UT6-TFU

Ingeniería en Informática

Análisis y Diseño de Aplicaciones I

Grupo 5

Integrantes del grupo:

Rafael Alonso

Federico Ferreira

Walter Taño

Nestor Martinez

Rafael Filardi

Docentes: José Abadie, Javier Yannone

Conceptos Clave:

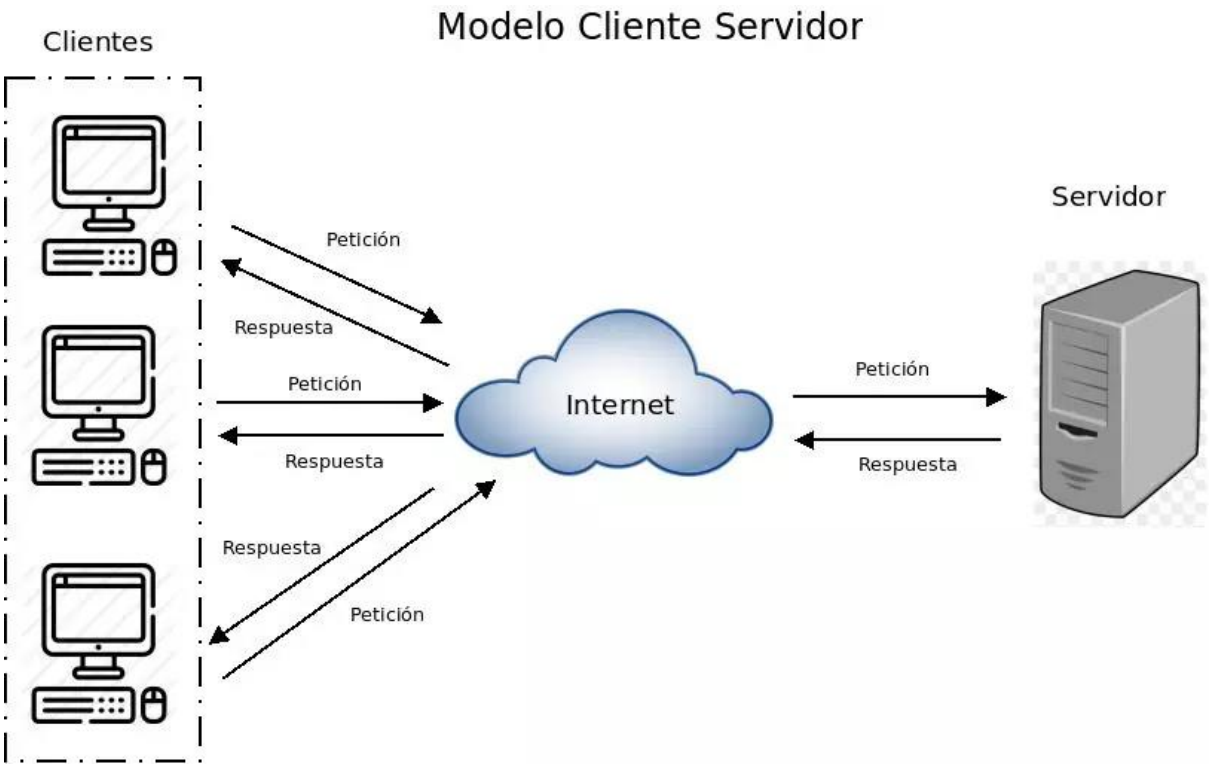
Separación Lógica

Se refiere a una separación a nivel de implementación de software, haciendo que este interactúe de manera como sí estuviese en hardware diferente, aunque pueda no necesariamente realmente estarlo. Podemos hablar de separación lógica cuando hablamos por ejemplo de un cliente y un servidor que se hallan en máquinas virtuales separadas pero sobre el mismo host, o cuando físicamente se hallan en hardware diferente, lo importante es que se utilizan medios de comunicación protocolizados entre ellos como sí los fuesen.

Arquitectura Cliente Servidor

La arquitectura cliente-servidor es un modelo de diseño de software y redes en el que las tareas se distribuyen entre dos tipos de entidades, clientes y servidores. Los clientes se definen como dispositivos o programas que solicitan y consumen recursos o servicios, y los servidores son los dispositivos o programas que proporcionan y administran dichos servicios.

Los clientes envían peticiones a los servidores, los servidores procesan esas peticiones y devuelven las respuestas correspondientes. Los clientes y los servidores se comunican a través de una red, por ejemplo internet, y utilizan protocolos de comunicación como HTTP, FTP, SMTP, etc.

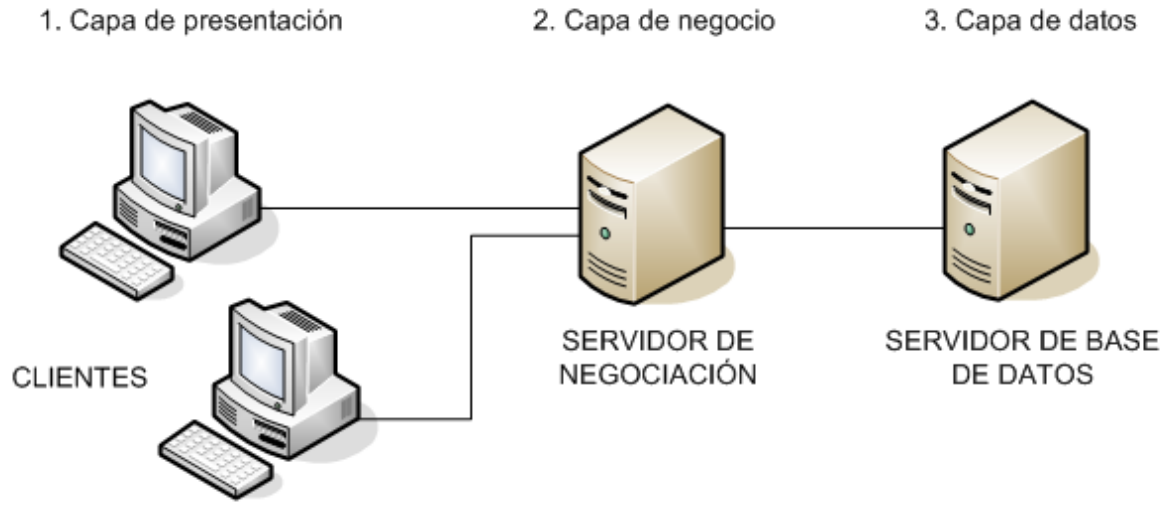


Idealmente los servidores son diseñados para ser confiables, escalables y capaces de manejar múltiples solicitudes de clientes simultáneamente. Pueden proporcionar una variedad de servicios tales como almacenamiento de datos, procesamiento de datos, servicios web, correo electrónico, etc

Por otro lado, los clientes usualmente son dispositivos como computadoras personales, teléfonos móviles, u otros programas que se comunican con servidores. Estos clientes pueden tener interfaces de usuario para permitir que los usuarios interactúen con los servidores y vean las respuestas obtenidas

Arquitectura En Capas

La arquitectura en capas consiste en la separación de un programa en capas lógicas con interfaces definidas, donde cada una tiene una función o responsabilidad compartida. Este modelo suele conformarse en 3 capas principales: capa de interfaz de usuario, capa de servicios o de negocio, y capa de acceso de datos o repositorio. Similarmente se dividen en niveles, que refieren a las ubicaciones físicas de las capas. Por ejemplo una aplicación donde la presentación y lógica se realiza en el cliente y el acceso de datos se delega a un servidor se puede considerar una arquitectura en 3 capas y 2 niveles.



[1]

Las capas que se utilizan y la cantidad dependen de la naturaleza de la aplicación, pueden tenerse capas que se encargan de balanceo de cargas, de seguridad o de tareas o servicios puntuales,, pero sí es necesario que estén todas claramente diferenciadas en las responsabilidades que abarcan y las interfaces por la que interactúan. El valor de esta estructura se da en que cada capa sólo debería saber interactuar con las capas directamente “arriba” o “abajo”, y mientras estas cumplan las interfaces dadas son fácilmente extensibles, modificables e incluso reemplazables por diferentes implementaciones.
Existen variedades de esta arquitectura también basadas en que tanta interacción o acoplamiento existen entre las capas. Por ejemplo podemos hablar de capas de caja negra, donde la capa presenta interfaces claras y específicas para su uso y no permite interacción por fuera de estas, como también opuestamente de caja blanca, donde la capa está más expuesta a sus vecinas. Estas variaciones tientan diferente rigidez sobre los principios OCP y SRP en el que esta arquitectura se basan.

[1] Representación de arquitectura en 3 capas. Recuperado de Wikipedia, por William Fernando(2003)

Por el lado de la interacción tenemos donde las capas están estrictamente limitadas a interactuar con la capa debajo y responder a la superior, o de Capas Estrictas o Cerradas, como también donde las capas pueden saltarse capas e interactuar con otras más profundas que la vecina, conocido como layer-skipping o capas abiertas.

Casos de uso

Casos de Uso de Arquitectura en Capas

Esta arquitectura admite una amplia flexibilidad, por lo que los usos pueden ser sumamente variados. Un uso clásico es en sistemas de servicios Web, donde se identifican capas claras de presentación y lógica básica en el frontend y de definición de endpoints, manejo de solicitudes, acceso a base de datos y lógicas agregadas, y otras no tan visibilizadas como proxies, middlewares o balanceadores de carga a nivel de backend.
De manera general, podríamos decir que para evaluar la necesidad de utilizar esta arquitectura debemos tener en cuenta los tres factores principales para cualquier proyecto: tiempo, costo y alcance. Sí tenemos un proyecto pequeño y que no pretendamos escalar o conectarlo en una red de sistemas, no vale la complejidad agregada. Sí tenemos una solución con partes claras que deben poder abstraerse entre sí y ser reemplazables y reutilizables en el futuro, seguramente sea la mejor idea.

Casos de Uso de Arquitectura Cliente-Servidor

La arquitectura cliente-servidor es utilizada en una amplia variedad de casos de uso, algunos ejemplos incluyen aplicaciones web, en este caso los clientes, por ejemplo, navegadores web, solicitan páginas web y recursos a servidores a través de los protocolos HTTP y HTTPS, el servidor procesa dichas solicitudes, accede a los datos almacenados y devuelve las respuestas correctas, las cuales son renderizadas y mostradas en el navegador del cliente. Muy similar al ejemplo anterior, esta arquitectura es utilizada en aplicaciones mobile, el cliente es el dispositivo móvil, el cual manda solicitudes al backend de la aplicación que se está utilizando y muestra las respuestas del servidor al usuario. Otro ejemplo común es el uso de bases de datos, se utiliza esta arquitectura para permitir que los clientes accedan y manipulen los datos almacenados en el servidor de la base de datos.

Ventajas y Desventajas

Ventajas y Desventajas de la Arquitectura en Capas

Pros:

- Modular: Cada capa es individual y cohesiva, y se pueden trabajar de manera independiente y ser fácilmente sustituidas por diferentes implementaciones, que son altamente reutilizables. Asegura separación de responsabilidades (SRP)
- Escalable: Se pueden agregar y sacar capas en medida de necesidad.

Cons:

- Consumo de Recursos: Al necesitar algún medio de comunicación entre las capas, se requieren más recursos para soportarlos, estos también pueden terminar afectando el rendimiento.

- Complejidad: La necesidad de interfaces y comunicación ordenada y correcta complejiza el sistema, y puede aumentar el tiempo de desarrollo.
- Fallas en Cadena: En caso de una falla de una de las capas inferiores, puede causar el colapso de todas las superiores.

Ventajas y Desventajas de la Arquitectura Cliente Servidor

Pros:

- Escalable: Es fácil escalar tanto los clientes como los servidores, se pueden agregar más servidores para manejar un aumento en la carga de trabajo o más clientes sin afectar el rendimiento general.
- Centralizado: Permite centralizar el procesamiento y gestión de recursos en los servidores, lo que facilita el control y administración sobre los mismos.
- Reutilizable: Es posible utilizar un servidor solo para brindar servicios compartidos a varios clientes.

Contras:

- Dependencia de Red: La comunicación entre clientes y servidores se realiza a través de una red, en caso de que la red experimente problemas, la experiencia del usuario y el rendimiento del sistema se verán afectados.
- Punto Único de fallo: Si se da un error en el servidor, todos los clientes que dependen de él se van a ver afectados, es necesario implementar redundancia y tolerancia de fallos.
- Costo: En caso de tener una carga de trabajo muy alta y necesitar servidores potentes, el costo para adquirir y mantener la infraestructura puede ser significativamente alto en comparación con otras arquitecturas.

¿Se pueden combinar?

La naturaleza de ambas se centran en conceptos similares, la división de tareas de una aplicación en entes lógicos diferentes que interactúan entre sí para ofrecer un servicio íntegro y consistente, por lo que su combinación es común.
El sistema centralizado del cliente servidor permite tener un lugar centralizado donde administrar y gestionar las capas de negocio y de repositorio de datos de un programa de manera consistente entre todos los usuarios, de manera segura y sin exponer datos sensibles como accesos a bases de datos. La interfaz de usuario se puede desarrollar de manera totalmente separada al desarrollo de las capas de negocio, y existen protocolos de comunicación eficientes por internet (http) que permiten integrar fácilmente los clientes al servicio que necesitan. Similarmente la modularidad de las capas permite fácilmente reutilizar medidas típicas de redundancia o de mejoras de performance, como capas de caching, de manera sencilla e invisible para el sistema en general.
Los beneficios de reutilización, escalabilidad y modularidad se acumulan a hacer un sistema claramente definido y que permite su expansión y cambio de manera fácil y desacoplada, y las desventajas son muy similares entre ambas arquitecturas, por lo que su aplicabilidad es en general, la misma.
La arquitectura en capas podría ser considerada como una abstracción, con capas que muestran y solicitan, capas que procesan y capas que guardan información ubicadas físicamente o no en lugares diferentes y comunicadas por un medio, y la arquitectura cliente-servidor un implementación física de esto mismo.

Bibliografía Referenciada

Qué es la arquitectura en capas, ventajas y ejemplos.

Cliente-servidor - Wikipedia, la enciclopedia libre

Arquitectura en Capas

Arquitectura basada en capas. – Blog de Juan Peláez en Geeks.ms.

Acosta Gonzaga, E., Álvarez Cedillo, J. A., & Gordillo Mejía, A. (2006). Arquitecturas en n-Capas: Un Sistema Adaptivo. Polibits, (34), 34-37.

Estilo de arquitectura de n niveles - Azure Architecture Center | Microsoft Learn