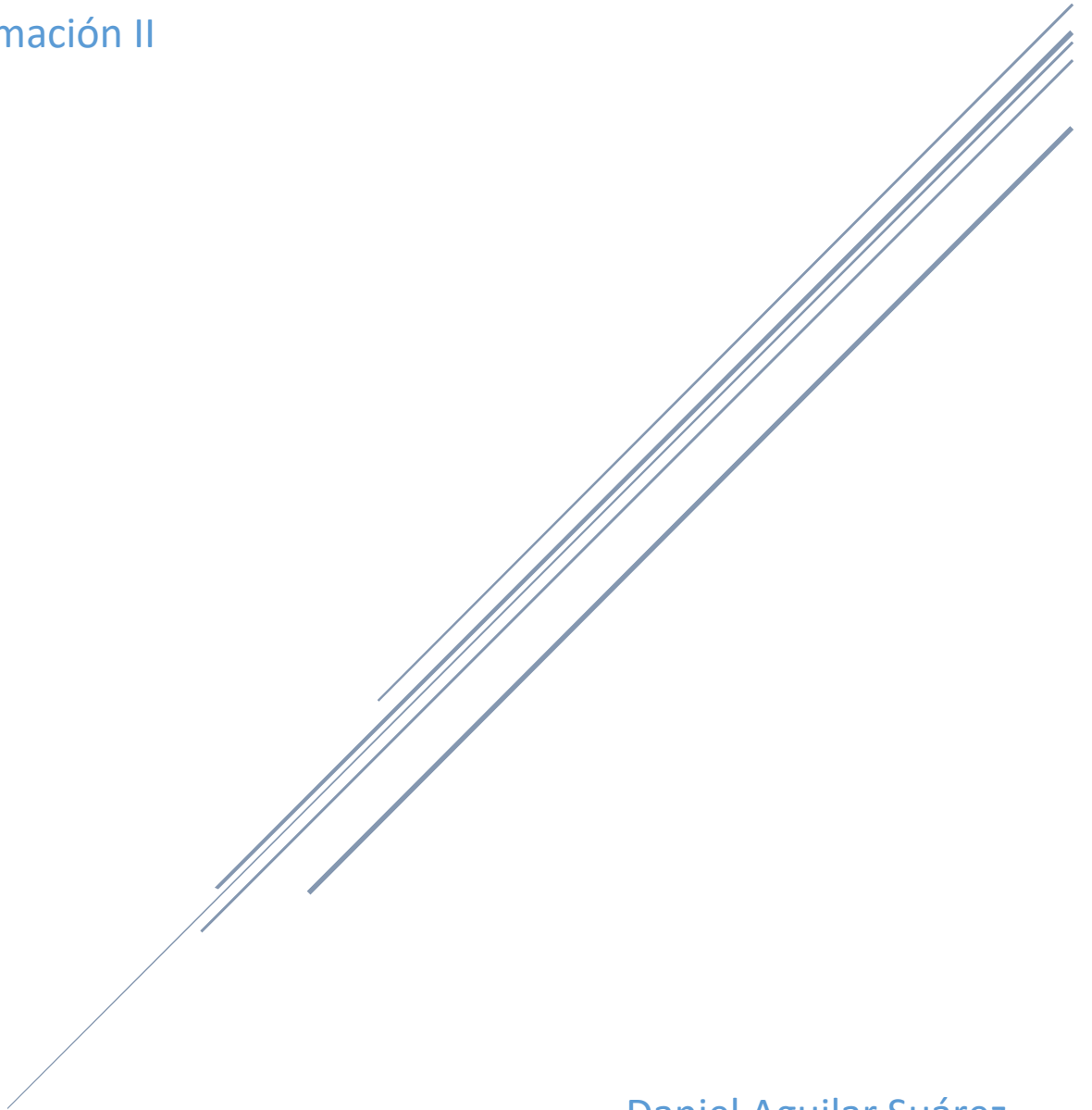


DISEÑO DE PROGRAMAS PARA LA GESTIÓN DE BASES DE DATOS RELACIONALES Y PERISTENCIA DE OBJETOS

Programación II



Daniel Aguilar Suárez
DAW

Tabla de contenido

1. Crear el package actividad.09.filmoteca	5
1.1. Crear una base de datos con una tabla Película que almacene los siguientes campos: título, director, país, duración, género. Introduce los datos de varias películas.....	5
1.2. Crear una tabla Director en la base de datos que almacene el nombre y los apellidos del director	7
1.3. Modificar la tabla Película de tal manera que el campo director sea una referencia a la tabla Director.....	8
1.4. Crear interfaz gráfica, menú principal	9
1.5. Insertar, eliminar y modificar película	11
1.5.1. Insertar Pelicula	11
1.5.2. Eliminar película	15
1.5.3. Modificar película	17
1.6. Insertar, eliminar y modificar director	21
1.6.1. Insertar director	21
1.6.2. Eliminar director	23
1.6.3. Modificar director	25
1.7. Listados	29
1.8. Visualización y botones de Siguiente y atrás	34
2. Crear package actividad09.tiendas para gestionar información en objectDB	37
2.1. Clase Empleado:.....	37
2.2. Clase Tiendas:	38
2.3. Clase GestionaTiendas y crear tres tiendas y tres empleados, luego insertarlos en la BBDD	39
2.4. Mostrar el menú y configurar las diferentes opciones	40
2.4.1. Mostrar empleados	41
2.4.2. Mostrar tiendas	41
2.4.3. Mostrar tiendas ordenadas por ventas	42
2.4.4. Editar un empleado	43
2.4.5. Crear una nueva tienda	44
3. CODIGO COMPLETO EJERCICIO FILMOTECA	47
3.1. DelPelicula	47
3.2. EliminarDirector.....	48
3.3. Funciones.....	49
3.4. GestionarBBDD	50
3.5. InsertDirector	54

3.6.	InsertPelicula	55
3.7.	Listados	57
3.8.	Main	63
3.9.	MenuPrincipal	63
3.10.	ModDirector	64
3.11.	ModPelicula	66
4.	CODIGO COMPLETO EJERCICIO TIENDAS	68
4.1.	Empleado.....	68
4.2.	Funciones.....	69
4.3.	GestionaTiendas	72
4.4.	Tienda.....	74

Daniel Aguilar Suarez
DISEÑO DE PROGRAMAS PARA LA GESTIÓN DE BASES DE DATOS RELACIONALES Y PERISTENCIA DE
OBJETOS

Información del alumno

Nombre	Daniel
Apellidos	Aguilar Suárez
Módulo/Crédito	M03II
UF (solo ciclos LOE)	UF6
Título de la actividad	Diseño de programas para la gestión de bases de datos relacionales y persistencia de objetos

Antes de continuar

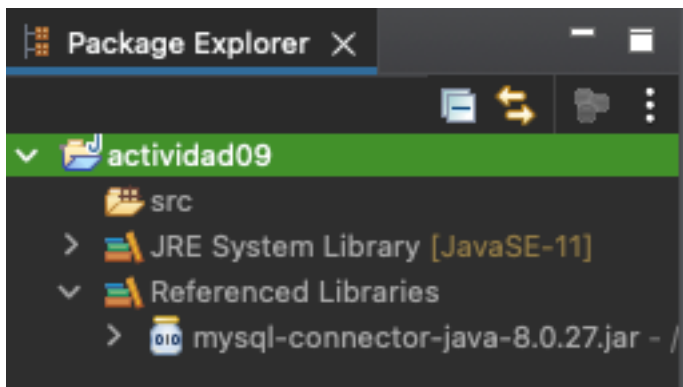
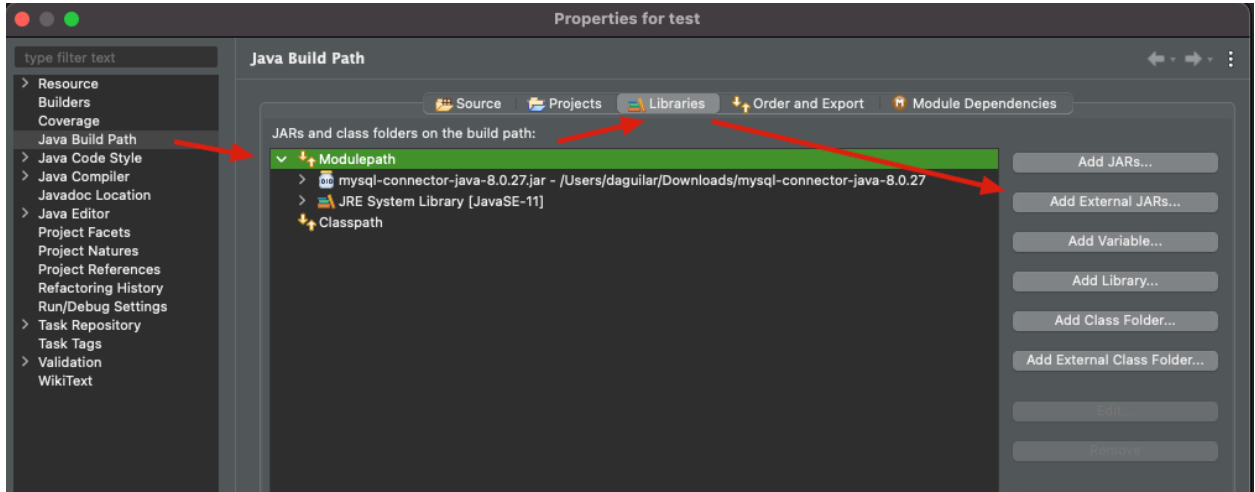
Debido a que no quedaba del todo claro en el enunciado, no sabía si la BBDD la tenía que crear desde MySQL o desde Java, yo la he creado desde Java, por lo que al ejecutar el proyecto va a crear directamente todo, BBDD, tablas, registros, etc. No obstante añadido el archivo SQL para crear la BBDD también, aunque no sea necesario ya que como digo, lo va a crear la ejecución del programa al principio.

Otro apunte son los puertos y a contraseña, tiene configurada una contraseña por defecto y un puerto que creo que no era el que venía por defecto en su momento ya que me suena que lo cambié cuando estudié BBDD, por lo que para poder ejecutar el proyecto tendrás que cambiarlo a mano.

Los códigos completos están al final del documento.

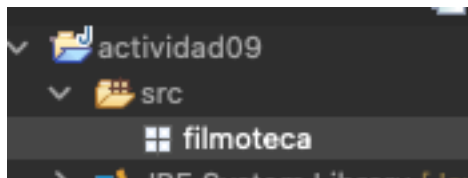
0. Importar mysql_connector

Antes de nada, procedo a importar la librería para continuar con los siguientes ejercicios, para ello, botón derecho encima del proyecto > Propiedades:



1. Crear el package actividad.09.filmoteca

Lo dejaré dentro de src todo para tenerlo mas organizado:



1.1. Crear una base de datos con una tabla Película que almacene los siguientes campos: título, director, país, duración, género. Introduce los datos de varias películas

Lo primero de todo creo una clase llamada GestionBBDD. En ella añadiré todas las funciones necesarias.

Para la conexión a la BBDD hago la siguiente configuración:

```
private static String das_datosConexion ="jdbc:mysql://127.0.0.1:8889/";
private static String das_baseDatos = "FILMOTECA";
private static String das_usuario = "root";
private static String das_password = "4246";
```

```
private static Connection con;
```

Y para crearla:

```
private void crearBDD() throws Exception{
    String query = "create database if not exists "+ das_baseDatos +"";
    Statement stmt = null;
    try{
        stmt = con.createStatement();
        stmt.executeUpdate(query);
        con = DriverManager.getConnection(das_datosConexion+das_baseDatos,
das_usuario, das_password);
    }catch (SQLException e){
        e.printStackTrace();
    }finally{
        stmt.close();
    }
}
```

Para la función que añadirá la tabla películas lo haré de la siguiente forma:

```
private void crearTablaPelicula() throws Exception {
    String query = "create table if not exists PELICULA("
        + "Titulo VARCHAR(30), "
        + "Director int, "
        + "Pais VARCHAR(30),"
        + "Duracion INTEGER,"
        + "Genero VARCHAR(15));";
    Statement stmt = null;
    try{
        stmt = con.createStatement();
        stmt.executeUpdate(query);
    }catch (SQLException e){
        e.printStackTrace();
    }finally{
        stmt.close();
    }
}
```

Para añadir películas:

```
public void insertarPelicula(String titulo, int director, String pais, int
duracion, String genero) throws SQLException{
    String query = "insert into PELICULA values('" + titulo + "', " + director
+ ", '" + pais + "', " + duracion + ", '" + genero + "')";
    Statement stmt = null;
    try {
        stmt = con.createStatement();
        stmt.executeUpdate(query);
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        stmt.close();
    }
}
```

Dentro del constructor habrá las siguientes líneas, al final del todo añadiré el código completo del constructor:

```
        insertarPelícula("Terminator", "Dani", 1, "EEUU", 100, "Acción");  
        insertarPelícula("ESDLA", 2, "EEUU", 180, "Fantasía");  
        insertarPelícula("Torrente", 3, "España", 120, "Comedia");  
        insertarPelícula("Otra Película", 4, "Francia", 200, "Amor");  
        insertarPelícula("Programadores", 5, "Italia", 120, "Drama");  
    }  
}
```

1.2. Crear una tabla Director en la base de datos que almacene el nombre y los apellidos del director

Para crear la tabla director:

```
private void crearTablaDirector() throws Exception {  
    String query = "create table if not exists DIRECTOR(" +  
        "id_director int PRIMARY KEY auto_increment," +  
        "Nombre VARCHAR(50)," +  
        "Apellidos VARCHAR(100));";  
    Statement stmt = null;  
    try {  
        stmt = con.createStatement();  
        stmt.executeUpdate(query);  
    } catch (SQLException e) {  
        e.printStackTrace();  
    } finally {  
        stmt.close();  
    }  
}
```

Funcion para insertar directores:

```
public void insertarDirector(int id, String nombre, String apellido) throws  
SQLException {  
    String query = "insert into DIRECTOR values(" + id + ", '" + nombre + "',  
    "'" + apellido + "')";  
    Statement stmt = null;  
    try {  
        stmt = con.createStatement();  
        stmt.executeUpdate(query);  
    } catch (SQLException e) {  
        e.printStackTrace();  
    } finally {  
        stmt.close();  
    }  
}
```

Insertcion de directores:

```
        insertarDirector(1, "Dani", "Aguilar");  
        insertarDirector(2, "Alicia", "Revilla");  
        insertarDirector(3, "Marco", "Polo");  
        insertarDirector(4, "Lolo", "Perro");  
        insertarDirector(5, "Coco", "Loco");  
    }  
}
```


1.3. Modificar la tabla Película de tal manera que el campo director sea una referencia a la tabla Director.

Para modificar la tabla programo la siguiente función:

```
public void updateRel() throws SQLException{
    String query = "ALTER TABLE PELICULA ADD CONSTRAINT fk_película_director
foreign key (Director) references DIRECTOR(id_director);";
    Statement stmt = null;
    try {
        stmt = con.createStatement();
        stmt.executeUpdate(query);
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        stmt.close();
    }
}
```

La añado al final del constructor para que se ejecute junto con lo demás.

CÓDIGO COMPLETO DEL CONSTRUCTOR:

```
public GestionBBDD(){
    try {
        con = DriverManager.getConnection(das_datosConexion, das_usuario,
das_password);
        try {
            // CREO LA BASE DE DATOS SI NO EXISTE
            crearBDD();
            // CREO LA TABLA PELICULA SI NO EXISTE
            crearTablaPelícula();
            // CREO LA TABLA DIRECTOR SI NO EXISTE
            crearTablaDirector();
            // INSERTO PELICULAS
            insertarPelícula("Terminator",1,"EEUU",100,"Accion");
            insertarPelícula("ESDLA",2,"EEUU",180,"Fantasia");
            insertarPelícula("Torrente",3,"España",120,"Comedia");
            insertarPelícula("Otra Película",4,"Francia",200,"Amor");
            insertarPelícula("Programadores",5,"Italia",120,"Drama");
            // INSERTO DIRECTORES
            insertarDirector(1,"Dani","Aguilar");
            insertarDirector(2,"Alicia","Revilla");
            insertarDirector(3,"Marco","Polo");
            insertarDirector(4,"Lolo","Perro");
            insertarDirector(5,"Coco","Loco");
            // ACTUALIZAR RELACION
            updateRel();
        } catch (Exception e) {
            e.printStackTrace();
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

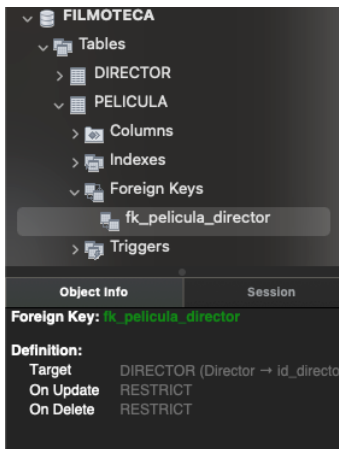
Añado la siguiente linea en el main antes de la ejecución:

```
public static GestionBBDD baseDatos = new GestionBBDD();
```

Ahora ejecuto el programa y compruebo que no tiene errores, me voy a MySQL y compruebo que se han creado correctamente las tablas, las relaciones y los registros:

	Titulo	Director	Pais	Duracion	Genero
▶	Terminator	1	EEUU	100	Accion
	ESDLA	2	EEUU	180	Fantasia
	Torrente	3	España	120	Comedia
	Otra Pelicula	4	Francia	200	Amor
	Programadores	5	Italia	120	Drama

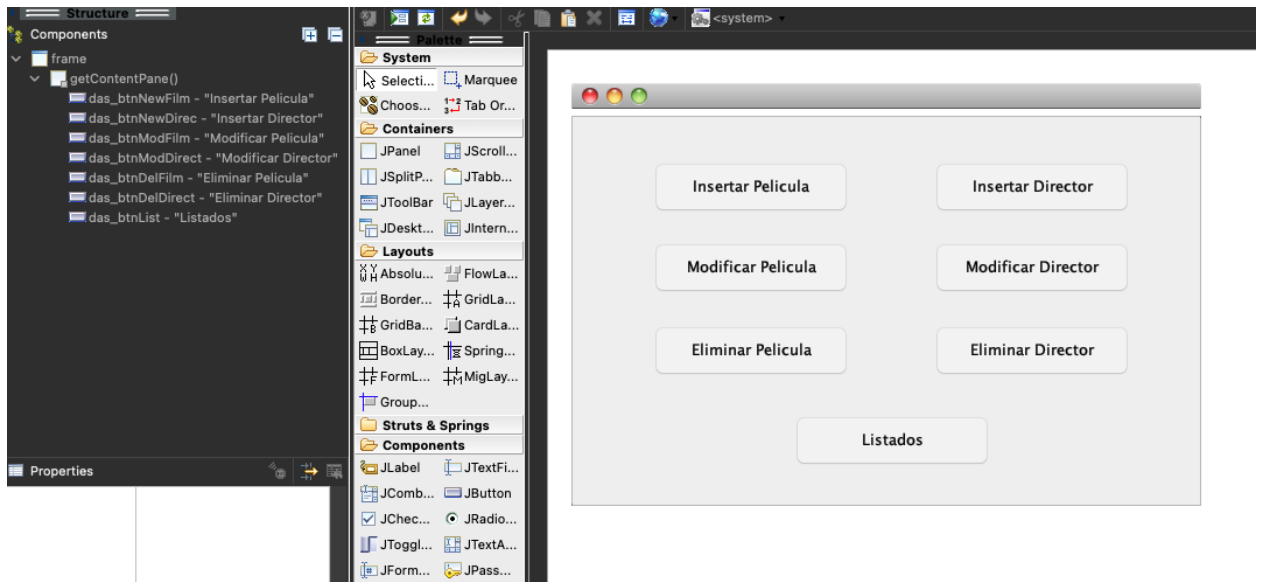
	id_director	Nombre	Apellidos
▶	1	Dani	Aguilar
	2	Alicia	Revilla
	3	Marco	Polo
	4	Lolo	Perro
	5	Coco	Loco
	NULL	NULL	NULL



1.4. Crear interfaz gráfica, menú principal

El menú se compone de un JFrame con 7 JButtons. He creado una clase llamada MenuPrincipal.java dentro de un package gui. La ejecución del programa la he dejado dentro de un Main el package filmoteca.

El menú principal y la estructura de variables quedaría de la siguiente forma:



Y el código actual sería el siguiente:

```
package filмотeca.gui;

import javax.swing.JFrame;
import javax.swing.JButton;

public class MenuPrincipal {

    public JFrame frame;

    public MenuPrincipal() {
        initialize();
    }

    /**
     * Initialize the contents of the frame.
     */
    private void initialize() {
        frame = new JFrame();
        frame.setBounds(100, 100, 540, 362);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.getContentPane().setLayout(null);

        JButton das_btnNewFilm = new JButton("Insertar Pelicula");
        das_btnNewFilm.setBounds(70, 39, 168, 45);
        frame.getContentPane().add(das_btnNewFilm);

        JButton das_btnNewDirec = new JButton("Insertar Director");
        das_btnNewDirec.setBounds(311, 39, 168, 45);
        frame.getContentPane().add(das_btnNewDirec);

        JButton das_btnModFilm = new JButton("Modificar Pelicula");
        das_btnModFilm.setBounds(70, 108, 168, 45);
        frame.getContentPane().add(das_btnModFilm);

        JButton das_btnModDirect = new JButton("Modificar Director");
        das_btnModDirect.setBounds(311, 108, 168, 45);
        frame.getContentPane().add(das_btnModDirect);
    }
}
```

```

JButton das_btnDelFilm = new JButton("Eliminar Pelicula");
das_btnDelFilm.setBounds(70, 179, 168, 45);
frame.getContentPane().add(das_btnDelFilm);

JButton das_btnDelDirect = new JButton("Eliminar Director");
das_btnDelDirect.setBounds(311, 179, 168, 45);
frame.getContentPane().add(das_btnDelDirect);

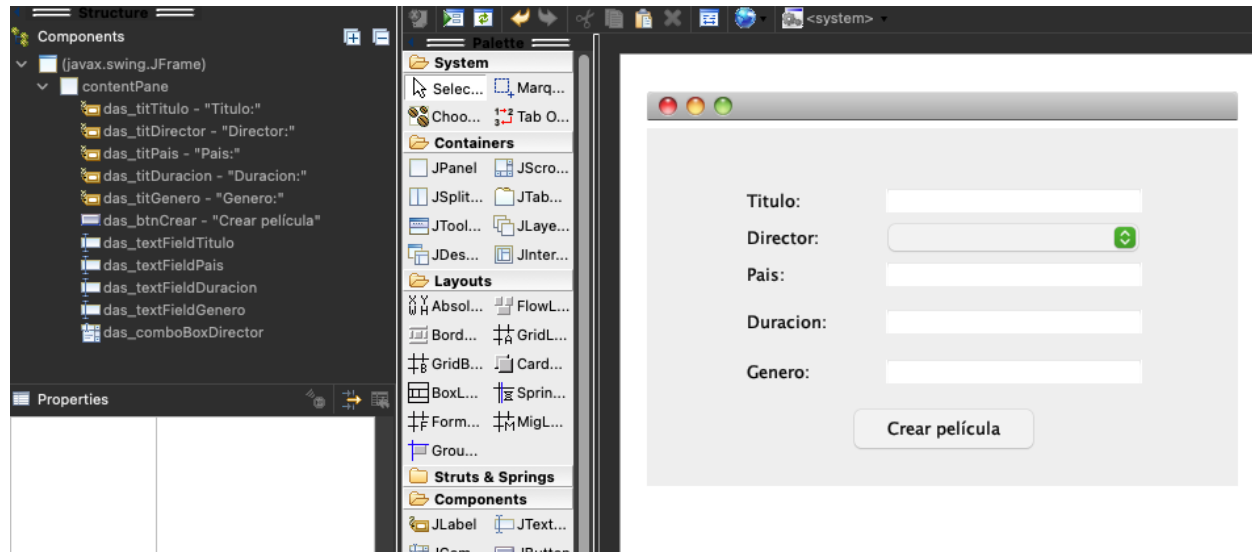
JButton das_btnList = new JButton("Listados");
das_btnList.setBounds(191, 256, 168, 45);
frame.getContentPane().add(das_btnList);
}

```

1.5. Insertar, eliminar y modificar película

1.5.1. Insertar Pelicula

Lo primero de todo creamos la ventana de insertar películas. Lo forman 5 JLabels, 1 JButton, 4 textFields y 1 ComboBox. Queda de la siguiente forma con las variables cambiadas:



Y el código actual sería el siguiente:

```

package filmoteca;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.JButton;
import javax.swing.JTextField;
import javax.swing.JComboBox;

public class InsertPelicula extends JFrame {

    private JPanel contentPane;
    private JTextField das_textFieldTitulo;
    private JTextField das_textFieldPais;
    private JTextField das_textFieldDuracion;

```

```

private JTextField das_textFieldGenero;

public InsertPelicula() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 450, 300);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    setContentPane(contentPane);
    contentPane.setLayout(null);

    JLabel das_titTitulo = new JLabel("Titulo:");
    das_titTitulo.setBounds(76, 47, 61, 16);
    contentPane.add(das_titTitulo);

    JLabel das_titDirector = new JLabel("Director:");
    das_titDirector.setBounds(76, 75, 61, 16);
    contentPane.add(das_titDirector);

    JLabel das_titPais = new JLabel("Pais:");
    das_titPais.setBounds(76, 103, 61, 16);
    contentPane.add(das_titPais);

    JLabel das_titDuracion = new JLabel("Duracion:");
    das_titDuracion.setBounds(76, 139, 61, 16);
    contentPane.add(das_titDuracion);

    JLabel das_titGenero = new JLabel("Genero:");
    das_titGenero.setBounds(76, 177, 61, 16);
    contentPane.add(das_titGenero);

    JButton das_btnCrear = new JButton("Crear película");
    das_btnCrear.setBounds(155, 211, 142, 36);
    contentPane.add(das_btnCrear);

    das_textFieldTitulo = new JTextField();
    das_textFieldTitulo.setBounds(179, 42, 201, 26);
    contentPane.add(das_textFieldTitulo);
    das_textFieldTitulo.setColumns(10);

    das_textFieldPais = new JTextField();
    das_textFieldPais.setBounds(179, 98, 201, 26);
    contentPane.add(das_textFieldPais);
    das_textFieldPais.setColumns(10);

    das_textFieldDuracion = new JTextField();
    das_textFieldDuracion.setBounds(179, 134, 201, 26);
    contentPane.add(das_textFieldDuracion);
    das_textFieldDuracion.setColumns(10);

    das_textFieldGenero = new JTextField();
    das_textFieldGenero.setBounds(179, 172, 201, 26);
    contentPane.add(das_textFieldGenero);
    das_textFieldGenero.setColumns(10);

    JComboBox das_comboBoxDirector = new JComboBox();
    das_comboBoxDirector.setBounds(179, 71, 201, 27);
    contentPane.add(das_comboBoxDirector);
}

```

Empezare por el combobox, primero creare una función en la clase GestionBBDD, esta función hara una consulta en la tabla directores y devolverá un arrayList con los directores actuales:

```

public static ArrayList<String> listarDatos() throws Exception {
    ArrayList<String> lista = new ArrayList<String>();
}

```

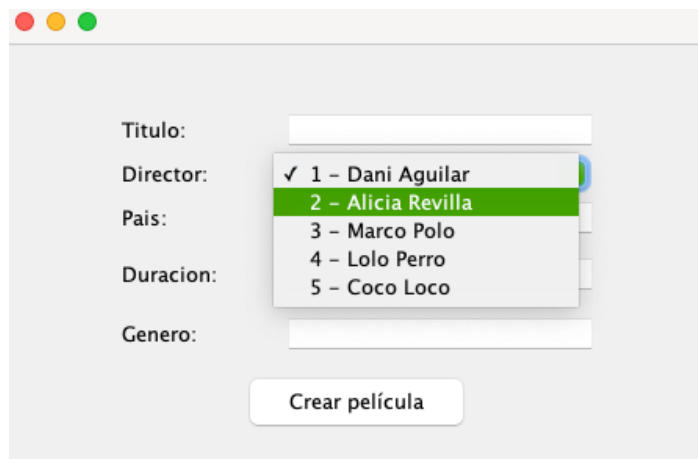
```
String query = "select * from DIRECTOR;";
Statement stmt = null;
ResultSet rs = null;
try {
    stmt = con.createStatement();
    rs = stmt.executeQuery(query);
    while(rs.next()){
        lista.add(rs.getInt(1)+ " - " + rs.getString(2) + " " +
rs.getString(3)); }
    } catch (SQLException e) { e.printStackTrace();
    } finally{
        rs.close();
        stmt.close(); }
return lista;
```

Ahora de nuevo en la clase InsertPelicula, creo un ArrayList, extraigo la información de la función que he creado, lo convierto en un array de String y con el creo el combobox actualizado:

```
ArrayList<String> directores = new ArrayList<String>();
try {
    directores = GestionBBDD.listarDatos();
} catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
// Conversion a array String del ArrayList
String[] prueba = directores.toArray(new String[directores.size()]);
// Creacion del combobox
JComboBox das_comboBoxDirector = new JComboBox(prueba);
das_comboBoxDirector.setBounds(179, 71, 201, 27);

contentPane.add(das_comboBoxDirector); }
```

Resultado:



Ahora procedo a la configuración para insertar películas. Usara la función de GestionBBDD insertarPelicula. El proceso pasara por una función externa que extraerá el primer carácter del director para poder usarla de identificador, la función es la siguiente:

```
public static int retornaNumDir(Object director) {  
    String direNum;  
    char caracter;  
    int numero;  
    direNum = (String)director;  
    caracter = direNum.charAt(0);  
    numero = Character.getNumericValue(caracter);  
    return numero;  
}
```

Después completa los datos necesarios de la función para crear la película, pasando por el parseado a int de la duración, quedando finalmente la función de la siguiente forma:

```
JButton das_btnCrear = new JButton("Crear película");  
das_btnCrear.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
        // Seleccion primer caracter director  
        int numero;  
        numero =  
Funciones.retornaNumDir(das_comboBoxDirector.getSelectedItem());  
        // Añadir película  
        try {  
            GestionBBDD.insertarPelicula(das_textFieldTitulo.getText(),  
numero, das_textFieldPais.getText(), Integer.parseInt(das_textFieldDuracion.getText()),  
das_textFieldGenero.getText());  
        } catch (NumberFormatException e1) {  
            // TODO Auto-generated catch block  
            e1.printStackTrace();  
        } catch (SQLException e1) {  
            // TODO Auto-generated catch block  
            e1.printStackTrace();  
        }  
    }  
});
```

RESULTADO:



The screenshot shows a Java Swing window with a light gray background and a title bar with red, yellow, and green buttons. The window contains a form with the following fields and values:

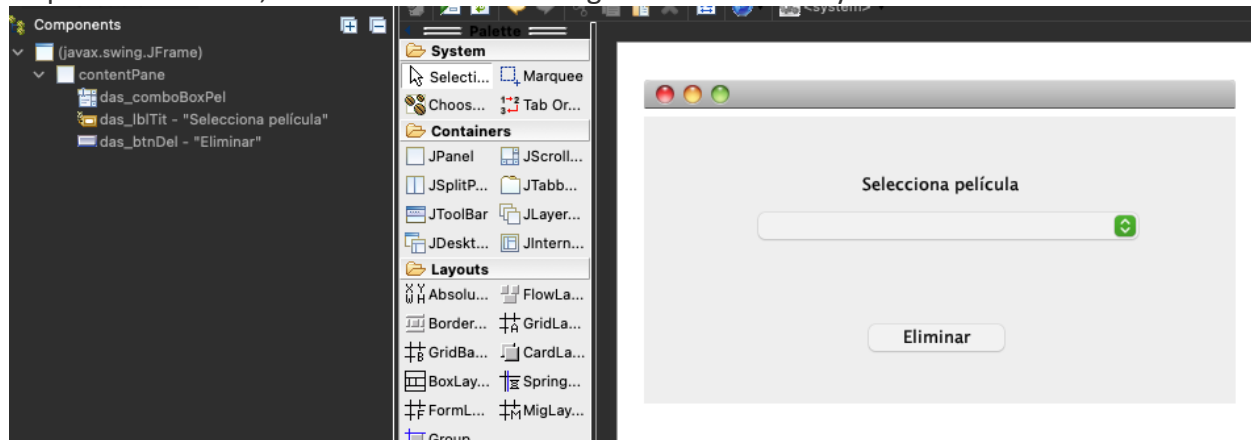
- Titulo: Nueva
- Director: 1 - Dani Aguilar (with a dropdown arrow)
- Pais: España
- Duracion: 200
- Genero: Comedia

At the bottom of the form is a button labeled "Crear película".

Título	Director	Pais	Duracion	Genero	
Terminator	1	EEUU	100	Accion	
ESDLA	2	EEUU	180	Fantasia	
Torrente	3	España	120	Comedia	
Otra Pelicula	4	Francia	200	Amor	
Programadores	5	Italia	120	Drama	
Nueva	1	España	200	Comedia	

1.5.2. Eliminar película

Lo primero de todo, creo la ventana con la siguiente estructura y variables:



El código queda de la siguiente forma:

```
package filmoteca;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JComboBox;
import javax.swing.JLabel;
import javax.swing.SwingConstants;
import javax.swing.JButton;

public class DelPelicula extends JFrame {

    private JPanel contentPane;

    public DelPelicula() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 450, 246);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        JComboBox das_comboBoxPel = new JComboBox();
        das_comboBoxPel.setBounds(82, 71, 300, 27);
        contentPane.add(das_comboBoxPel);

        JLabel das_lblTit = new JLabel("Selecciona película");
```



```
das_lblTit.setHorizontalAlignment(SwingConstants.CENTER);
das_lblTit.setBounds(6, 43, 438, 16);
contentPane.add(das_lblTit);

JButton das_btnDel = new JButton("Eliminar");
das_btnDel.setBounds(164, 154, 117, 29);
contentPane.add(das_btnDel);
```

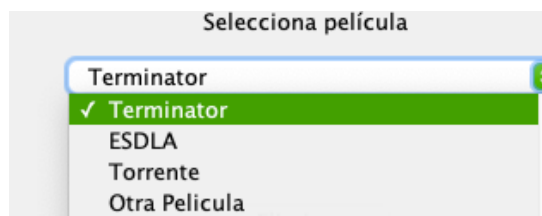
Para listar las películas para el comboBox, será necesaria una función, estará en GestionBBDD, será la siguiente:

```
public static ArrayList<String> listarPelículas() throws Exception {
    ArrayList<String> lista = new ArrayList<String>();
    String query = "select * from PELICULA;";
    Statement stmt = null;
    ResultSet rs = null;
    try {
        stmt = con.createStatement();
        rs = stmt.executeQuery(query);
        while(rs.next()){
            lista.add(rs.getString(1)); }
        } catch (SQLException e) { e.printStackTrace(); }
        finally{
            rs.close();
            stmt.close(); }
        return lista;
    }
```

Y para listar el ComboBox:

```
// COMBOBOX
// Extraccion de peliculas
ArrayList<String> peliculas = new ArrayList<String>();
try {
    peliculas = GestionBBDD.listarPelículas();
} catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
// Conversion a array String del ArrayList
String[] arrayCombo = peliculas.toArray(new String[peliculas.size()]);
// Creacion del comboBox
JComboBox das_comboBoxPel = new JComboBox(arrayCombo);
das_comboBoxPel.setBounds(82, 71, 300, 27);

contentPane.add(das_comboBoxPel);
```



Ahora voy a añadir la función de eliminar a la clase de GestiónBBDD, esta recibe por parámetro el nombre de la película y lo usa para borrar el registro:

```
public static void delPelícula(String titulo) throws SQLException{
```

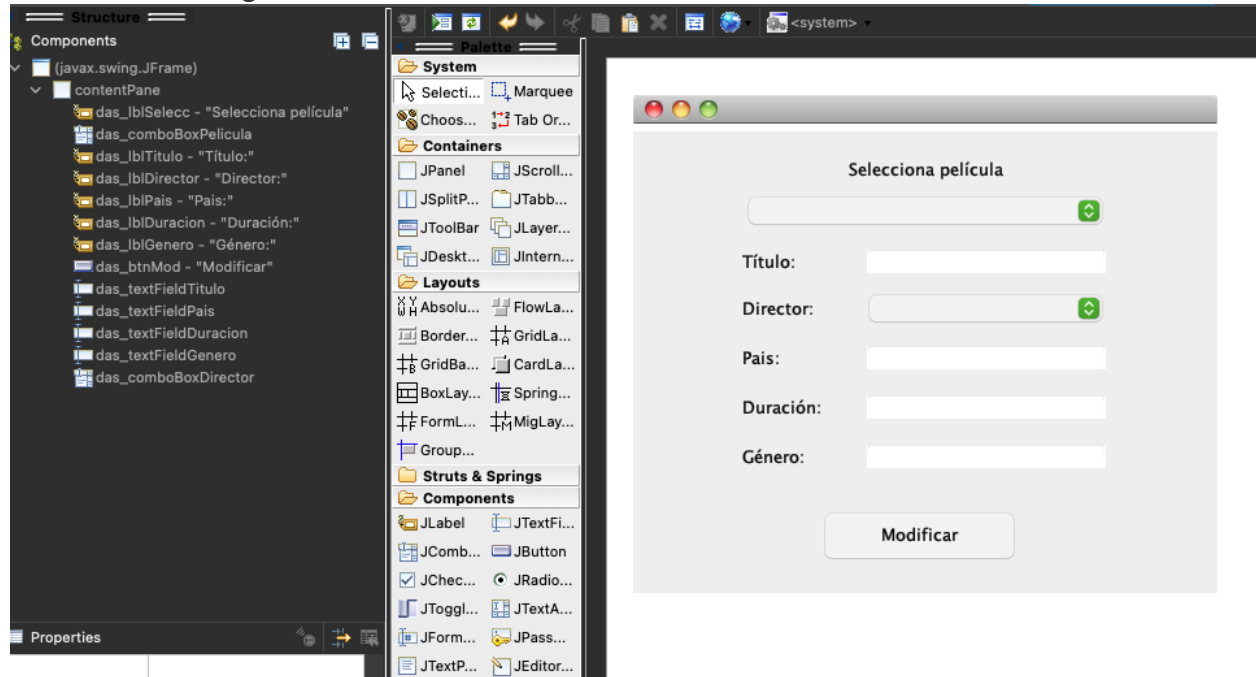
```
String query = "DELETE FROM PELICULA WHERE Titulo='"+ titulo +"'";
Statement stmt = null;
try {
    stmt = con.createStatement();
    stmt.executeUpdate(query);
} catch (SQLException e) {
    e.printStackTrace();
} finally {
    stmt.close();
}
```

En la ventana de DelPelicula, lo configuro de la siguiente forma, y una vez borrada, muestra un mensaje de aviso y se cierra la ventana:

```
public void actionPerformed(ActionEvent e) {
    try {
        GestionBBDD.delPelicula((String)das_comboBoxPel.getSelectedItem());
        JDialog das_avisos = new JDialog();
        JLabel das_mensaje = new JLabel("Película eliminada");
        das_avisos.getContentPane().add(das_mensaje);
        das_avisos.setSize(300, 100);
        das_avisos.setLocationRelativeTo(null);
        das_avisos.setVisible(true);
        das_avisos.dispose();
    } catch (SQLException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }
}
```

1.5.3. Modificar película

Para modificar películas lo primero creo la ventana correspondiente quedando la estructura de la siguiente manera:



Y el código:

```
package filmoteca;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.SwingConstants;
import javax.swing.JComboBox;
import javax.swing.JButton;
import javax.swing.JTextField;

public class ModPelicula extends JFrame {

    private JPanel contentPane;
    private JTextField das_textFieldTitulo;
    private JTextField das_textFieldPais;
    private JTextField das_textFieldDuracion;
    private JTextField das_textFieldGenero;

    public ModPelicula() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 450, 383);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        JLabel das_lblSelecc = new JLabel("Selecciona película");
        das_lblSelecc.setHorizontalAlignment(SwingConstants.CENTER);
        das_lblSelecc.setBounds(6, 21, 438, 16);
        contentPane.add(das_lblSelecc);

        JComboBox das_comboBoxPelicula = new JComboBox();
        das_comboBoxPelicula.setBounds(84, 49, 283, 27);
        contentPane.add(das_comboBoxPelicula);

        JLabel das_lblTitulo = new JLabel("Título:");
        das_lblTitulo.setBounds(84, 92, 61, 16);
        contentPane.add(das_lblTitulo);

        JLabel das_lblDirector = new JLabel("Director:");
        das_lblDirector.setBounds(84, 128, 61, 16);
        contentPane.add(das_lblDirector);

        JLabel das_lblPais = new JLabel("País:");
        das_lblPais.setBounds(84, 166, 61, 16);
        contentPane.add(das_lblPais);

        JLabel das_lblDuracion = new JLabel("Duración:");
        das_lblDuracion.setBounds(84, 204, 81, 16);
        contentPane.add(das_lblDuracion);

        JLabel das_lblGenero = new JLabel("Género:");
        das_lblGenero.setBounds(84, 242, 61, 16);
        contentPane.add(das_lblGenero);

        JButton das_btnMod = new JButton("Modificar");
        das_btnMod.setBounds(145, 291, 151, 41);
        contentPane.add(das_btnMod);

        das_textFieldTitulo = new JTextField();
        das_textFieldTitulo.setBounds(177, 87, 190, 26);
```

```

contentPane.add(das_textFieldTitulo);
das_textFieldTitulo.setColumns(10);

das_textFieldPais = new JTextField();
das_textFieldPais.setBounds(177, 161, 190, 26);
contentPane.add(das_textFieldPais);
das_textFieldPais.setColumns(10);

das_textFieldDuracion = new JTextField();
das_textFieldDuracion.setBounds(177, 199, 190, 26);
contentPane.add(das_textFieldDuracion);
das_textFieldDuracion.setColumns(10);

das_textFieldGenero = new JTextField();
das_textFieldGenero.setBounds(177, 237, 190, 26);
contentPane.add(das_textFieldGenero);
das_textFieldGenero.setColumns(10);

JComboBox das_comboBoxDirector = new JComboBox();
das_comboBoxDirector.setBounds(177, 124, 190, 27);
contentPane.add(das_comboBoxDirector);
}

```

Para el combobox de las películas uso el código que utilice anteriormente:

```

// COMBOBOX PELICULAS
// Extraccion de peliculas
ArrayList<String> peliculas = new ArrayList<String>();
try {
    peliculas = GestionBBDD.listarPeliculas();
} catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
// Conversion a array String del ArrayList
String[] arrayCombo = peliculas.toArray(new
String[peliculas.size()]);

```

Lo mismo para el de directores:

```

// COMBOBOX
// Extraccion de directores
ArrayList<String> directores = new ArrayList<String>();
try {
    directores = GestionBBDD.listarDirectores();
} catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
// Conversion a array String del ArrayList
String[] arrayComboDirectores = directores.toArray(new String[directores.size()]);
// Creacion del combobox
JComboBox das_comboBoxDirector = new JComboBox(arrayComboDirectores);
das_comboBoxDirector.setBounds(177, 124, 190, 27);

contentPane.add(das_comboBoxDirector);

```

En GestionBBDD creo la función que se encargara de modificar, será de la siguiente forma, que recibirá por parámetros la selección de la película, el título, el director, el país, la duración y el género:

```

        public static void modPelicula(String seleccion, String titulo, int director,
String pais, int duracion, String genero) throws SQLException {
        String query = "UPDATE PELICULA SET Titulo='"+ titulo + "', Director='"+
director + ", Pais='"+ pais + "', Duracion=" + duracion + ", Genero='" + genero + "' WHERE
Titulo='"+seleccion+"'";
        Statement stmt = null;
        try {
            stmt = con.createStatement();
            stmt.executeUpdate(query);
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            stmt.close();
        }
    }
}

```

Y por último, programo el botón para realizar la acción, de nuevo utilizo la función para extraer el primer carácter del director y con ello su identificador. Queda la función de la siguiente forma:

```

JButton das_btnMod = new JButton("Modificar");
das_btnMod.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        // Seleccion primer caracter director
        int numero;
        numero =
Funciones.retornaNumDir(das_comboBoxDirector.getSelectedItem());
        // Modificar pelicula
        try {
            GestionBBDD.modPelicula((String)das_comboBoxPelicula.getSelectedItem(),
das_textFieldTitulo.getText(), numero, das_textFieldPais.getText(),
Integer.parseInt(das_textFieldDuracion.getText()), das_textFieldGenero.getText());
            JDialog das_avisos = new JDialog();
            JLabel das_mensaje = new JLabel("Película modificada");
            das_avisos.getContentPane().add(das_mensaje);
            das_avisos.setSize(300, 100);
            das_avisos.setLocationRelativeTo(null);
            das_avisos.setVisible(true);
            das_avisos.dispose();
        } catch (NumberFormatException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        } catch (SQLException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
    }
});
}
}

```

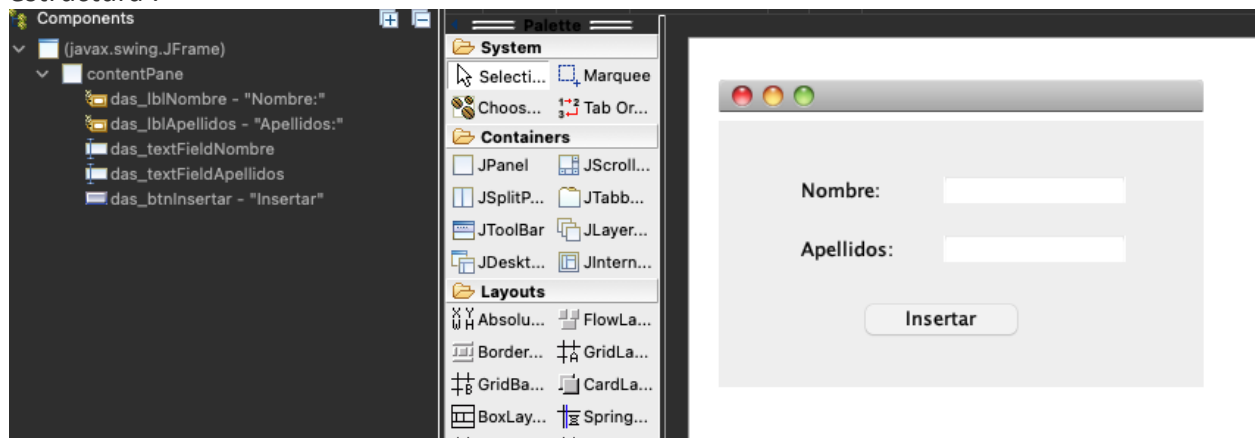
Para comprobar que funciona, modifico la siguiente película y compruebo su estado en MySQL:

Título	Director	Pais	Duracion	Genero
Terminator	1	EEUU	100	Accion
El Señor de los Anillos	4	España	300	Aventura
Torrente	3	España	120	Comedia
Otra Pelicula	4	Francia	200	Amor

1.6. Insertar, eliminar y modificar director

1.6.1. Insertar director

Lo primero de todo creo la ventana de insertar director, tendrá la siguiente estructura :



Y el código será el siguiente:

```
package filmoteca;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.JTextField;
```

```
import javax.swing.JButton;

public class InsertDirector extends JFrame {

    private JPanel contentPane;
    private JTextField das_textFieldNombre;
    private JTextField das_textFieldApellidos;

    public InsertDirector() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 330, 209);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        JLabel das_lblNombre = new JLabel("Nombre:");
        das_lblNombre.setBounds(56, 39, 61, 16);
        contentPane.add(das_lblNombre);

        JLabel das_lblApellidos = new JLabel("Apellidos:");
        das_lblApellidos.setBounds(56, 79, 82, 16);
        contentPane.add(das_lblApellidos);

        das_textFieldNombre = new JTextField();
        das_textFieldNombre.setBounds(150, 34, 130, 26);
        contentPane.add(das_textFieldNombre);
        das_textFieldNombre.setColumns(10);

        das_textFieldApellidos = new JTextField();
        das_textFieldApellidos.setBounds(150, 74, 130, 26);
        contentPane.add(das_textFieldApellidos);
        das_textFieldApellidos.setColumns(10);

        JButton das_btnInsertar = new JButton("Insertar");
        das_btnInsertar.setBounds(93, 121, 117, 29);
        contentPane.add(das_btnInsertar);
    }
}
```

Para la función de insertar, usamos la existente de GestionBBDD, la configuramos en el botón insertando en la función la información recibida de los textField:

NOTA*

He visto que estaba utilizando la función pasándole directamente por parámetro el ID siendo auto-increment, lo he cambiado a Null, ahora mismo la función para añadir directores es así:

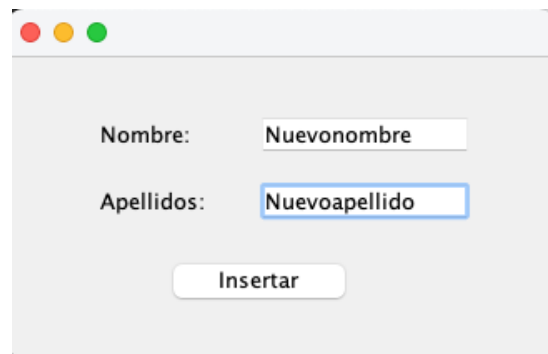
```
// DIRECTOR
public void insertarDirector(String nombre, String apellido) throws SQLException{
    String query = "insert into DIRECTOR values(null, '" + nombre + "', '" +
apellido + "')";

    Statement stmt = null;
    try {
        stmt = con.createStatement();
        stmt.executeUpdate(query);
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        stmt.close();
    }
}
```

Una vez cambiado esto, añado la función al botón:

```
JButton das_btnInsertar = new JButton("Insertar");
das_btnInsertar.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        try {
            GestionBBDD.insertarDirector(das_textFieldNombre.getText(),
das_textFieldApellidos.getText());
        } catch (SQLException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
    }
});
```

Compruebo que funcione:



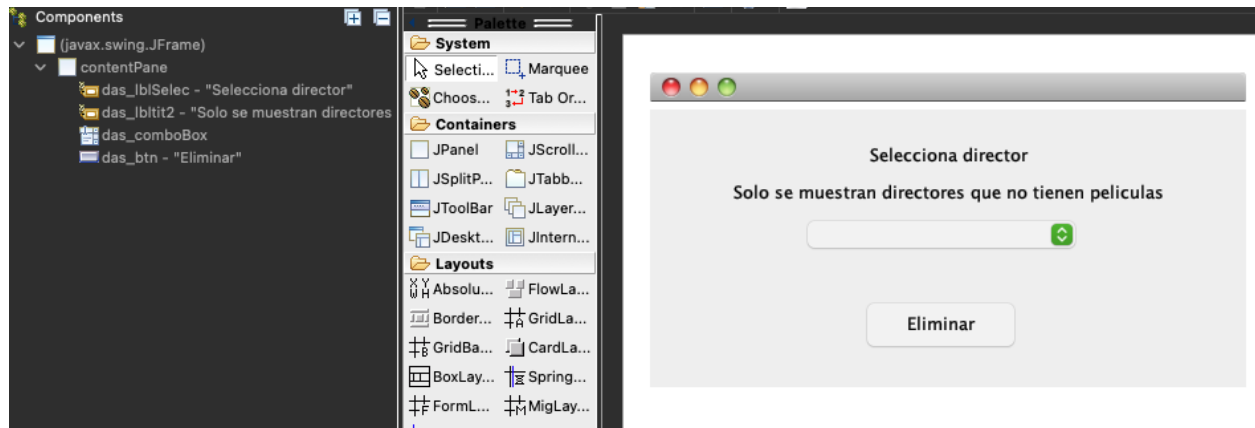
A screenshot of a Java Swing window. It contains two text input fields. The first is labeled 'Nombre:' and contains the text 'Nuevonombre'. The second is labeled 'Apellidos:' and contains the text 'Nuevoapellido'. Below these fields is a button labeled 'Insertar'.



A screenshot of a Java Swing window showing a list of directors. The list is titled 'Director:' and contains six items: '1 - Dani Aguilar', '2 - Alicia Revilla', '3 - Marco Polo', '4 - Lolo Perro', '5 - Coco Loco', and '6 - Nuevonombre Nuevoapellido'. The last item is highlighted in green. Below the list are labels for 'Pais:', 'Duración:', and 'Género:'.

1.6.2. Eliminar director

Lo primero, creo la interfaz y su estructura:



Para el combobox, será necesario un select que filtre por los directores que no tengan películas, la función que lo realizara será la siguiente:

```
public static ArrayList<String> listarDirectoresSinPelículas() throws Exception {
    ArrayList<String> lista = new ArrayList<String>();
    String query = "SELECT DIRECTOR.* FROM DIRECTOR WHERE DIRECTOR.id_director
NOT IN (SELECT DIRECTOR.id_director FROM PELICULA);";
    Statement stmt = null;
    ResultSet rs = null;
    try {
        stmt = con.createStatement();
        rs = stmt.executeQuery(query);
        while(rs.next()){
            lista.add(rs.getString(1) + " - " + rs.getString(2) + " " +
rs.getString(3)); }
        } catch (SQLException e) { e.printStackTrace();
        } finally{
            rs.close();
            stmt.close(); }
    return lista;
}
```

El combobox quedaría de la siguiente manera:

```
// COMBOBOX DIRECTORES
// Extraccion de directores
ArrayList<String> directores = new ArrayList<String>();
try {
    directores = GestionBBDD.listarDirectoresSinPelículas();
} catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

// Conversion a array String del ArrayList
String[] arrayComboDirectores = directores.toArray(new String[directores.size()]);
// Creacion del combobox
JComboBox das_comboBox = new JComboBox(arrayComboDirectores);
das_comboBox.setBounds(114, 82, 213, 27);

contentPane.add(das_comboBox);
```

Y nos quedaría configurar una función que borre registros de directores, lo haremos en GestionBBDD:

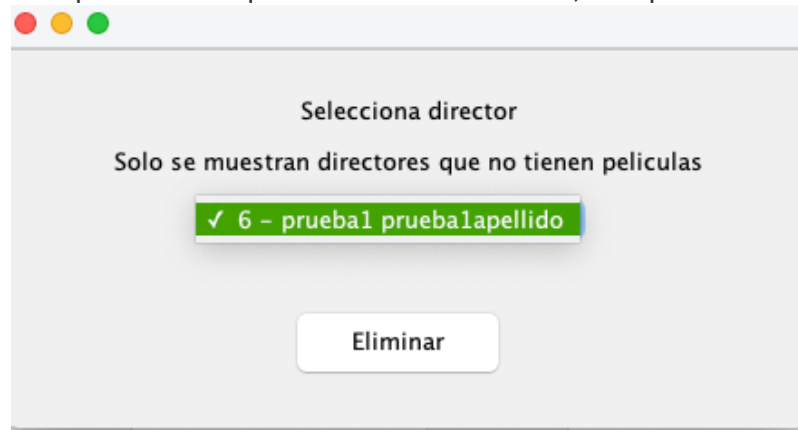
```
public static void delDirector(int id) throws SQLException{
```

```
String query = "DELETE FROM DIRECTOR WHERE id_director="+ id +"";
Statement stmt = null;
try {
    stmt = con.createStatement();
    stmt.executeUpdate(query);
} catch (SQLException e) {
    e.printStackTrace();
} finally {
    stmt.close();
}
```

De nuevo hare uso de la función para convertir el primer carácter del combobox en un int para pasarlo a la función y borrarlo con el ID, la función del botón quedaría de la siguiente forma:

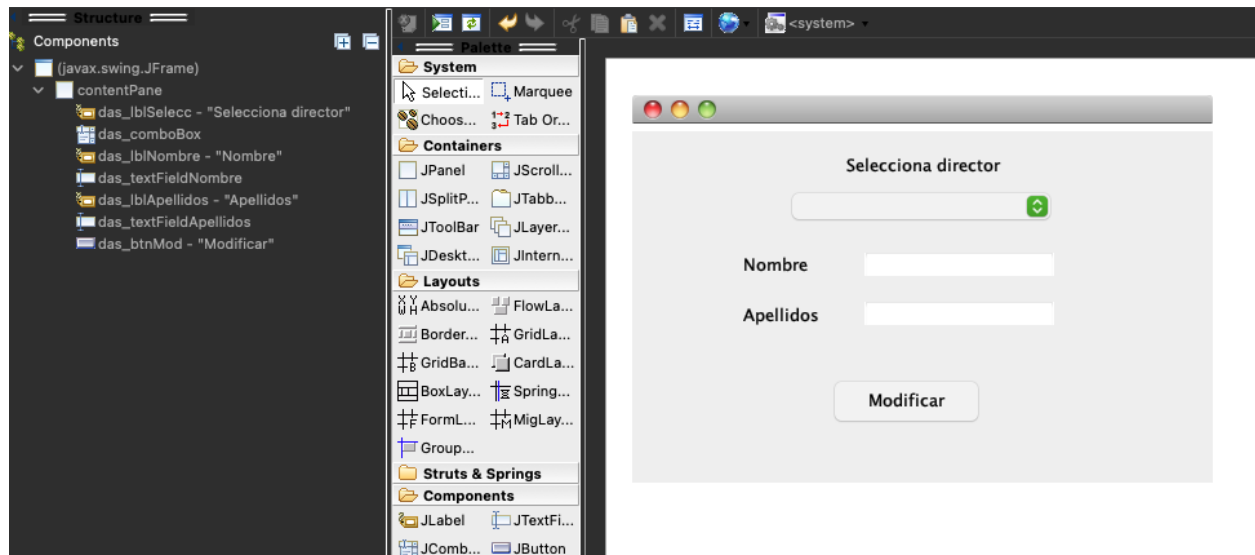
```
public void actionPerformed(ActionEvent e) {
    // Selecccion primer caracter director
    int numero;
    numero = Funciones.retornaNumDir(das_comboBox.getSelectedIndex());
    try {
        GestionBBDD.delDirector(numero);
        JDialog das_aviso = new JDialog();
        JLabel das_mensaje = new JLabel("Director eliminado");
        das_aviso.getContentPane().add(das_mensaje);
        das_aviso.setSize(300, 100);
        das_aviso.setLocationRelativeTo(null);
        das_aviso.setVisible(true);
        das_aviso.dispose();
    } catch (SQLException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }
}
```

Hago una prueba para comprobar que funcióne, primero creo un director y ahora veo que solo me aparece el director creado, sin aparecer los demás directores:



1.6.3. Modificar director

Lo primero creo la estructura de la ventana, queda de la siguiente manera:



Y el código:

```
package filmoteca;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.SwingConstants;
import javax.swing.JComboBox;
import javax.swing.JTextField;
import javax.swing.JButton;

public class ModDirector extends JFrame {

    private JPanel contentPane;
    private JTextField das_textFieldNombre;
    private JTextField das_textFieldApellidos;

    public ModDirector() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 450, 300);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        JLabel das_lblSelecc = new JLabel("Selecciona director");
        das_lblSelecc.setHorizontalAlignment(SwingConstants.CENTER);
        das_lblSelecc.setBounds(6, 18, 438, 16);
        contentPane.add(das_lblSelecc);

        JComboBox das_comboBox = new JComboBox();
        das_comboBox.setBounds(119, 46, 211, 27);
        contentPane.add(das_comboBox);

        JLabel das_lblNombre = new JLabel("Nombre");
        das_lblNombre.setBounds(86, 95, 61, 16);
        contentPane.add(das_lblNombre);

        das_textFieldNombre = new JTextField();
        das_textFieldNombre.setBounds(177, 90, 153, 26);
    }
}
```

```

contentPane.add(das_textFieldNombre);
das_textFieldNombre.setColumns(10);

JLabel das_lblApellidos = new JLabel("Apellidos");
das_lblApellidos.setBounds(86, 134, 61, 16);
contentPane.add(das_lblApellidos);

das_textFieldApellidos = new JTextField();
das_textFieldApellidos.setBounds(177, 128, 153, 26);
contentPane.add(das_textFieldApellidos);
das_textFieldApellidos.setColumns(10);

JButton das_btnMod = new JButton("Modificar");
das_btnMod.setBounds(154, 191, 117, 37);
contentPane.add(das_btnMod);

```

Para el combobox:

```

// COMBOBOX DIRECTORES
// Extraccion de directores
ArrayList<String> directores = new ArrayList<String>();
try {
    directores = GestionBBDD.listarDirectores();
} catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

// Conversion a array String del ArrayList
String[] arrayComboDirectores = directores.toArray(new String[directores.size()]);
// Creacion del combobox
JComboBox das_comboBox = new JComboBox(arrayComboDirectores);
das_comboBox.setBounds(119, 46, 211, 27);
contentPane.add(das_comboBox);

```

Ahora para modificar, necesito una función nueva en GestionBBDD que reciba el nombre, los apellidos, y el id de director que lo extraerá de nuevo de la función utilizada anteriormente, la creo de la siguiente forma:

```

public static void modDirector(String nombre, String apellidos, int id) throws
SQLException {
    String query = "UPDATE DIRECTOR SET Nombre='" + nombre + "', Apellidos='" +
apellidos + "' WHERE id_director=" + id + ";";
    Statement stmt = null;
    try {
        stmt = con.createStatement();
        stmt.executeUpdate(query);
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        stmt.close();
    }
}

```

Y por último, la vinculo con el botón:

```

public void actionPerformed(ActionEvent e) {
    // Seleccion primer caracter director
    int numero;
    numero = Funciones.retornaNumDir(das_comboBox.getSelectedItem());
    try {

```

```
        GestionBBDD.modDirector(das_textFieldNombre.getText(),  
das_textFieldApellidos.getText(), numero);  
        JDialog das_avisos = new JDialog();  
        JLabel das_mensaje = new JLabel("Director modificado");  
        das_avisos.getContentPane().add(das_mensaje);  
        das_avisos.setSize(300, 100);  
        das_avisos.setLocationRelativeTo(null);  
        das_avisos.setVisible(true);  
        das_avisos.dispose();  
    } catch (SQLException e1) {  
        // TODO Auto-generated catch block  
        e1.printStackTrace();  
    }  
}
```

Para hacer la prueba, modifico el director con el número 1 y compruebo que funciona:



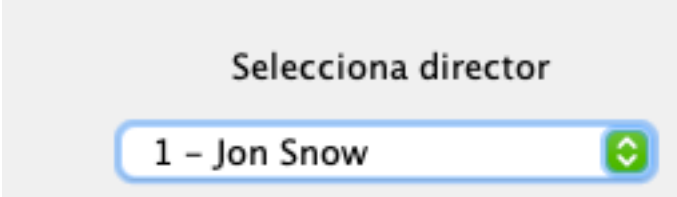
Selecciona director

1 - Daniel Aguilar

Nombre Jon

Apellidos Snow

Modificar

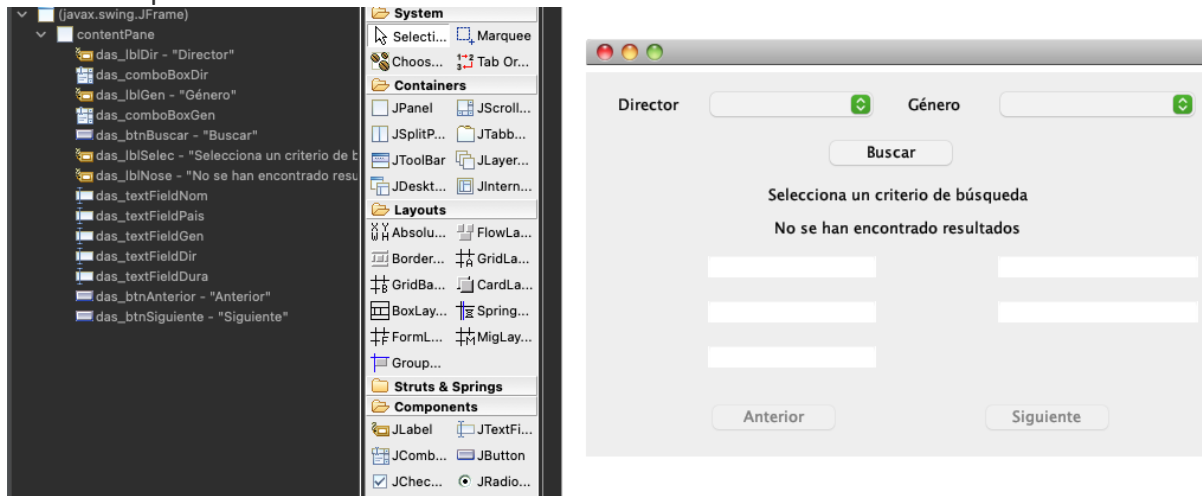


Selecciona director

1 - Jon Snow

1.7. Listados

Lo primero creo la estructura de la ventana:



Empezamos por los comboBox, para el de directores:

```
// COMBOBOX DIRECTORES
// Extraccion de directores
ArrayList<String> directores = new ArrayList<String>();
try {
    directores = GestionBBDD.listarDirectores();
} catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

// Conversion a array String del ArrayList
String[] arrayComboDirectores = directores.toArray(new String[directores.size()]);
// Creacion del combobox
JComboBox das_comboBoxDir = new JComboBox(arrayComboDirectores);
```

Para el de género hay que crear una función nueva en GestionBBDD:

```
// LISTAR GENEROS
// Funcion que devuelve un array con la informacion de la tabla PELICULA
public static ArrayList<String> listarGeneros() throws Exception {
    ArrayList<String> lista = new ArrayList<String>();
    String query = "select * from PELICULA;";
    Statement stmt = null;
    ResultSet rs = null;
    try {
        stmt = con.createStatement();
        rs = stmt.executeQuery(query);
        while(rs.next()){
            lista.add(rs.getString(5));
        } catch (SQLException e) { e.printStackTrace(); }
    } finally{
        rs.close();
        stmt.close();
    }
    return lista;
}
```

Y para crear el comboBox

```
// COMBOBOX GENEROS
// Extraccion de directores
ArrayList<String> generos = new ArrayList<String>();
try {
    generos = GestionBBDD.listarGeneros();
} catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

// Conversion a array String del ArrayList
String[] arrayComboGeneros = generos.toArray(new String[generos.size()]);
// Creacion del combobox
JComboBox das_comboBoxGen = new JComboBox(arrayComboGeneros);
```

Para poder hacer las búsquedas, añado una conexión y un resultSet en esta ventana:

```
// CONEXION
// Añado una conexion a la BBDD y a ResultSet aqui
Connection con=null;
ResultSet rs=null;
Statement stmt = null;
private static String das_datosConexion = "jdbc:mysql://127.0.0.1:8889/";
private static String das_baseDatos = "FILMOTECA";
private static String das_usuario = "root";
private static String das_password = "4246";
```

Empiezo con los diferentes SELECT, hay 4 casuísticas:

- Se selecciona solo director
- Se selecciona solo género
- Se selecciona Director y género
- No se selecciona nada

En el caso de que las búsquedas no arrojen resultados, se muestra un mensaje en la pantalla.

Ahora voy a crear 4 filtrados diferentes:

Si no se selecciona nada:

```
if (das_comboBoxDir.getSelectedIndex() == -1 &&
das_comboBoxGen.getSelectedIndex() == -1) {
    query = "SELECT P.Titulo, P.Pais, P.Genero, P.Duracion,
CONCAT(D.Nombre,' ',D.Apellidos) FROM DIRECTOR D INNER JOIN PELICULA P ON
D.id_director=P.Director";
    Statement stmt =
con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                    ResultSet.CONCUR_READ_ONLY);
    rs = stmt.executeQuery(query);

    if (rs.next()) {
        das_btnSiguiente.setEnabled(true);
        das_textFieldNom.setText(rs.getString(1));
        das_textFieldPais.setText(rs.getString(2));
        das_textFieldGen.setText(rs.getString(3));
    }
}
```

```

        das_textFieldDura.setText(rs.getString(4));
        das_textFieldDir.setText(rs.getString(5));
        das_lblNose.setVisible(false);
    } else {
        das_btnSiguiente.setEnabled(false);
        das_lblNose.setVisible(true);
        das_textFieldNom.setText("");
        das_textFieldPais.setText("");
        das_textFieldGen.setText("");
        das_textFieldDura.setText("");
        das_textFieldDir.setText("");
    }
}

```

Si se selecciona solo director:

```

    } else if (das_comboBoxDir.getSelectedIndex() >= 0 &&
das_comboBoxGen.getSelectedIndex() == -1) {
        int numero;
        numero =
Funciones.retornaNumDir(das_comboBoxDir.getSelectedItem());
        query = "SELECT P.Titulo, P.Pais, P.Genero, P.Duracion,
CONCAT(D.Nombre, ' ', D.Apellidos) FROM DIRECTOR D INNER JOIN PELICULA P ON
D.id_director=P.Director WHERE D.id_director=" + numero + ";";
        Statement stmt =
con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                    ResultSet.CONCUR_READ_ONLY);
        rs = stmt.executeQuery(query);
        if (rs.next()) {
            das_btnSiguiente.setEnabled(true);
            das_textFieldNom.setText(rs.getString(1));
            das_textFieldPais.setText(rs.getString(2));
            das_textFieldGen.setText(rs.getString(3));
            das_textFieldDura.setText(rs.getString(4));
            das_textFieldDir.setText(rs.getString(5));
            das_lblNose.setVisible(false);
        } else {
            das_btnSiguiente.setEnabled(false);
            das_lblNose.setVisible(true);
            das_textFieldNom.setText("");
            das_textFieldPais.setText("");
            das_textFieldGen.setText("");
            das_textFieldDura.setText("");
            das_textFieldDir.setText("");
        }
    }
}

```

Si se selecciona solo genero:

```

    } else if (das_comboBoxDir.getSelectedIndex() == -1 &&
das_comboBoxGen.getSelectedIndex() >= 0) {
        query = "SELECT P.Titulo, P.Pais, P.Genero, P.Duracion,
CONCAT(D.Nombre, ' ', D.Apellidos) FROM DIRECTOR D INNER JOIN PELICULA P ON
D.id_director=P.Director WHERE P.Genero='" + (String)das_comboBoxGen.getSelectedItem() + "';";
        Statement stmt =
con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                    ResultSet.CONCUR_READ_ONLY);
        rs = stmt.executeQuery(query);
        if (rs.next()) {
            das_btnSiguiente.setEnabled(true);
            das_textFieldNom.setText(rs.getString(1));
            das_textFieldPais.setText(rs.getString(2));
            das_textFieldGen.setText(rs.getString(3));
            das_textFieldDura.setText(rs.getString(4));
            das_textFieldDir.setText(rs.getString(5));
            das_lblNose.setVisible(false);
        } else {
            das_btnSiguiente.setEnabled(false);
            das_lblNose.setVisible(true);
        }
    }
}

```



```
das_textFieldNom.setText("");  
das_textFieldPais.setText("");  
das_textFieldGen.setText("");  
das_textFieldDura.setText("");  
das_textFieldDir.setText("");
```

Si están seleccionados genero y director:

```
    } else {  
        int numero;  
        numero =  
Funciones.retornaNumDir(das_comboBoxDir.getSelectedIndex());  
        query = "SELECT P.Titulo, P.Pais, P.Genero, P.Duracion,  
CONCAT(D.Nombre, ' ', D.Apellidos) FROM DIRECTOR D INNER JOIN PELICULA P ON  
D.id_director=P.Director WHERE D.id_director="+ numero + " AND P.Genero='"+  
(String)das_comboBoxGen.getSelectedIndex() +"'";  
        Statement stmt =  
con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,  
                    ResultSet.CONCUR_READ_ONLY);  
        rs = stmt.executeQuery(query);  
        if (rs.next()) {  
            das_btnSiguiente.setEnabled(true);  
            das_textFieldNom.setText(rs.getString(1));  
            das_textFieldPais.setText(rs.getString(2));  
            das_textFieldGen.setText(rs.getString(3));  
            das_textFieldDura.setText(rs.getString(4));  
            das_textFieldDir.setText(rs.getString(5));  
            das_lb1Nose.setVisible(false);  
        } else {  
            das_btnSiguiente.setEnabled(false);  
            das_lb1Nose.setVisible(true);  
            das_textFieldNom.setText("");  
            das_textFieldPais.setText("");  
            das_textFieldGen.setText("");  
            das_textFieldDura.setText("");  
            das_textFieldDir.setText("");  
        }  
    }
```

CODIGO COMPLETO DEL BOTÓN:

```
public void actionPerformed(ActionEvent e) {  
    String query;  
    try {  
        // Si los dos combobox están vacíos  
        if (das_comboBoxDir.getSelectedIndex() == -1 &&  
das_comboBoxGen.getSelectedIndex() == -1) {  
            query = "SELECT P.Titulo, P.Pais, P.Genero, P.Duracion,  
CONCAT(D.Nombre, ' ', D.Apellidos) FROM DIRECTOR D INNER JOIN PELICULA P ON  
D.id_director=P.Director";  
            Statement stmt =  
con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,  
                    ResultSet.CONCUR_READ_ONLY);  
            rs = stmt.executeQuery(query);  
            if (rs.next()) {  
                das_btnSiguiente.setEnabled(true);  
                das_textFieldNom.setText(rs.getString(1));  
                das_textFieldPais.setText(rs.getString(2));  
                das_textFieldGen.setText(rs.getString(3));  
                das_textFieldDura.setText(rs.getString(4));  
                das_textFieldDir.setText(rs.getString(5));  
                das_lb1Nose.setVisible(false);  
            } else {  
                das_btnSiguiente.setEnabled(false);  
                das_lb1Nose.setVisible(true);  
                das_textFieldNom.setText("");  
            }  
        }  
    } catch (SQLException ex) {  
        ex.printStackTrace();  
    }  
}
```

```

        das_textFieldPais.setText("");
        das_textFieldGen.setText("");
        das_textFieldDura.setText("");
        das_textFieldDir.setText("");
    }
    // Si esta seleccionado solo director
    } else if (das_comboBoxDir.getSelectedIndex() >= 0 &&
das_comboBoxGen.getSelectedIndex() == -1) {
        int numero;
        numero =
Funciones.retornaNumDir(das_comboBoxDir.getSelectedItem());
        query = "SELECT P.Titulo, P.Pais, P.Genero, P.Duracion,
CONCAT(D.Nombre, ' ', D.Apellidos) FROM DIRECTOR D INNER JOIN PELICULA P ON
D.id_director=P.Director WHERE D.id_director=" + numero + ";";
        Statement stmt =
con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                    ResultSet.CONCUR_READ_ONLY);
        rs = stmt.executeQuery(query);
        if (rs.next()) {
            das_btnSiguiente.setEnabled(true);
            das_textFieldNom.setText(rs.getString(1));
            das_textFieldPais.setText(rs.getString(2));
            das_textFieldGen.setText(rs.getString(3));
            das_textFieldDura.setText(rs.getString(4));
            das_textFieldDir.setText(rs.getString(5));
            das_lblNose.setVisible(false);
        } else {
            das_btnSiguiente.setEnabled(false);
            das_lblNose.setVisible(true);
            das_textFieldNom.setText("");
            das_textFieldPais.setText("");
            das_textFieldGen.setText("");
            das_textFieldDura.setText("");
            das_textFieldDir.setText("");
        }
    }
    // Si esta seleccionado solo genero
    } else if (das_comboBoxDir.getSelectedIndex() == -1 &&
das_comboBoxGen.getSelectedIndex() >= 0) {
        query = "SELECT P.Titulo, P.Pais, P.Genero, P.Duracion,
CONCAT(D.Nombre, ' ', D.Apellidos) FROM DIRECTOR D INNER JOIN PELICULA P ON
D.id_director=P.Director WHERE P.Genero='" + (String)das_comboBoxGen.getSelectedItem() + "'";
        Statement stmt =
con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                    ResultSet.CONCUR_READ_ONLY);
        rs = stmt.executeQuery(query);
        if (rs.next()) {
            das_btnSiguiente.setEnabled(true);
            das_textFieldNom.setText(rs.getString(1));
            das_textFieldPais.setText(rs.getString(2));
            das_textFieldGen.setText(rs.getString(3));
            das_textFieldDura.setText(rs.getString(4));
            das_textFieldDir.setText(rs.getString(5));
            das_lblNose.setVisible(false);
        } else {
            das_btnSiguiente.setEnabled(false);
            das_lblNose.setVisible(true);
            das_textFieldNom.setText("");
            das_textFieldPais.setText("");
            das_textFieldGen.setText("");
            das_textFieldDura.setText("");
            das_textFieldDir.setText("");
        }
    }
    // Si están seleccionados genero y director
    } else {
        int numero;
        numero =
Funciones.retornaNumDir(das_comboBoxDir.getSelectedItem());

```

Ahora para terminar, voy a configurar los botones de siguiente y atrás.

desactiva:

```

}
}
};

```

Y para el botón de atrás:

```
public void actionPerformed(ActionEvent e) {
    try {
        if (rs.previous()) {
            das_btnAnterior.setEnabled(true);
            das_btnSiguiente.setEnabled(true);
            das_textFieldNom.setText(rs.getString(1));
            das_textFieldPais.setText(rs.getString(2));
            das_textFieldGen.setText(rs.getString(3));
            das_textFieldDura.setText(rs.getString(4));
            das_textFieldDir.setText(rs.getString(5));
        } else {
            das_btnAnterior.setEnabled(false);
        }
    } catch (SQLException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }
}
```

Ahora hago una prueba para comprobar que funciona correctamente:

The screenshot shows a Java Swing application window with a light gray background. At the top, there are two dropdown menus labeled "Director" and "Género", both with green arrow icons. Below them is a "Buscar" button. Underneath the button is the text "Selecciona un criterio de búsqueda". Below this text, there are three text input fields: "Terminator", "EEUU", and "Accion". To the right of these fields, there are two more text input fields: "Dani Aguilar" and "100". At the bottom of the window, there are two buttons: "Anterior" and "Siguiente". The "Siguiente" button is highlighted with a blue border.

The image shows a web application interface for searching movies. It features a search form with two dropdown menus for 'Director' and 'Género', a 'Buscar' button, and a section titled 'Selecciona un criterio de búsqueda'. This section contains three input fields: 'Programadores' (with the value 'Coco Loco'), 'Italia' (with the value '120'), and 'Drama'. At the bottom, there are two buttons: 'Anterior' and 'Siguiente'.

Director Género

Buscar

Selecciona un criterio de búsqueda

Programadores Coco Loco

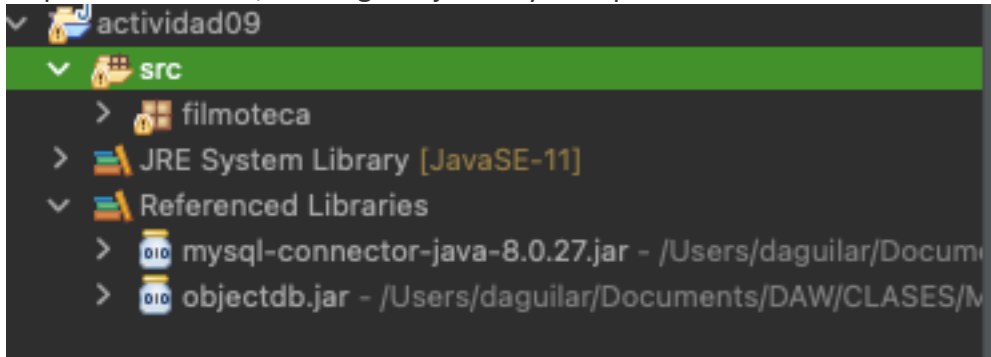
Italia 120

Drama

Anterior Siguiente

2. Crear package actividad09.tiendas para gestionar información en objectDB

Lo primero de todo, descargo ObjectDB y lo importo a la librería



2.1. Clase Empleado:

```
package tiendas;

import javax.persistence.*;

@Entity
public class Empleado {

    @Id
    @GeneratedValue
    private long id;
    private String nombre;
    private String apellido;

    // CONSTRUCTOR
    public Empleado(long id, String nombre, String apellido) {
        super();
        this.id = id;
        this.nombre = nombre;
        this.apellido = apellido;
    }

    // GETTERS AND SETTERS
    public long getId() {
        return id;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getApellido() {
        return apellido;
    }
}
```

```

    public void setApellido(String apellido) {
        this.apellido = apellido;
    }

    // toString
    @Override
    public String toString() {
        return "Empleado [id=" + id + ", nombre=" + nombre + ", apellido=" + apellido +
    "];";
}

```

2.2. Clase Tiendas:

```

package tiendas;

import java.util.ArrayList;

import javax.persistence.*;

@Entity
public class Tienda {

    @Id
    @GeneratedValue
    private long id;
    private String direccion;
    private int ventas;
    @OneToOne(cascade = CascadeType.ALL, orphanRemoval=true)
    private ArrayList<Empleado> empleados;

    // CONSTRUCTOR
    public Tienda(long id, String direccion, int ventas, ArrayList<Empleado> empleados) {
        super();
        this.id = id;
        this.direccion = direccion;
        this.ventas = ventas;
        this.empleados = empleados;
    }

    // GETTERS AND SETTERS
    public long getId() {
        return id;
    }

    public String getDireccion() {
        return direccion;
    }

    public void setDireccion(String direccion) {
        this.direccion = direccion;
    }

    public int getVentas() {
        return ventas;
    }

    public void setVentas(int ventas) {
        this.ventas = ventas;
    }

    public ArrayList<Empleado> getEmpleados() {
        return empleados;
    }
}

```

```

    }

    public void setEmpleados(ArrayList<Empleado> empleados) {
        this.empleados = empleados;
    }

    // toString
    @Override
    public String toString() {
        return "Tienda [id=" + id + ", direccion=" + direccion + ", ventas=" + ventas + ",
empleados=" + empleados
                + "]\n";
    }
}
}
});

```

2.3. Clase GestionaTiendas y crear tres tiendas y tres empleados, luego insertarlos en la BBDD

```

public class GestionaTiendas {
    public static void main(String[] args) throws Exception {
        // Creacion de la BBDD
        EntityManagerFactory emf =
Persistence.createEntityManagerFactory("objectdb:db/tiendasDB.tmp;drop");
        EntityManager em = emf.createEntityManager();

        // Inicio de transaccion
        em.getTransaction().begin();

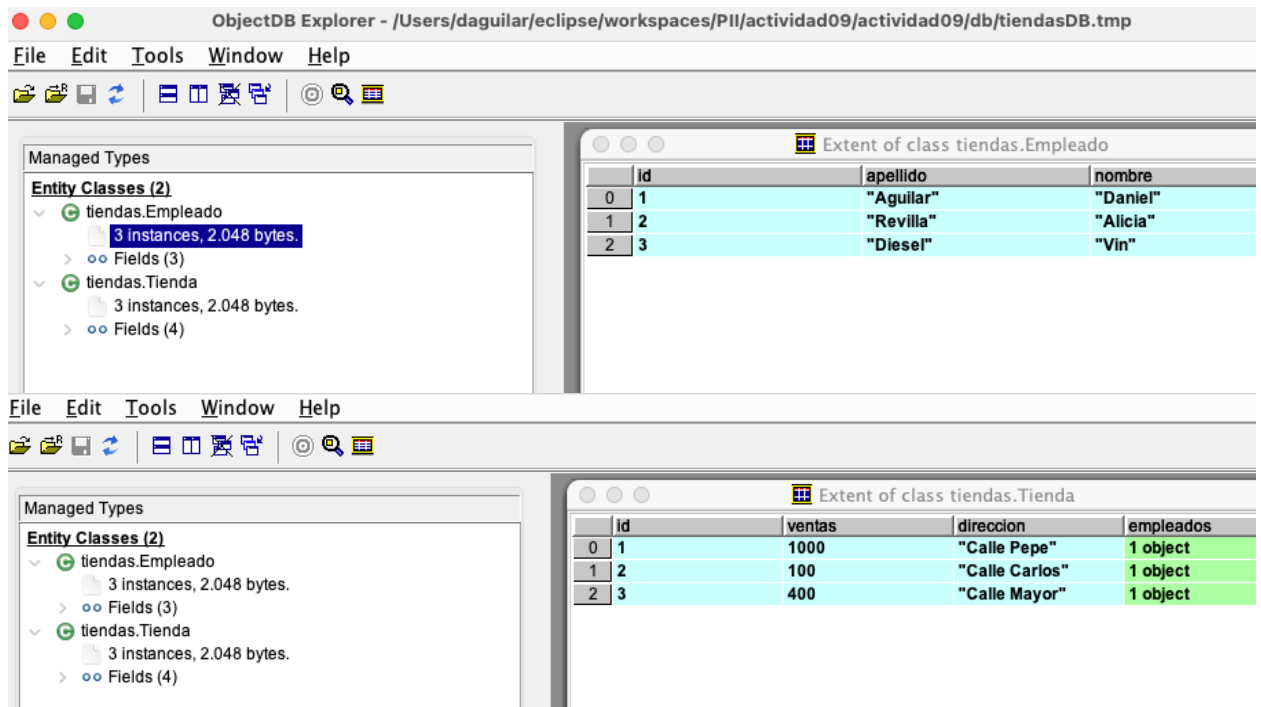
        // Creacion de tres empleados
        Empleado empleado1 = new Empleado(1,"Daniel","Aguilar");
        Empleado empleado2 = new Empleado(2,"Alicia","Revilla");
        Empleado empleado3 = new Empleado(3,"Vin","Diesel");

        // Creacion de tres tiendas
        ArrayList<Empleado> listEmpleadost1 = new ArrayList<Empleado>();
        ArrayList<Empleado> listEmpleadost2 = new ArrayList<Empleado>();
        ArrayList<Empleado> listEmpleadost3 = new ArrayList<Empleado>();
        listEmpleadost1.add(empleado1);
        listEmpleadost2.add(empleado2);
        listEmpleadost3.add(empleado3);
        Tienda tienda1 = new Tienda(1,"Calle Pepe",1000,listEmpleadost1);
        Tienda tienda2 = new Tienda(2,"Calle Carlos",100,listEmpleadost2);
        Tienda tienda3 = new Tienda(3,"Calle Mayor",400,listEmpleadost3);

        // Insertar datos en la BBDD
        em.persist(tienda1);
        em.persist(tienda2);
        em.persist(tienda3);
        em.getTransaction().commit();
    }
}

```

Compruebo desde el explorer de ObjectDB que efectivamente se han creado:



2.4. Mostrar el menú y configurar las diferentes opciones

Primero configuro el Switch case completo, para ello uso una función de seleccionar enteros, de esta forma me aseguro que el usuario siempre introducirá un número entero:

```
// Funcion para pedir un numero entero
public static Integer pideEntero() {
    boolean salir = false;
    Integer entero = 0;
    Scanner scanner = new Scanner(System.in);
    do {
        try {
            entero = scanner.nextInt();
            salir = true;
        } catch (Exception exc1) {
            System.out.println("Valor introducido invalido, vuelve a intentarlo");
            scanner.next();
        }
    } while (!salir);
    return entero;
}
```

También configuro un menú en una función dentro de un Array:

```
public static void menu() {
    String[] muestraMenu;
    muestraMenu = new String[9];
    muestraMenu[0] = "*****";
    muestraMenu[1] = "Introduzca la operacion a realizar del siguiente menú de opciones";
    muestraMenu[2] = "1. Muestra los empleados";
    muestraMenu[3] = "3. Muestra las tiendas";
    muestraMenu[4] = "4. Mostrar las tiendas ordenadas por ventas";
    muestraMenu[5] = "5. Modificar un empleado";
    muestraMenu[6] = "6. Añade una tienda";
    muestraMenu[7] = "0. Salir";
    muestraMenu[8] = "*****";
}
```

```
for (String opcion : muestraMenu) {  
    System.out.println(opcion);  
}
```

Ahora preparo el switch case, empezamos por mostrar empleados.

2.4.1. Mostrar empleados

Usare una función externa, la función utiliza el TypedQuery y recibe por parámetro el EntityManager, después recorre los resultados con un for:

```
public static void verEmpleados(EntityManager em) {  
    TypedQuery<Empleado> query = em.createQuery("SELECT e FROM Empleado e", Empleado.class);  
    ArrayList<Empleado> empleados = (ArrayList<Empleado>) query.getResultList();  
    for (Empleado e : empleados) {  
        System.out.println(e);  
    }  
}
```

Hago una prueba para comprobar que funciona correctamente:

```
1. Muestra los empleados  
2. Muestra las tiendas  
3. Mostrar las tiendas ordenadas por ventas  
4. Modificar un empleado  
5. Añade una tienda  
0. Salir  
*****  
1  
Empleado [id=1, nombre=Daniel, apellido=Aguilar]  
Empleado [id=2, nombre=Alicia, apellido=Revilla]  
Empleado [id=3, nombre=Vin, apellido=Diesel]
```

2.4.2. Mostrar tiendas

Configuro la función usando de nuevo el TypedQuery recibiendo por parámetro el EntityManager

```
public static void verTiendas(EntityManager em) {  
    TypedQuery<Tienda> query = em.createQuery("SELECT t FROM Tienda t", Tienda.class);  
    ArrayList<Tienda> tiendas = (ArrayList<Tienda>) query.getResultList();  
    for (Tienda t : tiendas) {  
        System.out.println(t);  
    }  
}
```

Hago una prueba para comprobar que funciona correctamente:

```

GestionaTiendas [Java Application] /Library/Java/JavaVirtualMachines/amazon-corretto-11.jdk/Contents/Home/bin/java (16 dic. 2021 15:37:34)
2. Muestra las tiendas
3. Mostrar las tiendas ordenadas por ventas
4. Modificar un empleado
5. Añade una tienda
0. Salir
*****
2
Tienda [id=1, direccion=Calle Pepe, ventas=1000, empleados=[Empleado [id=1, nombre=Daniel, apellido=Aguilar]]]
Tienda [id=2, direccion=Calle Carlos, ventas=100, empleados=[Empleado [id=2, nombre=Alicia, apellido=Revilla]]]
Tienda [id=3, direccion=Calle Mayor, ventas=400, empleados=[Empleado [id=3, nombre=Vin, apellido=Diesel]]]
*****

```

2.4.3. Mostrar tiendas ordenadas por ventas

Configuro una función usando de nuevo el TypedQuery recibiendo por parámetro el EntityManager, en este caso uso una Query diferente usando el order by según selección. La función tiene dentro un switch case, que además hace uso de la función pideEntero usada anteriormente. Quedaría así:

```

public static void verTiendasOrdenadas(EntityManager em) {
    System.out.println("Ordenar segun ventas ascendentes o descendentes?");
    System.out.println("1- Ascendente");
    System.out.println("2- Descendente");
    switch (pideEntero()) {
        case 1 :
            TypedQuery<Tienda> query1 = em.createQuery("SELECT t FROM Tienda t ORDER
BY t.ventas ASC", Tienda.class);

            ArrayList<Tienda> tiendas1 = (ArrayList<Tienda>) query1.getResultList();

            for (Tienda t1 : tiendas1) {
                System.out.println(t1);
            }
            break;

        case 2 :
            TypedQuery<Tienda> query2 = em.createQuery("SELECT t FROM Tienda t ORDER
BY t.ventas DESC", Tienda.class);

            ArrayList<Tienda> tiendas2 = (ArrayList<Tienda>) query2.getResultList();

            for (Tienda t2 : tiendas2) {
                System.out.println(t2);
            }
            break;
    }
}

```

Hago una prueba para comprobar que funciona correctamente:

```

Console X
GestionaTiendas [Java Application] /Library/Java/JavaVirtualMachines/amazon-corretto-11.jdk/Contents/Home/bin/java (16 dic. 2021 15:55:13)
3. Mostrar las tiendas ordenadas por ventas
4. Modificar un empleado
5. Añade una tienda
0. Salir
*****
3
Ordenar segun ventas ascendentes o descendentes?
1- Ascendente
2- Descendente
1
Tienda [id=2, direccion=Calle Carlos, ventas=100, empleados=[Empleado [id=2, nombre=Alicia, apellido=Revilla]]]
Tienda [id=3, direccion=Calle Mayor, ventas=400, empleados=[Empleado [id=3, nombre=Vin, apellido=Diesel]]]
Tienda [id=1, direccion=Calle Pepe, ventas=1000, empleados=[Empleado [id=1, nombre=Daniel, apellido=Aguilar]]]

```

```

Console X
GestionaTiendas [Java Application] /Library/Java/JavaVirtualMachines/amazon-corretto-11.jdk/Contents/Home/bin/java (16 dic. 2021 15:55:13)
3. Mostrar las tiendas ordenadas por ventas
4. Modificar un empleado
5. Añade una tienda
0. Salir
*****
3
Ordenar segun ventas ascendentes o descendentes?
1- Ascendente
2- Descendente
2
Tienda [id=1, direccion=Calle Pepe, ventas=1000, empleados=[Empleado [id=1, nombre=Daniel, apellido=Aguilar]]]
Tienda [id=3, direccion=Calle Mayor, ventas=400, empleados=[Empleado [id=3, nombre=Vin, apellido=Diesel]]]
Tienda [id=2, direccion=Calle Carlos, ventas=100, empleados=[Empleado [id=2, nombre=Alicia, apellido=Revilla]]]

```

2.4.4. Editar un empleado

Para modificar el empleado, usare una función que recibirá por parámetro el EntityManager y además el ID del empleado para poder buscarlo en la BBDD (Usando la función `findById` de nuevo), una vez lo reciba, lo mostrara por pantalla y con un switch case podrá seleccionar que quiere modificar:

```

public static void modEmpleado(EntityManager em, int id) {
    Scanner scanner = new Scanner(System.in);
    String nombre;
    String apellido;
    TypedQuery<Empleado> query = em.createQuery("SELECT e FROM Empleado e WHERE e.id="+id,
Empleado.class);
    Empleado e1 = query.getSingleResult();
    System.out.println("El empleado seleccionado es: " + e1.toString());
    System.out.println("Indica el atributo a modificar:");
    System.out.println("1- Nombre");
    System.out.println("2- Apellido");
    System.out.println("3- Ninguno");

    switch(pideEntero()) {
        case 1:
            em.getTransaction().begin();
            System.out.println("Indica el nuevo nombre");
            nombre = scanner.next();
            e1.setNombre(nombre);
            em.getTransaction().commit();

            break;

        case 2:
            em.getTransaction().begin();
            System.out.println("Indica el nuevo apellido");
            apellido = scanner.next();
            e1.setApellido(apellido);

```

```
        em.getTransaction().commit();  
break;  
  
case 3:  
    break;
```

Hago una prueba para comprobar que funciona:

Actuales:

```
1  
Empleado [id=1, nombre=Daniel, apellido=Aguilar]  
Empleado [id=2, nombre=Alicia, apellido=Revilla]  
Empleado [id=3, nombre=Vin, apellido=Diesel]  
  
Introduzca la operacion a realizar del siguiente menú de opciones  
1. Muestra los empleados  
2. Muestra las tiendas  
3. Mostrar las tiendas ordenadas por ventas  
4. Modificar un empleado  
5. Añade una tienda  
0. Salir  
*****  
4  
Introduce el ID del empleado a modificar:  
1  
El empleado seleccionado es: Empleado [id=1, nombre=Daniel, apellido=Aguilar]  
Indica el atributo a modificar:  
1- Nombre  
2- Apellido  
3- Ninguno  
1  
Indica el nuevo nombre  
Carlos  
*****  
Introduzca la operacion a realizar del siguiente menú de opciones  
1. Muestra los empleados  
2. Muestra las tiendas  
3. Mostrar las tiendas ordenadas por ventas  
4. Modificar un empleado  
5. Añade una tienda  
0. Salir  
*****  
1  
Empleado [id=1, nombre=Carlos, apellido=Aguilar]  
Empleado [id=2, nombre=Alicia, apellido=Revilla]  
Empleado [id=3, nombre=Vin, apellido=Diesel]
```

2.4.5. Crear una nueva tienda

Para la creación de tiendas usare una función que primero inicia la transacción, despues pedirá los datos al usuario, primero el ID para la tienda, luego la dirección, luego las ventas, y luego el ID del empleado.

Después hará una búsqueda por el ID y lo añadirá al array de empleados.

Preguntara si quiere añadir un nuevo empelado, si quiere añadirlo, usando un switch case, podrá añadir otro, o salir. Una vez salga hará un persist y un commit.

En el caso de que el ID esté duplicado, dara una excepción:

```
public static void nuevaTienda(EntityManager em) {
    try {
        // Inicia la transaccion
        em.getTransaction().begin();
        // Variables
        boolean salir = false;
        Scanner scanner = new Scanner(System.in);
        int idTienda;
        String dire;
        int ventas;
        int id;
        ArrayList<Empleado> listEmpleados = new ArrayList<Empleado>();
        // Peticiones
        System.out.println("Indica un id para la tienda");
        idTienda = pideEntero();
        System.out.println("Indica una direccion");
        dire = scanner.next();
        System.out.println("Indica las ventas");
        ventas = pideEntero();
        verEmpleados(em);
        System.out.println("Indica el ID del empleado que quieras añadir a la tienda");
        id = pideEntero();
        // Busqueda del empleado para añadirlo al ArrayList
        TypedQuery<Empleado> query = em.createQuery("SELECT e FROM Empleado e WHERE e.id="+id,
Empleado.class);
        listEmpleados.add(query.getSingleResult());
        System.out.println("Empleado añadido correctamente");

        do {
            System.out.println("Quiere añadir otro empleado?");
            System.out.println("1- Si");
            System.out.println("2- No");

            switch (pideEntero()) {
                case 1:
                    System.out.println("Indica el ID del empleado que quieras añadir a
la tienda");
                    id = pideEntero();
                    query = em.createQuery("SELECT e FROM Empleado e WHERE e.id="+id,
Empleado.class);
                    listEmpleados.add(query.getSingleResult());
                    System.out.println("Empleado añadido correctamente");
                    break;
                case 2:
                    salir = true;
                    break;
            }
        } while (!salir);

        // Crear tienda
        Tienda nuevaTienda = new Tienda(idTienda,dire,ventas,listEmpleados);
        // Commit
        em.persist(nuevaTienda);
        em.getTransaction().commit();
    } catch (Exception e) {
        System.out.println("El ID de tienda ya existe");
    }
}
```

Prueba:

```

0. Salir
*****
5
Indica un id para la tienda
6
Indica una direccion
afds
Indica las ventas
4
Empleado [id=1, nombre=Daniel, apellido=Aguilar]
Empleado [id=2, nombre=Alicia, apellido=Revilla]
Empleado [id=3, nombre=Vin, apellido=Diesel]
Indica el ID del empleado que quieras añadir a la tienda
1
Empleado añadido correctamente
Quiere añadir otro empleado?
1- Si
1- No
1
Indica el ID del empleado que quieras añadir a la tienda
2
Empleado añadido correctamente
Quiere añadir otro empleado?
1- Si
1- No
1
Indica el ID del empleado que quieras añadir a la tienda
3
Empleado añadido correctamente
Quiere añadir otro empleado?
1- Si
1- No
2
*****
Introduzca la operacion a realizar del siguiente menú de opciones
1. Muestra los empleados
2. Muestra las tiendas
3. Mostrar las tiendas ordenadas por ventas
4. Modificar un empleado
5. Añade una tienda
0. Salir
*****
2
Tienda [id=1, direccion=Calle Pepe, ventas=1000, empleados=[Empleado [id=1, nombre=Daniel, apellido=Aguilar]]]
Tienda [id=2, direccion=Calle Carlos, ventas=100, empleados=[Empleado [id=2, nombre=Alicia, apellido=Revilla]]]
Tienda [id=3, direccion=Calle Mayor, ventas=400, empleados=[Empleado [id=3, nombre=Vin, apellido=Diesel]]]
Tienda [id=6, direccion=afds, ventas=4, empleados=[Empleado [id=1, nombre=Daniel, apellido=Aguilar], Empleado [id=2
*****
Introduzca la operacion a realizar del siguiente menú de opciones

```

3. CODIGO COMPLETO EJERCICIO FILMOTECA

3.1. DelPelicula

```
package filмотeca;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JComboBox;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.SwingConstants;

import java.util.ArrayList;

import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.sql.SQLException;
import java.awt.event.ActionEvent;

public class DelPelicula extends JFrame {

    private JPanel contentPane;

    public DelPelicula() {
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setBounds(100, 100, 450, 246);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);
        this.setLocationRelativeTo(null);

        // COMBOBOX
        // Extraccion de peliculas
        ArrayList<String> peliculas = new ArrayList<String>();
        try {
            peliculas = GestionBBDD.listarPeliculas();
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        // Conversion a array String del ArrayList
        String[] arrayCombo = peliculas.toArray(new String[peliculas.size()]);
        // Creacion del comboBox
        JComboBox das_comboBoxPel = new JComboBox(arrayCombo);
        das_comboBoxPel.setBounds(82, 71, 300, 27);
        contentPane.add(das_comboBoxPel);

        JLabel das_lblTit = new JLabel("Selecciona película");
        das_lblTit.setHorizontalAlignment(SwingConstants.CENTER);
        das_lblTit.setBounds(6, 43, 438, 16);
        contentPane.add(das_lblTit);

        JButton das_btnDel = new JButton("Eliminar");
        das_btnDel.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {

                GestionBBDD.delPelicula((String)das_comboBoxPel.getSelectedItem());
                JDialog das_aviso = new JDialog();
                JLabel das_mensaje = new JLabel("Película eliminada");
                das_aviso.getContentPane().add(das_mensaje);
                das_aviso.setSize(300, 100);
```



```

        das_avisos.setLocationRelativeTo(null);
        das_avisos.setVisible(true);
        dispose();
    } catch (SQLException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }
}

});
das_btnDel.setBounds(164, 154, 117, 29);
contentPane.add(das_btnDel);
}
}

```

3.2. EliminarDirector

```

package filmoteca;

import java.awt.BorderLayout;
import java.awt.EventQueue;
import java.util.ArrayList;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.SwingConstants;
import javax.swing.JComboBox;
import javax.swing.JDialog;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.sql.SQLException;
import java.awt.event.ActionEvent;

public class EliminarDirector extends JFrame {

    private JPanel contentPane;

    public EliminarDirector() {
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setBounds(100, 100, 450, 237);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);
        this.setLocationRelativeTo(null);

        JLabel das_lblSelec = new JLabel("Selecciona director");
        das_lblSelec.setHorizontalAlignment(SwingConstants.CENTER);
        das_lblSelec.setBounds(6, 26, 438, 16);
        contentPane.add(das_lblSelec);

        JLabel das_lbltit2 = new JLabel("Solo se muestran directores que no tienen películas");
        das_lbltit2.setHorizontalAlignment(SwingConstants.CENTER);
        das_lbltit2.setBounds(6, 54, 438, 16);
        contentPane.add(das_lbltit2);

        // COMBOBOX DIRECTORES
        // Extraccion de directores
        ArrayList<String> directores = new ArrayList<String>();
        try {
            directores = GestionBBDD.listarDirectoresSinPeliculas();
        }
    }
}

```

```

        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        // Conversion a array String del ArrayList
        String[] arrayComboDirectores = directores.toArray(new String[directores.size()]);
        // Creacion del combobox
        JComboBox das_comboBox = new JComboBox(arrayComboDirectores);
        das_comboBox.setBounds(114, 82, 213, 27);
        contentPane.add(das_comboBox);

        JButton das_btn = new JButton("Eliminar");
        das_btn.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // Seleccion primer caracter director
                int numero;
                numero = Funciones.retornaNumDir(das_comboBox.getSelectedItem());
                try {
                    GestionBBDD.delDirector(numero);
                    JDialog das_avisos = new JDialog();
                    JLabel das_mensaje = new JLabel("Director eliminado");
                    das_avisos.getContentPane().add(das_mensaje);
                    das_avisos.setSize(300, 100);
                    das_avisos.setLocationRelativeTo(null);
                    das_avisos.setVisible(true);
                    das_avisos.dispose();
                } catch (SQLException e1) {
                    // TODO Auto-generated catch block
                    e1.printStackTrace();
                }
            }
        });
        das_btn.setBounds(161, 143, 117, 39);
        contentPane.add(das_btn);
    }
}

```

3.3. Funciones

```

package filmoteca;

public class Funciones {

    // Funcion que recibe por parametro el selector del combobox de directores, primero lo
    // convierte en un String y lo almacena en una variable
    // Luego almacena en otra variable char el primer valor, por ultimo lo pasa a int y lo
    // retorna
    public static int retornaNumDir(Object director) {
        String direNum;
        char caracter;
        int numero;
        direNum = (String)director;
        caracter = direNum.charAt(0);
        numero = Character.getNumericValue(caracter);
        return numero;
    }
}

```

3.4. GestionarBBDD

```

package filmoteca;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

public class GestionBBDD {

    private static String das_datosConexion = "jdbc:mysql://127.0.0.1:8889/";
    private static String das_baseDatos = "FILMOTECA";
    private static String das_usuario = "root";
    private static String das_password = "4246";
    private static Connection con;

    // CONSTRUCTOR BBDD
    public GestionBBDD(){
        try {
            con = DriverManager.getConnection(das_datosConexion, das_usuario,
das_password);

            try {
                // CREO LA BASE DE DATOS SI NO EXISTE
                crearBDD();
                // CREO LA TABLA PELICULA SI NO EXISTE
                crearTablaPelicula();
                // CREO LA TABLA DIRECTOR SI NO EXISTE
                crearTablaDirector();
                // INSERTO PELICULAS
                insertarPelicula("Terminator",1,"EEUU",100,"Accion");
                insertarPelicula("ESDLA",2,"EEUU",180,"Fantasia");
                insertarPelicula("Torrente",3,"España",120,"Comedia");
                insertarPelicula("Otra Pelicula",4,"Francia",200,"Amor");
                insertarPelicula("Programadores",5,"Italia",120,"Drama");
                // INSERTO DIRECTORES
                insertarDirector("Dani","Aguilar");
                insertarDirector("Alicia","Revilla");
                insertarDirector("Marco","Polo");
                insertarDirector("Lolo","Perro");
                insertarDirector("Coco","Loco");
                // ACTUALIZAR RELACION
                updateRel();
            } catch (Exception e) {
                e.printStackTrace();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    // **** CREACION DE BBDD **** //
    // CREAR BBDD
    private void crearBDD() throws Exception{
        String query = "create database if not exists "+ das_baseDatos +"";
        Statement stmt = null;
        try{
            stmt = con.createStatement();
            stmt.executeUpdate(query);
            con = DriverManager.getConnection(das_datosConexion+das_baseDatos,
das_usuario, das_password);
        }catch (SQLException e){
            e.printStackTrace();
        }finally{

```

```
        stmt.close();
    }
}

// **** CREACION DE TABLAS **** //
// TABLA PELICULA
private void crearTablaPelicula() throws Exception {
    String query = "create table if not exists PELICULA("
        + "Titulo VARCHAR(30), "
        + "Director int, "
        + "Pais VARCHAR(30),"
        + "Duracion INTEGER,"
        + "Genero VARCHAR(15));";

    Statement stmt = null;
    try{
        stmt = con.createStatement();
        stmt.executeUpdate(query);
    }catch (SQLException e){
        e.printStackTrace();
    }finally{
        stmt.close();
    }
}

//TABLA DIRECTOR
private void crearTablaDirector() throws Exception {
    String query = "create table if not exists DIRECTOR("
        + "id_director int PRIMARY KEY auto_increment,"
        + "Nombre VARCHAR(50),"
        + "Apellidos VARCHAR(100));";

    Statement stmt = null;
    try{
        stmt = con.createStatement();
        stmt.executeUpdate(query);
    }catch (SQLException e){
        e.printStackTrace();
    }finally{
        stmt.close();
    }
}

// **** INSERCCIONES **** //
// PELICULAS
public static void insertarPelicula(String titulo, int director, String pais, int
duracion, String genero) throws SQLException{
    String query = "insert into PELICULA values('" + titulo + "', " + director
+ ", '" + pais + "', " + duracion + ", '" + genero + "')";
    Statement stmt = null;
    try {
        stmt = con.createStatement();
        stmt.executeUpdate(query);
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        stmt.close();
    }
}

// DIRECTOR
public static void insertarDirector(String nombre, String apellido) throws
SQLException{
    String query = "insert into DIRECTOR values(null, '" + nombre + "', '" +
apellido + "')";
    Statement stmt = null;
    try {
        stmt = con.createStatement();
```

```

        stmt.executeUpdate(query);
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        stmt.close();
    }
}

// **** ACTUALIZAR RELACION PELICULA-DIRECTORES **** //
public void updateRel() throws SQLException{
    String query = "ALTER TABLE PELICULA ADD CONSTRAINT fk_película_director
foreign key (Director) references DIRECTOR(id_director);";
    Statement stmt = null;
    try {
        stmt = con.createStatement();
        stmt.executeUpdate(query);
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        stmt.close();
    }
}

// **** SELECT **** //
// LISTAR DIRECTORES
// Funcion que devuelve un array con la información de la tabla DIRECTOR
public static ArrayList<String> listarDirectores() throws Exception {
    ArrayList<String> lista = new ArrayList<String>();
    String query = "select * from DIRECTOR;";
    Statement stmt = null;
    ResultSet rs = null;
    try {
        stmt = con.createStatement();
        rs = stmt.executeQuery(query);
        while(rs.next()){
            lista.add(rs.getInt(1)+ " - " + rs.getString(2) + " " +
rs.getString(3)); }
        } catch (SQLException e) { e.printStackTrace();
    } finally{
        rs.close();
        stmt.close(); }

    return lista;
}

// LISTAR DIRECTORES SIN PELICULAS
// Funcion que devuelve un array con la información de la tabla DIRECTOR de los
directores que no tengan películas asignadas
public static ArrayList<String> listarDirectoresSinPelículas() throws Exception {
    ArrayList<String> lista = new ArrayList<String>();
    String query = "SELECT DIRECTOR.* FROM DIRECTOR WHERE DIRECTOR.id_director
NOT IN (SELECT DIRECTOR FROM PELICULA);";
    Statement stmt = null;
    ResultSet rs = null;
    try {
        stmt = con.createStatement();
        rs = stmt.executeQuery(query);
        while(rs.next()){
            lista.add(rs.getInt(1)+ " - " + rs.getString(2) + " " +
rs.getString(3)); }
        } catch (SQLException e) { e.printStackTrace();
    } finally{
        rs.close();
        stmt.close(); }

    return lista;
}

```

```

// LISTAR PELICULAS
// Funcion que devuelve un array con la informacion de la tabla PELICULA
unicamente con los titulos
public static ArrayList<String> listarPelículas() throws Exception {
    ArrayList<String> lista = new ArrayList<String>();
    String query = "select * from PELICULA;";
    Statement stmt = null;
    ResultSet rs = null;
    try {
        stmt = con.createStatement();
        rs = stmt.executeQuery(query);
        while(rs.next()){
            lista.add(rs.getString(1)); }
        } catch (SQLException e) { e.printStackTrace();
        } finally{
            rs.close();
            stmt.close(); }
    return lista;
}

// LISTAR GENEROS
// Funcion que devuelve un array con la informacion de la tabla PELICULA
unicamente con los generos
public static ArrayList<String> listarGeneros() throws Exception {
    ArrayList<String> lista = new ArrayList<String>();
    String query = "select * from PELICULA;";
    Statement stmt = null;
    ResultSet rs = null;
    try {
        stmt = con.createStatement();
        rs = stmt.executeQuery(query);
        while(rs.next()){
            lista.add(rs.getString(5)); }
        } catch (SQLException e) { e.printStackTrace();
        } finally{
            rs.close();
            stmt.close(); }
    return lista;
}

// **** ELIMINAR **** //
// BORRAR PELICULAS
// Funcion que recibe por parametro el nombre de una película, y borra el registro
que coincida con esta
public static void delPelícula(String titulo) throws SQLException{
    String query = "DELETE FROM PELICULA WHERE Titulo='"+ titulo +"';";
    Statement stmt = null;
    try {
        stmt = con.createStatement();
        stmt.executeUpdate(query);
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        stmt.close();
    }
}

// BORRAR PELICULAS
// Funcion que recibe por parametro el id de un director, y borra el registro que
coincida con este
public static void delDirector(int id) throws SQLException{
    String query = "DELETE FROM DIRECTOR WHERE id_director="+ id +";";
    Statement stmt = null;
    try {

```

```

        stmt = con.createStatement();
        stmt.executeUpdate(query);
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        stmt.close();
    }
}

// **** MODIFICAR **** //
// MODIFICAR PELICULA
// Funcion que recibe por parametro la seleccion, el nombre, director, pais,
// duracion y genero, para modificar una pelicula existente
public static void modPelicula(String seleccion, String titulo, int director,
String pais, int duracion, String genero) throws SQLException {
    String query = "UPDATE PELICULA SET Titulo='"+ titulo +"', Director='"+
director + ", Pais='"+ pais + "', Duracion=" + duracion + ", Genero='" + genero + "' WHERE
Titulo='"+seleccion+"';";
    Statement stmt = null;
    try {
        stmt = con.createStatement();
        stmt.executeUpdate(query);
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        stmt.close();
    }
}

// MODIFICAR PELICULA
// Funcion que recibe por parametro el nombre, apellidos e id de un director y
// borra el que coincide con este
public static void modDirector(String nombre, String apellidos, int id) throws
SQLException {
    String query = "UPDATE DIRECTOR SET Nombre='"+ nombre +"', Apellidos='"+
apellidos + "' WHERE id_director=" + id + ";";
    Statement stmt = null;
    try {
        stmt = con.createStatement();
        stmt.executeUpdate(query);
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        stmt.close();
    }
}

```

3.5. InsertDirector

```
package filmoteca;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JButton;
import javax.swing.JDialog;

import java.awt.event.ActionListener;
import java.sql.SQLException;
import java.awt.event.ActionEvent;

public class InsertDirector extends JFrame {
```

```

private JPanel contentPane;
private JTextField das_textFieldNombre;
private JTextField das_textFieldApellidos;

public InsertDirector() {
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    setBounds(100, 100, 330, 209);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    setContentPane(contentPane);
    contentPane.setLayout(null);
    this.setLocationRelativeTo(null);

    JLabel das_lblNombre = new JLabel("Nombre:");
    das_lblNombre.setBounds(56, 39, 61, 16);
    contentPane.add(das_lblNombre);

    JLabel das_lblApellidos = new JLabel("Apellidos:");
    das_lblApellidos.setBounds(56, 79, 82, 16);
    contentPane.add(das_lblApellidos);

    das_textFieldNombre = new JTextField();
    das_textFieldNombre.setBounds(150, 34, 130, 26);
    contentPane.add(das_textFieldNombre);
    das_textFieldNombre.setColumns(10);

    das_textFieldApellidos = new JTextField();
    das_textFieldApellidos.setBounds(150, 74, 130, 26);
    contentPane.add(das_textFieldApellidos);
    das_textFieldApellidos.setColumns(10);

    JButton das_btnInsertar = new JButton("Insertar");
    das_btnInsertar.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            try {
                GestionBBDD.insertarDirector(das_textFieldNombre.getText(),
das_textFieldApellidos.getText());

                JDialog das_avisos = new JDialog();
                JLabel das_mensaje = new JLabel("Director añadido");
                das_avisos.getContentPane().add(das_mensaje);
                das_avisos.setSize(300, 100);
                das_avisos.setLocationRelativeTo(null);
                das_avisos.setVisible(true);
                das_avisos.dispose();
            } catch (SQLException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
            }
        }
    });
    das_btnInsertar.setBounds(93, 121, 117, 29);
    contentPane.add(das_btnInsertar);
}
}

```

3.6. InsertPelicula

```

package filмотeca;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.JButton;
import javax.swing.JTextField;
import javax.swing.JComboBox;

```



```
import javax.swing.JDialog;
import java.util.ArrayList;

import javax.swing.DefaultComboBoxModel;
import java.awt.event.ActionListener;
import java.sql.SQLException;
import java.awt.event.ActionEvent;

public class InsertPelicula extends JFrame {

    private JPanel contentPane;
    private JTextField das_textFieldTitulo;
    private JTextField das_textFieldPais;
    private JTextField das_textFieldDuracion;
    private JTextField das_textFieldGenero;

    public InsertPelicula() {

        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setBounds(100, 100, 450, 300);
        this.setLocationRelativeTo(null);

        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        JLabel das_titTitulo = new JLabel("Titulo:");
        das_titTitulo.setBounds(76, 47, 61, 16);
        contentPane.add(das_titTitulo);

        JLabel das_titDirector = new JLabel("Director:");
        das_titDirector.setBounds(76, 75, 61, 16);
        contentPane.add(das_titDirector);

        JLabel das_titPais = new JLabel("Pais:");
        das_titPais.setBounds(76, 103, 61, 16);
        contentPane.add(das_titPais);

        JLabel das_titDuracion = new JLabel("Duracion:");
        das_titDuracion.setBounds(76, 139, 61, 16);
        contentPane.add(das_titDuracion);

        JLabel das_titGenero = new JLabel("Genero:");
        das_titGenero.setBounds(76, 177, 61, 16);
        contentPane.add(das_titGenero);

        das_textFieldTitulo = new JTextField();
        das_textFieldTitulo.setBounds(179, 42, 201, 26);
        contentPane.add(das_textFieldTitulo);
        das_textFieldTitulo.setColumns(10);

        das_textFieldPais = new JTextField();
        das_textFieldPais.setBounds(179, 98, 201, 26);
        contentPane.add(das_textFieldPais);
        das_textFieldPais.setColumns(10);

        das_textFieldDuracion = new JTextField();
        das_textFieldDuracion.setBounds(179, 134, 201, 26);
        contentPane.add(das_textFieldDuracion);
        das_textFieldDuracion.setColumns(10);

        das_textFieldGenero = new JTextField();
        das_textFieldGenero.setBounds(179, 172, 201, 26);
        contentPane.add(das_textFieldGenero);
```

```

das_textFieldGenero.setColumns(10);

// COMBOBOX
// Extraccion de directores
ArrayList<String> directores = new ArrayList<String>();
try {
    directores = GestionBBDD.listarDirectores();
} catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
// Conversion a array String del ArrayList
String[] arrayCombo = directores.toArray(new String[directores.size()]);
// Creacion del combobox
JComboBox das_comboBoxDirector = new JComboBox(arrayCombo);
das_comboBoxDirector.setBounds(179, 71, 201, 27);
contentPane.add(das_comboBoxDirector);

JButton das_btnCrear = new JButton("Crear película");
das_btnCrear.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        // Seleccion primer caracter director
        int numero;
        numero =
Funciones.retornaNumDir(das_comboBoxDirector.getSelectedItem());
        // Añadir película
        try {
            GestionBBDD.insertarPelicula(das_textFieldTitulo.getText(),
numero, das_textFieldPais.getText(), Integer.parseInt(das_textFieldDuracion.getText()),
das_textFieldGenero.getText());

            JDialog das_aviso = new JDialog();
            JLabel das_mensaje = new JLabel("Película creada");
            das_aviso.getContentPane().add(das_mensaje);
            das_aviso.setSize(300, 100);
            das_aviso.setLocationRelativeTo(null);
            das_aviso.setVisible(true);
            das_aviso.dispose();
        } catch (NumberFormatException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        } catch (SQLException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
    }
});
das_btnCrear.setBounds(155, 211, 142, 36);
contentPane.add(das_btnCrear);

```

3.7. Listados

```

package filmoteca;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.JComboBox;

```

```

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

import javax.swing.JButton;
import javax.swing.SwingConstants;
import javax.swing.JTextField;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class Listados extends JFrame {

    // CONEXION
    // Añado una conexion a la BBDD y a ResultSet aqui
    Connection con=null;
    ResultSet rs=null;
    Statement stmt = null;
    private static String das_datosConexion ="jdbc:mysql://127.0.0.1:8889/";
    private static String das_baseDatos = "FILMOTECA";
    private static String das_usuario = "root";
    private static String das_password = "4246";

    private JPanel contentPane;
    private JTextField das_textFieldNom;
    private JTextField das_textFieldPais;
    private JTextField das_textFieldGen;
    private JTextField das_textFieldDir;
    private JTextField das_textFieldDura;

    public Listados() {

        try {
            con = DriverManager.getConnection(das_datosConexion+das_baseDatos,
das_usuario, das_password);
        } catch (SQLException e2) {
            // TODO Auto-generated catch block
            e2.printStackTrace();
        }

        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setBounds(100, 100, 526, 352);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);
        this.setLocationRelativeTo(null);

        JLabel das_lblDir = new JLabel("Director");
        das_lblDir.setBounds(27, 19, 61, 16);
        contentPane.add(das_lblDir);

        // COMBOBOX DIRECTORES
        // Extraccion de directores
        ArrayList<String> directores = new ArrayList<String>();
        try {
            directores = GestionBBDD.listarDirectores();
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        // Conversion a array String del ArrayList

```

```
String[] arrayComboDirectores = directores.toArray(new String[directores.size()]);
// Creacion del combobox
JComboBox das_comboBoxDir = new JComboBox(arrayComboDirectores);
das_comboBoxDir.setBounds(100, 15, 148, 27);
contentPane.add(das_comboBoxDir);
// Esto deja el combobox vacio
das_comboBoxDir.setSelectedIndex(-1);

JLabel das_lblGen = new JLabel("Género");
das_lblGen.setBounds(272, 19, 61, 16);
contentPane.add(das_lblGen);

// COMBOBOX GENEROS
// Extraccion de directores
ArrayList<String> generos = new ArrayList<String>();
try {
    generos = GestionBBDD.listarGeneros();
} catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

// Conversion a array String del ArrayList
String[] arrayComboGeneros = generos.toArray(new String[generos.size()]);
// Creacion del combobox
JComboBox das_comboBoxGen = new JComboBox(arrayComboGeneros);
das_comboBoxGen.setBounds(345, 15, 175, 27);
contentPane.add(das_comboBoxGen);
das_comboBoxGen.setSelectedIndex(-1);

JLabel das_lblSelec = new JLabel("Selecciona un criterio de búsqueda");
das_lblSelec.setHorizontalAlignment(SwingConstants.CENTER);
das_lblSelec.setBounds(6, 95, 514, 16);
contentPane.add(das_lblSelec);

JLabel das_lblNose = new JLabel("No se han encontrado resultados");
das_lblNose.setHorizontalAlignment(SwingConstants.CENTER);
das_lblNose.setBounds(6, 123, 514, 16);
das_lblNose.setVisible(false);
contentPane.add(das_lblNose);

das_textFieldNom = new JTextField();
das_textFieldNom.setEnabled(false);
das_textFieldNom.setEditable(false);
das_textFieldNom.setBounds(100, 151, 148, 26);
contentPane.add(das_textFieldNom);
das_textFieldNom.setColumns(10);

das_textFieldPais = new JTextField();
das_textFieldPais.setEnabled(false);
das_textFieldPais.setEditable(false);
das_textFieldPais.setBounds(100, 189, 148, 26);
contentPane.add(das_textFieldPais);
das_textFieldPais.setColumns(10);

das_textFieldGen = new JTextField();
das_textFieldGen.setEnabled(false);
das_textFieldGen.setEditable(false);
das_textFieldGen.setBounds(100, 227, 148, 26);
contentPane.add(das_textFieldGen);
das_textFieldGen.setColumns(10);

das_textFieldDir = new JTextField();
das_textFieldDir.setEnabled(false);
das_textFieldDir.setEditable(false);
das_textFieldDir.setBounds(345, 151, 175, 26);
```

```

contentPane.add(das_textFieldDir);
das_textFieldDir.setColumns(10);

das_textFieldDura = new JTextField();
das_textFieldDura.setEnabled(false);
das_textFieldDura.setEditable(false);
das_textFieldDura.setBounds(345, 189, 175, 26);
contentPane.add(das_textFieldDura);
das_textFieldDura.setColumns(10);

// DECLARACION BOTON SIGUIENTE
JButton das_btnSiguiente = new JButton("Siguiente");
das_btnSiguiente.setEnabled(false);
das_btnSiguiente.setBounds(331, 277, 117, 29);
contentPane.add(das_btnSiguiente);

JButton das_btnBuscar = new JButton("Buscar");
das_btnBuscar.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String query;
        try {
            // Si los dos combobox están vacíos
            if (das_comboBoxDir.getSelectedIndex() == -1 &&
das_comboBoxGen.getSelectedIndex() == -1) {
                query = "SELECT P.Titulo, P.Pais, P.Genero, P.Duracion,
CONCAT(D.Nombre,' ',D.Apellidos) FROM DIRECTOR D INNER JOIN PELICULA P ON
D.id_director=P.Director";
                Statement stmt =
con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                    ResultSet.CONCUR_READ_ONLY);
                rs = stmt.executeQuery(query);

                if (rs.next()) {
                    das_btnSiguiente.setEnabled(true);
                    das_textFieldNom.setText(rs.getString(1));
                    das_textFieldPais.setText(rs.getString(2));
                    das_textFieldGen.setText(rs.getString(3));
                    das_textFieldDura.setText(rs.getString(4));
                    das_textFieldDir.setText(rs.getString(5));
                    das_lblNose.setVisible(false);
                } else {
                    das_btnSiguiente.setEnabled(false);
                    das_lblNose.setVisible(true);
                    das_textFieldNom.setText("");
                    das_textFieldPais.setText("");
                    das_textFieldGen.setText("");
                    das_textFieldDura.setText("");
                    das_textFieldDir.setText("");
                }
                // Si esta seleccionado solo director
            } else if (das_comboBoxDir.getSelectedIndex() >= 0 &&
das_comboBoxGen.getSelectedIndex() == -1) {
                int numero;
                numero =
Funciones.retornaNumDir(das_comboBoxDir.getSelectedItem());
                query = "SELECT P.Titulo, P.Pais, P.Genero, P.Duracion,
CONCAT(D.Nombre,' ',D.Apellidos) FROM DIRECTOR D INNER JOIN PELICULA P ON
D.id_director=P.Director WHERE D.id_director=" + numero + ";";
                Statement stmt =
con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                    ResultSet.CONCUR_READ_ONLY);
                rs = stmt.executeQuery(query);
                if (rs.next()) {
                    das_btnSiguiente.setEnabled(true);

```

```

        das_textFieldNom.setText(rs.getString(1));
        das_textFieldPais.setText(rs.getString(2));
        das_textFieldGen.setText(rs.getString(3));
        das_textFieldDura.setText(rs.getString(4));
        das_textFieldDir.setText(rs.getString(5));
        das_lblNose.setVisible(false);
    } else {
        das_btnSiguiente.setEnabled(false);
        das_lblNose.setVisible(true);
        das_textFieldNom.setText("");
        das_textFieldPais.setText("");
        das_textFieldGen.setText("");
        das_textFieldDura.setText("");
        das_textFieldDir.setText("");
    }
    // Si esta seleccionado solo genero
    } else if (das_comboBoxDir.getSelectedIndex() == -1 &&
das_comboBoxGen.getSelectedIndex() >= 0) {
        query = "SELECT P.Titulo, P.Pais, P.Genero, P.Duracion,
CONCAT(D.Nombre, ' ', D.Apellidos) FROM DIRECTOR D INNER JOIN PELICULA P ON
D.id_director=P.Director WHERE P.Genero='" + (String)das_comboBoxGen.getSelectedItem() + "';";
        Statement stmt =
con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                    ResultSet.CONCUR_READ_ONLY);
        rs = stmt.executeQuery(query);
        if (rs.next()) {
            das_btnSiguiente.setEnabled(true);
            das_textFieldNom.setText(rs.getString(1));
            das_textFieldPais.setText(rs.getString(2));
            das_textFieldGen.setText(rs.getString(3));
            das_textFieldDura.setText(rs.getString(4));
            das_textFieldDir.setText(rs.getString(5));
            das_lblNose.setVisible(false);
        } else {
            das_btnSiguiente.setEnabled(false);
            das_lblNose.setVisible(true);
            das_textFieldNom.setText("");
            das_textFieldPais.setText("");
            das_textFieldGen.setText("");
            das_textFieldDura.setText("");
            das_textFieldDir.setText("");
        }
    }
    // Si están seleccionados genero y director
    } else {
        int numero;
        numero =
Funciones.retornaNumDir(das_comboBoxDir.getSelectedItem());
        query = "SELECT P.Titulo, P.Pais, P.Genero, P.Duracion,
CONCAT(D.Nombre, ' ', D.Apellidos) FROM DIRECTOR D INNER JOIN PELICULA P ON
D.id_director=P.Director WHERE D.id_director="+ numero + " AND P.Genero='" +
(String)das_comboBoxGen.getSelectedItem() + "';";
        Statement stmt =
con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                    ResultSet.CONCUR_READ_ONLY);
        rs = stmt.executeQuery(query);
        if (rs.next()) {
            das_btnSiguiente.setEnabled(true);
            das_textFieldNom.setText(rs.getString(1));
            das_textFieldPais.setText(rs.getString(2));
            das_textFieldGen.setText(rs.getString(3));
            das_textFieldDura.setText(rs.getString(4));
            das_textFieldDir.setText(rs.getString(5));
            das_lblNose.setVisible(false);
        } else {
            das_btnSiguiente.setEnabled(false);
            das_lblNose.setVisible(true);
            das_textFieldNom.setText("");

```

```

        das_textFieldPais.setText("");
        das_textFieldGen.setText("");
        das_textFieldDura.setText("");
        das_textFieldDir.setText("");
    }
}
} catch (SQLException e1) {
    e1.printStackTrace();
    das_lb1Nose.setVisible(true);
}
}
});
das_btnBuscar.setBounds(199, 54, 117, 29);
contentPane.add(das_btnBuscar);

// FUNCION BOTON ANTERIOR
JButton das_btnAnterior = new JButton("Anterior");
das_btnAnterior.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        try {
            if (rs.previous()) {
                das_btnAnterior.setEnabled(true);
                das_btnSiguiente.setEnabled(true);
                das_textFieldNom.setText(rs.getString(1));
                das_textFieldPais.setText(rs.getString(2));
                das_textFieldGen.setText(rs.getString(3));
                das_textFieldDura.setText(rs.getString(4));
                das_textFieldDir.setText(rs.getString(5));
            } else {
                das_btnAnterior.setEnabled(false);
            }
        } catch (SQLException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
    }
});
das_btnAnterior.setEnabled(false);
das_btnAnterior.setBounds(100, 277, 117, 29);
contentPane.add(das_btnAnterior);

// FUNCION BOTON SIGUIENTE
das_btnSiguiente.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        try {
            if (rs.next()) {
                das_btnSiguiente.setEnabled(true);
                das_btnAnterior.setEnabled(true);
                das_textFieldNom.setText(rs.getString(1));
                das_textFieldPais.setText(rs.getString(2));
                das_textFieldGen.setText(rs.getString(3));
                das_textFieldDura.setText(rs.getString(4));
                das_textFieldDir.setText(rs.getString(5));
            } else {
                das_btnSiguiente.setEnabled(false);
            }
        } catch (SQLException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
    }
});
}
});

```

3.8. Main

```
package filмотeca;

import java.awt.EventQueue;

public class Main {

    public static GestionBBDD baseDatos = new GestionBBDD();

    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    MenuPrincipal window = new MenuPrincipal();
                    window.frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }
}
```

3.9. MenuPrincipal

```
package filмотeca;

import javax.swing.JFrame;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class MenuPrincipal {

    public JFrame frame;

    public MenuPrincipal() {

        initialize();
    }

    /**
     * Initialize the contents of the frame.
     */
    private void initialize() {
        frame = new JFrame();
        frame.setBounds(100, 100, 540, 362);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.getContentPane().setLayout(null);
        frame.setLocationRelativeTo(null);

        JButton das_btnNewFilm = new JButton("Insertar Pelicula");
        das_btnNewFilm.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                InsertPelicula ventanaInsertarPel = new InsertPelicula();
                ventanaInsertarPel.setVisible(true);
            }
        });
        das_btnNewFilm.setBounds(70, 39, 168, 45);
        frame.getContentPane().add(das_btnNewFilm);

        JButton das_btnNewDirec = new JButton("Insertar Director");
        das_btnNewDirec.addActionListener(new ActionListener() {
```



```

        public void actionPerformed(ActionEvent e) {
            InsertDirector ventanaInserDir = new InsertDirector();
            ventanaInserDir.setVisible(true);
        }
    });
    das_btnNewDirec.setBounds(311, 39, 168, 45);
    frame.getContentPane().add(das_btnNewDirec);

    JButton das_btnModFilm = new JButton("Modificar Pelicula");
    das_btnModFilm.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            ModPelicula ventanaModPel = new ModPelicula();
            ventanaModPel.setVisible(true);
        }
    });
    das_btnModFilm.setBounds(70, 108, 168, 45);
    frame.getContentPane().add(das_btnModFilm);

    JButton das_btnModDirect = new JButton("Modificar Director");
    das_btnModDirect.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            ModDirector ventanaModDir = new ModDirector();
            ventanaModDir.setVisible(true);
        }
    });
    das_btnModDirect.setBounds(311, 108, 168, 45);
    frame.getContentPane().add(das_btnModDirect);

    JButton das_btnDelFilm = new JButton("Eliminar Pelicula");
    das_btnDelFilm.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            DelPelicula ventanaDelPel = new DelPelicula();
            ventanaDelPel.setVisible(true);
        }
    });
    das_btnDelFilm.setBounds(70, 179, 168, 45);
    frame.getContentPane().add(das_btnDelFilm);

    JButton das_btnDelDirect = new JButton("Eliminar Director");
    das_btnDelDirect.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            EliminarDirector ventanaDelDir = new EliminarDirector();
            ventanaDelDir.setVisible(true);
        }
    });
    das_btnDelDirect.setBounds(311, 179, 168, 45);
    frame.getContentPane().add(das_btnDelDirect);

    JButton das_btnList = new JButton("Listados");
    das_btnList.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            Listados ventanaList = new Listados();
            ventanaList.setVisible(true);
        }
    });
    das_btnList.setBounds(191, 256, 168, 45);
    frame.getContentPane().add(das_btnList);
}

```

3.10. ModDirector

```

package filмотeca;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;

```

```

import javax.swing.JLabel;
import javax.swing.SwingConstants;
import javax.swing.JComboBox;
import javax.swing.JDialog;
import javax.swing.JTextField;

import java.util.ArrayList;

import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.sql.SQLException;
import java.awt.event.ActionEvent;

public class ModDirector extends JFrame {

    private JPanel contentPane;
    private JTextField das_textFieldNombre;
    private JTextField das_textFieldApellidos;

    public ModDirector() {
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setBounds(100, 100, 450, 300);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);
        this.setLocationRelativeTo(null);

        JLabel das_lblSelecc = new JLabel("Selecciona director");
        das_lblSelecc.setHorizontalAlignment(SwingConstants.CENTER);
        das_lblSelecc.setBounds(6, 18, 438, 16);
        contentPane.add(das_lblSelecc);

        // COMBOBOX DIRECTORES
        // Extraccion de directores
        ArrayList<String> directores = new ArrayList<String>();
        try {
            directores = GestionBBDD.listarDirectores();
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        // Conversion a array String del ArrayList
        String[] arrayComboDirectores = directores.toArray(new String[directores.size()]);
        // Creacion del combobox
        JComboBox das_comboBox = new JComboBox(arrayComboDirectores);
        das_comboBox.setBounds(119, 46, 211, 27);
        contentPane.add(das_comboBox);

        JLabel das_lblNombre = new JLabel("Nombre");
        das_lblNombre.setBounds(86, 95, 61, 16);
        contentPane.add(das_lblNombre);

        das_textFieldNombre = new JTextField();
        das_textFieldNombre.setBounds(177, 90, 153, 26);
        contentPane.add(das_textFieldNombre);
        das_textFieldNombre.setColumns(10);

        JLabel das_lblApellidos = new JLabel("Apellidos");
        das_lblApellidos.setBounds(86, 134, 61, 16);
        contentPane.add(das_lblApellidos);

        das_textFieldApellidos = new JTextField();
        das_textFieldApellidos.setBounds(177, 128, 153, 26);
        contentPane.add(das_textFieldApellidos);
        das_textFieldApellidos.setColumns(10);
    }
}

```

```

        JButton das_btnMod = new JButton("Modificar");
        das_btnMod.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // Seleccion primer caracter director
                int numero;
                numero = Funciones.retornaNumDir(das_comboBox.getSelectedItem());
                try {
                    GestionBBDD.modDirector(das_textFieldNombre.getText(),
das_textFieldApellidos.getText(), numero);
                    JDialog das_aviso = new JDialog();
                    JLabel das_mensaje = new JLabel("Director modificado");
                    das_aviso.getContentPane().add(das_mensaje);
                    das_aviso.setSize(300, 100);
                    das_aviso.setLocationRelativeTo(null);
                    das_aviso.setVisible(true);
                    das_aviso.dispose();
                } catch (SQLException e1) {
                    // TODO Auto-generated catch block
                    e1.printStackTrace();
                }
            }
        });
        das_btnMod.setBounds(154, 191, 117, 37);
        contentPane.add(das_btnMod);
    }
}

```

3.11. ModPelicula

```

package filmoteca;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.SwingConstants;
import javax.swing.JComboBox;
import javax.swing.JDialog;

import java.util.ArrayList;

import javax.swing.JButton;
import javax.swing.JTextField;
import java.awt.event.ActionListener;
import java.sql.SQLException;
import java.awt.event.ActionEvent;

public class ModPelicula extends JFrame {

    private JPanel contentPane;
    private JTextField das_textFieldTitulo;
    private JTextField das_textFieldPais;
    private JTextField das_textFieldDuracion;
    private JTextField das_textFieldGenero;

    public ModPelicula() {
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setBounds(100, 100, 450, 383);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);
        this.setLocationRelativeTo(null);
    }
}

```

```
JLabel das_lblSelecc = new JLabel("Selecciona película");
das_lblSelecc.setHorizontalAlignment(SwingConstants.CENTER);
das_lblSelecc.setBounds(6, 21, 438, 16);
contentPane.add(das_lblSelecc);

// COMBOBOX PELICULAS
// Extraccion de peliculas
ArrayList<String> peliculas = new ArrayList<String>();
try {
    peliculas = GestionBBDD.listarPeliculas();
} catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
// Conversion a array String del ArrayList
String[] arrayComboPeliculas = peliculas.toArray(new String[peliculas.size()]);
// Creacion de Combobox
JComboBox das_comboBoxPelicula = new JComboBox(arrayComboPeliculas);
das_comboBoxPelicula.setBounds(84, 49, 283, 27);
contentPane.add(das_comboBoxPelicula);

// COMBOBOX DIRECTORES
// Extraccion de directores
ArrayList<String> directores = new ArrayList<String>();
try {
    directores = GestionBBDD.listarDirectores();
} catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
// Conversion a array String del ArrayList
String[] arrayComboDirectores = directores.toArray(new String[directores.size()]);
// Creacion del combobox
JComboBox das_comboBoxDirector = new JComboBox(arrayComboDirectores);
das_comboBoxDirector.setBounds(177, 124, 190, 27);
contentPane.add(das_comboBoxDirector);

JLabel das_lblTitulo = new JLabel("Título:");
das_lblTitulo.setBounds(84, 92, 61, 16);
contentPane.add(das_lblTitulo);

JLabel das_lblDirector = new JLabel("Director:");
das_lblDirector.setBounds(84, 128, 61, 16);
contentPane.add(das_lblDirector);

JLabel das_lblPais = new JLabel("País:");
das_lblPais.setBounds(84, 166, 61, 16);
contentPane.add(das_lblPais);

JLabel das_lblDuracion = new JLabel("Duración:");
das_lblDuracion.setBounds(84, 204, 81, 16);
contentPane.add(das_lblDuracion);

JLabel das_lblGenero = new JLabel("Género:");
das_lblGenero.setBounds(84, 242, 61, 16);
contentPane.add(das_lblGenero);

das_textFieldTitulo = new JTextField();
das_textFieldTitulo.setBounds(177, 87, 190, 26);
contentPane.add(das_textFieldTitulo);
das_textFieldTitulo.setColumns(10);

das_textFieldPais = new JTextField();
das_textFieldPais.setBounds(177, 161, 190, 26);
```

```

        contentPane.add(das_textFieldPais);
        das_textFieldPais.setColumns(10);

        das_textFieldDuracion = new JTextField();
        das_textFieldDuracion.setBounds(177, 199, 190, 26);
        contentPane.add(das_textFieldDuracion);
        das_textFieldDuracion.setColumns(10);

        das_textFieldGenero = new JTextField();
        das_textFieldGenero.setBounds(177, 237, 190, 26);
        contentPane.add(das_textFieldGenero);
        das_textFieldGenero.setColumns(10);

        JButton das_btnMod = new JButton("Modificar");
        das_btnMod.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // Seleccion primer caracter director
                int numero;
                numero =
Funciones.retornaNumDir(das_comboBoxDirector.getSelectedItem());
                // Modificar pelicula
                try {

                    GestionBBDD.modPelicula((String)das_comboBoxPelicula.getSelectedItem(),
das_textFieldTitulo.getText(), numero, das_textFieldPais.getText(),
Integer.parseInt(das_textFieldDuracion.getText()), das_textFieldGenero.getText());
                    JDialog das_avisos = new JDialog();
                    JLabel das_mensaje = new JLabel("Película modificada");
                    das_avisos.getContentPane().add(das_mensaje);
                    das_avisos.setSize(300, 100);
                    das_avisos.setLocationRelativeTo(null);
                    das_avisos.setVisible(true);
                    das_avisos.dispose();
                } catch (NumberFormatException e1) {
                    // TODO Auto-generated catch block
                    e1.printStackTrace();
                } catch (SQLException e1) {
                    // TODO Auto-generated catch block
                    e1.printStackTrace();
                }
            }
        });
        das_btnMod.setBounds(145, 291, 151, 41);
        contentPane.add(das_btnMod);

```

4. CODIGO COMPLETO EJERCICIO TIENDAS

4.1. Empleado

```

package tiendas;

import javax.persistence.*;

@Entity
public class Empleado {

    @Id
    @GeneratedValue
    private long id;

```

```

private String nombre;
private String apellido;

// CONSTRUCTOR
public Empleado(long id, String nombre, String apellido) {
    super();
    this.id = id;
    this.nombre = nombre;
    this.apellido = apellido;
}

// GETTERS AND SETTERS
public long getId() {
    return id;
}

public String getNombre() {
    return nombre;
}

public void setNombre(String nombre) {
    this.nombre = nombre;
}

public String getApellido() {
    return apellido;
}

public void setApellido(String apellido) {
    this.apellido = apellido;
}

// toString
@Override
public String toString() {
    return "Empleado [id=" + id + ", nombre=" + nombre + ", apellido=" + apellido +
"]";
}

```

4.2. Funciones

```

package tiendas;

import java.util.ArrayList;
import java.util.Scanner;

import javax.persistence.EntityManager;
import javax.persistence.TypedQuery;

public class Funciones {

    // Funcion para pedir un numero entero
    public static Integer pideEntero() {
        boolean salir = false;

```

```

Integer entero = 0;
Scanner scanner = new Scanner(System.in);
do {
    try {
        entero = scanner.nextInt();
        salir = true;
    } catch (Exception exc1) {
        System.out.println("Valor introducido invalido, vuelve a intentarlo");
        scanner.next();
    }
} while (!salir);
return entero;
}

// MENU
public static void menu() {
    String[] muestraMenu;
    muestraMenu = new String[9];
    muestraMenu[0] = "*****";
    muestraMenu[1] = "Introduzca la operacion a realizar del siguiente menú de opciones";
    muestraMenu[2] = "1. Muestra los empleados";
    muestraMenu[3] = "2. Muestra las tiendas";
    muestraMenu[4] = "3. Mostrar las tiendas ordenadas por ventas";
    muestraMenu[5] = "4. Modificar un empleado";
    muestraMenu[6] = "5. Añade una tienda";
    muestraMenu[7] = "0. Salir";
    muestraMenu[8] = "*****";

    for (String opcion : muestraMenu) {
        System.out.println(opcion);
    }
}

// SELECTS
// 1. Mostrar empleados
public static void verEmpleados(EntityManager em) {
    TypedQuery<Empleado> query = em.createQuery("SELECT e FROM Empleado e", Empleado.class);

    ArrayList<Empleado> empleados = (ArrayList<Empleado>) query.getResultList();

    for (Empleado e : empleados) {
        System.out.println(e);
    }
}

// 2. Mostrar tiendas
public static void verTiendas(EntityManager em) {
    TypedQuery<Tienda> query = em.createQuery("SELECT t FROM Tienda t", Tienda.class);

    ArrayList<Tienda> tiendas = (ArrayList<Tienda>) query.getResultList();

    for (Tienda t : tiendas) {
        System.out.println(t);
    }
}

// 3. Mostrar tiendas por orden ascendente o descendente segun seleccion
public static void verTiendasOrdenadas(EntityManager em) {
    System.out.println("Ordenar segun ventas ascendentes o descendentes?");
    System.out.println("1- Ascendente");
    System.out.println("2- Descendente");
    switch (pideEntero()) {
        case 1 :
            TypedQuery<Tienda> query1 = em.createQuery("SELECT t FROM Tienda t ORDER
BY t.ventas ASC", Tienda.class);

```

```

        ArrayList<Tienda> tiendas1 = (ArrayList<Tienda>) query1.getResultList();

        for (Tienda t1 : tiendas1) {
            System.out.println(t1);
        }
        break;

        case 2 :
            TypedQuery<Tienda> query2 = em.createQuery("SELECT t FROM Tienda t ORDER
BY t.ventas DESC", Tienda.class);

            ArrayList<Tienda> tiendas2 = (ArrayList<Tienda>) query2.getResultList();

            for (Tienda t2 : tiendas2) {
                System.out.println(t2);
            }
            break;
        }
    }

    // MODIFICAR EMPLEADOS
    // 4. Modificar un empleado
    public static void modEmpleado(EntityManager em, int id) {
        Scanner scanner = new Scanner(System.in);
        String nombre;
        String apellido;
        TypedQuery<Empleado> query = em.createQuery("SELECT e FROM Empleado e WHERE e.id="+id,
Empleado.class);
        Empleado e1 = query.getSingleResult();
        System.out.println("El empleado seleccionado es: " + e1.toString());
        System.out.println("Indica el atributo a modificar:");
        System.out.println("1- Nombre");
        System.out.println("2- Apellido");
        System.out.println("3- Ninguno");

        switch(pideEntero()) {
            case 1:
                em.getTransaction().begin();
                System.out.println("Indica el nuevo nombre");
                nombre = scanner.next();
                e1.setNombre(nombre);
                em.getTransaction().commit();

                break;

            case 2:
                em.getTransaction().begin();
                System.out.println("Indica el nuevo apellido");
                apellido = scanner.next();
                e1.setApellido(apellido);
                em.getTransaction().commit();

                break;

            case 3:
                break;

        }
    }

    // CREACION DE TIENDAS
    // 5. Crear una nueva tienda
    public static void nuevaTienda(EntityManager em) {
        try {
            // Inicia la transaccion
            em.getTransaction().begin();
            // Variables
            boolean salir = false;

```



```

Scanner scanner = new Scanner(System.in);
int idTienda;
String dire;
int ventas;
int id;
ArrayList<Empleado> listEmpleados = new ArrayList<Empleado>();
// Peticiones
System.out.println("Indica un id para la tienda");
idTienda = pideEntero();
System.out.println("Indica una direccion");
dire = scanner.next();
System.out.println("Indica las ventas");
ventas = pideEntero();
verEmpleados(em);
System.out.println("Indica el ID del empleado que quieras añadir a la tienda");
id = pideEntero();
// Búsqueda del empleado para añadirlo al ArrayList
TypedQuery<Empleado> query = em.createQuery("SELECT e FROM Empleado e WHERE e.id="+id,
Empleado.class);
listEmpleados.add(query.getSingleResult());
System.out.println("Empleado añadido correctamente");

do {
    System.out.println("Quiere añadir otro empleado?");
    System.out.println("1- Si");
    System.out.println("2- No");

    switch (pideEntero()) {
        case 1:
            System.out.println("Indica el ID del empleado que quieras añadir a
la tienda");
            id = pideEntero();
            query = em.createQuery("SELECT e FROM Empleado e WHERE e.id="+id,
Empleado.class);
            listEmpleados.add(query.getSingleResult());
            System.out.println("Empleado añadido correctamente");
            break;
        case 2:
            salir = true;
            break;
    }

} while (!salir);

// Crear tienda
Tienda nuevaTienda = new Tienda(idTienda,dire,ventas,listEmpleados);
// Commit
em.persist(nuevaTienda);
em.getTransaction().commit();

} catch (Exception e) {
    System.out.println("El ID de tienda ya existe");
}
}
}

```

4.3. GestionaTiendas

```

package tiendas;

import java.util.ArrayList;

import javax.persistence.*;

public class GestionaTiendas {

```

```

public static void main(String[] args) throws Exception {
    // Creacion de la BBDD
    EntityManagerFactory emf =
Persistence.createEntityManagerFactory("objectdb:db/tiendasDB.tmp;drop");
    EntityManager em = emf.createEntityManager();

    // Inicio de transaccion
    em.getTransaction().begin();

    // Creacion de tres empleados
    Empleado empleado1 = new Empleado(1,"Daniel","Aguilar");
    Empleado empleado2 = new Empleado(2,"Alicia","Revilla");
    Empleado empleado3 = new Empleado(3,"Vin","Diesel");

    // Creacion de tres tiendas
    ArrayList<Empleado> listEmpleadost1 = new ArrayList<Empleado>();
    ArrayList<Empleado> listEmpleadost2 = new ArrayList<Empleado>();
    ArrayList<Empleado> listEmpleadost3 = new ArrayList<Empleado>();
    listEmpleadost1.add(empleado1);
    listEmpleadost2.add(empleado2);
    listEmpleadost3.add(empleado3);
    Tienda tienda1 = new Tienda(1,"Calle Pepe",1000,listEmpleadost1);
    Tienda tienda2 = new Tienda(2,"Calle Carlos",100,listEmpleadost2);
    Tienda tienda3 = new Tienda(3,"Calle Mayor",400,listEmpleadost3);

    // Insertar datos en la BBDD
    em.persist(tienda1);
    em.persist(tienda2);
    em.persist(tienda3);
    em.getTransaction().commit();

    // MENU
    boolean salir = false;
    do {
        Funciones.menu();
        switch(Funciones.pideEntero()) {

            case 1:
                Funciones.verEmpleados(em);
                break;

            case 2:
                Funciones.verTiendas(em);
                break;

            case 3:
                Funciones.verTiendasOrdenadas(em);
                break;

            case 4:
                System.out.println("Introduce el ID del empleado a modificar: ");
                Funciones.modEmpleado(em, Funciones.pideEntero());
                break;

            case 5:
                Funciones.nuevaTienda(em);
                break;

            case 0:
                System.out.println("Gracias por usar el programa");
                salir = true;
                break;

        }
    } while (!salir);
}

```

4.4. Tienda

```

package tiendas;

import java.util.ArrayList;
import javax.persistence.*;

@Entity
public class Tienda {

    @Id
    @GeneratedValue
    private long id;
    private String direccion;
    private int ventas;
    @OneToOne(cascade = CascadeType.ALL, orphanRemoval=true)
    private ArrayList<Empleado> empleados;

    // CONSTRUCTOR
    public Tienda(long id, String direccion, int ventas, ArrayList<Empleado> empleados) {
        super();
        this.id = id;
        this.direccion = direccion;
        this.ventas = ventas;
        this.empleados = empleados;
    }

    // GETTERS AND SETTERS
    public long getId() {
        return id;
    }

    public String getDireccion() {
        return direccion;
    }

    public void setDireccion(String direccion) {
        this.direccion = direccion;
    }

    public int getVentas() {
        return ventas;
    }

    public void setVentas(int ventas) {
        this.ventas = ventas;
    }

    public ArrayList<Empleado> getEmpleados() {
        return empleados;
    }

    public void setEmpleados(ArrayList<Empleado> empleados) {
        this.empleados = empleados;
    }

    // toString
    @Override
    public String toString() {
        return "Tienda [id=" + id + ", direccion=" + direccion + ", ventas=" + ventas + ", empleados=" + empleados + "]\n";
    }
}

```