U-tad
CENTRO PROFESIONAL
DE TECNOLOGÍA Y ARTE DIGITAL

U-tad
CENTRO UNIVERSITARIO
DE TECNOLOGÍA Y ARTE DIGITAL

TAREA ADAT_UD1_02_JDBC

Autor: Marta_Molina

Fecha: 3_10_2021

CFGS DAMP

Índice

1. Introducción

La aplicación presentada en esta tarea, es una la cual servirá para dar de alta, baja y modificar artículos de una base de datos.

Esta reducida aplicación ha sido desarrollada en Java. El IDE usado para ello ha sido eclipse con el plugin de WindowBuilder para facilitar el desarrollo de la interfaz gráfica.
He creado las clases siguiendo el modelo MVC.

Las principales clases extendidas han sido:

- javax.swing
- java.awt
- java.sql
- com.mysql.cj.jdbc.result

2. Características

La aplicación cuenta con las siguientes características:

FrontEnd	
Vistas.java	Crea la ventana del programa y añade los elementos visuales.
martaMolina.png	Imagen con extensión ".png" del logo de la aplicación.
rectangle1.png	Imagen con extensión ".png" de un rectángulo.

Tabla 1 Front End

BackEnd	
Launcher.java	Contiene el main. Lanza la aplicación. Crea una instancia del Contolador, la Vista y el Modelo.
Moldelo.java	Contiene la lógica de la aplicación.
Controlador.java	Crea la interacción entre la vista y el Modelo.

Tabla 2 BackEnd

3. Desarrollo

3.1. Launcher.java

Lanzadera de la aplicación, crea la comunicación entre clases a través de los setters.

3.1.1 Main

Clase principal que contiene el main. Es la encargada de lanzar la aplicación. Main de la clase la cual crea un objeto de Vista, Modelo y Controlador. Crea y hace visible la ventana.

```
 * @author Marta Molina Aguilera
package Controlador;

import Modelo.Modelo;
import Vista.Vista;

public class Launcher {
    @SuppressWarnings("unused")
    private Modelo miModelo;
    @SuppressWarnings("unused")
    private Vista miVista;

     * Main de la clase la cual crea un objeto de Vista, Modelo y Controlador. Hace
    public static void main(String[] args) {
        Vista miVista = new Vista();
        Modelo miModelo = new Modelo();
        Controlador miControlador = new Controlador();

        miControlador.setModelo(miModelo);
        miControlador.setVista(miVista);

        miModelo.setVista(miVista);

        miVista.setModelo(miModelo);
        miVista.setControlador(miControlador);

        miVista.setVisible(true);

    }
```

Captura 1 Launcher.java – Main

3.1.1 setModelo(Modelo miModelo)

Setter del modelo, el cual contiene logica del código. Consiguiendo así la comunicación entre clases. Es llamado desde la lanzadera.

```
public void setModelo(Modelo miModelo) {
    this.miModelo = miModelo;
}
```

Captura 2 Launcher.java -setModelo()

3.2 Modelo.java

Clase la cual contiene la lógica de la aplicación.

3.2.1 Declaración de variables y Constructor.

Constructor de la clase Modelo. Nada más ser creado un objeto Modelo, este llama desde el constructor a la función de crear una conexión con la BD y cargar la tabla con la información que esta le proporciona.

```
25  public class Modelo {
26      private Vista miVista;
27      private DefaultTableModel modelo;
28
29      private String bd = "Books";
30      private String login = "root";
31      private String pwd = "";
32      private String url = "jdbc:mysql://localhost/" + bd;
33      private Connection conexion;
34      private PreparedStatement pstmt;
35      private String sqlTabla1 = "SELECT * FROM BooksTable;";
36
38+     * Constructor de la clase Modelo. Nada más ser creado un objeto Modelo, este
42-     public Modelo() {
43          // Cargar pantalla al empezar
44          crearConexion();
45          cargarTabla1();
46      }
```

Captura 3 Modelo.java-Constructor

3.2.2 crearConexion()

Función que crea la conexión con la base de datos.

```
56-     private void crearConexion() {
57          try {
58              Class.forName("com.mysql.cj.jdbc.Driver");
59              conexion = DriverManager.getConnection(url, login, pwd);
60          } catch (ClassNotFoundException cnfe) {
61              System.err.println("Driver JDBC no encontrado");
62              cnfe.printStackTrace();
63          } catch (SQLException sqle) {
64              System.err.println("Error al conectarse a al BD");
65              sqle.printStackTrace();
66          } catch (Exception e) {
67              System.err.println("Error general");
68              e.printStackTrace();
69          }
70      }
```

Captura 4 Modelo.java-crearConexion( )

- Lanza una *ClassNotFoundException cnfe*: si no encuentra el driver mysql-connector-java.
- Lanza una *SQLException sqle:* Si no puede conectarse a la base de datos.
- Lanza *Exception e:* Con cualquier error de otro tipo.

3.2.3 cargarTabla1()

Función la cual carga los datos de la tabla BooksTable y hace que la tabla de la ventana no sea editable. Llama a las funciones getNumColumnas(sqlTabla1) y getNumFilas(sqlTabla1). Que le devuelven el numero de resultados y columnas para asignar las dimensiones a la tabla.

- Lanza una *SQLException e*: Si hubiera problemas accediendo a la Base de Datos.

```
79      private void cargarTabla1() {
80          modelo = new DefaultTableModel();
81          int numColumnas = getNumColumnas(sqlTabla1);
82          int numFilas = getNumFilas(sqlTabla1);
83
84          String[] cabecera = new String[numColumnas];
85          Object[][] contenido = new Object[numFilas][numColumnas];
86          try {
87              pstmt = conexion.prepareStatement(sqlTabla1);
88              ResultSet rset = pstmt.executeQuery();
89              ResultSetMetaData rsmd = (ResultSetMetaData) rset.getMetaData();
90              for (int i = 0; i < numColumnas; i++) {
91                  cabecera[i] = rsmd.getColumnName(i + 1);
92              }
93              int fila = 0;
94              while (rset.next()) {
95                  for (int col = 1; col <= numColumnas; col++) {
96                      contenido[fila][col - 1] = rset.getString(col);
97                  }
98                  fila++;
99              }
100         } catch (SQLException e) {
101             e.printStackTrace();
102         }
103         modelo = new DefaultTableModel(contenido, cabecera) {
104             public boolean isCellEditable(int row, int column) {
105                 return false;// This causes all cells to be not editable
106             }
107         };
108
```

Captura 5 Modelo.java - cargarTabla1()

3.2.4 getNumColumnas(String sql) y getNumFilas(String sql)

Funciones las cuales retornan el número de columnas/filas generado por la consulta ejecutada.

Toman como parámetro la query a ejecutar. Retornan un entero con el nuevo devuelto tras la consulta.

Lanzan:

- *SQLException e:* si hay error al consultar la base de datos.

```java
    private int getNumColumnas(String sql) {
        int num = 0;
        try {
            PreparedStatement pstmt = conexion.prepareStatement(sql);
            ResultSet rset = pstmt.executeQuery();
            ResultSetMetaData rsmd = (ResultSetMetaData) rset.getMetaData();
            num = rsmd.getColumnCount();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return num;
    }

     * Función la cual retorna el numero de filas generado por la consulta
    private int getNumFilas(String sql) {
        int numFilas = 0;
        try {
            PreparedStatement pstmt = conexion.prepareStatement(sql);
            ResultSet rset = pstmt.executeQuery();
            while (rset.next())
                numFilas++;
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return numFilas;
    }
```

Captura 6 Modelo.java - getNumColumnas(String sql) y getNumFilas(String sql)

3.2.5 annadirRegistro(String titulo, String autor, String categoria, String precio)

Función la cual hace un insert a la base de datos y añade una fila con estos a la tabla. Si hay un error, muestra un mensaje en la ventana. Toma como parámetros los string pasados por el controlador en la función con el mismo nombre.

Lanza:

- *SQLException e:* si hay error al consultar la base de datos y llama a la acción de la vista de cambiarError( )

```java
168     public void annadirRegistro(String titulo, String autor, String categoria, String precio) {
169         try {
170             Statement stmt = conexion.createStatement();
171             String qery = "INSERT INTO `BooksTable` (`Titulo`, `Autor`, `Categoria`, `Precio`) VALUES ('" + titulo
172                     + "','" + autor + "','" + categoria + "','" + precio + "')";
173             stmt.executeUpdate(qery);
174             stmt.close();
175             modelo.insertRow(modelo.getRowCount(), new String[] { titulo, autor, categoria, precio.toString() });
176             miVista.cambiarMsgResultado("Los datos han sido agregados");
177             miVista.setVisible(false);
178             miVista.setVisible(true);
179         } catch (SQLException e) {
180             miVista.cambiarError("Error al añadir registro. Cierre y abara la app de nuevo.");
181             System.err.println(e);
182         }
183     }
```

Captura 7 Modelo.java - annadirRegistro(String titulo, String autor, String categoria, String precio)

3.2.6 modificarRegistro(String[] datos)

Método el cual modifica los datos del registro seleccionado en la ventana. Toma cómo parámetro el array de Strings enviado por el controaldor en la función de mismo nombre. Mediante un ResultSetMetaData toma los nombres de las columnas de las tablas de la base de Datos y con un PrepareStatement actualiza la fila seleccionada.
Lanza:

- *SQLException e:* si hay error al consultar la base de datos y llama a la acción de la vista de cambiarError( )

```
192    public void modificarRegistro(String[] datos) {
193
194        try {
195            Statement stmt = conexion.createStatement();
196            ResultSet rset = pstmt.executeQuery();
197            ResultSetMetaData rsmd = (ResultSetMetaData) rset.getMetaData();
198            String[] cabecera = new String[getNumColumnas(sqlTabla1)];
199            String seleccion = datos[0];
200
201            // Conseguir nombre de las columnas
202            for (int i = 0; i < getNumColumnas(sqlTabla1); i++) {
203                cabecera[i] = rsmd.getColumnName(i + 1);
204            }
205            String qry = "";
206
207            for (int i = 1; i < cabecera.length; ++i) {
208                qry = "UPDATE BooksTable SET " + cabecera[i] + "='" + datos[i + 1] + "' WHERE Titulo ='" + seleccion
209                        + "';";
210                PreparedStatement pst;
211                pst = conexion.prepareStatement(qry);
212                stmt.executeUpdate(qry);
213            }
214            // Para que la modificación del titulo no pise el resto
215            qry = "UPDATE BooksTable SET " + cabecera[0] + "='" + datos[1] + "' WHERE Titulo ='" + seleccion + "';";
216            pstmt = conexion.prepareStatement(qry);
217            stmt.executeUpdate(qry);
218            stmt.close();
219            miVista.cambiarMsgResultado(
220                    "Los datos han sido modificados.");
221            cargarTabla1();
222            miVista.setVisible(false);
223            miVista.setVisible(true);
224
225        } catch (SQLException e) {
226            miVista.cambiarError("Error al añadir registro. Cierre y abara la app de nuevo.");
227            System.err.println(e);
228        }
229    }
```

Captura 8 Modelo.java - modificarRegistro(String[] datos)

3.2.7 borrarRegistro(String titulo)

Método el cual borra el registro seleccionado en la tabla de miVista y en la base de datos. Toma cómo parámetro título que equivale al id “Título” que es el ID de la tabla BooksTable de la base de datos.

Lanza:

- *SQLException e:* si hay error al consultar la base de datos y llama a la acción de la vista de cambiarError( )

```
238    public void borrarRegistro(String titulo) {
239        try {
240            Statement stmt = conexion.createStatement();
241            ResultSet rset = pstmt.executeQuery();
242            String qry = "DELETE FROM BooksTable WHERE Titulo='" + titulo + "';";
243            PreparedStatement pst;
244            pst = conexion.prepareStatement(qry);
245            stmt.executeUpdate(qry);
246            stmt.close();
247            cargarTabla1();
248            miVista.cambiarMsgResultado("El registro ha sido eliminado.");
249            miVista.borrarFila();
250        } catch (SQLException e) {
251            miVista.cambiarError("Error al añadir registro. Cierre y abara la app de nuevo.");
252            System.err.println(e);
253        }
254    }
```

Captura 9 Modelo.java - borrarRegistro(String titulo)

3.2.8 mostrarSelección(String[] titutlo)

Método el cual llama a la Vista y hace visible en pantalla el título del registro seleccionado. Asigna en los campos de texto los valores del artículo seleccionado. Toma cómo parámetro el título de la fila seleccionada en la vista.

```
264    public void mostrarSeleccion(String titulo) {
265        miVista.setSeleccion(titulo);
266    }
267
```

Captura 10 Modelo.java - mostrarSeleccion(String titulo)

3.2.9 getModelo()

Función la cual retorna un objeto de la clase actual.

```
273    public DefaultTableModel getModelo() {
274        return modelo;
275    }
```

Captura 11 Modelo.java - getModelo()

3.2.10 setVista(Vista miVista)

Método el cual asigna los valores de la clase Vista a la instancia declarada en esta clase. Consiguiendo así la comunicación entre clases. Toma cómo parámetro un objeto de la clase vista.

```
283     public void setVista(Vista miVista) {
284         this.miVista = miVista;
285     }
286
```

Captura 12 setVista(Vista miVista)

3.3 Controlador.java

Clase la cual comunica la vista con el modelo.

3.3.1 setModelo(Modelo miModelo) y setVista(Vista miVista)

Setters del modelo y la vista, los cuales crean la comunicación entre el MVC.

```
13  public class Controlador {
14      private Modelo miModelo;
15      private Vista miVista;
16
18      * Setter del Modelo para crear comunicación con él.
22      public void setModelo(Modelo miModelo) {
23          this.miModelo = miModelo;
24      }
25
27      * Setter de la Vista para crear comunicación con él.
31      public void setVista(Vista miVista) {
32          this.miVista = miVista;
33      }
```

Captura 13 Controlador.java- setModelo(Modelo miModelo) y setVista(Vista miVista)

3.3.2 annadirRegistro()

Método el cual llama a la función del modelo "annadirRegistro" pasa los datos de los getters de los campos de texto a este a través de los parámetros.

```
41      public void annadirRegistro() {
42          miModelo.annadirRegistro(miVista.getTitulo(), miVista.getAutor(), miVista.getCategoria(), miVista.getPrecio());
43      }
```

Captura 14 Controlador.java - annadirRegistro()

3.3.3 mostrarSeleccion()

Método el cual llama a la función del modelo del mismo nombre y le pasa cómo parámetro el título de la fila seleccionada de la vista.

```
49    public void mostrarSeleccion() {
50        miModelo.mostrarSeleccion(miVista.getSeleccionTitulo());
51    }
52
```

Captura 15 Controlador.java - mostrarSeleccion()

3.3.4 modificarRegistro()

Método el cual llama a la función del modelo del mismo nombre, la cual pasa en un array de Strings el título seleccionado y los datos escritos en los campos de texto.

```
58    public void modificarRegistro() {
59        String[] datos = { miVista.getTituloSeleccionado(), miVista.getTitulo(), miVista.getAutor()
60                miVista.getCategoria(), miVista.getPrecio() };
61        miModelo.modificarRegistro(datos);
62    }
```

Captura 16 Controlador.java - modificarRegistro()

3.3.5 borrarRegistro()

Método el cual llama a la función con el mismo nombre del controlador, le pasa cómo parámetro el título del elemento seleccionado en la tabla.

```
58    public void borrarRegistro() {
59        miModelo.borrarRegistro(miVista.getTituloSeleccionado());
70    }
```

Captura 17 Controlador.java - borrarRegistro()

3.4 Vista

Clase la cual implementa todos los atributos visuales de la aplicación.

3.4.1 Constructor

Constructor de la clase, aplica todos los atributos visuales a la ventana, le otorga un título y un icono.

```java
public Vista() {
    setResizable(false);
    addWindowListener(new WindowAdapter() {
        @Override
        public void windowActivated(WindowEvent e) {
            table.setModel(miModelo.getModelo());
        }
    });
    setTitle("Marta MA - UD1_02_JDBC");
    setIconImage(img.getImage());
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 941, 690);
    contentPane = new JPanel();
    contentPane.setBackground(new Color(242, 232, 207));
    contentPane.setBorder(new EmptyBorder(2, 2, 2, 2));
    setContentPane(contentPane);
    contentPane.setLayout(null);
    UIManager.put("Button.select", new Color(56, 102, 65));

    JScrollPane scrollPane = new JScrollPane();
    scrollPane.setBounds(60, 307, 805, 304);
    scrollPane.setBorder(new EmptyBorder(getInsets()));
    contentPane.add(scrollPane);

    table = new JTable();
    table.addMouseListener(new MouseAdapter() {
        @Override
        public void mouseClicked(MouseEvent e) {
            miControlador.mostrarSeleccion();
        }
    });
    table.setBorder(new EmptyBorder(getInsets()));
    table.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
```

Captura 18 Vista.java - Constructor pte.1

```java
    scrollPane.setViewportView(table);

    JLabel lblBBDD = new JLabel("Base de Datos - Librería Colinas");
    lblBBDD.setForeground(new Color(56, 102, 65));
    lblBBDD.setFont(new Font("Tahoma", Font.BOLD, 32));
    lblBBDD.setBounds(60, 45, 543, 68);
    contentPane.add(lblBBDD);

    JLabel lblTitulo = new JLabel("Título:");
    lblTitulo.setForeground(new Color(56, 102, 65));
    lblTitulo.setBounds(60, 140, 100, 30);
    contentPane.add(lblTitulo);

    txtTitulo = new JTextField();
    txtTitulo.addFocusListener(new FocusAdapter() {
        public void focusGained(FocusEvent e) {
            cambiarError("");
        }
    });
    txtTitulo.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
    txtTitulo.setMargin(new Insets(2, 22, 2, 22));
    txtTitulo.setBounds(119, 140, 300, 30);
    contentPane.add(txtTitulo);
    txtTitulo.setColumns(10);

    JLabel lblAutor = new JLabel("Autor:");
    lblAutor.setForeground(new Color(56, 102, 65));
    lblAutor.setBounds(470, 140, 100, 30);
    contentPane.add(lblAutor);

    txtAutor = new JTextField();
    txtAutor.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
    txtAutor.addFocusListener(new FocusAdapter() {
        public void focusGained(FocusEvent e) {
            cambiarError("");
        }
```

Captura 19 Vista.java - Constructor pte.2

```
127         });
128         txtAutor.setColumns(10);
129         txtAutor.setBounds(556, 140, 300, 30);
130         contentPane.add(txtAutor);
131
132         JLabel lblCategora = new JLabel("Categoría:");
133         lblCategora.setForeground(new Color(56, 102, 65));
134         lblCategora.setBounds(60, 200, 100, 30);
135         contentPane.add(lblCategora);
136
137         txtCategoria = new JTextField();
138         txtCategoria.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
139         txtCategoria.addFocusListener(new FocusAdapter() {
140             public void focusGained(FocusEvent e) {
141                 cambiarError("");
142             }
143         });
144         txtCategoria.setColumns(10);
145         txtCategoria.setBounds(119, 200, 300, 30);
146         contentPane.add(txtCategoria);
147
148         JLabel lblPrecio = new JLabel("Precio:");
149         lblPrecio.setForeground(new Color(56, 102, 65));
150         lblPrecio.setBounds(470, 200, 100, 30);
151         contentPane.add(lblPrecio);
152
153         txtPrecio = new JTextField();
154         txtPrecio.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
155         txtPrecio.addFocusListener(new FocusAdapter() {
156             public void focusGained(FocusEvent e) {
157                 cambiarError("");
158             }
159         });
160         txtPrecio.setColumns(10);
161         txtPrecio.setBounds(556, 200, 300, 30);
162         contentPane.add(txtPrecio);
```

Captura 20 Vista.java - Constructor pte.3

```
164         JButton btnAgregar = new JButton("Agregar");
165         btnAgregar.setToolTipText("Agregar");
166         btnAgregar.setBackground(new Color(106, 153, 78));
167         btnAgregar.setForeground(new Color(242, 232, 207));
168         btnAgregar.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
169         btnAgregar.addActionListener(new ActionListener() {
170             public void actionPerformed(ActionEvent e) {
171                 miControlador.annadirRegistro();
172
173             }
174         });
175         btnAgregar.setBounds(119, 260, 95, 30);
176         contentPane.add(btnAgregar);
177
178         lblError = new JLabel("");
179         lblError.setHorizontalAlignment(SwingConstants.LEFT);
180         lblError.setBounds(119, 238, 508, 14);
181         contentPane.add(lblError);
182
183         JButton btnModificar = new JButton("Modificar");
184         btnModificar.setToolTipText("Modificar");
185         btnModificar.setBorder(new EmptyBorder(getInsets()));
186         btnModificar.setBackground(new Color(106, 153, 78));
187         btnModificar.setForeground(new Color(242, 232, 207));
188         btnModificar.addActionListener(new ActionListener() {
189             public void actionPerformed(ActionEvent e) {
190                 miControlador.modificarRegistro();
191             }
192         });
```

Captura 21 Vista.java - Constructor pte.4

```
193         btnModificar.setBounds(222, 260, 95, 30);
194         contentPane.add(btnModificar);
195
196         JButton btnEliminar = new JButton("Eliminar");
197         btnEliminar.setToolTipText("Eliminar");
198         btnEliminar.setBorder(new EmptyBorder(getInsets()));
199         btnEliminar.setBackground(new Color(106, 153, 78));
200         btnEliminar.setForeground(new Color(242, 232, 207));
201         btnEliminar.addActionListener(new ActionListener() {
202             public void actionPerformed(ActionEvent e) {
203                 miControlador.borrarRegistro();
204             }
205         });
206         btnEliminar.setBounds(324, 260, 95, 30);
207         contentPane.add(btnEliminar);
208
209         lblSeleccion = new JLabel("Selección:");
210         lblSeleccion.setForeground(new Color(56, 102, 65));
211         lblSeleccion.setBounds(470, 260, 386, 30);
212         contentPane.add(lblSeleccion);
213
214         lblRectangle = new JLabel("");
215         ImageIcon rectangle1 = new ImageIcon("imgs/Rectangle1.png");
216         Image newRecImg1 = rectangle1.getImage().getScaledInstance(885, 619, java.awt.Image.SCALE_SMOOTH);
217         lblRectangle.setIcon(new ImageIcon(newRecImg1));
218         lblRectangle.setBounds(20, 22, 885, 619);
219         contentPane.add(lblRectangle);
220     }
```

Captura 22 Vista.java - Constructor pte.5

3.4.1 borrarFila()

Método el cual quita la fila seleccionada de la tabla.

```
225     public void borrarFila() {
226         int fila = table.getSelectedRow();
227         ((DefaultTableModel) table.getModel()).removeRow(fila);
228     }
```

Captura 23 Vista.java - borrarFila()

3.4.1 getters de los campos de texto

Métodos los cuales recogen y devuelven el texto escrito en el respectivo campo de texto.

```
237     public String getTitulo() {
238         return txtTitulo.getText();
239     }
240
242      * Método el cual devuelve lo escrito en el campo de texto del autor.
246     public String getAutor() {
247         return txtAutor.getText();
248     }
249
251      * Método el cual devuelve lo escrito en el campo de texto de la categoría.
255     public String getCategoria() {
256         return txtCategoria.getText();
257     }
258
260      * Método el cual devuelve lo escrito en el campo de texto del precio.
264     public String getPrecio() {
265         return txtPrecio.getText();
266     }
```

Captura 24 Vista.java - get de campos de texto.

3.4.2 getSeleccionTitulo()

Método el cual devuelve el título del elemento seleccionado y asigna su valor.

```
275    public String getSeleccionTitulo() {
276        int fila = table.getSelectedRow();
277        String seleccion = table.getValueAt(fila, 0).toString();
278        tituloSeleccionado = seleccion;
279        return seleccion;
280    }
```

Captura 25 Vista.java - getTitulo()

3.4.3 getTituloSeleccionado()

Método el cual devulve el String del tituloSeleccionado (variable de String, no el campo de texto).

```
288    public String getTituloSeleccionado() {
289        return tituloSeleccionado;
290    }
291
```

Captura 26 Vista.java - getSeleccionTitulo()

3.4.4 setSeleccion(String seleccion)

Función la cual asigna el String pasado por parámetro a la lblSeleccion. Toma cómo parámetro seleccion: Título del elemento seleccionado, tipo String.

```
299    public void setSeleccion(String seleccion) {
300        lblSeleccion.setText("Selección: " + seleccion);
301    }
302
```

Captura 27 Vista.java - getTituloSeleccionado()

3.4.5 cambiarError(String msgError)

Función la cual asigna color y mensaje a la etiqueta de error. Toma cómo parámetro msgError: Tipo String, mensaje de error/resultado.

```
308    public void cambiarError(String msgError) {
309        lblError.setText(msgError);
310        lblError.setForeground(new Color(188, 71, 73));
311    }
```

Captura 28 Vista.java - cambiarError()

3.4.5 cambiarMsgResultado(String resultado)

Función la cual asigna color y mensaje a la etiqueta de error. Toma cómo parámetro msgError: Tipo String, mensaje de error/resultado.

```
318    public void cambiarMsgResultado(String resultado) {
319        lblError.setText(resultado);
320        lblError.setForeground(new Color(106, 153, 78));
321    }
```

Captura 29 Vista.java - cambiarMsgResultado

3.4.6 setModelo(Modelo miModelo)

Método el cual asigna las propiedades del objeto Modelo y hace posible la comunicación entre clases. Toma cómo parámetro miModelo: Objeto de la clase Modelo.

```
330    public void setModelo(Modelo miModelo) {
331        this.miModelo = miModelo;
332    }
```

Captura 30 Vista.java - setModelo

3.4.7 setControlador(Controlador miControlador)

Método el cual asigna las propiedades del objeto Controlador y hace posible la comunicación entre clases. Toma cómo parámetro miControlador.

```
340    public void setControlador(Controlador miControlador) {
341        this.miControlador = miControlador;
342    }
343 }
```

Captura 31 Vista.java - setControlador()

Bibliografía (Enlaces consultados)

- Solucionar un error que me daba el PrepareStatement: https://stackoverflow.com/questions/10896151/java-sql-sqlexception-parameter-index-out-of-range-1-number-of-parameters-wh
- Paleta de colores: https://coolors.co/386641-6a994e-a7c957-f2e8cf-bc4749
- Porqué el botón se ponía gris al clicar sobre él y no lo podía cambiar: https://community.oracle.com/tech/developers/discussion/1375618/why-does-jbutton-background-color-change-when-clicking
- Como hacer una tabla no editable: https://stackoverflow.com/questions/1990817/how-to-make-a-jtable-non-editable

Índice de figuras

Índice de tablas

GLOSARIO

CFGS	Ciclo Formativo de Grado Superior
UD	Unidad Didáctica
MVC	Modelo Vista Controlador