JUA CERQUERA
HECTOR FABIAN
DANIEL LEYTON
MARTHA LUCIA
YANUARD STEVIN

MODELO

Contenido

Versión preliminar de la tecnología 5G

Modelo Vista Controlador (MVC)

5G es la quinta generación de tecnología inalámbrica que ofrece mucho más que solo velocidades más rápidas. Ofrece frecuencias más altas para brindar una conexión rápida, menos latencia y más dispositivos.

El patrón MVC consta de tres componentes principales

Modelo (Model):

Vista (View):

Controlador (Controller):

Modelo Vista Vista Modelo (MVVM)

El Modelo-Vista-Vista-Modelo (MVVM) es un patrón de arquitectura de software utilizado en el desarrollo de aplicaciones de software, especialmente en el contexto de aplicaciones de interfaz de usuario (UI). MVVM se utiliza comúnmente en aplicaciones de plataforma como WPF (Windows Presentation Foundation), Xamarin, y aplicaciones móviles que utilizan frameworks como Flutter o React Native. Este patrón se enfoca en separar la lógica de la interfaz de usuario de la lógica de negocio y los datos de la aplicación.

El MVVM consta de tres componentes principales:

1G **MODELO (MODEL)**

2G **VISTA (VIEW)**

3G **MODELO DE VISTA (VIEWMODEL):**

Modelo Vista Presentador (MVP)

El Modelo-Vista-Presentador (MVP) es un patrón de arquitectura de software que se utiliza para diseñar y desarrollar aplicaciones de software, especialmente en el contexto de aplicaciones de interfaz de usuario (UI). El MVP es una evolución del patrón Modelo-Vista-Controlador (MVC) y se enfoca en la separación de las responsabilidades en una aplicación.

El patrón MVP se divide en tres componentes principales:

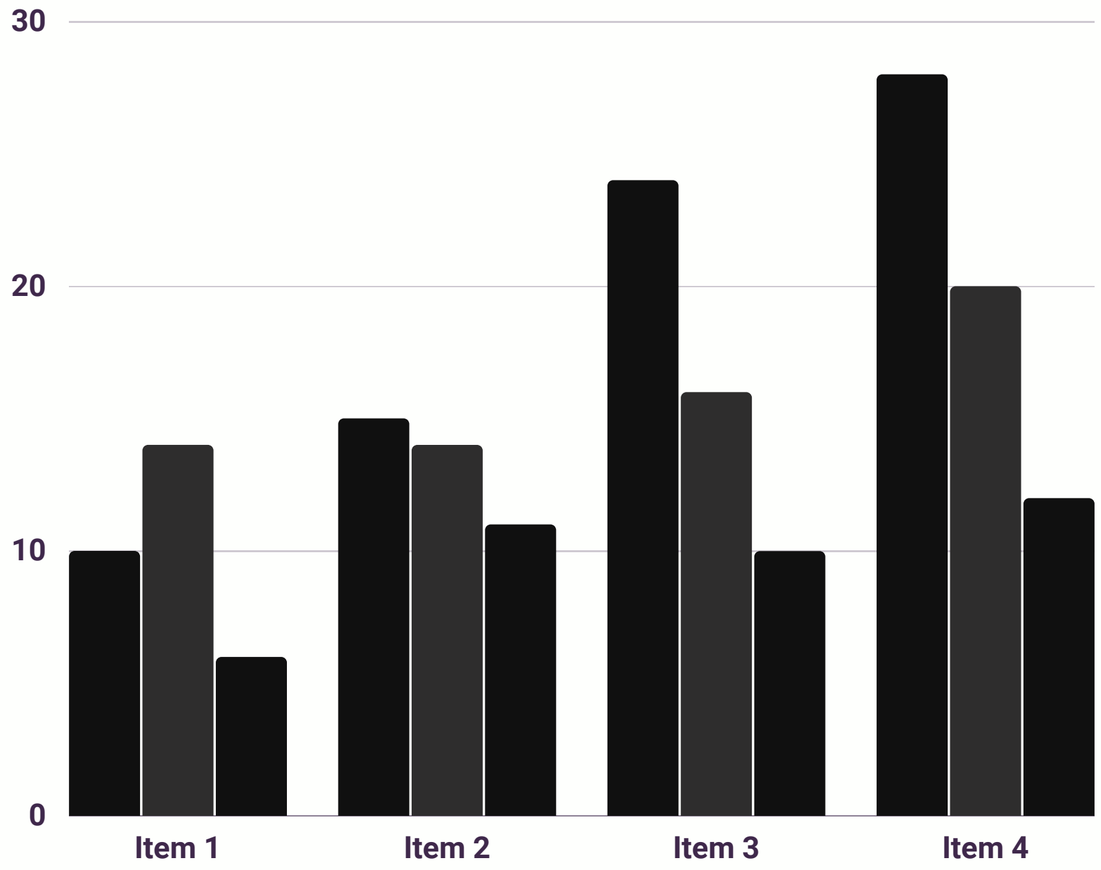
MODELO (MODEL):

VISTA (VIEW)

PRESENTADOR (PRESENTER):

Modelo Vista Servicio (MVS)

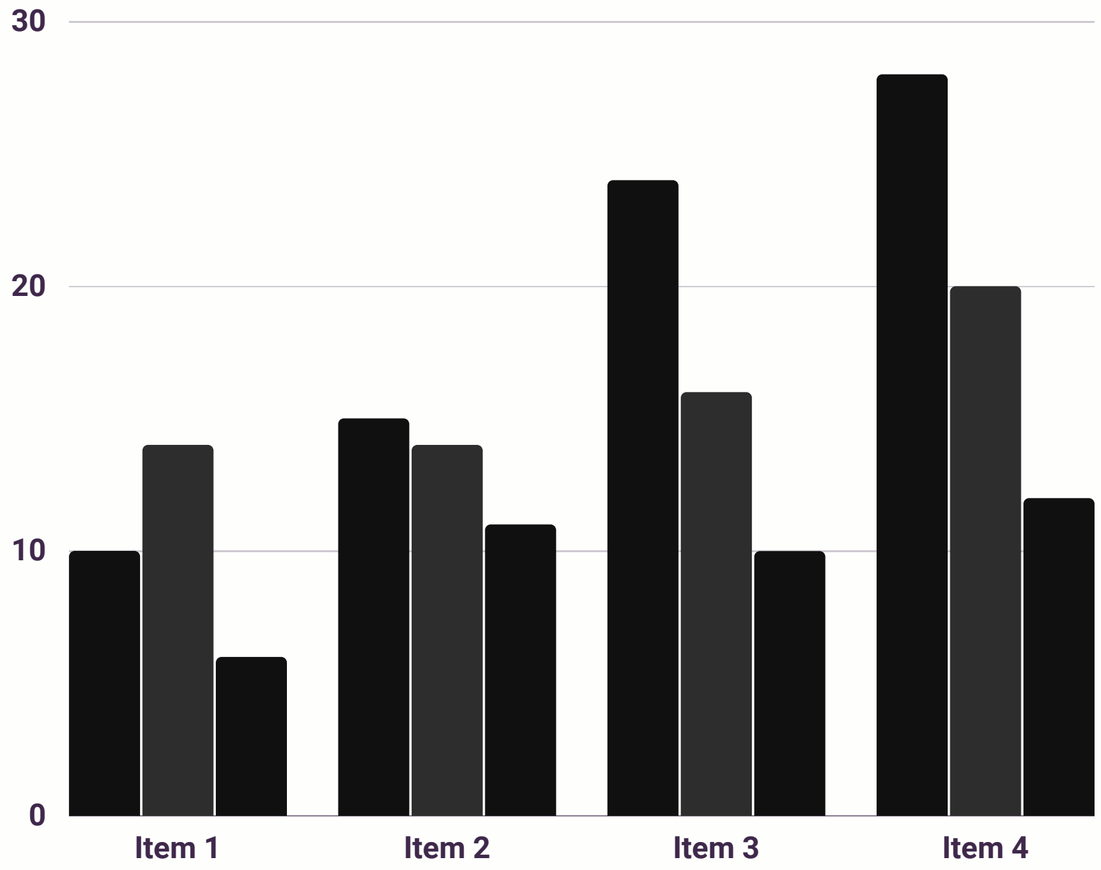
El término "Modelo Vista Servicio" (MVS) no es un patrón de arquitectura de software ampliamente reconocido o estandarizado como el Modelo-Vista-Controlador (MVC), el Modelo-Vista-Presentador (MVP) o el Modelo-Vista-Modelo (MVVM). En su lugar, parece ser un término específico o una variante que se utiliza en ciertos contextos o proyectos particulares.



Sin una definición estándar, el término "Modelo Vista Servicio" podría referirse a una arquitectura o patrón personalizado diseñado para un proyecto específico. En una arquitectura de este tipo, se podrían definir componentes llamados "Modelo", "Vista" y "Servicio", pero los detalles precisos de cómo se organizan y cómo se comunican estos componentes dependerían de la implementación específica.

Modelo Vista Adaptador (MVA)

El término "Modelo Vista Adaptador" (MVA) no es un patrón de arquitectura de software ampliamente reconocido ni estandarizado, al menos hasta mi última actualización en septiembre de 2021. Puede ser una variante de un patrón más ampliamente reconocido, como el Modelo-Vista-Controlador (MVC) o el Modelo-Vista-Presentador (MVP), o simplemente una nomenclatura utilizada en un contexto particular.



En arquitecturas de software como el MVC o el MVP, el "Modelo" representa la lógica de negocio y los datos, la "Vista" representa la interfaz de usuario y la presentación, y el "Controlador" o "Presentador" actúa como intermediario entre el modelo y la vista. El controlador o presentador se encarga de manejar las interacciones del usuario y actualizar la vista según los cambios en el modelo.

Modelo Vista Interactor (MVI)

El Modelo Vista Interactor (MVI) es un patrón de arquitectura de software que se utiliza en el desarrollo de aplicaciones, particularmente en el contexto de aplicaciones de interfaz de usuario (UI). El MVI es especialmente popular en el desarrollo de aplicaciones Android, donde se utiliza para mantener una separación clara entre la lógica de negocio, la interfaz de usuario y la gestión del estado de la aplicación.

El patrón MVI consta de tres componentes principales:

MODELO (MODEL)

VISTA (VIEW):

INTERACTOR (INTERACTOR):

conexión a una base de datos MySQL desde Java 17

Asegúrate de que tienes el controlador JDBC de MySQL en tu proyecto

MySQL Community Downloads

Connector/J

General Availability (GA) Releases | Archives

Connector/J 8.1.0

Select Operating System:

Select Operating System…

Importar clases

```
package coneccionsql;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
```

Establecer una conexión a la base de datos

```
    public static void main(String[] args) {
        // Parámetros de conexión a la base de datos
        String url = "jdbc:mysql://localhost:3306/primerbase";
        String usuario = "root";
        String contraseña = "juda";

        try {
            // Cargar el controlador JDBC
            Class.forName(className: "com.mysql.cj.jdbc.Driver");

            // Establecer la conexión
            Connection conexion = DriverManager.getConnection(url, user: usuario, password: contraseña);

            if (conexion != null) {
                System.out.println(x: "Conexión exitosa a la base de datos MySQL");

                // Aquí puedes realizar operaciones en la base de datos

                // Cerrar la conexión cuando hayas terminado
                conexion.close();
            }
        } catch (SQLException e) {
            System.err.println("Error de SQL: " + e.getMessage());
        } catch (ClassNotFoundException e) {
            System.err.println("Error al cargar el controlador JDBC: " + e.getMessage());
        }
    }
}
```