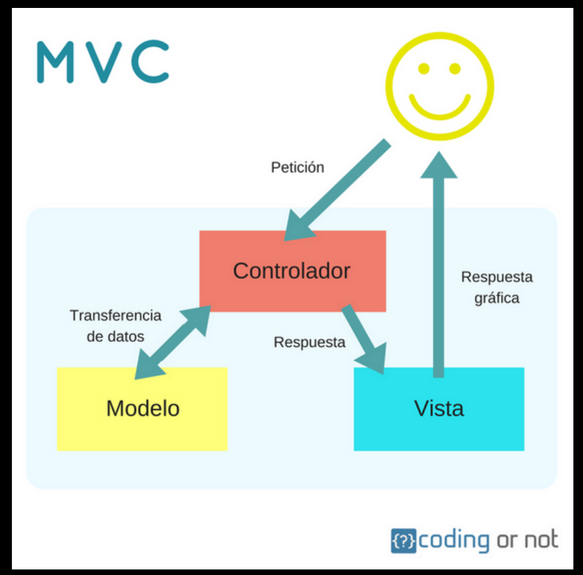
Tienda de Manolo

Grupo
Carlos Andres Pantoja
Karen Yulieth Espinosa
John Alexander Corredor
Heyder Santiago
Rodríguez Galviz

Que es Modelo Vista Controlador (MVC)



El Modelo-Vista-Controlador (MVC) es un patrón de diseño de software ampliamente utilizado en el desarrollo de aplicaciones web y de software en general. Su objetivo principal es separar y organizar el código en tres componentes distintos para mejorar la modularidad, la escalabilidad y la mantenibilidad de una aplicación.

Cuales son sus componentes Modelo Vista Controlador (MVC)

1. **Modelo: Administra los datos y la lógica de negocio de la aplicación.**
2. **Vista: Se encarga de mostrar la información al usuario.**
3. **Controlador: Actúa como intermediario entre el Modelo y la Vista, maneja las interacciones del usuario y actualiza el Modelo y la Vista en consecuencia.**

Ventajas y Desventajas Modelo Vista Controlador (MVC)

La principal ventaja del MVC separa una aplicación en partes, lo que facilita su gestión y cambios sin afectar en exceso a las otras partes. una desventaja es MVC puede complicar aplicaciones pequeñas al dividirlas en tres partes, lo que no siempre es necesario. Funciona mejor en proyectos grandes y complejos.

Caso practico Modelo Vista Controlador (MVC)

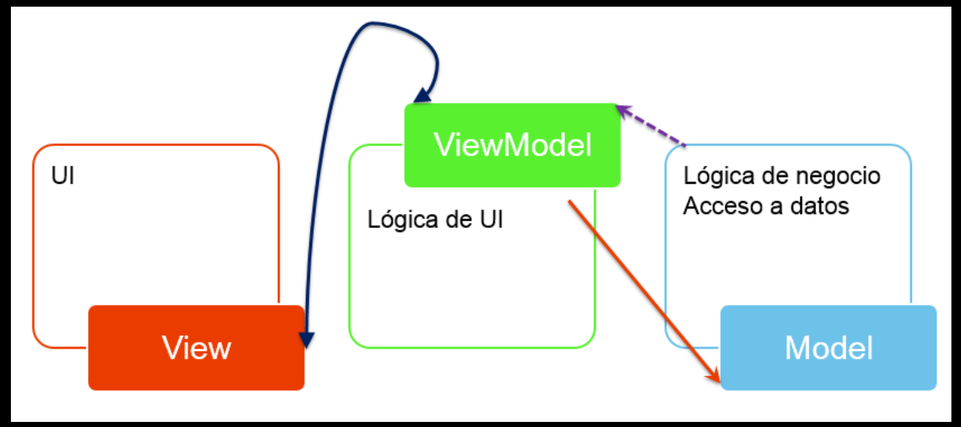
Supongamos que estás desarrollando una aplicación web simple para administrar una lista de tareas. En este caso, el MVC se puede implementar de la siguiente manera:

Modelo (Model): El modelo representa los datos y la lógica de la aplicación. En este caso, el modelo podría ser una clase que almacena las tareas y realiza operaciones en ellas.

La Vista es la interfaz de usuario con la que los usuarios interactúan. En el contexto de una aplicación web, la Vista podría ser una página web que muestra la lista de tareas, los detalles de cada tarea y permite a los usuarios interactuar con ellas.

Controlador (Controller): El Controlador actúa como intermediario entre el Modelo y la Vista. Se encarga de recibir las acciones del usuario desde la Vista, como agregar o eliminar una tarea, y luego interactuar con el Modelo para realizar las operaciones correspondientes. También actualiza la Vista para reflejar cualquier cambio en los datos del Modelo.

Que es Modelo Vista vista Modelo (MVVM)



es un patrón de diseño de software que se utiliza comúnmente en el desarrollo de aplicaciones de interfaz de usuario (UI), especialmente en el contexto de aplicaciones móviles y de escritorio. MVVM es una evolución del patrón Modelo-Vista-Controlador (MVC) y se centra en la separación de las responsabilidades y la lógica de presentación de la interfaz de usuario

Cuales son sus componentes Modelo Vista vista Modelo (MVVM)

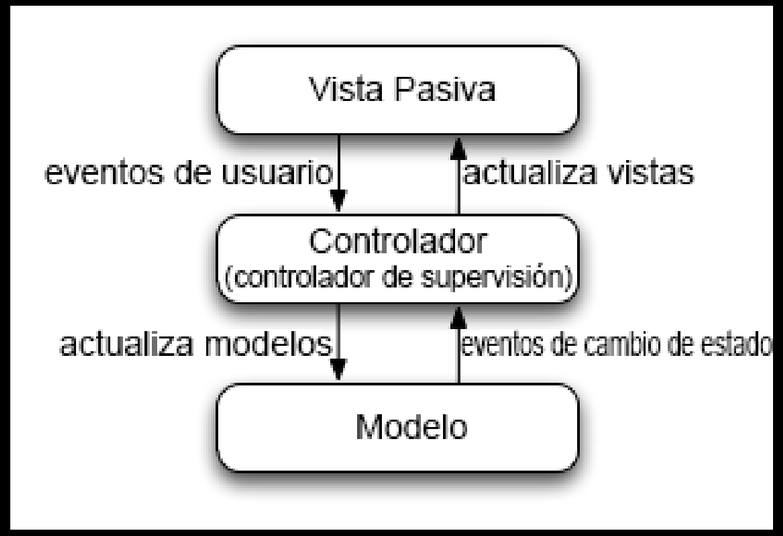
1. **Modelo: Representa los datos y la lógica de negocio.**
2. **Vista: La interfaz de usuario que muestra información y recibe interacciones.**
3. **Vista Modelo: Actúa como intermediario entre Modelo y Vista, proporcionando datos y lógica para la presentación, además de manejar las interacciones del usuario. Facilita la actualización automática de la interfaz de usuario.**

4.

Ventajas y Desventajas Modelo Vista vista Modelo (MVVM)

La principal ventaja los ViewModel pueden ser reutilizados en diferentes vistas, lo que reduce la duplicación de código y mejora la coherencia.
la desventaja es si no se planifica cuidadosamente, un ViewModel puede volverse muy complejo con demasiada lógica de presentación, lo que dificultaría entenderlo y mantenerlo.

Que es Modelo Vista Presentador (MVP)



El patrón Modelo-Vista-Presentador (MVP) es un patrón de diseño de software que se utiliza para separar y organizar las responsabilidades en aplicaciones de interfaz de usuario. Es especialmente común en el desarrollo de aplicaciones de escritorio y en el desarrollo de aplicaciones web en el lado del servidor (como aplicaciones basadas en tecnologías web como ASP.NET WebForms)

Cuales son sus componentes Modelo Vista Presentador (MVP)

1. **Modelo: Contiene la lógica de negocio y los datos de la aplicación.**
2. **Vista: Es la interfaz de usuario que muestra información y recibe interacciones del usuario.**
3. **Presentador: Actúa como intermediario entre el Modelo y la Vista, manejando la lógica de presentación y la interacción con el usuario, actualizando directamente la Vista.**

Ventajas y Desventajas Modelo Vista Presentador (MVP)

La principal ventaja MVP es útil en proyectos heredados o en entornos que no admiten enlace de datos (Data Binding), ya que no depende de esta característica.
la desventaja es que puede haber una curva de aprendizaje para comprender y aplicar correctamente el patrón MVP, especialmente para desarrolladores nuevos en su uso.

Que es Modelo Vista Servicio (MVS)

El patrón Modelo-Vista-Servicio (MVS) es una arquitectura de software que organiza una aplicación en tres componentes clave: el "Modelo" representa los datos y la lógica de negocio, el "Vista" maneja la presentación de la información al usuario, y el "Servicio" contiene lógica adicional relacionada con la aplicación, como operaciones de acceso a datos o servicios externos. Este patrón se utiliza comúnmente en aplicaciones web y permite una clara separación de responsabilidades: el Modelo gestiona los datos y la lógica subyacente, la Vista se encarga de la interfaz de usuario, y el Servicio proporciona funciones adicionales que pueden ser compartidas por diferentes partes de la aplicación.

Cuales son sus componentes Modelo Vista Servicio (MVS)

1. **Modelo: Contiene la lógica de negocio y los datos de la aplicación.**
2. **Vista: Es la interfaz de usuario que muestra información y recibe interacciones del usuario.**
3. **Servicio (Service): El Servicio es una capa que contiene lógica adicional relacionada con la aplicación. Proporciona funciones y operaciones que no pertenecen directamente al Modelo ni a la Vista. Esto puede incluir operaciones de acceso a datos, integración con servicios externos, lógica de negocio especializada y más**

Ventajas y Desventajas Modelo Vista Servicio (MVS)

Una ventaja del patrón Modelo-Vista-Servicio (MVS) es la alta modularidad y reutilización de código que ofrece. Al separar la lógica de negocio en la capa de Servicio, es más fácil agregar nuevas características o servicios sin afectar el Modelo ni la Vista principal de la aplicación.

Sin embargo, una desventaja de MVS puede ser la complejidad adicional que introduce en la arquitectura, ya que requiere la implementación y gestión de una capa de Servicio adicional, lo que podría aumentar la complejidad del desarrollo y el mantenimiento del código en aplicaciones pequeñas o simples.

Que es Modelo Vista Adaptador (MVA)

El patrón Modelo-Vista-Adaptador (MVA) es una variante del patrón Modelo-Vista-Controlador (MVC) que se utiliza para separar y organizar la lógica de una aplicación. En MVA, el "Adaptador" actúa como intermediario entre el "Modelo" (que gestiona los datos y la lógica de negocio) y la "Vista" (que se encarga de la interfaz de usuario). El Adaptador se utiliza para adaptar o traducir los datos del Modelo en un formato que la Vista pueda entender y viceversa, lo que facilita la comunicación y la presentación de datos.

Cuales son sus componentes Modelo Vista Adaptador (MVA)

1. **Modelo: Contiene la lógica de negocio y los datos de la aplicación.**
2. **Vista: Es la interfaz de usuario que muestra información y recibe interacciones del usuario.**
3. **Adaptador (Adapter): El Adaptador actúa como un intermediario entre el Modelo y la Vista. Se encarga de traducir o adaptar los datos del Modelo en un formato adecuado para la Vista y viceversa. Facilita la comunicación entre el Modelo y la Vista, asegurando que los datos se presenten de manera correcta y coherente en la interfaz de usuario sin que la Vista necesite conocer detalles internos del Modelo.**

Ventajas y Desventajas Modelo Vista Adaptador (MVA)

Una ventaja del patrón Modelo-Vista-Adaptador (MVA) es que permite la separación clara de responsabilidades entre el Modelo, la Vista y el Adaptador, lo que facilita la adaptación de datos y la presentación en la interfaz de usuario, además de mejorar la modularidad y la mantenibilidad del código.
Sin embargo, una desventaja es que introduce una capa adicional (el Adaptador), lo que puede aumentar la complejidad de la arquitectura y del desarrollo de la aplicación, especialmente en casos donde la adaptación de datos no es muy compleja, lo que podría ser innecesario en aplicaciones pequeñas o simples.

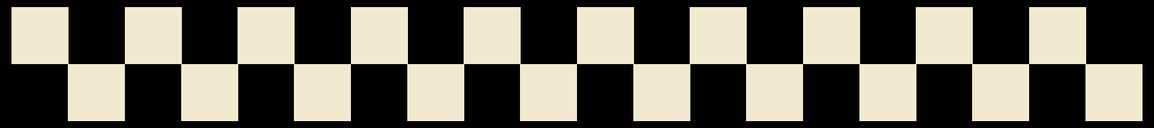
Que es Modelo Vista Interactor (MVI)

El patrón Modelo-Vista-Interactor (MVI) es un enfoque de arquitectura de software que se centra en la gestión de la lógica de la interfaz de usuario y la interacción del usuario. En MVI, el "Modelo" representa el estado de la aplicación, la "Vista" muestra el estado actual al usuario y el "Interactor" maneja las interacciones del usuario y actualiza el Modelo en consecuencia. Este patrón promueve una clara separación de responsabilidades y un flujo de datos unidireccional, lo que facilita el mantenimiento y la escalabilidad del código en aplicaciones de interfaz de usuario complejas.

Cuales son sus componentes Modelo Vista Interactor (MVI)

1. **Modelo: Contiene la lógica de negocio y los datos de la aplicación.**
2. **Vista: Es la interfaz de usuario que muestra información y recibe interacciones del usuario.**
3. **Interactor (Interactor): El Interactor es el componente que maneja las interacciones del usuario. Recibe las acciones del usuario desde la Vista, procesa estas interacciones y actualiza el Modelo en consecuencia. Se encarga de la lógica de negocio y las operaciones relacionadas con la interacción del usuario.**

Ventajas y Desventajas Modelo Vista Interactor (MVI)



Una ventaja clave del patrón Modelo-Vista-Interactor (MVI) es su capacidad para mantener una separación sólida entre la lógica de negocio y la interfaz de usuario, lo que facilita la reutilización de componentes y el mantenimiento del código en aplicaciones complejas. Esto promueve una estructura modular y un flujo de datos unidireccional, lo que simplifica la escalabilidad y la depuración.

Sin embargo, una desventaja potencial del MVI es que puede introducir una mayor complejidad en aplicaciones más simples o pequeñas, ya que requiere la implementación de componentes adicionales, como el Interactor. Esto puede aumentar el tiempo de desarrollo y requerir una mayor cantidad de código en comparación con enfoques más simples en ciertos casos.

La conexion a base de datos (My SQL) desde java (17)

porta las bibliotecas necesarias: En tu proyecto Java, importa las bibliotecas JDBC relevantes. Estas bibliotecas contienen las clases y métodos necesarios para interactuar con la base de datos.

Crea una cadena de conexión: Debes proporcionar una cadena de conexión que incluye detalles como el tipo de base de datos, la dirección del servidor, el puerto y las credenciales de acceso.

Establece la conexión: Utiliza la cadena de conexión para establecer una conexión con la base de datos. Puedes hacerlo mediante la clase Connection proporcionada por JDBC.

```
static Connection conexion;
static String host = "jdbc:mysql://localhost:3306/";
static String user = "root";
static String pass = "";
static String bd = "tienda";

static {
    try {
        Conexion.conexion = DriverManager.getConnection(host + bd, user, password:pass);
        System.out.println(x: "Conexion exitosa :)");
    } catch (SQLException ex) {
        System.out.println(x: ex.getMessage());
        Logger.getLogger(name:Main.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
}
```