El modelo Cliente/Servidor.

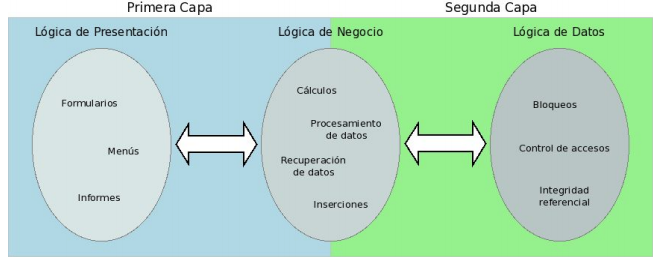
Es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Las aplicaciones Clientes realizan peticiones a una o varias aplicaciones Servidores, que deben encontrarse en ejecución para atender dichas demandas. Una disposición muy común son los sistemas multicapa en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes computadoras aumentando así el grado de distribución del sistema.

Arquitecturas de dos capas.

Uno de los objetivos de las aplicaciones de 2 capas es separar la lógica de acceso a los datos de lo que es la interfaz de usuario y trasladarla al servidor. Habitualmente, se implementan servicios como procedimientos almacenados en el sistema gestor de datos; con esto se pretende reducir la carga de los clientes y centralizar las operaciones comunes de acceso a los datos. En este modelo intervienen únicamente dos entidades: El Cliente y El Servidor.

El papel de cliente lo desempeña la aplicación final del usuario, que implementará todas las funciones necesarias como validaciones de datos y condiciones de recuperación. Y el servido lo desempeña el propio SGBD (Sistema de Gestión de Bases de Datos), el que se ocupa de todas las funciones correspondientes restantes. Un ejemplo podria ser una aplicación que se comunica directamente con la base de datos.

Modelo general:



Arquitecturas de tres capas.

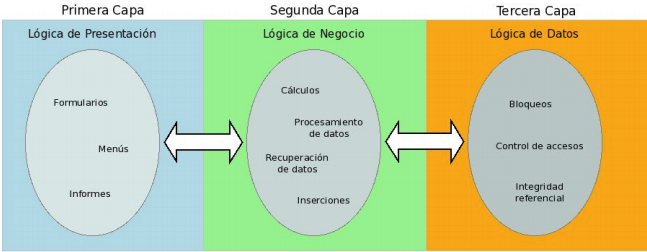
Disponen de tres tipos de nodos:

* Clientes que interactúan con los usuarios finales.
* Servidores de aplicación que procesan los datos para los clientes.
* Servidores de la base de datos que almacenan los datos para los servidores de aplicación.

Aporta una flexibilidad adicional en la construcción de aplicaciones cuando aumentan su complejidad. La arquitectura en tres capas lo que hace es dividir el sistema en tres partes diferenciadas, de tal forma que cada capa solo se comunique con la inferior. Esas tres capas se denominan:

* **Persistencia:** Esta capa se encarga de guardar los datos.
* **Negocio:** En esta capa se gestiona la lógica de la aplicación.
* **Presentación:** En esta capa se crea la interfaz de usuario.

Un ejemplo seria una aplicación de tienda en línea, donde a traves de la capa de presentación escoja un producto lo cual lleve a la transacción o procesamiento de datos y posteriormente a la base de datos para registrar la transacción.



Arquitectura de tres capas con tecnología de monitoreado en el procesamiento de transacciones.

Esta arquitectura, la más común en la actualidad, está compuesta por las siguientes capas:

* Presentación: comprende la lógica para manejar la interacción entre el usuario y la aplicación.
* Fuentes de Datos: tiene que ver con la comunicación con otros sistemas que llevan a cabo tareas en nombre de la aplicación. Monitores de transacciones, otras aplicaciones, sistemas de mensajería, pero generalmente base de datos.
* Lógica de Dominio: es la funcionalidad específica que la aplicación debe hacer para el dominio de trabajo. Los cálculos sobre la base de las entradas y los datos almacenados, la validación de los datos de entrada o determinar la fuente de datos implicada son ejemplos de funcionalidades en la capa de dominio.

**Transacciones de Negocio y de Sistema**

Las transacciones de sistema están soportadas por bases de datos y monitores de transacciones. Se componen de un grupo de comandos SQL delimitados por instrucciones para abrir y cerrar la transacción.

* Atomicidad: cada paso en la secuencia de acciones de la transacción debe completarse satisfactoriamente o todo lo realizado hasta el momento debe deshacerse.
* Consistencia: los recursos del sistema deben estar en un estado consistente tanto al principio como al final de la transacción.
* Aislamiento: el resultado de una transacción no deber ser visible para el resto de transacciones hasta que no se haya completado con éxito.
* Persistencia: el resultado de una transacción exitosa debe ser permanente.

Un ejemplo serial el sistema que tiene un banco para validad la información del cliente y aceptar y procesas transacciones.