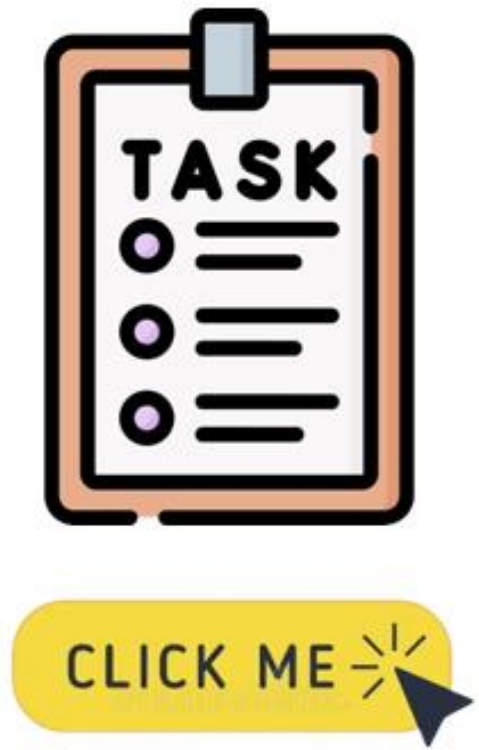
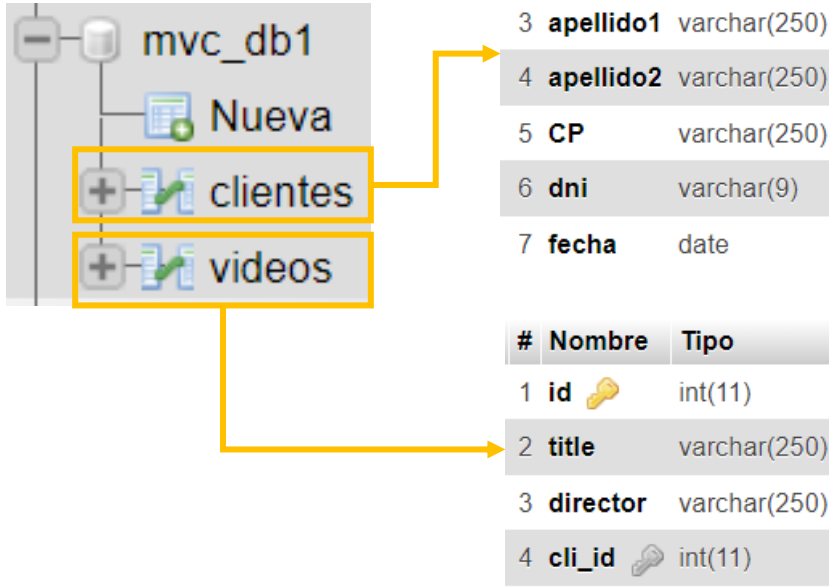
T22 – Patrón MVC



1

DB mvc_db1 creada para realizar el ejercicio:



#	Nombre	Tipo
1	id	int(11)
2	nombre	varchar(250)
3	apellido1	varchar(250)
4	apellido2	varchar(250)
5	CP	varchar(250)
6	dni	varchar(9)
7	fecha	date

#	Nombre	Tipo
1	id	int(11)
2	title	varchar(250)
3	director	varchar(250)
4	cli_id	int(11)

Controlador para la conexión con la DB:

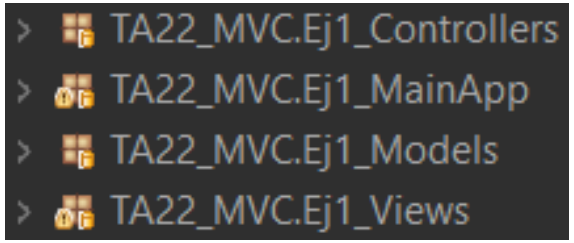
```java
package TA22_MVC.Ej1_Controllers;

import java.sql.Connection;

public class conexion_database {
    private static final String URL = "jdbc:mysql://localhost:3306/mvc_db1";
    private static final String USER = "root";
    private static final String PASSWORD = "";

    public static Connection getConnection() {
        Connection connection = null;
        try {
            // Cargar el driver de MySQL
            Class.forName("com.mysql.cj.jdbc.Driver");
            // Establecer la conexión
            connection = DriverManager.getConnection(URL, USER, PASSWORD);
            System.out.println("Conexión exitosa a la base de datos");
        } catch (ClassNotFoundException e) {
            System.out.println("Error: Clase del driver no encontrada");
            e.printStackTrace();
        } catch (SQLException e) {
            System.out.println("Error: No se pudo establecer la conexión con la base de datos");
            e.printStackTrace();
        }
        return connection;
    }
```

```java
public static void main(String[] args) {
    // Probar la conexión
    Connection conn = conexion_database.getConnection();
    if (conn != null) {
        try {
            conn.close();
            System.out.println("Conexión cerrada");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

A partir de las necesidades del enunciado se crea la base de datos necesaria para la gestión de los datos.

Establecemos la conexión a la DB con el controlador apropiado.

La estructura del proyecto en Eclipse es la siguiente:



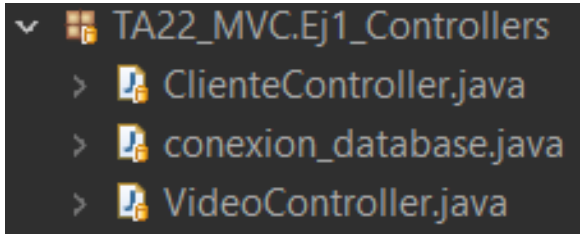
CONTROLADORES

Objetivo: ser el vínculo entre el código del Back-end y el Front-end.

Nuestro back serán los Modelos que generemos, que contendrán los métodos necesarios.

El front será cada una de las View que generemos.

Los controladores que hemos generado son los siguientes:



La función que realizan es llamar a los métodos del Modelo para poder aplicarlos. Por ejemplo:

```java
package TA22_MVC.Ej1_Controllers;

import java.util.List;

public class ClienteController {
    public void addCliente(Cliente cliente) {
        cliente.addCliente();
    }
```

```java
    public List<Cliente> getAllClientes() {
        return Cliente.getAllClientes();
    }

    public Cliente getClienteById(int id) {
        return Cliente.getClienteById(id);
    }
```

MAIN-APP

Objetivo: generar la View inicial del programa. Su función es permitir la interacción con el usuario para que se pueda acceder a cada una de las View que se han creado.
En nuestro caso, el acceso se da mediante Jbutton, cuyos eventos generan las View indicadas acorde al texto mostrado en el propio botón.

El MainFrame generado ha sido el siguiente:

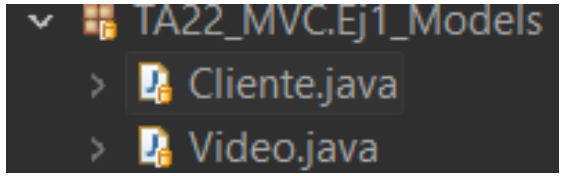

```java
package TA22_MVC.Ej1_MainApp;

import javax.swing.*;

public class MainFrame extends JFrame {
    public MainFrame() {
        setTitle("Sistema de Gestión de Clientes y Videos");
        setSize(550, 350);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(null);

        // Menú para CRUD de Clientes
        JButton clienteCreateButton = new JButton("Agregar Cliente");
        clienteCreateButton.setBounds(50, 50, 200, 30);
        clienteCreateButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                new ClienteCreateView();
            }
        });
        add(clienteCreateButton);
```

Como podemos ver, en el MainFrame se genera la ventana donde colocaremos todos los Jbutton necesarios para acceder a todas las View generadas. Cada botón está identificado acorde a la View que genera. La vista global sería:

MODELS

Objetivo: contener el código necesario para que, cuando el controlador lo aplique, se pueda generar el Objeto deseado y manipular esa información con los métodos relacionados del mismo.

Los modelos que hemos generado han sido:




```java
package TA22_MVC.Ej1_Models;

import java.sql.Connection;

public class Cliente {
    // Atributos del cliente
    private int id;
    private String nombre;
    private String apellido1;
    private String apellido2;
    private String CP;
    private String dni;
    private Date fecha;
```

```java
package TA22_MVC.Ej1_Models;

import java.sql.Connection;

public class Video {
    // Atributos del video
    private int id;
    private String title;
    private String director;
    private int cli_id;
```

Los modelos contienen:

- Constructores

```java
// Constructor con parámetros
public Cliente(int id, String nombre, String apellido1,
        String apellido2, String CP, String dni, Date fecha) {
    this.id = id;
    this.nombre = nombre;
    this.apellido1 = apellido1;
    this.apellido2 = apellido2;
    this.CP = CP;
    this.dni = dni;
    this.fecha = fecha;
}
```

- Getters y setters

```java
// Getters y setters
public int getId() {

public void setId(int id) {
```

- Métodos de acceso a datos

```java
// Métodos de acceso a datos
public void addCliente() {

public static List<Cliente> getAllClientes() {

public static Cliente getClienteById(int id) {

public void updateCliente() {

public static void deleteCliente(int id) {
```

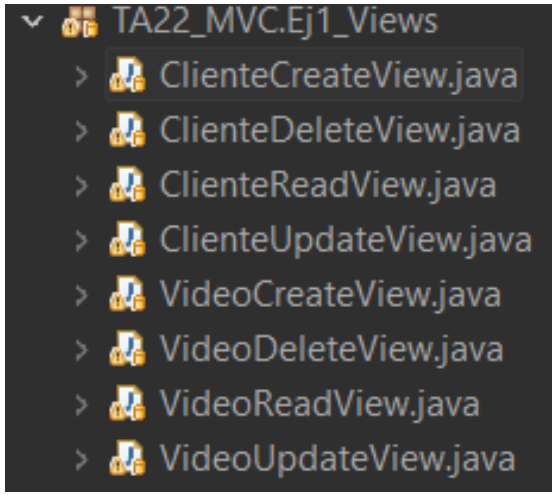
VIEWS

1

Objetivo: generar y visualizar el frame acorde a la acción deseada que ha sido seleccionada en el MainFrame.

La View consta de todos aquellos atributos y campos necesarios para generar una estructura clara y funcional para el usuario

Las vistas que hemos generado han sido:



Estos son algunos elementos típicos que se utilizan para...

- Rellenar campos con información

```
JTextField nombreField = new JTextField();
JTextField apellido1Field = new JTextField();
JTextField apellido2Field = new JTextField();
JTextField direccionField = new JTextField();
JTextField dniField = new JTextField();
JTextField fechaField = new JTextField();
```

- Identificar los campos de la view y agregar campos:

```
add(new JLabel("Nombre**:"));
add(nombreField);
add(new JLabel("Apellido1**:"));
add(apellido1Field);
add(new JLabel("Apellido2:"));
add(apellido2Field);
```

- Establecer conexión con los controladores:

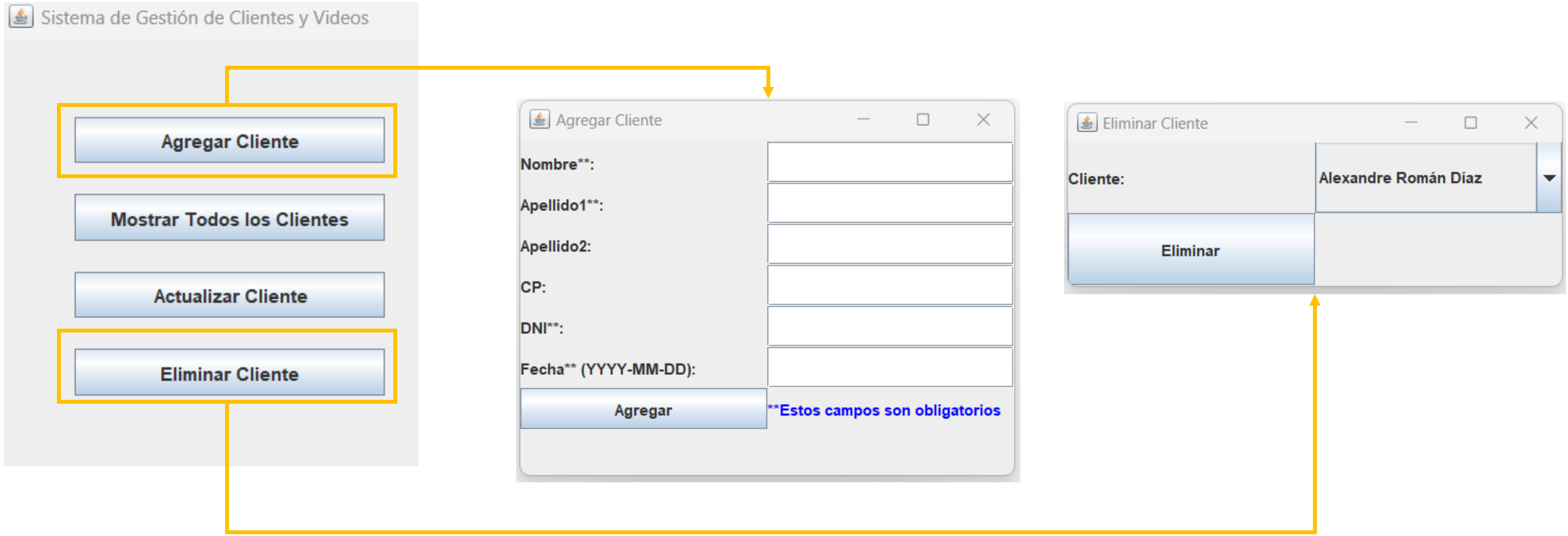
```
JButton addButton = new JButton("Agregar");
addButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
```

- Indicar información adicional:

```
JLabel obligatoriosLabel = new JLabel(
        "**Estos campos son obligatorios");
obligatoriosLabel.setForeground(Color.BLUE);
add(obligatoriosLabel);
```

VIEWS: ejemplos

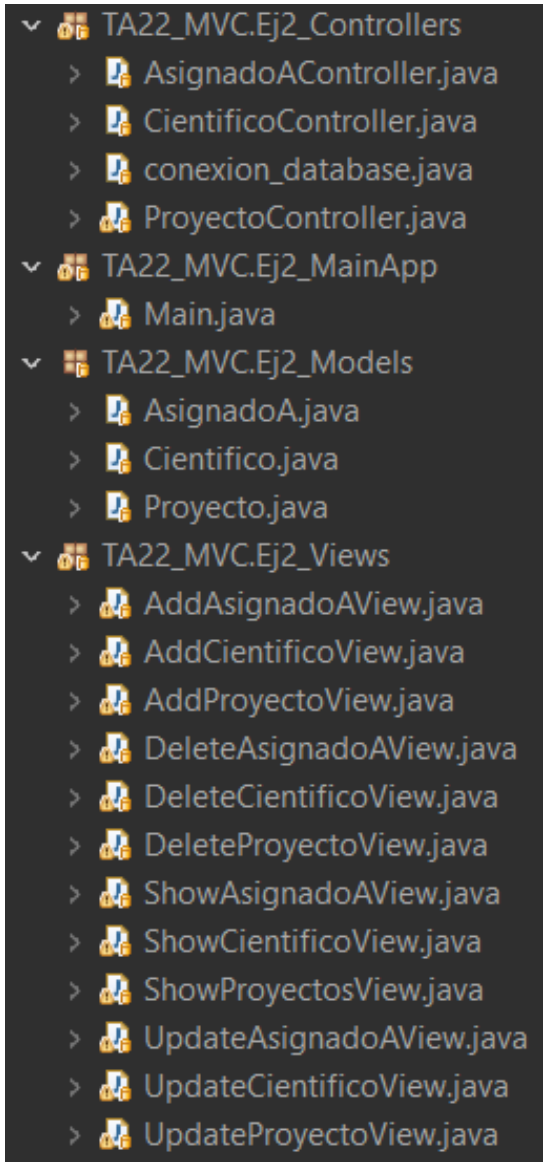
MainFrame



2

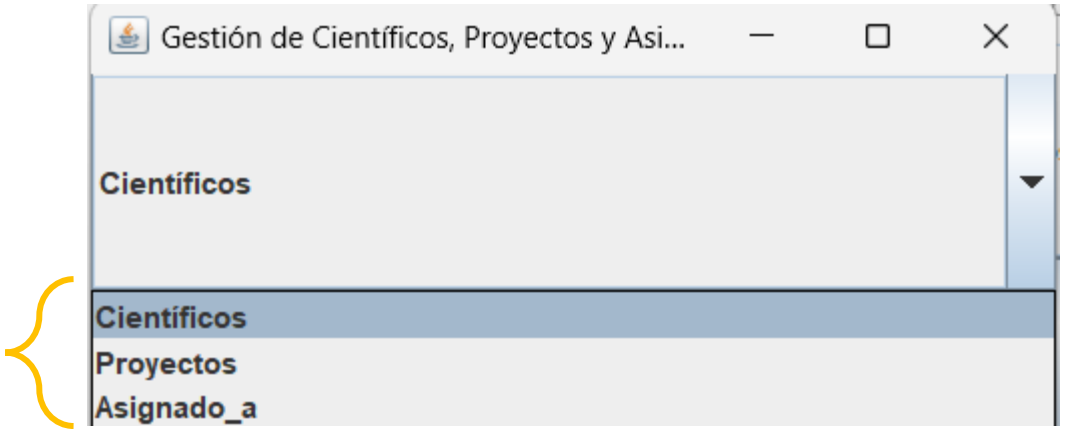
DB utilizada:

Estructura del proyecto y archivos generados:

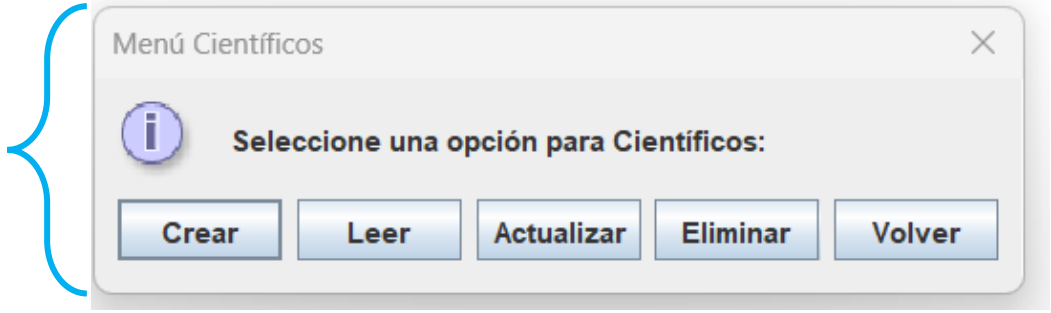


MainFrame

Selector de tablas (DB)



Selector acciones a realizar



Selector acciones a realizar

2