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.

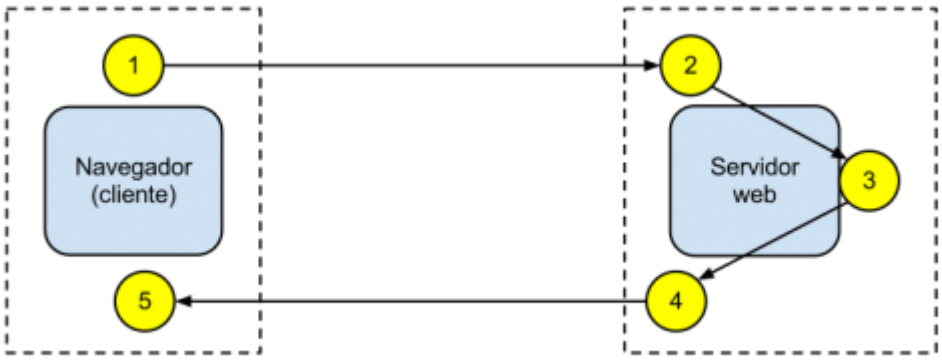
Modelo Cliente-Servidor

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**.

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.

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
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.



El servidor web envía al cliente el recurso solicitado sin hacer ningún tratamiento sobre él. Es decir, envía de manera indiferente una página HTML, una imagen, un archivo de sonido o cualquier otro tipo de archivo.

Ventajas

- **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).

- **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.

Desventajas

- 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.

- 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.

- 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.

- El cliente **no dispone de los recursos** que puedan existir en el servidor. Por ejemplo, si la aplicación es una Web, no podemos escribir en el disco duro del cliente o imprimir directamente sobre las impresoras sin sacar antes la ventana previa de impresión de los navegadores.

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.
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.

Ventajas

- **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.

Desventajas

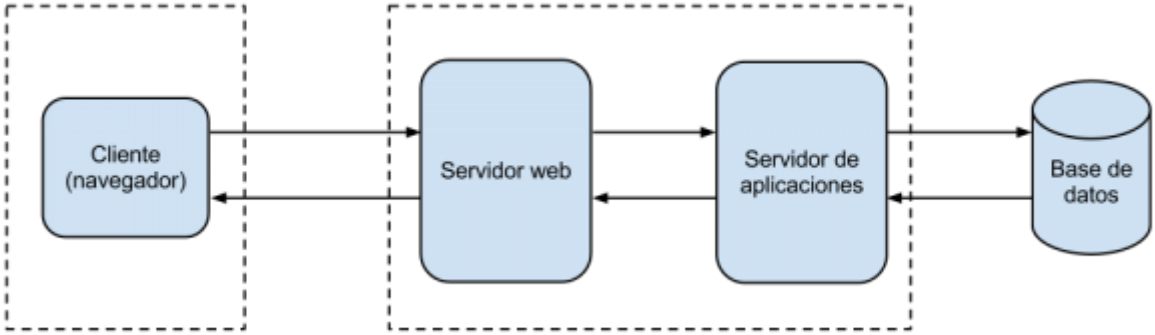
- **Falta de fiabilidad de los recursos**: No hay garantías de que los recursos obtenidos desde un nodo sean realmente los deseados. En el modelo C/S, el recurso obtenido del servidor central es único, y no ha podido ser modificado por otro nodo.
- **Difícil mantenimiento**: La actualización de los recursos compartidos debe realizarse en todos los nodos.

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.

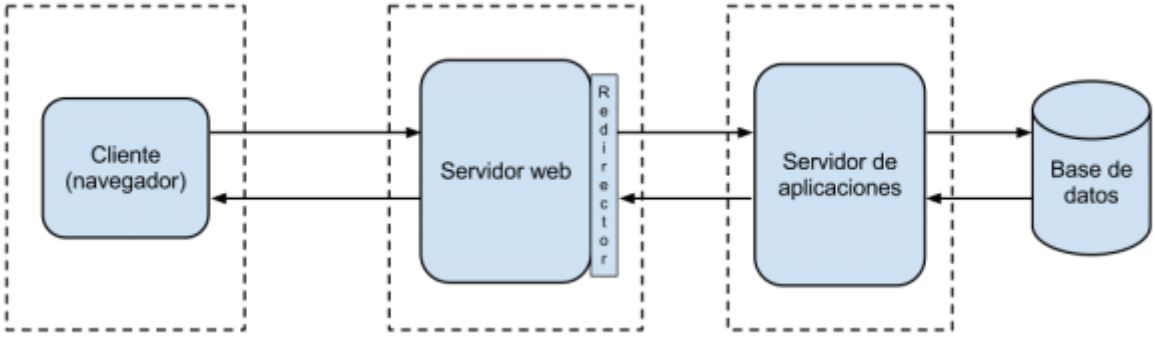
El servidor web y el servidor de aplicaciones pueden residir en una misma máquina como se refleja en el siguiente esquema. El servidor web recibe la petición de un recurso, y si éste corresponde a un recurso dinámico, transfiere la parte correspondiente al servidor de aplicaciones el cual devolverá de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

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.



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. 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.



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