élogos

ARQUITECTURA MODELO VISTA CONTROLADOR (MVC)

élogos

ÍNDICE

ARQUITECTURA MODELO VISTA CONTROLADOR (MVC)

1. Modelo, Vista, Controlador

Qué es el MVC

El Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

El patrón MVC se aplica en la construcción de aplicaciones Web, donde:

- La vista se encarga de la generación de las páginas Web utilizadas para presentación de datos e interacción con la aplicación
- El modelo se encarga de implementa la lógica de negocio, incluyendo el acceso a los datos

El controlador recibe las peticiones desde la capa cliente y es el responsable de coordinar las acciones del resto de bloques.

Características del MVC

El patrón fue descrito por primera vez en 1979 por Trygve Reenskaug, que trabajaba entonces en Smalltalk, en los laboratorios de investigación de Xerox.

La implementación original está descrita a fondo en Programación de Aplicaciones en Smalltalk-80(TM):

¿Cómo utilizar Modelo Vista Controlador?

Modelo

Ésta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de éstos y permite derivar nuevos datos, por ejemplo, no permitiendo comprar un número de unidades negativo, calculando si hoy es el cumpleaños del usuario o, los impuestos o importes totales en un carrito de la compra.

Vista

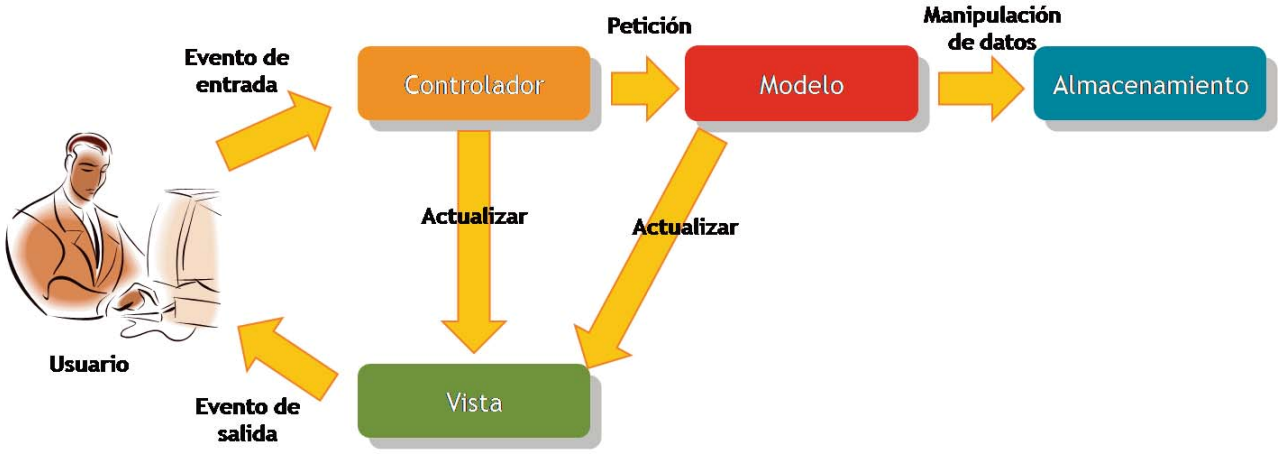
Éste presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.

Controlador

Éste responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

Flujo MVC

Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo que sigue el control generalmente es el siguiente:



Usuario

El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.)

Controlador

El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.

Modelo

El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.

Vista

El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se reflejan los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra).

El controlador no pasa objetos de dominio (el modelo) a la vista aunque puede dar la orden a la vista para que se actualice.

Nota: en algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista.

Tareas del Modelo, Controlador y Vista

A continuación veremos las responsabilidades y tareas del Modelo, del Controlador y la Vista

El Modelo es el responsable de:

- Acceder a la capa de almacenamiento.
- Define las reglas de negocio(la funcionalidad del sistema).
- Lleva un registro de las vistas y controladores del sistema.
- Notificará a las vistas cambios que en los datos pueda producir un agente externo.

El Controlador es el responsable de:

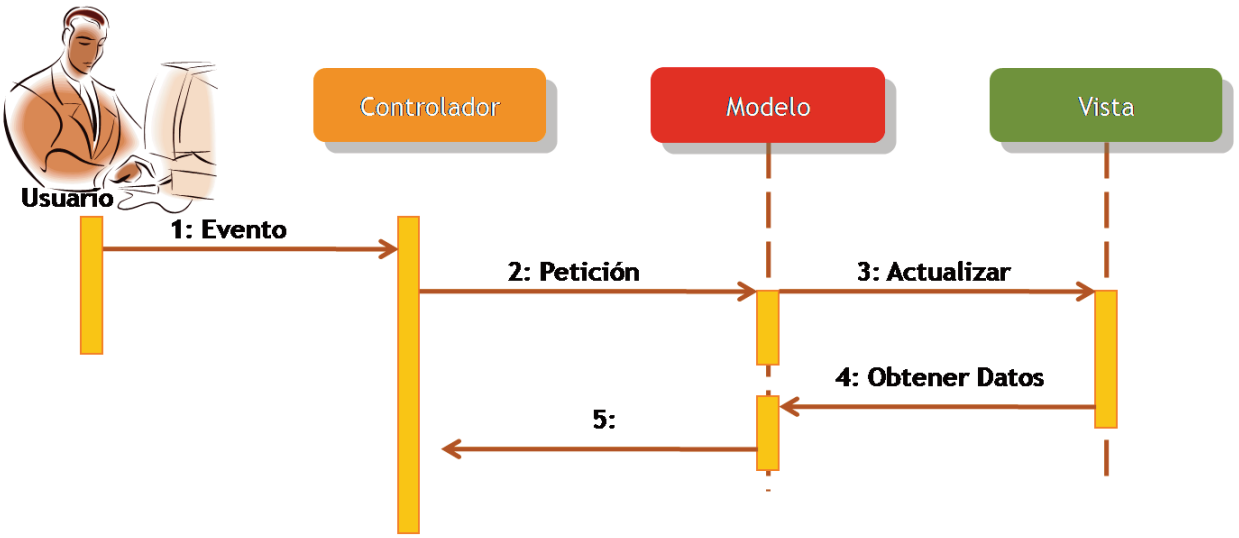
- Recibir los eventos de entrada.
- Contener reglas de gestión de eventos del tipo “SI Evento Z, entonces Acción W”

Las Vistas son responsables de:

- Recibir datos del modelo y mostrarlos al usuario.
- Dirigir todas las peticiones desde la capa cliente al controlador.
- Dar el servicio de “Actualización()”

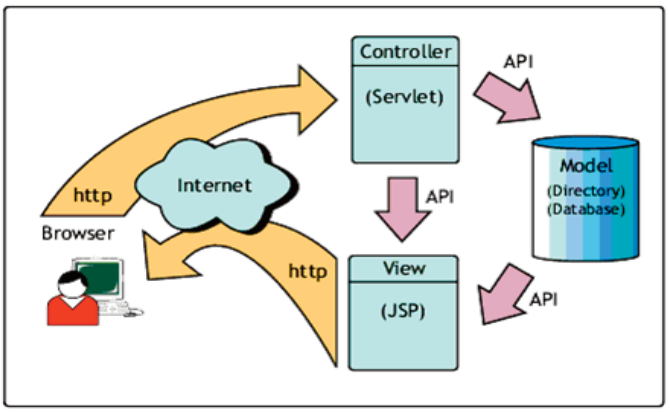
Ejemplo de MVC como modelo pasivo

Un ejemplo de MVC con un modelo pasivo (aquel que no notifica cambios en los datos) es la navegación Web, que responde a las entradas del usuario, pero no detecta los cambios en datos del servidor.



Arquitectura MVC

Los patrones de presentaciones del catálogo J2EE están basados en la arquitectura MVC. El MVC se aplica en proyectos de desarrollo de software en un esfuerzo por separar los datos de la aplicación de la presentación de los mismos.



Esta separación permite a la interfaz o visualización, adoptar diferentes formas con una ligera modificación del código. Por ejemplo, utilizando un patrón MVC, una interfaz de usuario puede presentarse tanto como una página HTML(para navegadores Web) como una página WML(para dispositivos móviles)m dependiendo del dispositivo que solicite la página. El controlador reorganizaría la fuente de la solicitud y plasmaría la visualización adecuada a los datos de la aplicación.

Orígenes de MVC

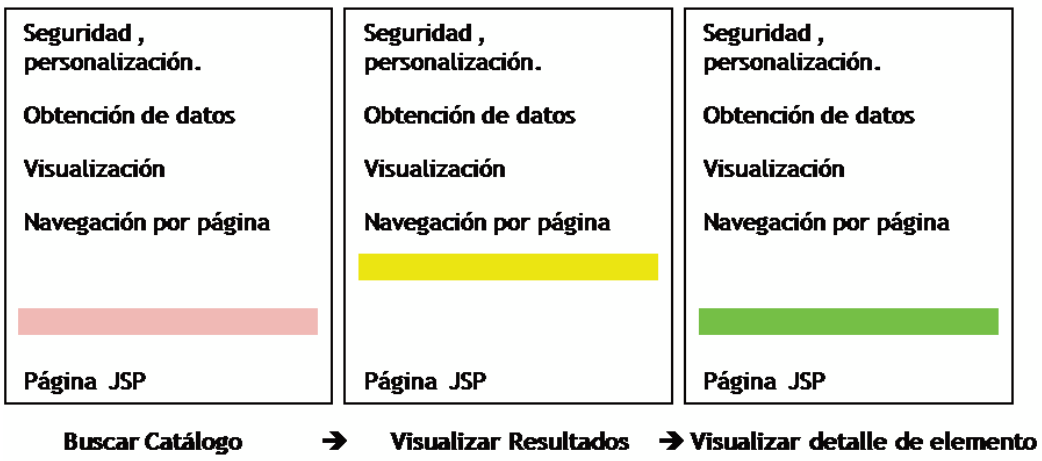
Por ejemplo, imagina una interfaz formada por muchos controladores de usuario distintos. Estos controladores contienen los datos, las instrucciones en cuanto al formato y el código que lanza un evento cuando se activa el controlador.

Esto genera una interfaz para una plataforma específica acoplada al propio código de la aplicación. Aplicando el patrón MVC y separando cada uno de sus componentes, la interfaz de usuario se convierte en algo reducido, fácilmente conectable y totalmente transferible entre plataformas. Java Swing API es un buen ejemplo de ello.

Aplicación de catálogo Sin MVC

Imaginemos una sencilla aplicación Web que muestra las páginas de un catálogo. Normalmente, hablaríamos de una búsqueda de página, una página de resultados y después la página que demuestra en detalle un elemento.

Aplicación de catálogo Sin MVC



Cada página tiene la responsabilidad de autentificar el usuario, recuperar las preferencias del mismo, obtener los datos requeridos y, finalmente, controlar la navegación por la página.

Aplicación de catálogo Con MVC

Observando la aplicación anterior, resulta fácil comprobar la existencia de bastante código redundante para visualizar cada página. Esto no solo introduce la posibilidad de errores, sino que ata la aplicación a la presentación incluyendo muchas funcionalidades no relacionadas con la presentación dentro del código de la misma.

Al aplicar un patrón MVC a esta aplicación, las funciones más comunes van a un controlador en el servidor. Ahora, el código de presentación es el único responsable de representar los datos de la aplicación en el formato más adecuado para un dispositivo en particular (normalmente un navegador Web).

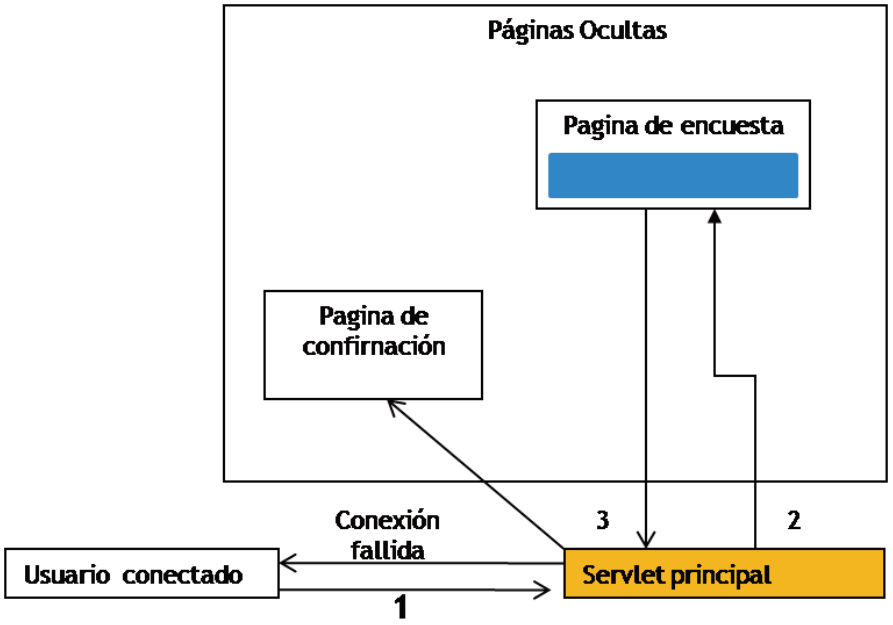
MVC en acción

Como ejemplo de patrón MVC, exponemos el diagrama de bloques de una sencilla aplicación Web, consistente en obtener información sanitaria de cliente para después almacenarla en una base de datos.

Además de recoger los datos, la aplicación solicitará al usuario conectarse al sistema antes de acceder a cualquiera de sus páginas.

Este ejemplo ilustra el beneficio que se obtiene al usar la arquitectura MVC, como centralizar la seguridad de la aplicación otorgando al usuario un único punto de acceso a la aplicación y estandarizar y compartir la conexión de la base de datos usando un mecanismo de asociación de conexiones en el servidor de la aplicación.

Arquitectura Modelo Vista Controlador (MVC)



La aplicación comienza con una página de conexión y entonces pasa a un servlet principal que actúa como controlador. El servlet determinará si proceder con la página siguiente basándose en el éxito del procedimiento de conexión. Una vez se haya conectado el usuario, irá a una página de encuestas donde añadirá su información y la enviará.

Una vez más, el servlet requerirá la solicitud y llevará al usuario a la página siguiente. Si los datos se han grabado satisfactoriamente, se lleva al usuario a una página de confirmación.