email del administrador del sistema, para recibir información de errores.
- **ErrorLog**: ruta del archivo **log** de errores (permite examinar los problemas ocurridos).
- **LogLevel**: permite indicar que eventos se almacenan en el archivo log.

10. Módulos y componentes necesarios

- **PHP**

Es el lenguaje de scripts de servidor más popular.
Es gratuito, de código abierto, y tiene muy buena relación con Apache, MySQL y Linux, aunque también con Windows.

- **Instalación de PHP**

Windows:

Se puede realizar la descarga desde la dirección http://windows.php.net/download

- Si se instala con **Apache**, hay que descargar la versión **Thread Safe**.
- Si se instala con IIS, hay que descargar la versión Non-Thread Safe.

Además, hay que tener instalado el **Visual C++ Redistribuitable para Visual Studio**.

- Descomprimir el **zip** en la ruta deseada.
- Añadir el directorio raíz de PHP y el directorio ext a **PATH** del sistema.
- Generar archivo de configuración **php.ini** a partir de **php.ini-development** o **php.ini-production**, y completarlo con extensiones deseadas.
- Añadir el módulo PHP al archivo de configuración de Apache (**httpd.conf**)
- Comprobar la instalación cargando en el navegador una página que llame a **phpinfo()**.

10. Módulos y componentes necesarios

Linux:

- Se puede realizar la instalación **a partir del código fuente** de PHP, descargando el archivo comprimido desde la dirección http://php.net/downloads.php
- O a través del **gestor de paquetes de Linux**, que será más cómodo, pero es necesario conocer el nombre de los paquetes necesarios.

En cualquier caso, tras la instalación se debe reiniciar Apache y comprobar que la instalación ha sido correcta al igual que se realizaba en Windows.

- **Configuración de PHP**

- Hay un archivo ejecutable que permite interpretar código PHP. Sus opciones: **php --help**

  Para conocer los archivos de configuración que usa PHP: **php --ini**

- El archivo de configuración general suele ser: **php.ini**

  Hay numerosas opciones de configuración.

11. Sistema gestor de base de datos: instalación y configuración

- **MySQL**

Es un sistema gestor de base de datos, que unido a Apache y PHP forman una pila de software orientada al desarrollo de aplicaciones web.

Perteneciente a la empresa Oracle, mantiene una licencia de código abierto GPL, y otra comercial con soporte completo para empresas.

- **Instalación de MySQL**

Windows:

Se puede instalar mediante la descarga de un archivo comprimido zip, o mediante un archivo instalador: http://dev.mysql.com/downloads

Este instala:

- El servidor MySQL (se instalará como servicio de Windows).
- MySQL Workbench, entorno de trabajo para administrar nuestras bases de datos.
- Conectores para lenguajes que quieran acceder a las bases de datos: Java, Python, PHP.
- Herramientas, documentación y ejemplos.

11. Sistema gestor de base de datos: instalación y configuración

Linux:

Se puede instalar mediante la descarga de archivos binarios genéricos, o a través del gestor de paquetes de Linux.

Establecimiento de la seguridad en MySQL:

Por defecto, en la mayoría de las instalaciones MySQL, el usuario root no tiene contraseña. Cualquier persona podría hacerse administrador y ocasionar problemas. Para ello, MySQL proporciona el archivo **mysql_secure_installation** que permite asegurar la instalación realizando ciertas acciones entre las que se incluyen asignar una contraseña de root.

- **Configuración de MySQL**

Para establecer la configuración del sistema: archivos de texto **win.ini** o **win.cnf**.

- **Maria DB**

Se creó con total compatibilidad con MySQL para asegurar que siempre hubiera disponible una versión abierta de MySQL. https://downloads.mariadb.org

12. Soluciones integradas

La tarea de instalar y configurar Apache, PHP y MySQL es tediosa y compleja. Las soluciones integradas son paquetes de software que instalan fácilmente estos componentes y algunos más, y lo dejan preconfigurado para que funcione correctamente.

En la fase de producción de una aplicación web no se debe instalar de esta forma el servidor de aplicaciones y de base de datos, sino que es mejor realizar una instalación por componentes más personal y optimizada. Esto solo es válido para el entorno de desarrollo.

Existen plataformas que integran: sistema operativo, servidor web, servidor de base de datos y lenguajes del lado del servidor; orientadas al desarrollo de aplicaciones web:

- **LAMP**: Linux, Apache, MySQL y PHP. Es enteramente software libre.
- **WAMP**: Windows como sistema operativo, Apache como servidor web, MySQL como servidor de base de datos y PHP, Perl o Python como lenguajes de programación.
- **MAMP**: Mac Os X como sistema operativo, Apache como servidor web, MySQL sistema gestor de bases de datos y PHP, Perl o Python como lenguajes de programación.
- **WISA**: Es una solución propietaria de Microsoft: Windows, IIS, SQL Server y ASP .NET.
- **XAMPP**: X - disponible para cualquiera de los sistemas operativos, incluye: Apache, MySQL/MariaDB y PHP/Perl.

12. Soluciones integradas

- **XAMPP**

Es la solución integrada más popular. Es multiplataforma, ya que funciona con **Windows**, **Linux** y **OSX**, e instala **Apache**, **MySQL**, **PHP**, **phpMyAdmin**, **Perl**, **FileZilla** (servidor FTP) y un servidor de correo.

En su página están disponibles los enlaces de descarga del sistema y de la documentación:
http://www.apachefriends.org

Instalación en Windows

- Lo más cómodo es pulsar: *XAMPP para Windows*, y se descargará el instalador. También se puede descargar versiones en formato ZIP que no requieren instalación.
- No se necesita instalar todos los componentes, pero si se hace se pueden activar o no. Lo imprescindible es: Apache, PHP y MySQL. Conviene instalar Perl, para poder activar la seguridad de MySQL, y phpMyAdmin para facilitar la gestión de MySQL.
- Hay que dar paso a Apache a través del cortafuegos de Windows.
- Se creará una carpeta **c:\xampp**, y un grupo de programas **XAMPP** en el menú Inicio.
- Desde cualquiera de los dos se podrá acceder al panel de control de XAMPP.

12. Soluciones integradas

Instalación en Linux

- Descargar el archivo de instalación correspondiente al sistema.
- Desde una ventana de terminal acceder al directorio de descarga y modificar los permisos del archivo:

  chmod 755 xampp-Linux-*-installer.run

- Lanzar el instalador:

  sudo ./xampp-Linux-*-installer.run

- XAMPP se instala en el directorio **/opt/lampp**, y desde ahí se accede a todos los servicios.
- Arrancar tanto Apache como MySQL:

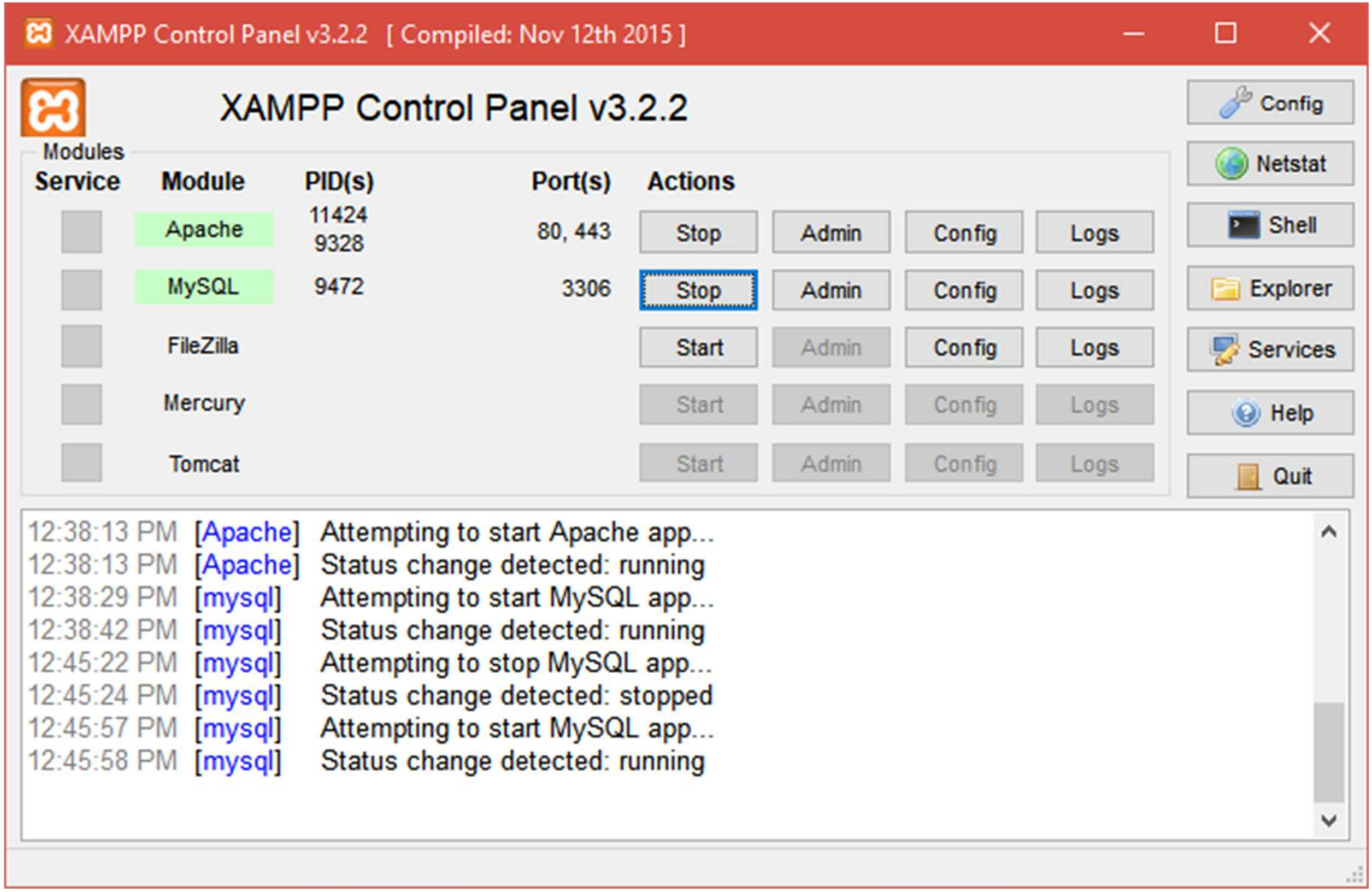
  sudo /opt/lampp/lampp start

- Parar los servidores:

  sudo /opt/lampp/lampp stop

12. Soluciones integradas

Configuración desde el panel de control



12. Soluciones integradas

Configuración desde el panel de control

- **Lista de servicios**: A la izquierda del panel aparece la lista de servidores instalados, y su estado de funcionamiento (verde: en funcionamiento).
- **Botones de arranque y parada**: Start y Stop permiten arrancar y detener el servicio.
- **Botones Service**: Permite instalar el módulo (Apache, MySQL..) como servicio de Windows, ello permite el arranque automático del servicio con el inicio del sistema.
- **Botones Config**: Permite acceder a los archivos de configuración de los servicios.
- **Botones Logs**: Permite acceder a los archivos de registro de cada módulo.
- **Botón Config** (llave inglesa): permite realizar acciones de configuración sobre el software instalado: P.ej.: indicar editor predeterminado, indicar módulos que arrancan al abrir el panel, cambiar nombre y puertos de los servicios instalados.
- **Botón Shell**: abre ventana en línea de comandos sobre el directorio raíz de XAMPP.
- **Botón Explorer**: abre ventana de navegación sobre el directorio raíz de XAMPP.
- **Botón Services**: proporciona la lista de servicios de Windows.
- **Botón Help**: ayuda de XAMPP.
- **Botón Quit**: cierra el panel de control.

12. Soluciones integradas

Configuración manual

En caso de instalar XAMPP en sistemas sin entorno gráfico no se dispondrá del panel de control. Hay que tener en cuenta:

- El directorio raíz suele ser: **c:\xampp** en Windows, y **/opt/lampp** en Linux.

  En él se encuentran los siguientes directorios:

- **apache** o **apache2**: raíz de instalación de Apache.
- **htdocs**: raíz de documentos del servidor Apache.
- **php**: raíz de PHP.
- **mysql**: raíz de MySQL.
- **bin** y **sbin**: (solo en Linux) contienen los ejecutables de los servicios instalados: httpd, mysqld, php, mysql_secure_admin, htpasswd,..