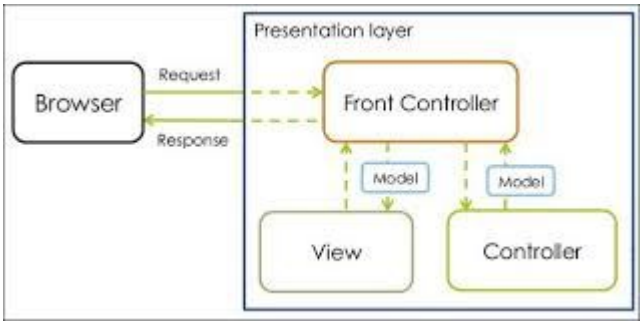
Modelo Vista Controlador :

El propósito de este patrón es simplificar la implementación de aplicaciones de acuerdo a las peticiones de los usuarios y los datos a desplegar" . El Model-View-Controller o Modelo–Vista–Controlador (MVC) es un patrón de diseño de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos de forma que las modificaciones al componente de la vista, o a cualquier parte del sistema puedan ser hechas con un mínimo impacto en el componente del modelo de datos o en los otros componentes del sistema. Este patrón cumple perfectamente el cometido de modularizar un sistema.



El Modelo Vista Controlador (MVC) surge de la necesidad de crear software más robusto con un ciclo de vida más adecuado, donde se potencie la facilidad de mantenimiento, reutilización del código y la separación de conceptos. El patrón MVC fue una de las primeras ideas en el campo de las interfaces gráficas de usuario y uno de los primeros trabajos en describir e implementar aplicaciones software en términos de sus diferentes funciones. Controlador Vista Modelo 5 Fue descrito por primera vez en 1979 por Trygve Reenskaug, entonces trabajando en Smalltalk en laboratorios de investigación de Xerox.

Analogía del patrón de diseño MVC:

Imagina que estás construyendo una casa. En este caso, la casa representa tu aplicación de backend. MVC divide las responsabilidades en tres componentes principales:

1. Modelo (Model): El modelo es como los planos de la casa. Contiene toda la lógica de negocio y los datos de la aplicación. Es responsable de manejar la interacción con la base de datos y de gestionar los datos de la aplicación. En la analogía de la casa, el modelo serían los planos detallados que indican cómo se construye cada parte de la casa y qué materiales se utilizan.
2. Vista (View): La vista es como la apariencia de la casa. Es responsable de mostrar la información al usuario de una manera que sea comprensible y atractiva. En la analogía de la casa, la vista serían los acabados, muebles y decoraciones que hacen que la casa sea acogedora y funcional.

3. Controlador (Controller): El controlador es como el arquitecto que supervisa la construcción de la casa. Se encarga de manejar las solicitudes del usuario, procesar la entrada y coordinar las acciones entre el modelo y la vista. En la analogía de la casa, el controlador sería el encargado de coordinar a los trabajadores de la construcción, asegurándose de que sigan los planos y de que el resultado final cumpla con las expectativas del cliente.

En resumen, MVC en el backend divide la aplicación en tres partes: el modelo para la lógica y los datos, la vista para la presentación de la información, y el controlador para manejar las solicitudes y coordinar la interacción entre el modelo y la vista, de manera similar a cómo se construye una casa con una separación clara entre los planos, la apariencia y la coordinación de la construcción.

Componentes :

- Modelo: Representa los datos que el usuario está esperando ver, en algunos casos el Modelo consiste de Java Beans.

- Vista: Se encarga de transformar el modelo para que sea visualizado por el usuario, ya sea un archivo de texto normal o en una página Web (HTML o JSP) que el navegador pueda desplegar. El propósito de la Vista es convertir los datos para que el usuario le sean significativos y los pueda interpretar fácilmente; la Vista no debe trabajar directamente con los parámetros de request, debe delegar esta responsabilidad al controlador.

- Controlador: Es la parte lógica que es responsable de procesamiento y comportamiento de acuerdo a las peticiones (request) del usuario, construyendo un modelo apropiado, y pasándolo a la vista para su correcta visualización. En el caso de una aplicación Web Java en la mayoría de los casos el Controlador es implementado por un servlet.

Interacción de los componentes :

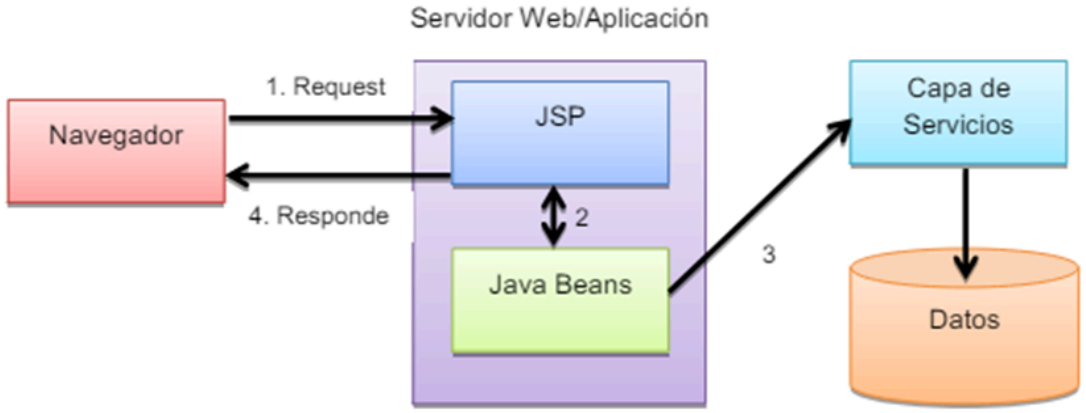
Se pueden encontrar muchas implementaciones de MVC, pero generalmente el flujo de datos se describe así:
1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.).

2. 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.

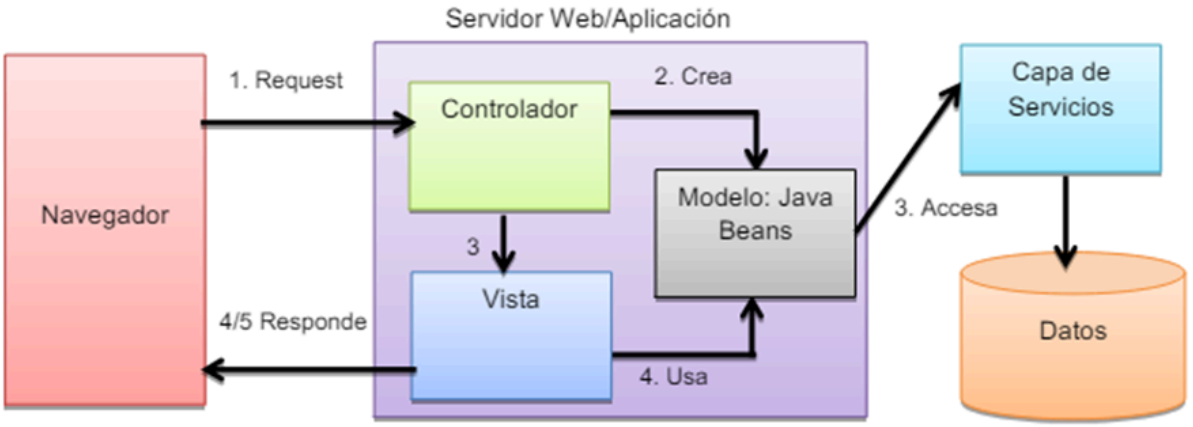
3. El controlador accede al modelo, actualizándose, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario. Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.

4. 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. El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, se podría utilizar el patrón Observador para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. 5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

Tipos de Patrones MVC Actualmente existen dos tipos de patrón MVC: MVC tipo 1: Las paginas JSP están en el centro de aplicación y contiene tanto la lógica de control como la de presentación. Este tipo de arquitectura funciona de la siguiente manera: el cliente hace una petición o una página JSP, se construye la lógica de la página, generalmente en objetos java y se transforma el modelo para ser desplegado una vez más.



MCV tipo 2: Aquí ya existe una clara separación entre el Controlador y el Vista, ya que ahora es directamente el Controlador quien recibe la petición, prepara el modelo y lo transforma para que sea desplegado en la vista. Esta arquitectura se utiliza para aplicaciones complejas.



MVC y bases de datos

Muchos sistemas informáticos utilizan un Sistema de Gestión de Base de Datos para gestionar los datos que debe utilizar la aplicación; en líneas generales del MVC dicha gestión corresponde al modelo. La unión entre capa de presentación y capa de negocio conocido en el paradigma de la Programación por capas representaría la integración entre la Vista y su correspondiente Controlador de eventos y acceso a datos, MVC no pretende discriminar entre capa de negocio y capa de presentación pero si pretende separar la capa visual gráfica de su correspondiente programación y acceso a datos, algo que mejora el desarrollo y mantenimiento de la Vista y el Controlador en paralelo, ya que ambos cumplen ciclos de vida muy distintos entre sí.

Uso en aplicaciones Web

Aunque originalmente MVC fue desarrollado para aplicaciones de escritorio, ha sido ampliamente adoptado como arquitectura para diseñar e implementar aplicaciones web en los principales lenguajes de programación. Se han desarrollado multitud de frameworks, comerciales y no comerciales, que implementan este patrón; estos frameworks se diferencian básicamente en la interpretación de como las funciones MVC se dividen entre cliente y servidor. Los primeros frameworks MVC para desarrollo web plantean un enfoque de cliente ligero en el que casi todas las funciones, tanto de la vista, el modelo y el controlador recaen en el servidor. En este enfoque, el cliente manda la petición de cualquier hiperenlace o formulario al controlador y después recibe de la vista una página completa y actualizada (u otro documento); tanto el modelo como el controlador (y buena parte de la vista) están completamente alojados en el servidor.

- Vista: la página HTML.
- Controlador: código que obtiene los datos dinámicamente y genera el contenido HTML.
- Modelo: la información almacenada en base de datos o en XML.

FrameWorks

Es un término utilizado en la computación en general, para referirse a un conjunto de bibliotecas utilizadas para implementar la estructura estándar de una aplicación. Todo esto es realizado con el propósito de promover la reutilización de código, con el fin de ahorrarse trabajo al desarrollador al no tener que reescribir ese código para cada nueva aplicación que se desea crear.

Existen varios FrameWorks para diferentes fines, algunos son orientados para aplicaciones web, otros para aplicaciones multiplataforma, sistemas operativos, etc.

Este determina la arquitectura de una aplicación, se encarga de definir la estructura general, sus particiones en clases y objetos, responsabilidades clave, así como la colaboración entre las clases objetos, esto evita que el usuario tenga que definirlo y se pueda enfocar en cosas específicas de su aplicación.

Los Frameworks utilizan un variado número de patrones de diseño, ya que así logran soportar aplicaciones de más alto nivel y que reutilizan una mayor cantidad de código, que uno que no utiliza dichos patrones. "Los patrones ayudan a hacer la arquitectura de los FrameWorks más adecuada para muchas y diferentes aplicaciones sin necesidad de rediseño".

Por esta razón es importante que se documenten qué patrones utiliza el FrameWorks para que los que se encuentren familiarizados con dichos patrones puedan tener una mejor visión y poder adentrarse en el FrameWorks más fácilmente. El patrón MVC es utilizado en múltiples Frameworks como:

- Java Swing, Java Enterprise Edition - XForms (formato XML estándar para la especificación de un modelo de proceso de datos XML e interfaces de usuario como formularios web).

- GTK+ (escrito en C, toolkit creado por Gnome para construir aplicaciones gráficas, inicialmente para el sistema X Window)

- Google Web Toolkit, Apache Struts, Ruby On Rails, entre otros.