

TEMA 1. ARQUITECTURA MULTICAPA

Indice

- Introducción
- Organización en capas
- Distribución en capas
- Lógica de negocio
- Presentación: arquitectura multicanal
- Fuentes de información: integración de fuentes heterogéneas.

1. Introducción

Arquitectura de una aplicación. Distintas partes que conforma nuestra aplicación y como interactúan entre ellas. La organización usual más típica es en capas. La arquitectura en Capas se basa en un patrón arquitectónico (layering).

Podemos dividir las capas en partes más pequeñas. Esto es lo que se denomina **componente**.

Patrón: Solución que sea eficaz y reusable ante un problema que se repite una y otra vez a la hora de realizar el diseño. Existen 3 tipos de patrones: de análisis, de diseño y de arquitectura.

Aplicación Corporativa:

Tiene las siguientes características.

- o Información persistente y en gran volumen (típicamente, bases de datos)
- o Distribuida (geográficamente Ξ separación física, trabajo simultáneo)
- o Múltiples usuarios concurrentes
- o Múltiples frontales de acceso Ξ distintas formas en las que el usuario puede interactuar con la aplicación.
- o Integración con otros sistemas.
- o Lógica de negocios compleja Ξ las acciones que debe realizar la aplicación son complejas (las funcionalidades lo son)
- o Seguras Ξ deben tener mecanismo para asegurar la integridad.

Ejemplo: Aplicación que gestione el stock de una empresa (gran empresa con un número importante de productos)

Aplicación no corporativa : Corporativa de textos

2. Organización en capas

Reflejada en el patrón layering. Útil en aplicaciones complejas como Networking (ftp, tcp, ip, ethernet) o Sistemas operativos (oscall, drivers, cpu instrucciones)

Ventajas

- o Independencia de capas.
- o Fácil sustitución de servicios con implementación diferentes.
- o Fácil reutilización por capas superiores.

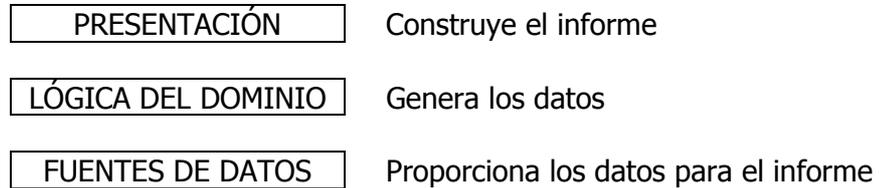
Inconvenientes

- o No se encapsula todo
- o Se degrada el rendimiento a la hora de la ejecución de la aplicación.

Principales capas:

- Presentación: Encargada de interactuar con el usuario, es la capa encargada de generar el interfaz de diálogo con el usuario. Trata interacción entre el usuario y el software
 - Tendencias: clientes pesados (API de java) vs clientes vuelo (ventanas)

- Lógica del dominio: Mecanismo / funciones que tiene que realizar nuestra aplicación, el código de la aplicación
- Fuente de datos: Encargada de acceder, recuperar, almacenar los datos, normalmente de otros sistemas como bases de datos (aplicación Legacy ≡ Legacy System)



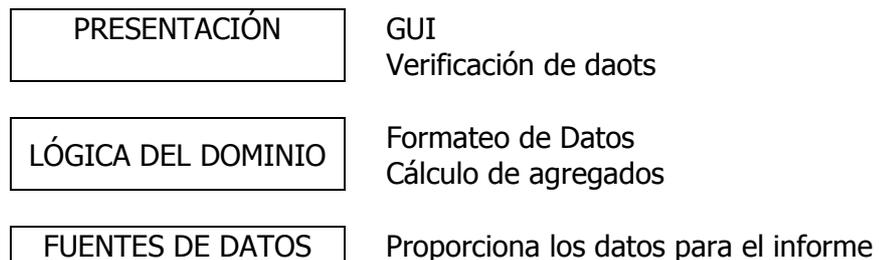
3. Distribución en Capas

Los Sistemas Complejos pueden incrementarse el número de capas, por ejemplo, en la construcción de una arquitectura basada en componentes

Los sistemas sencillos se dan en lógica de negocio contenida en la capa de presentación.

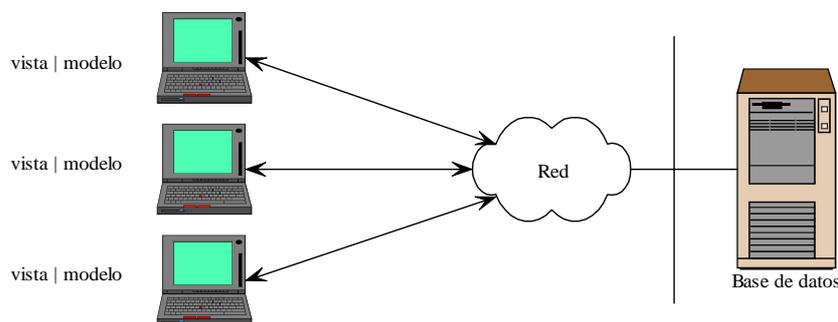
Problemas: No muy recomendables por su poca escalabilidad y por que no son fácilmente modificables. Se hace muy rápidamente, pero una vez desarrollado es difícil de mantener y modificar. Se usa en Internet en portales pequeños pero no en grandes aplicaciones.

Ejemplo: JSP + Java + Oracle (Presentación + Negocio + BBDD)



Distribución capas lógicas en elementos físicos

Distribución de capas en 2 niveles:



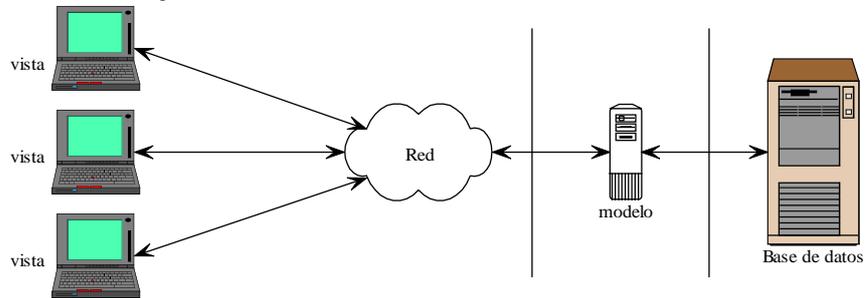
2 capas en un mismo PC: Presentación y lógica de negocio se encuentra en el cliente.

Inconvenientes:

- Consumo elevado de ancho de banda
- No reutilizar conexiones a la base de datos
- Difícil de mantener
- Difícil de escalar

El porcentaje entre abrir y cerrar conexiones. Si podemos mantener, usamos las ya abiertas. Aquí en este modelo, no se puede ya que está repartida en varios clientes.

Distribución de capas en 2 niveles:

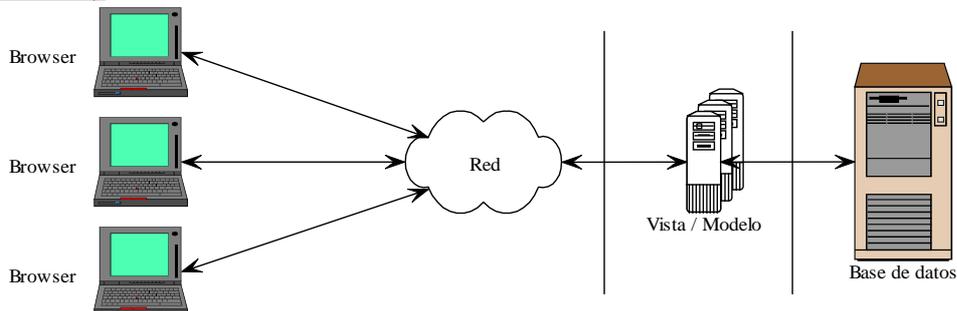


- Reducción del ancho de banda consumido
- Posibilitamos la reutilización
- Conexiones a la base de datos facilitan el mantenimiento

Pero dado que el modelo está instalada en un único punto, implica que hay una desventaja clave: escalabilidad en la vista.

Es más fácil de mantener y, sobre todo, escalar, todos los PCS clientes poseen un navegador que ejecuta la aplicación. La vista se genera en único PC y en los browsers solo se visualiza. Si, por el rendimiento, la aplicación necesita ser escalada, se realiza la siguiente distribución

Distribución de capas en 2 niveles con presentación web (escalado de servidores)



El software usado en bases de datos deben disponer de capacidades de sincronización de datos para mantener su integridad.

Tablas replicadas (mismo dato/ mismo momento)

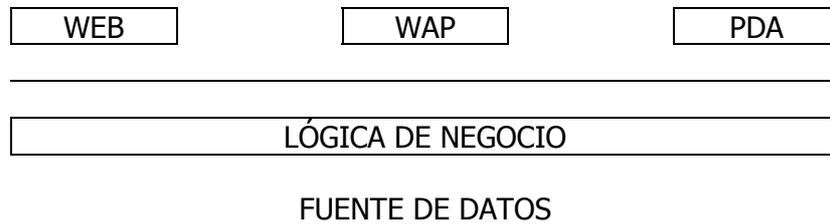
Igualmente en cada instancia del modelo y de la vista que se esté ejecutando deben estar los datos sincronizados.

También debe existir un balanceo de carga. Así al aumentar más hardware aumentamos el rendimiento de la aplicación

Capa de presentación: Arquitectura Multicanal.

La arquitectura consiste en la posesión de distintas formas de presentación a los usuarios funcionando de forma simultánea en el sistema.

Ejemplos: Web, PDA, móviles poseen una presentación adaptada a cada uno.



Dificultades: Los datos pueden tener diferentes representaciones en cada capa (compatibilidad).

Solución PASO DE DATOS:

Se usa para solucionar el problema un lenguaje abstracto XML. Especificamos lo que vamos a devolver, especificamos los datos que se pasan de forma abstracta. Evita las dependencias con la lógica de negocio.

Solución Web SERVICE:

Ofrecen servicio a través de un web (métodos para realizar cualquier operación definida y que devuelven cualquier resultado) Conjunto de métodos de la lógica de negocio.

Cuando introduzcamos nuevas capas de presentación (funcionalidades) es mayor la ganancia con web service y xml. En principio XML y web service suponen más líneas de código.

Solucion XLS

Hojas de estilo en las que estableces la forma de la interfaz de forma independiente del código. El formato recoge el fichero XML donde están los formatos que se aplican definitivamente a los datos.

Lógica del Dominio:

- Modelo del dominio (Domain Mode)
- Módulo Tabla (Table Module)

La lógica de negocio es la lógica del dominio. Contiene las funcionalidades que se ofrecen al usuario y presenta dos posibles formas (las anteriores). Por ejemplo, la lógica del dominio sería Agregar producto, Eliminar producto y Modificar producto.

Mapeamos cada fila de la tabla a un objeto / instancia de tipo producto.

Módulo TABLA

PRODUCTO
ID
Nombre
Precio

No se mapea una fila de la tabla, sino directamente toda la tabla.

La lógica debe manejar cada producto (agregar, modificar,...) y los metodos para "llevar" la tabla a la base de datos. Esta forma es más compleja (carga más el sistema, debido al tamaño de los datos)

Producto DAO (Método de acceso a datos)

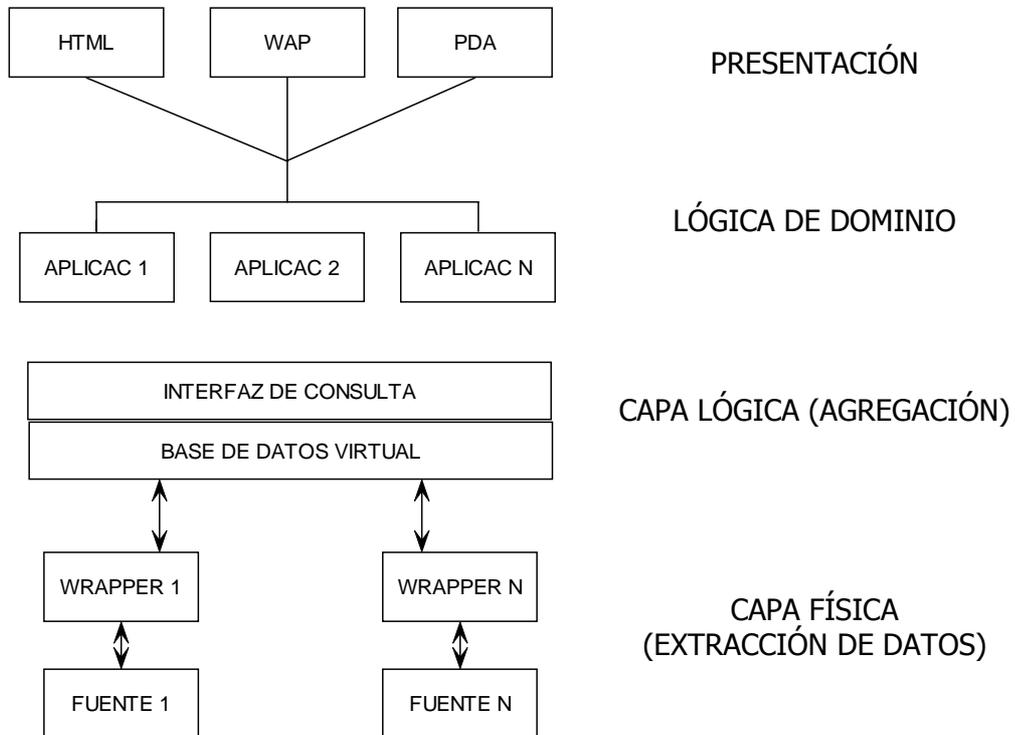
FUENTE DE DATOS: HETEROGENEIDAD.

El problema a solucionar en la capa de datos es la heterogeneidad de las fuentes de datos. Pueden existir varias fuentes que interactúen en nuestro sistema:

- o Bases de datos (casi todas relacionales)

- Documentos XML
- Systems "Legacy"
- Fuentes no estructuradas (ejemplo: paginas web donde la aplicación consulta otras páginas web para recopilar información que nos interese)
- Sistemas de directorio (LDAP)

FUENTES DE DATOS: INTEGRACIÓN FUENTES HETEROGENEAS



La base de datos virtual simula una base de datos para toda la inferior. La aplicación trata igual a todas las fuentes. Los detalles de las fuentes se resuelven en los wrappers. Las fuentes devuelven los datos a la base virtual.

La aplicación lanza la consulta a la interfaz. La base virtual la lanza a cada wrapper y este sabrá como lanzarla a cada fuente.

La base, una vez tiene los datos de vuelta, los empaqueta en una tabla que es recorrida por la aplicación.

Cuanto más funcionalidades hayamos implementado en la base, más cosas podremos realizar (funcionalidades SQL)