Despliegue de Aplicaciones Web

T.1 Implantación de arquitecturas web

Índice

1 Modelos de arquitecturas web

Modelos de arquitecturas web

Los modelos de arquitecturas web **describen la relación entre los distintos elementos que componen la estructura de funcionamiento de las páginas y aplicaciones web**.

Modelos de arquitecturas web

Modelo Punto a Punto (P2P – Peer to Peer)

Una red punto a punto (P2P - Peer to Peer) es un tipo de arquitectura de red **descentralizada y distribuida** en la que los **nodos** individuales de la red (peers) **actúan** tanto **como suministradores** como **clientes** de recursos, en contraste con el modelo cliente-servidor (C/S) en el que los nodos clientes acceden a los recursos que proporcionan los servidores centrales.

Modelos de arquitecturas web

En una red P2P, las tareas, como realizar un streaming de audio o vídeo, se **comparten** entre múltiples puntos interconectados que ponen una parte de sus **recursos** (CPU, almacenamiento, ancho de banda) disponibles directamente por otros puntos de la red, sin la necesidad de disponer de servidores que realicen la coordinación centralizada.

Modelos de arquitecturas web

Ventajas P2P:

- **Escalabilidad**: Millones de usuarios potenciales. Cuantos más nodos conectados, mejor será su funcionamiento, al contrario del modelo C/S.

- **Robustez**: Los clientes pueden localizar los recursos sin depender del correcto funcionamiento de un único servidor.

- **Descentralización**: Ningún nodo es imprescindible para el funcionamiento de la red.

- **Distribución de costes entre los nodos**: No hace la carga sobre un único nodo de la red.

Modelos de arquitecturas web

Inconvenientes P2P:

• **Falta de fiabilidad de los recursos**: No hay garantías de que los recursos obtenidos desde un nodo sean realmente los deseados. En el modelo C/S, el recurso obtenido del servidor central es único, y no ha podido ser modificado por otro nodo.

• **Difícil mantenimiento**: La actualización de los recursos compartidos debe realizarse en todos los nodos.

Modelos de arquitecturas web

Ejemplos P2P:

Modelos de arquitecturas web

Modelo Cliente-Servidor (C/S)

El modelo cliente-servidor es un modelo informático que actúa como una aplicación distribuida que particiona tareas o cargas de trabajo entre los **proveedores de** un recurso o **servicio**, llamados **servidores**, y los **solicitantes del servicio**, llamados **clientes**.

Modelos de arquitecturas web

Lo más frecuente es que los clientes y los servidores se comuniquen a través de una red informática, pero ambos pueden residir en la misma máquina.

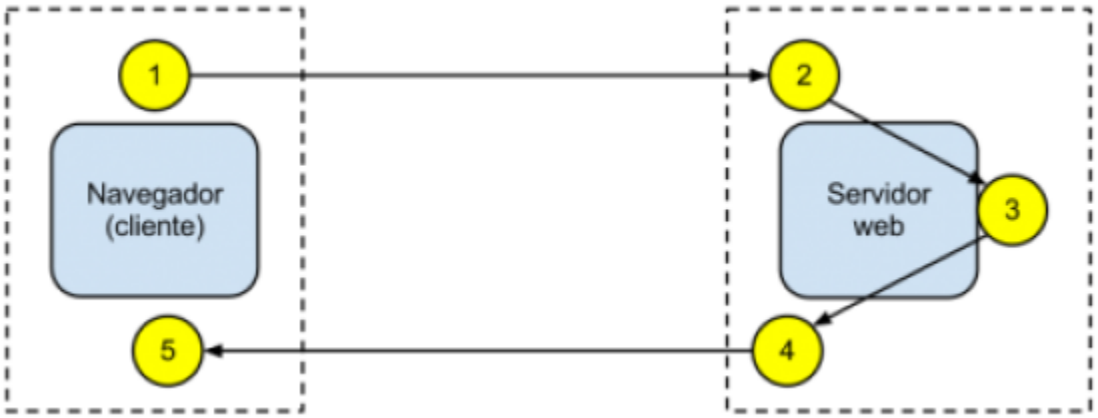
Un **servidor es un host que ejecuta** uno o más **programas servidores que comparten sus recursos con los clientes**, aunque no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa.

Modelos de arquitecturas web

El **esquema de funcionamiento** más básico del modelo cliente servidor para una arquitectura web está basado en uno o varios clientes que **solicitan una página web a un servidor web**:

1. Desde el **navegador web** (**cliente**) el usuario solicita la carga de una página web indicando su URL.

2. El **servidor** recibe la petición de la página web .

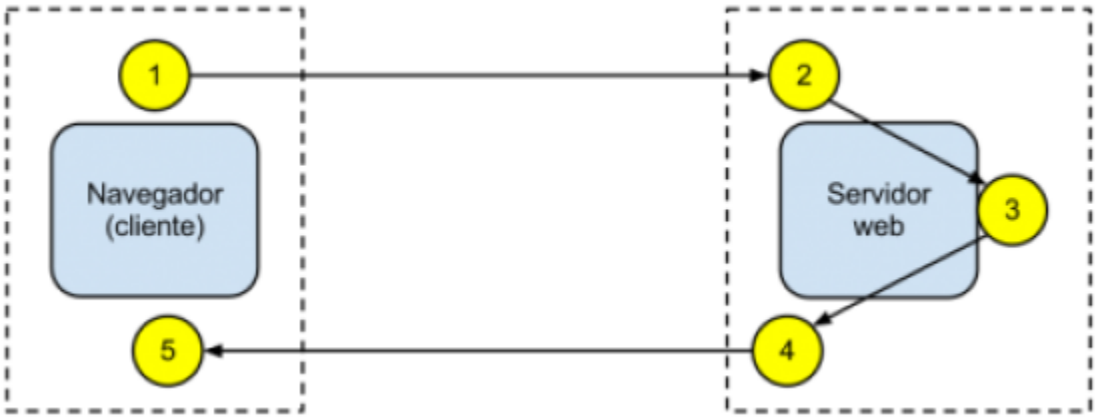


Modelos de arquitecturas web

3. **Busca** en su sistema de almacenamiento la página solicitada

4. **Envía** el contenido de la página web (**código fuente**) por el mismo medio por el que recibió la petición.

5. El **navegador web** recibe el código fuente de la página y lo interpreta **mostrando al usuario la página web**.



Modelos de arquitecturas web

Ventajas del modelo Cliente-Servidor:

• **Centralización del control**: los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema. Esta centralización también facilita la tarea de actualizar datos u otros recursos (mejor que en las redes P2P)…

• **Escalabilidad**: se puede aumentar la capacidad de clientes y servidores por separado. Cualquier elemento puede ser aumentado (o mejorado) en cualquier momento, o se pueden añadir nuevos nodos a la red (clientes y/o servidores).

Modelos de arquitecturas web

• **Fácil mantenimiento**: al estar distribuidas las funciones y responsabilidades entre varios ordenadores independientes, es posible reemplazar, reparar, actualizar, o incluso trasladar un servidor, mientras que sus clientes no se verán afectados por ese cambio (o se afectarán mínimamente). Esta **independencia de los cambios** también se conoce como **encapsulación**.

• Existen tecnologías, suficientemente desarrolladas, diseñadas para el paradigma de C/S que aseguran la seguridad en las transacciones, la amigabilidad de la interfaz, y la facilidad de empleo.

Modelos de arquitecturas web

Inconvenientes del modelo Cliente-Servidor:

• La **congestión del tráfico** ha sido siempre un problema en el paradigma de C/S. Cuando una gran cantidad de clientes envían peticiones simultaneas al mismo servidor, puede ser que cause muchos problemas para éste (a mayor número de clientes, más problemas para el servidor). Al contrario, en las redes P2P como cada nodo en la red hace también de servidor, cuantos más nodos hay, mejor es el ancho de banda que se tiene.

Ataques de DoS y DDoS →

Modelos de arquitecturas web

• El paradigma de C/S clásico **no tiene la robustez de una red P2P.** Cuando un servidor está caído, las peticiones de los clientes no pueden ser satisfechas. En la mayor parte de redes P2P, los recursos están generalmente distribuidos en varios nodos de la red. Aunque algunos salgan o abandonen la descarga; otros pueden todavía acabar de descargar consiguiendo datos del resto de los nodos en la red.

Modelos de arquitecturas web

• El **software y el hardware de un servidor son generalmente muy determinantes.** Un hardware regular de un ordenador personal puede no poder servir a cierta cantidad de clientes. Normalmente se necesita software y hardware específico, sobre todo en el lado del servidor, para satisfacer el trabajo. Por supuesto, esto aumentará el coste.

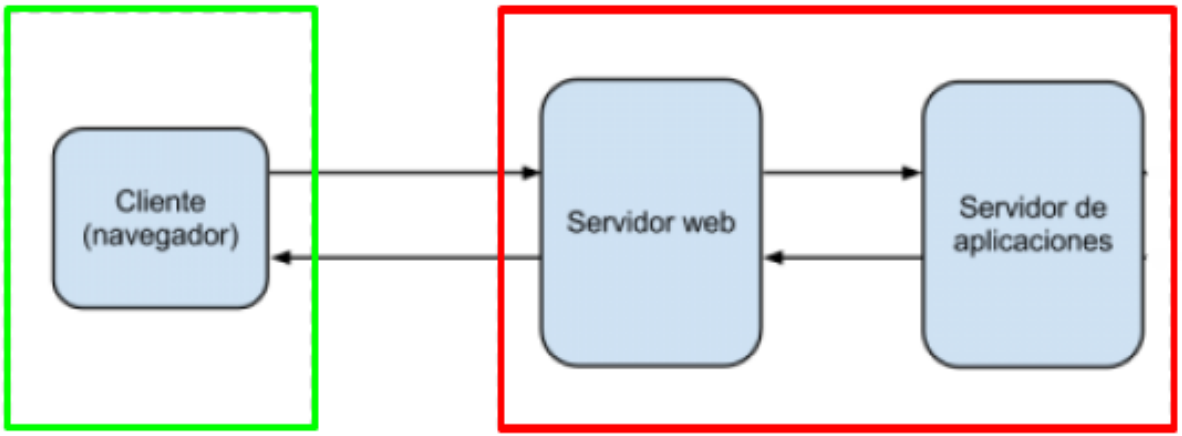
Modelos de arquitecturas web

Modelo con servidor de aplicaciones

La función que realiza un servidor de aplicaciones es diferente, ya que **los recursos** que va a manipular no son archivos estáticos, sino que **contienen el código que tiene que ejecutar**. Es decir, un servidor web en solitario enviaría al cliente el recurso solicitado tal cual, mientras que el servidor de aplicaciones lo ejecuta y envía al cliente el resultado a través del servidor web.

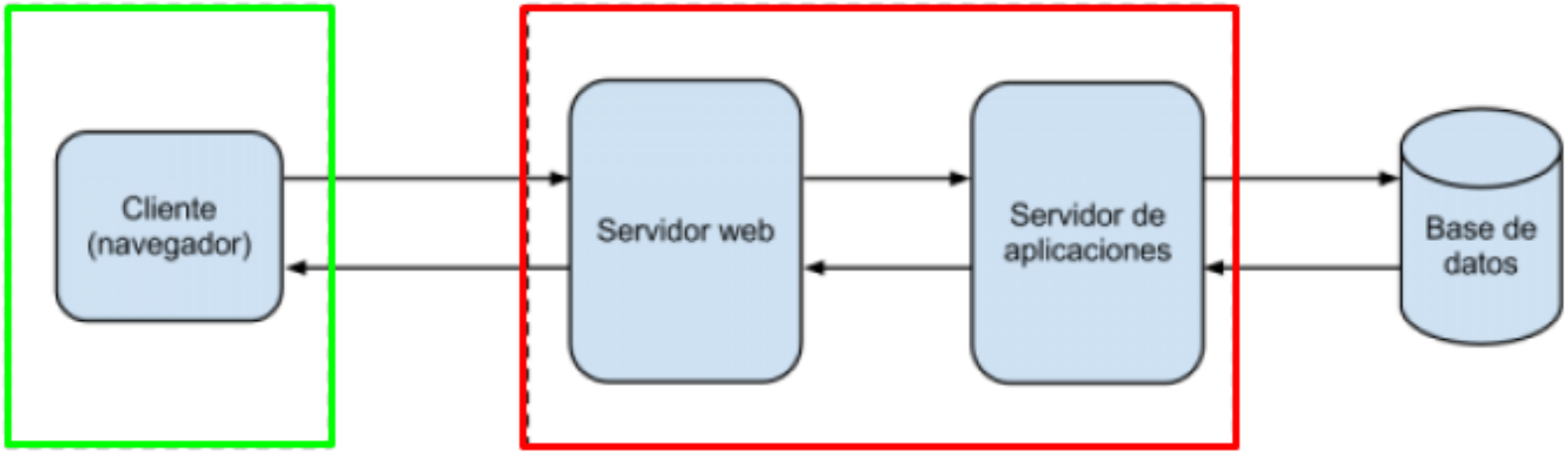
Modelos de arquitecturas web

El servidor web y el servidor de aplicaciones pueden residir en una misma máquina como se refleja en el siguiente esquema. El servidor web recibe la petición de un recurso, y si éste corresponde a un recurso dinámico, transfiere la parte correspondiente al servidor de aplicaciones el cual devolverá de nuevo al servidor web el recurso ejecutado. Finalmente será el servidor web el que envíe al cliente el resultado final.



Modelos de arquitecturas web

Es muy frecuente que el servidor de aplicaciones deba conectarse con una base de datos para obtener los datos solicitados durante la ejecución del código. Dicha base de datos puede residir en la misma máquina que el servidor web o en otro host conectado en red.



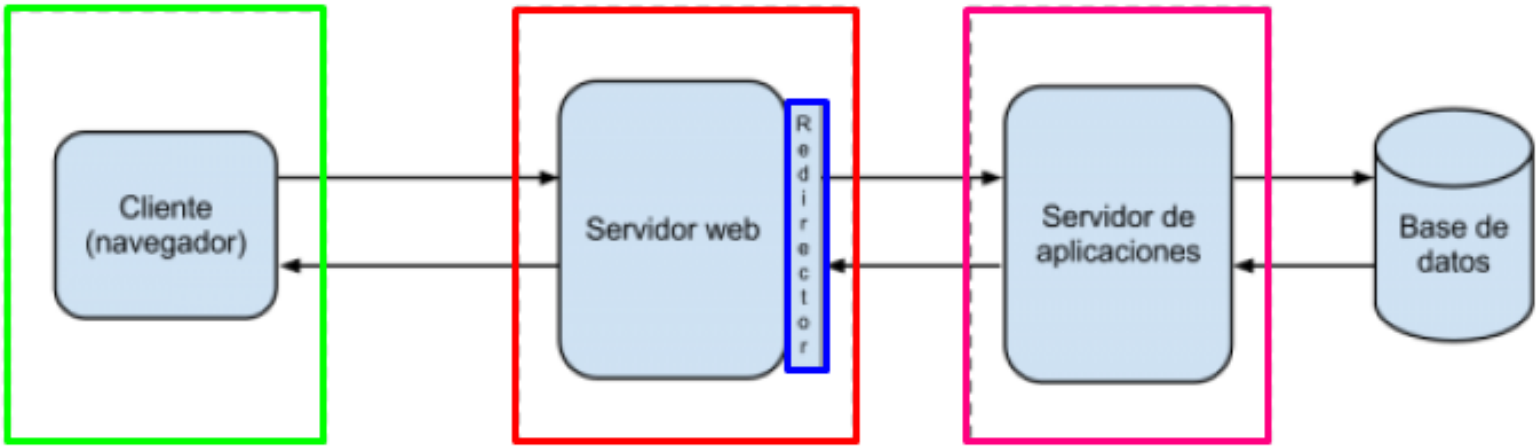
Modelos de arquitecturas web

Modelo con servidor de aplicaciones externo

La parte correspondiente al servidor web suele tener menos carga de trabajo que el servidor de aplicaciones por lo que se puede establecer una estructura en la que el **servidor web** sea **independiente** del **servidor de aplicaciones**, de manera que el servidor web se pueda implantar de forma dedicada precisando menos recursos.

Modelos de arquitecturas web

Un **redirector** será el encargado de transferir al servidor de aplicaciones los elementos que necesiten ser ejecutados, mientras que no pasarán del servidor web los recursos estáticos.



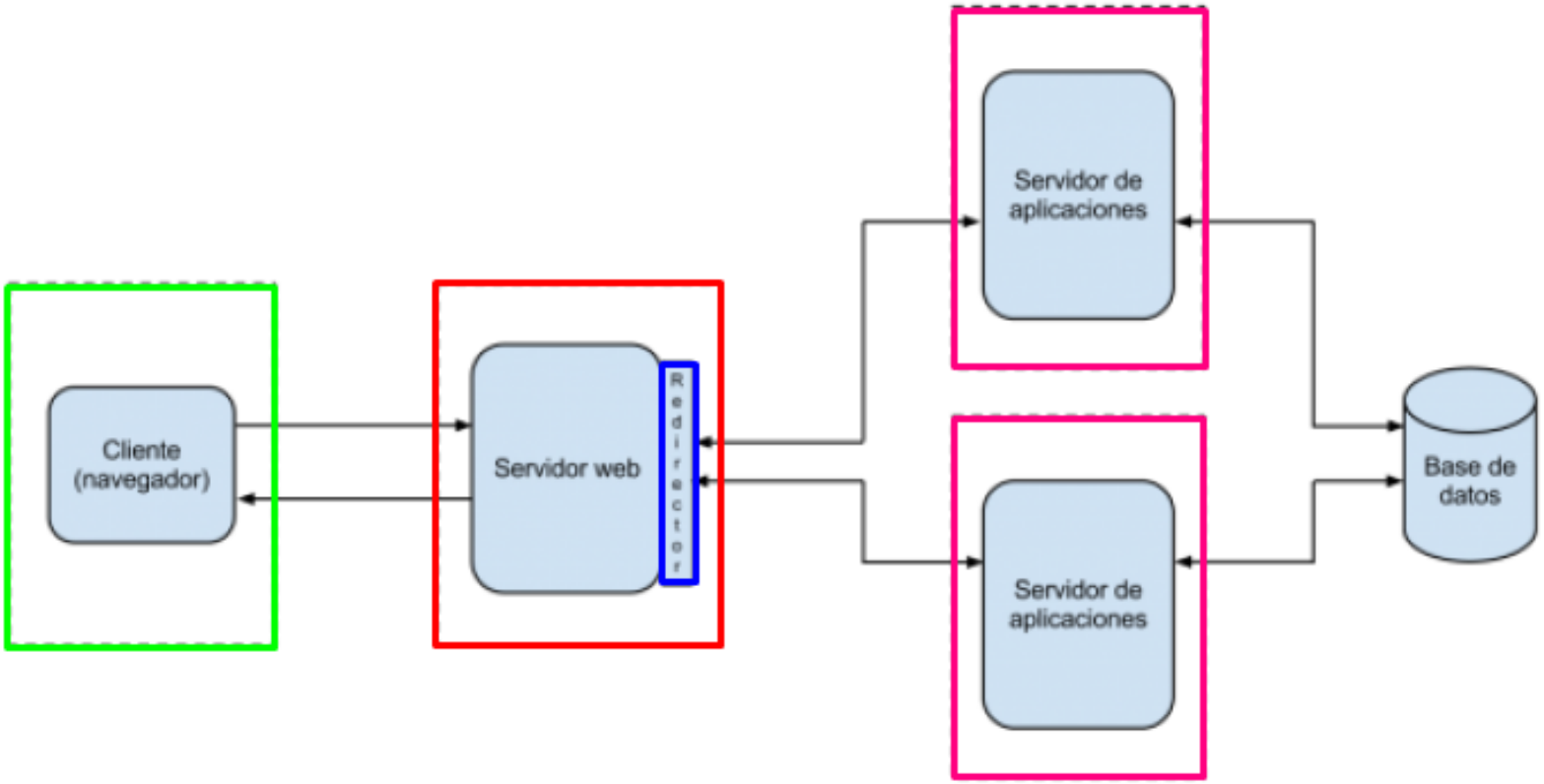
Modelos de arquitecturas web

Modelo con varios servidores de aplicaciones

Si la carga de trabajo del servidor de aplicaciones se estima que va a ser elevado, se puede implantar un sistema con varios servidores de aplicaciones unidos a un mismo servidor web que requiere menos rendimiento. La conexión se realizará a través de un **redirector** como en el caso anterior, que además **realizará las funciones de balanceador de carga** para determinar en un determinado momento qué servidor de aplicaciones debe ejecutar el código en función de la carga de trabajo que tenga cada uno.

Modelos de arquitecturas web

En este caso, todos los servidores de aplicaciones deben ser iguales.



2 Servidores web y de aplicaciones

Servidores web y de aplicaciones

Servidor Web

Un servidor web es un servidor que **permite el acceso a recursos mediante el protocolo HTTP** (HyperText Transfer Protocol) **de internet.**

Son servidores capaces de dar acceso y de permitir la gestión de un conjunto de **recursos estáticos** como respuesta a peticiones recibidas por los clientes. Es decir, que permiten consultar, cargar y eliminar recursos del servidor. Estos recursos suelen ser documentos de HTML o variantes de este formato y contenidos adjuntos o relacionados con estos documentos, como imágenes, vídeos, etc.

Servidores web y de aplicaciones

Estos recursos suelen estar guardados en forma de archivos en dispositivos de almacenamiento propios del servidor.

El concepto original de servidor web no contempla la posibilidad de generar de forma dinámica los contenidos a partir de la ejecución de código como respuesta de las peticiones. Pero, en la actualidad, la mayoría de servidores web admiten la instalación de módulos que permiten que se generen contenidos dinámicos a partir de la ejecución de programas escritos en diversos lenguajes de programación (PHP, Javascript, Python, Perl, etc.), aunque esta característica es más propia de los servidores de aplicaciones.

Servidores web y de aplicaciones

Ejemplos de servidores web:

- Apache HTTP Server
- Nginx
- Microsoft Internet Information Server

Servidores web y de aplicaciones

Servidores de aplicaciones

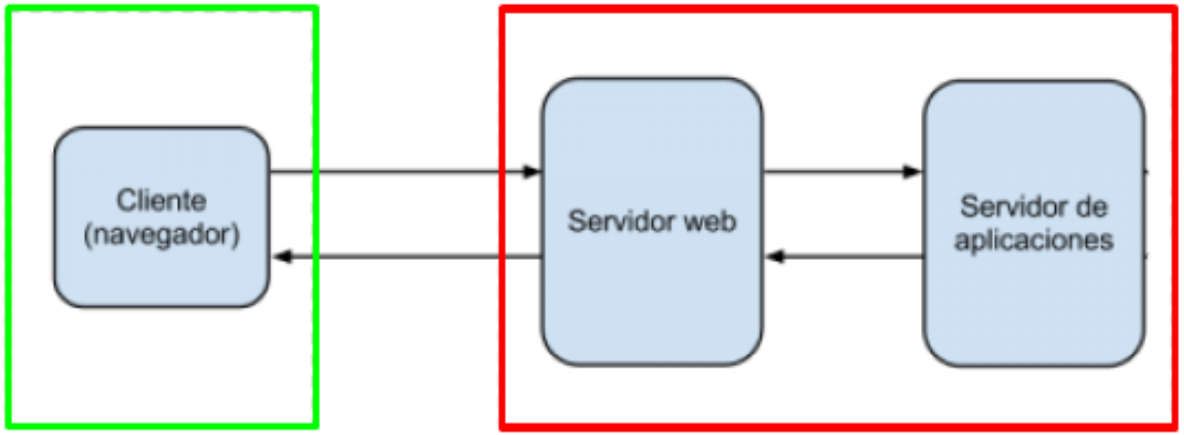
Un servidor de aplicaciones por lo general es un servidor que **ofrece a los clientes un servicio de ejecución de aplicaciones**. Si nos centramos en las aplicaciones web, un servidor de aplicaciones **es un software que controla la ejecución de programas**. Los clientes, desde un navegador (usando el protocolo HTTP), acceden a una interfaz web desde de dónde ejecutarán la aplicación.

Servidores web y de aplicaciones

Un servidor de aplicaciones web puede entenderse como un servidor orientado a la ejecución de programas que puede recibir las peticiones de servicio y devolver los resultados utilizando los mismos protocolos (HTTP) y formatos de datos que los servidores web (HTML).

Servidores web y de aplicaciones

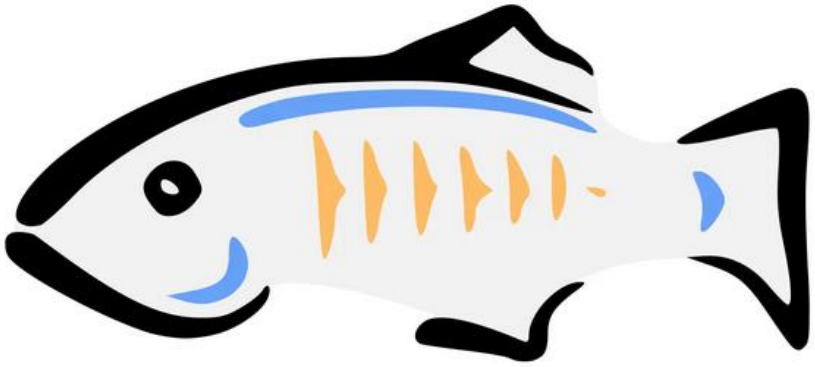
Si el propio servidor no tiene la capacidad de interactuar con estos protocolos pueden trabajar conjuntamente con el soporte de un servidor web que haga de intermediario entre el servidor de aplicaciones y el cliente. Los servidores de aplicaciones, además, suelen proporcionar un amplio conjunto de servicios complementarios orientados a la persistencia de datos, la seguridad, el control de transacciones y concurrencia, entre otros.

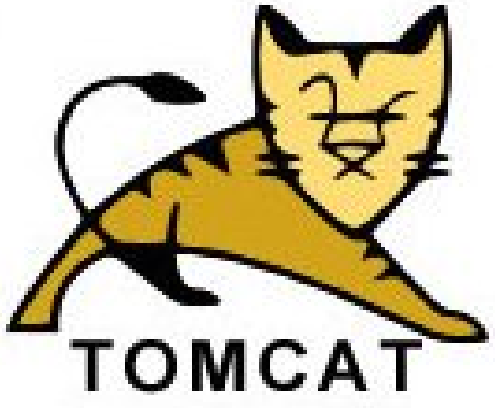


Servidores web y de aplicaciones

Ejemplos de servidores de aplicaciones :

- GlassFish (servidor Java EE, Oracle)
- TOMCAT
- Microsoft Internet Information Server (servidor .NET)





Servidores web y de aplicaciones

Servidor de bases de datos

Un servidor de bases de datos se utiliza para **almacenar, recuperar y administrar los datos de una base de datos**.

El servidor gestiona las actualizaciones de datos, permite el acceso simultáneo de muchos servidores o usuarios web y garantiza la seguridad y la integridad de los datos.

Servidores web y de aplicaciones

Entre sus funciones básicas, el software de servidores de bases de datos ofrece herramientas para facilitar y acelerar la administración de bases de datos.

Algunas funciones son la exportación de datos, la configuración del acceso de los usuarios y el soporte de datos.

Servidores web y de aplicaciones

Ejemplos de servidores de bases de datos:

- Oracle Database
- MySQL
- Microsoft SQL Server
- PostgreSQL
- MongoDB
- Firebase

PostgreSQL

Servidores web y de aplicaciones

Servidores de archivos

Un servidor de archivos es un servidor que **permite gestionar a través de red la carga, descarga, actualización y eliminación de archivos almacenados en los sus dispositivos desde ordenadores cliente.**

Servidores web y de aplicaciones

En el ámbito de las aplicaciones web, los servidores de archivos se utilizan principalmente para desplegar las aplicaciones sobre el servidor en el que se ejecutarán. El despliegue de una aplicación web sobre los servidores de producción comporta habitualmente la carga de grandes cantidades de archivos sobre estos servidores. Debido a que el desarrollo y mantenimiento de estas aplicaciones se hace en las máquinas de programadores, es necesario algún sistema de transferencia de archivos cada vez que se quiere actualizar la versión de producción de una aplicación.

Servidores web y de aplicaciones

Uno de los protocolos más usados para la transferencia de archivos en el despliegue de aplicacionesweb es el protocolo **FTP** (file transfer protocol), con sus variantes **FTPS** **y** **SFTP** para adaptarse a las necesidades actuales de seguridad. (FTPS vs SFTP)

Ejemplos de servidores de transferencia de archivos

- ProFTPD o vsftpd para Linux
- Microsoft Internet Information Server para Windows.

Servidores web y de aplicaciones

Servidores de directorio

Un servidor de directorio es un servidor que **permite gestionar información administrativa respecto al entorno de una aplicación web**, como pueden ser, por ejemplo, los usuarios autorizados con sus roles o permisos, etc.

Servidores web y de aplicaciones

La principal utilidad de los servidores de directorio es facilitar la gestión de información relativa a la explotación de aplicaciones web. La ventaja de gestionar esta información mediante este tipo de servidores es la centralización de datos y la facilidad acceso mediante protocolos estándar como LDAP.

Ejemplos de servidores de directorio:

- OpenLDAP para Linux
- Active Directory para Windows

3 Estructura y recursos de una aplicación web

Estructura y recursos de una aplicación web

Las aplicaciones web, además de presentar una arquitectura cliente-servidor (lo que no es necesario en el caso de las aplicaciones de escritorio), **suelen estar estructuradas con una gran cantidad de archivos y recursos de distintos tipos**.

Por eso es necesario establecer unas directrices para organizar la ubicación de estos componentes y su interrelación durante la fase de desarrollo, así como también en el momento de poner la aplicación en producción. De lo contrario, el desarrollo y mantenimiento de una aplicación de tamaño medio o grande se convertirá en una tarea casi imposible de manejar.

Estructura y recursos de una aplicación web

Olvidándonos de la organización o estructura que impone el hecho de escoger unas determinadas herramientas de desarrollo o un determinado servidor web o de aplicaciones, estas aplicaciones se pueden estructurar según varios modelos de organización de sus componentes y recursos. **Algunos** de los **modelos de estructuración de aplicaciones web** que podemos encontrar más habitualmente **son** las que se describen a continuación.

Estructura y recursos de una aplicación web

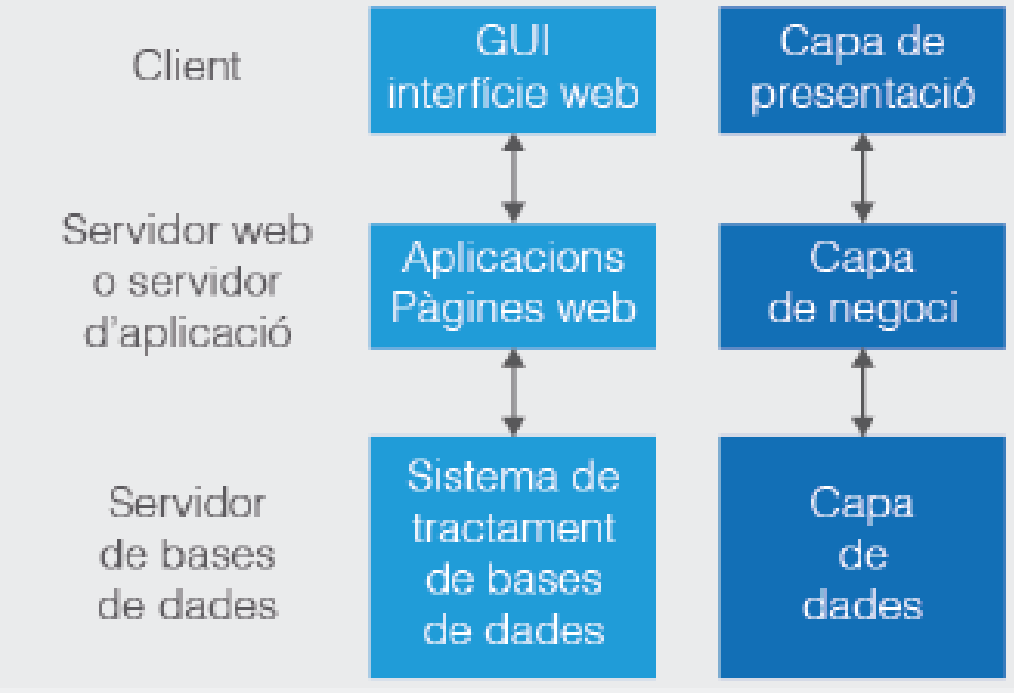
Arquitectura multinivel

La arquitectura multinivel (multitier architecture) es **un tipo concreto de la arquitectura cliente-servidor en la que los componentes y recursos de una aplicación se separan según su función.** Una de las divisiones más utilizadas es la que **separa el nivel de presentación, el nivel de lógica de aplicación y el nivel de gestión de datos**.

En este caso, la estructura concreta sería tres niveles (3-tier architecture). El modelo **se** define como N-tier architecture (multinivel), ya que **propone una división flexible de las aplicaciones en los niveles que sea necesario para hacer más eficiente su desarrollo, mantenimiento y explotación**.

Estructura y recursos de una aplicación web

En este modelo, **la división por niveles se hace de forma lineal**: el nivel 1 interactúa de forma directa y única con el nivel 2, el nivel 2 interactúa con el 3, y así sucesivamente



Estructura y recursos de una aplicación web

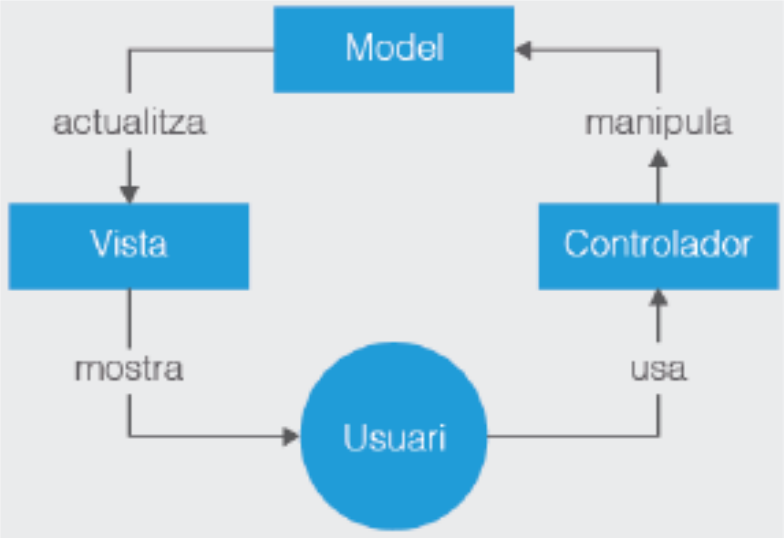
Hay que **diferenciar entre el concepto multinivel** (multitier N-tier) **y multicapa** (multilayer) N-layer).

En el **modelo multinivel** cada nivel, además de implementar una función concreta, es ejecutado por un hardware diferente del resto de niveles.

En el **modelo multicapa**, cada capa desarrolla una función concreta que puede ser ejecutada por un mismo ordenador que se encarga, también, de la ejecución de otras capas.

Arquitectura modelo-vista-controlador

La arquitectura modelo-vista-controlador (modelo view controller) es una arquitectura que **separa la representación de la información y la lógica de una aplicación de la interacción del usuario.**



Estructura y recursos de una aplicación web

Los tres elementos que define esta arquitectura son:

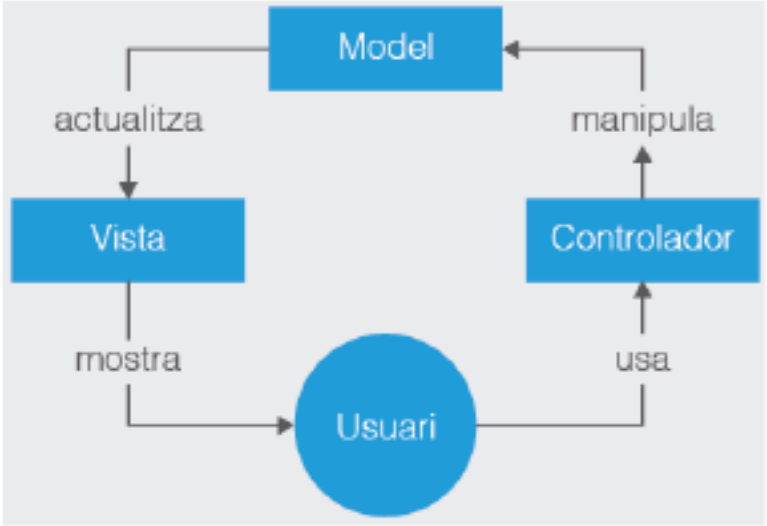
• **Modelo**: contiene los datos de la aplicación, las reglas de negocio o la lógica de la aplicación y sus funciones.

• **Vista**: es la representación visible de la aplicación, la salida de los datos hacia al usuario, es decir, la **interfaz**.

• **Controlador**: controla la interacción del usuario (entrada de datos) y convierte esta interacción en órdenes o pedidos para el modelo o la vista.

Estructura y recursos de una aplicación web

La interrelación entre los elementos de esta arquitectura no se realiza siguiendo un modelo lineal como el modelo multinivel, sino que se trata de un **modelo circular**.



Estructura y recursos de una aplicación web

Paralelamente a la estructura de la aplicación, debe tenerse en cuenta que cada nivel, capa o módulo puede estar formado por un gran número de componentes y recursos de varios tipos: archivos HTML, CSS, imágenes, etc.

Por eso es conveniente establecer un **sistema de organización coherente y eficiente** para estructurar todos estos componentes que se acaban generando durante el desarrollo de una aplicación web. La mayoría de plataformas de desarrollo avanzadas imponen mecanismos para organizar y describir de forma sistemática la localización, las características y la configuración de los componentes y recursos de las aplicaciones.

Estructura y recursos de una aplicación web

Entre estos mecanismos destacan dos:

• **Estructura de directorios**: las plataformas avanzadas de desarrollo de aplicaciones web suelen definir una estructura de directorios mínima que toda aplicación debe tener a partir de la cual se despliegan los diversos tipos de componentes. Los desarrolladores deben seguir las directrices de cada plataforma.

• **Descriptor de despliegue**: hay un archivo de configuración donde se puede especificar el nombre, la ubicación y los parámetros de configuración de los diversos componentes que forman una aplicación para tener esta información centralizada, accesible y actualizable sin necesidad de realizar modificaciones en el código fuente de la aplicación. Este descriptor describe cómo debe desplegar la aplicación en el servidor.