Arquitectura

1.- Aplicaciones Web

Contenidos

Definición

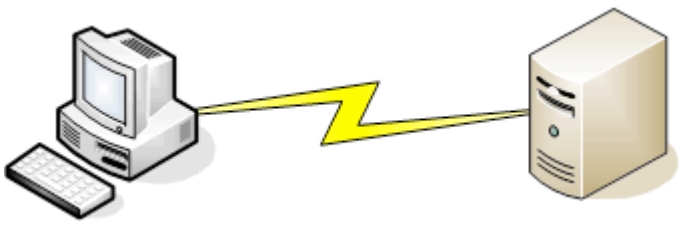
Lo que distingue una **aplicación Web** de una mero sitio Web reside en la posibilidad que ofrece al usuario de actuar sobre la **lógica de negocio en el servidor**. Por lógica de negocio se entiende un conjunto de procesos que implementan las reglas de funcionamiento de la aplicación Web. Por ejemplo, en el caso de una tienda Web, la lógica de negocio son los procesos que implementan el procedimiento de compra: selección de productos, carro de la compra, confirmación de compra, pago, seguimiento de la entrega,...

Pero lo importante es que a los **usuarios** se les ofrece la posibilidad de **actuar sobre la lógica de negocio**. Por ejemplo, en el caso anterior, el usuario puede registrarse con lo que se crea un perfil personalizado, se hace un seguimiento de sus compras a lo largo de una sesión (carro de la compra) pero también a través de sesiones (perfil de comprador), con sus acciones se modifican stocks, etc.,...

Ejemplo: no se considera aplicación Web un sitio de búsqueda que únicamente ofrece un formulario para introducir los criterios, realiza un consulta a base de datos y devuelve los resultados obtenidos, sin que esto cause cambios en el estado interno del buscador.

Arquitectura clásica

La arquitectura de una aplicación Web es similar a la de un sitio Web, se basa en el modelo **Cliente/Servidor**. Como en el caso del sitio Web, tenemos el navegador en la parte cliente, el servidor Web en la parte del servidor y una conexión de red. Pero en las aplicaciones Web hay que considerar que existe una **lógica de negocio sensible a las interacciones del usuario**.



Arquitectura básica Cliente/Servidor

Originalmente la Web únicamente contaba con **contenidos estáticos**. La necesidad de ofrecer servicios más sofisticados a través de este medio llevó a soluciones tecnológicas que permitiesen cierto grado de interacción con el servidor más allá de solicitar una página en concreto. Los **CGIs** fueron las primeras opciones en este sentido. Permitían interactuar con código ejecutable en el servidor desde el navegador. A través de la URL solicitada por el navegador se pasaban los parámetros de entrada para el código ejecutable y este devolvía una página HTML con la respuesta. Este mismo método se utiliza en tecnologías similares como ISAPI, NSAPI o Java

Servlets.

> Por ejemplo: http://mm.musicbrainz.org/newsearch.html?**limit=25&table=Track&search=hit**

A medida que la sofisticación de los servicios ofrecidos vía Web a ido aumentando, se ha ido añadiendo, al menos a nivel lógico y no necesariamente a nivel físico, **nuevos elementos en la arquitectura de las aplicaciones Web** (por ejemplo el servidor de aplicaciones, bases de datos,...). Las nuevas disposiciones de elementos se han basado en patrones arquitectónicos del contexto de las aplicaciones distribuidas.

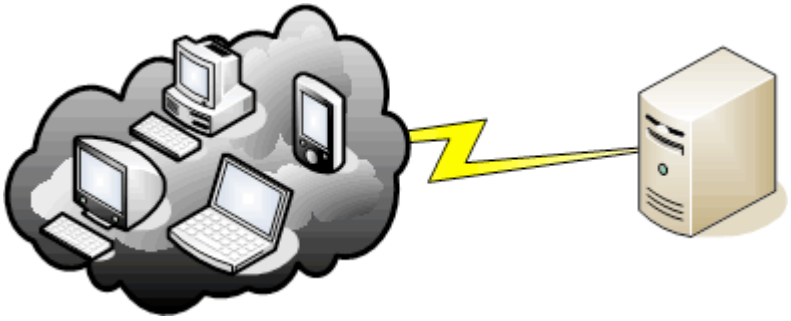
Patrón arquitectónico: un patrón arquitectónico es cualquier esquema relativo a la configuración de un sistema en su conjunto, no relativo a una parte de él. Por lo tanto aplica a una escala en la que la unidad de organización es mucho mayor que la clase y más próxima a los programas o ejecutables.

2.- Arquitectura de aplicaciones web

Arquitectura actual: Tres niveles

En las aplicaciones distribuidas, partiendo del modelo Cliente/Servidor, al aumentar la complejidad de los procesos se acaba con el denominado problema del "**Cliente Pesado** ". Al poner la mayor parte del código necesario para llevar a cabo los procesos en el cliente, este debe descargar del servidor los datos necesarios para llevarlos a cabo. Esto es muy ineficiente por dos razones principales:

- La red sufre una gran carga debido a las múltiples descargas de cada uno de los clientes.
- La gran dependencia en el rendimiento del hardware en el lado del cliente.



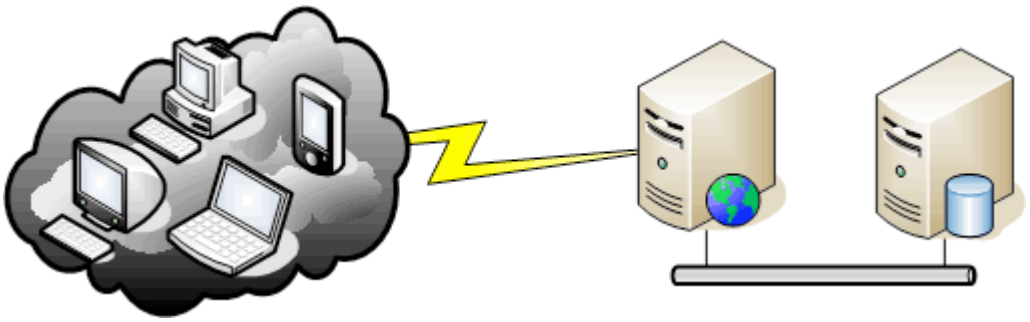
Cliente/Servidor con clientes heterogéneos

Las arquitectura en tres niveles **divide la funcionalidad** para optimizar el uso de recursos. Se consiguen soluciones mucho más **flexibles y escalables**. Los tres niveles son:

- **Cliente**
  Contiene los componentes de usuario que son únicos para cada uno de ellos. Esto es la lógica de aplicación específica del usuario y la interfaz.
- **Aplicación**
  Constituye un entorno multiusuario y mantiene las partes compartidas de la aplicación. Las operaciones con un uso intensivo de datos deben ejecutarse en este nivel. También es el

punto donde se puede llevar a cabo la coordinación de transacciones (operaciones de múltiples usuarios).

- **Almacenamiento**
  Es nivel de la base de datos. Se especializa en dar un servicio de persistencia a los datos de la aplicación y permite manejar grandes volúmenes de ellos.



Arquitectura en tres capas: cliente, aplicación y almacenamiento

Al mantener buena parte del código en el nivel intermedio, **la aplicación se aísla de la interfaz de usuario y de la base de datos**. Por lo tanto, se pueden hacer cambios en el código en el nivel de aplicación sin que esto afecte al resto. Además, los **datos** están **centralizados** y fácilmente accesibles sin necesidad de moverlos completamente hasta el cliente.

Este modelo se utiliza ampliamente en el entorno Web. Se incorpora la figura del **servidor de aplicaciones** que permite que el sistema gestione la lógica de negocio y el estado. Al ser una aplicación Web, la interfaz con el cliente sigue siendo básicamente **HTTP** y por lo tanto el servidor de aplicaciones incorpora un servidor Web o utiliza uno externo.

La división de la aplicación en niveles supone la necesidad de establecer interfaces entre ellas. Si además se quiere construir una arquitectura independiente de los fabricantes la mejor opción es utilizar elementos que implementen **interfaces estándar**.

Para la interfaz entre el servidor de aplicaciones y la base de datos las opciones son múltiples, dependiendo del lenguaje de programación (Java, C, PHP, VisualBasic,...), el tipo de base de datos (relacional, XML,...), la base de datos concreta (MySQL, Oracle, eXist,...), etc. En cualquier caso con cada configuración hay más de una opción y lo normal es plantearse la elección tras definir la BD o el lenguaje de programación a utilizar.

Por lo que respecta a la interfaz entre el cliente y el servidor de aplicaciones, la opción básica sigue siendo HTTP. Existen otras opciones que permiten un mayor grado de interactividad entre el código en el cliente y en el servidor. Estas opciones permiten definir tres arquitecturas de cliente Web [Conallen99].

Cliente Web ligero

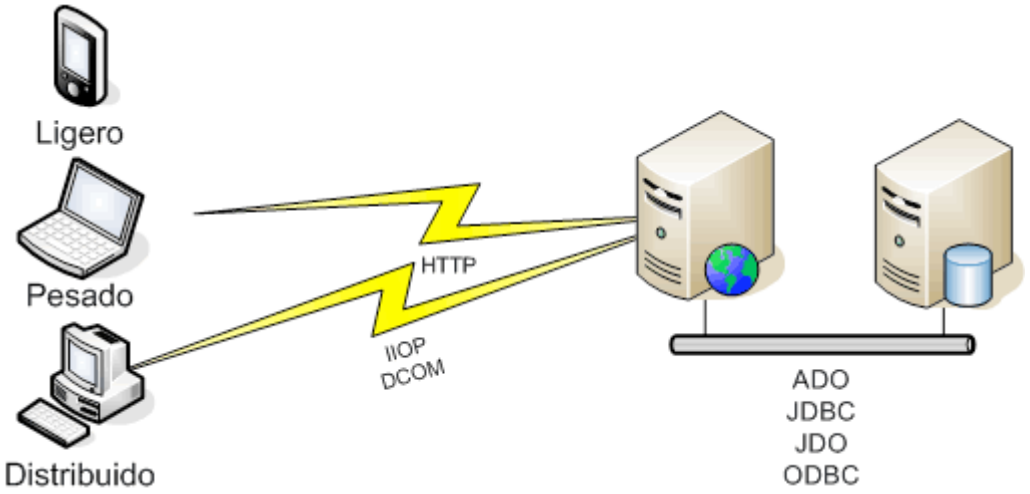
Toda la lógica de negocio se mantiene en el servidor. Esto permite tener clientes que sólo requieren el uso de un **navegador Web** sencillo. Básicamente **HTML**, **formularios** para la entrada de datos y **lenguajes de script** (VBScript, JavaScript,...) para el control previo de los datos introducidos, p.e. validar un NIF. Ésta es la solución que supone una mayor carga de proceso y comunicaciones, la interacción entre el cliente y el servidor Web se basa en una sucesión de pregunta/respuesta síncronas.

Cliente Web pesado

Parte de la **lógica de negocio se realiza en el cliente**. Applets, ActiveX,... para el encapsulado del código que permite su ejecución en el navegador Web del cliente. Para la comunicación con el servidor se sigue utilizando HTTP.

Cliente Web distribuido

La aplicación se desarrolla en base al uso de **objetos distribuidos**. Para la comunicación entre objetos distribuidos se utiliza CORBA/IIOP, DCOM o Java RMI. Esta arquitectura supone la mayor carga de proceso y comunicaciones.



Interfaces en la arquitectura de tres niveles según arquitectura cliente Web

La elección de la arquitectura de cliente Web debe hacerse teniendo en cuenta los aspectos de riesgo para el rendimiento comentados para el problema del "Cliente Pesado". La capacidad de la red de absorber la transferencia de datos hacia el cliente y la configuración hardware de éste.

Otras opciones arquitectónicas

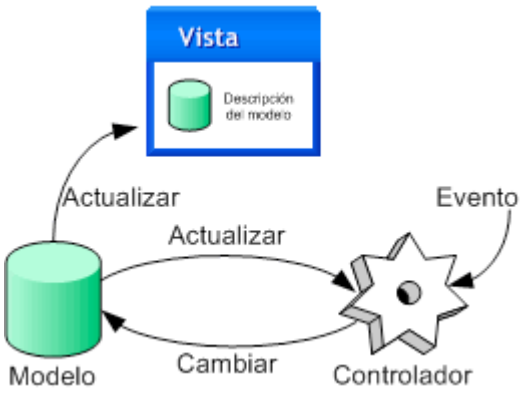
Cuatro niveles

Partiendo del modelo en tres niveles, se han definido arquitecturas derivadas de este patrón.

La arquitectura en cuatro niveles nació como una evolución de la de tres para aplicaciones con interfaces de usuario complejas. Para ello integra el **patrón de diseño MVC** (Modelo-Vista-Controlador). Los patrones de diseño se aplican sobre un nivel de granularidad más pequeño que los arquitectónicos, sobre conjuntos de clases. El patrón de diseño MVC define conjuntos de clases llamadas modelos, vistas y controladores respectivamente:

- **Modelo**: las clases de tipo modelo representan los objetos del dominio de la aplicación.
  - Ejemplo: producto, usuario, cesta de la compra,...
- **Vista**: las de tipo vista son elementos de presentación de información a los actores de la aplicación.
  - Ejemplo: compra actual,...

- **Controlador**: las de tipo controlador gestionan los cambios del modelo en respuesta a las acciones de los actores.
    - Ejemplo: añadir producto a la cesta de la compra,...



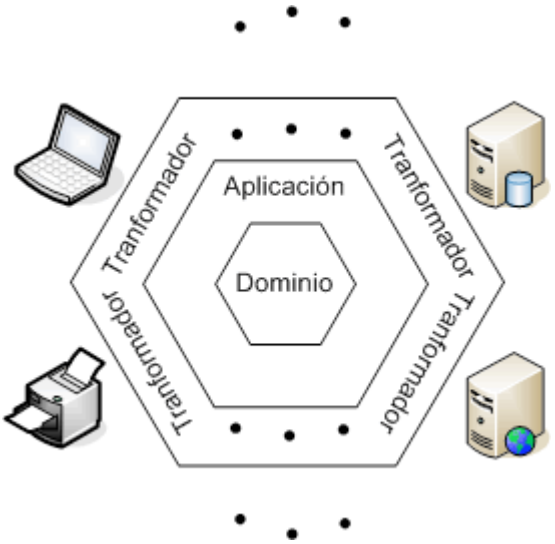
Patrón de diseño Modelo-Vista-Controlador

Al incorporar el patrón MVC a este patrón arquitectónico se obtienen cuatro niveles:

- **Vista**: contiene los elementos de interfaz que presentan la información. Se asocian a las vistas del patrón MVC, aunque este nivel también puede incluir elementos de tipo controlador sencillos.
    - Cuando se aplica al entorno Web se trata de páginas HTML (posiblemente con scripts), JSP, ASP,...
- **Controlador**: se trata de los componentes que intermedian entre los elementos de la interfaz y los del modelo del dominio. Son responsables del flujo de aplicación y controlan la navegación entre vistas.
    - En el entorno Web son CGIs, Servlets,...
- **Dominio**: son los objetos que modelan el dominio de la aplicación.
    - Hablando del contexto Web son EJBs o simples clases Java, C++,...
- **Infraestructura**: incluye el almacenamiento en bases de datos.

Arquitectura hexagonal

En esta arquitectura no se define una parte frontal hacia el usuario (front-end) y una parte trasera (back-end) como en los patrones arquitectónicos anteriores. Por el contrario, se considera que la aplicación se encuentra situada en el centro y que interacciona con el resto a través de una serie de transformadores que adaptan los mensajes de la aplicación al elemento con el cual se interacciona en cada caso. Esos elementos pueden ser terminales de usuario, listados, bases de datos, otras aplicaciones,...



Patrón arquitectónico hexagonal [Cockburn97]

Esta arquitectura se adapta muy bien a las soluciones basadas en el uso de XML, que proporciona un formato común de interacción de la aplicación con el exterior, y XSL, que facilita la implementación de los transformadores.

Referencias

[Conallen99] Jim Conallen: “Building Web Applications with UML”. Addison Wesley, 1999. ISBN: 0-201-61577-0 http://safari.awprofessional.com/0201615770

[Cockburn97] Alistair Cockburn: "HexagonalArchitecture". http://c2.com/cgi/wiki?HexagonalArchitecture