

CUANDO DICES PROGRAMACIÓN LO HAS DICHO
TODO

MANUAL TÉCNICO

VIRIDIANA LEAL RAMOS

CANAAN URIEL TÉLLEZ RODRÍGUEZ

AMÉRICA YARELI JIMÉNEZ VILCHES

JADE VIRIDIANA MORALES MARTÍNEZ

INDICE

BASE DE DATOS XAMPP.....	1
CREA UN PROYECTO EN ECLIPCE.....	
CREA UN PACKAGE Y LIBRERÍA.....	
CREA CLASE CONEXIÓN, DAO, MODULO Y VISTA.....	
CREAR JFRAME PANTALLA ALUMNO.....	
CREAR INTERFAZ PANTALLA ALUMNO.....	
VALIDAR LONGITUD DE CAMPOS.....	
METODOS DE GETTERS Y SETTERS.....	
CONEXIÓN A LA BASE DE DATOS.....	
PROGRAMAR BOTONES DE ALUMNO.....	
CASO 1: BOTON DE AGREGAR.....	
CASO 2: BOTON EDITAR.....	
CASO 3: BOTON ELIMINAR.....	
CASO 4: BOTON BORRAR.....	
PROBAR EL PROGRAMA.....	
DAO USUARIO.....	
MODELO USUARIO.....	
INTERFAZ DE USUARIO.....	
PROGRAMAR BOTONES DE USUARIO.....	
CASO 1: BOTON AGREGAR.....	
CASO 2: BOTON ELIMINAR.....	
CASO 3: EDITAR.....	
CASO 4:PDF.....	
DAO SEMESTRE.....	
MODELO SMESTRE.....	
INTERFAZ SEMESTRE.....	
PROGRAMAR BOTONES DE SEMESTRE.....	
CASO 1: BOTON AGREGAR.....	

CASO 2: BOTON DE EDITAR.....
CASO 3: BOTON DE ELIMINAR.....
CASO 4: BOTON DE LIMPIAR.....
CASO 5: BOTON DE PDF.....
DAO PROMEDIO.....
MODELO PROMEDIO.....
INTERFAZ DE PROMEDIO.....
PROGRAMAR LOS BOTONES.....
CASO 1: BOTON AGREGAR.....
CASO 2: BOTON EDITAR.....
CASO 3: BOTON ELIMINAR.....
CASO 4: BOTON PDF.....
DAO PROFESOR.....
MODELO PROFESOR.....
INTERFAZ DE PROFESOR.....
PROGRAMAR LOS BOTONES.....
CASO 1: BOTON AGREGAR.....
CASO 2: BOTON EDITAR.....
CASO 3: BOTON ELIMINAR.....
CASO 4: BOTON PDF.....
CASO 5: BOTON FOTO.....

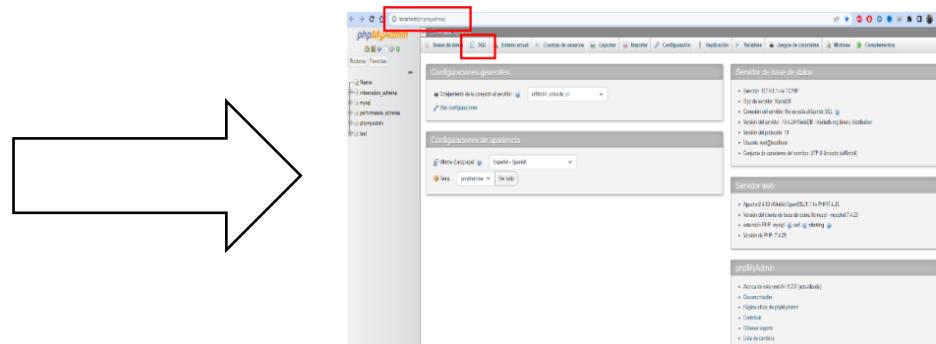
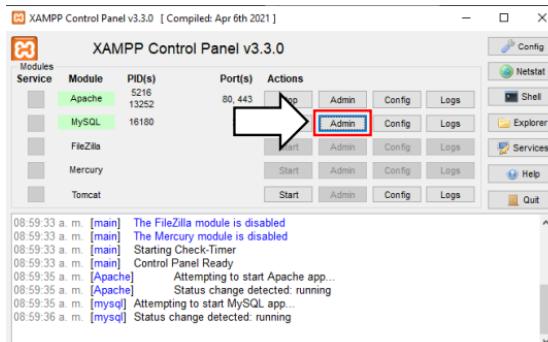
BASE DE DATOS XAMPP

Es necesario abrir el Control Panel y habilitarlos dos servicios Apache y MySQL.



BASE DE DATOS DESDE PHPMYADMIN

Da clic en el botón **admin** del Control Panel de XAMPP para abrir

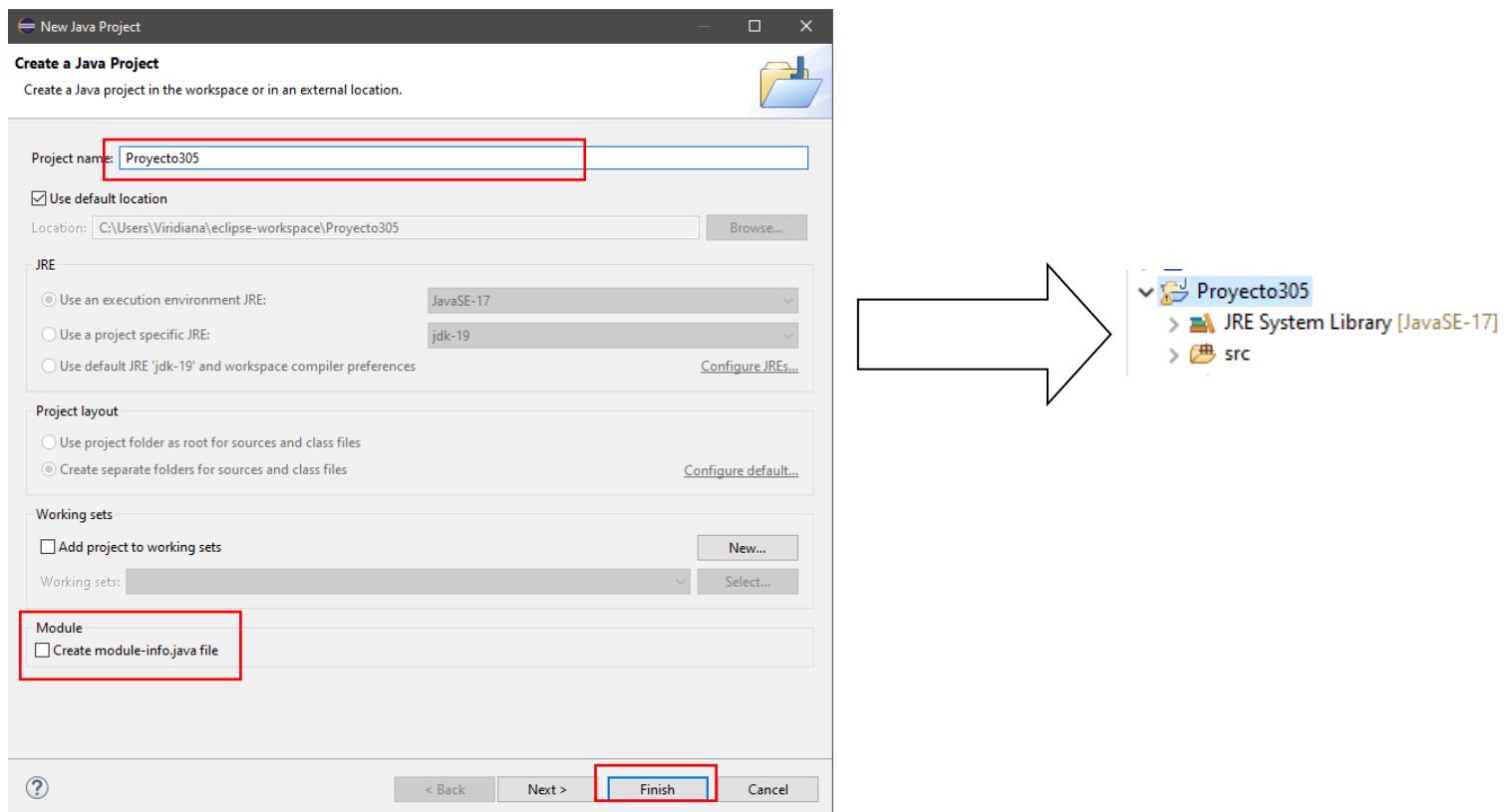


CREA UN PROYECTO EN ECLIPCE

Cuando abres **Eclipce**, seleccionar el **Espacio de trabajo**, que es la carpeta donde se almacenan los proyectos. Da clic en el Menú **File->New->Java Project**

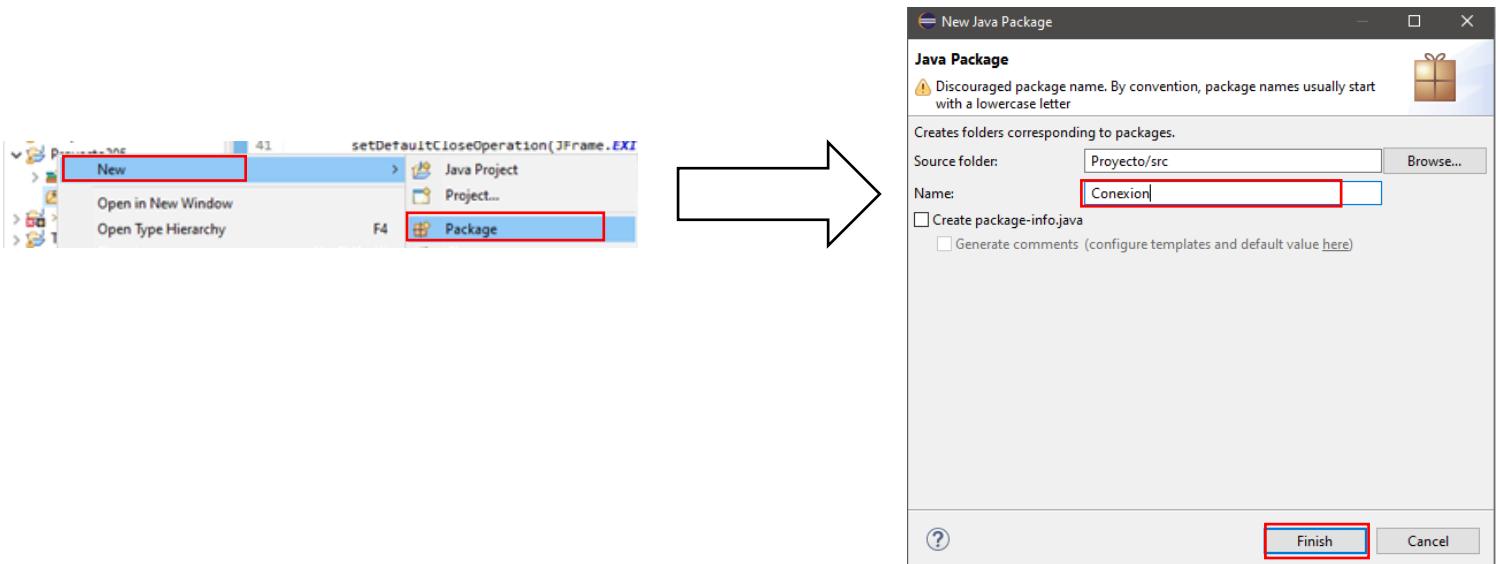


Coloca el nombre de tu proyecto, verifica que no esté seleccionado la opción **Create module-info.java file**, da clic en **Finish** y el proyecto se muestra creado del lado izquierdo.

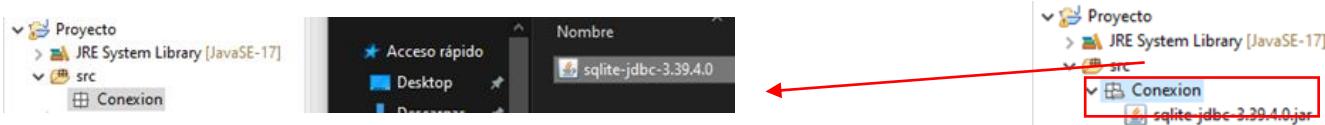


CREA UN PACKAGE Y LIBRERÍA

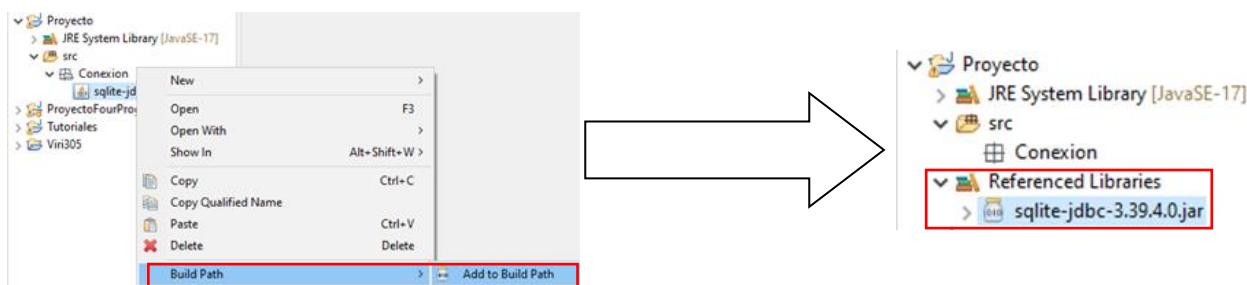
Dentro de la carpeta src dar clic derecho y crear un **package** con un nombre representativo y dar clic en **Finish**.



Ubica el archivo del **Controlador** **sqlite-jdbc-3.39.4.0.jar** dentro de la carpeta y arrastra hasta el **package** anterior creado, verifica que se encuentra dentro del **package**, con la extensión **.jar**.

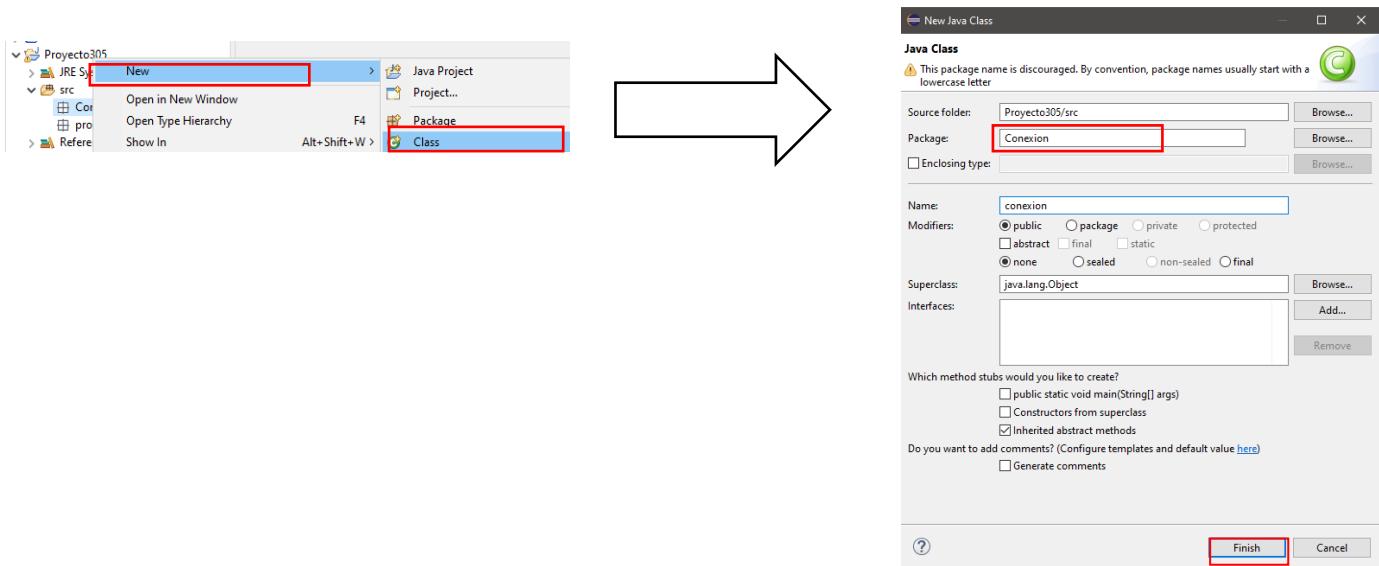


Para agregar el controlador como librería, dar clic derecho sobre el archivo, ir a **Build Path->Add to Build Path**, verificar que el controlador aparece dentro de **Referenced Libraries**.



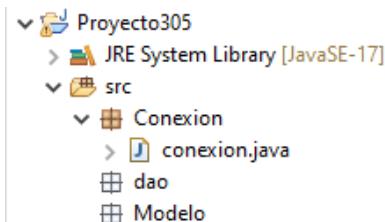
CREAR CLASE CONEXIÓN, DAO, MODULO Y VISTAS

Para crear la clase **Conexión**, da clic derecho sobre el **package**, ir a **New→Class**, escribe el nombre **“conexión”** y da clic en **Finish**.



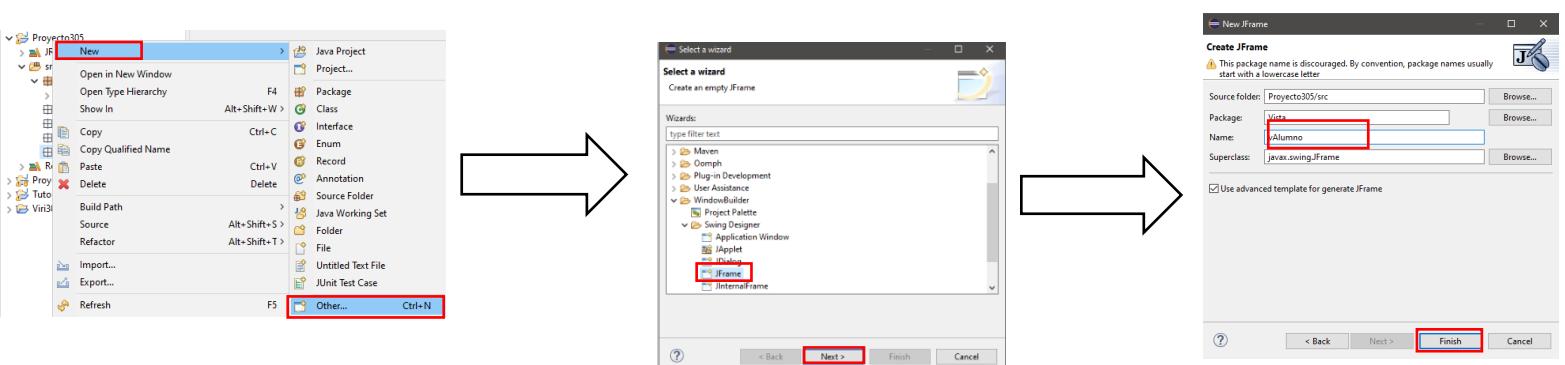
Repetimos el mismo procedimiento del Package de Conexión para hacer un Package de dao, Modulo y Vista.

Se tendría que ver de la siguiente manera:



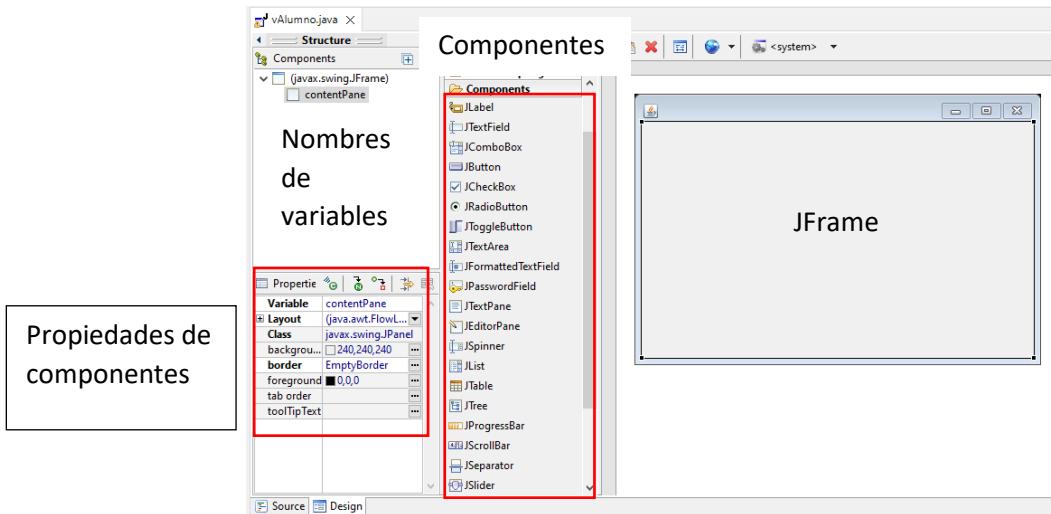
CREAR JFRAME PANTALLA ALUMNO

Dar clic derecho sobre el **Package Vista**, ir a **New→Other**, buscar dentro de **WindowsBuilder**, selecciona **JFrame** da clic en **Next**, escribe el nombre **vAlumno** y da clic en **Finish**.

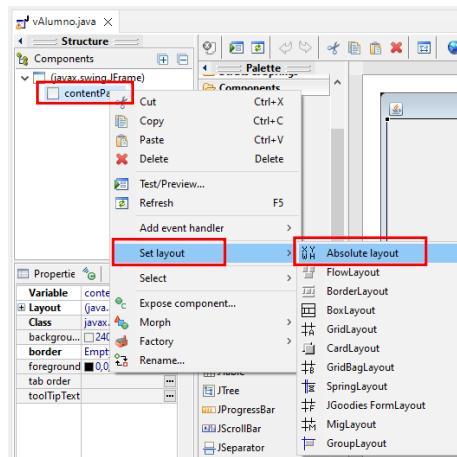


CREA INTERFAZ PANTALLA ALUMNO

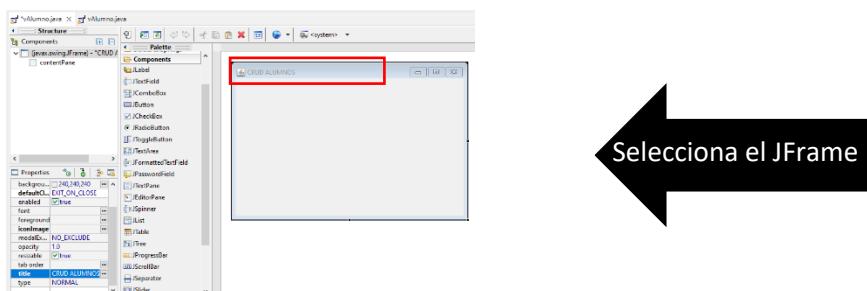
Selecciona y abre el archivo **Alumno**, da clic en la pestaña **Design** para cambiar el entorno de diseño de un **JFrame**.



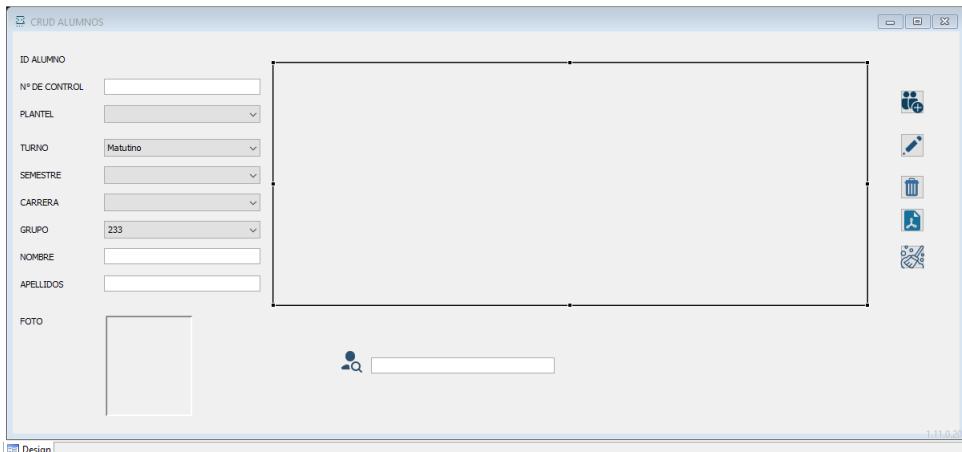
Para cambiar el **Layout** por **AbsolutLayout**, da clic derecho sobre **getContentPane()** → **Set layout** → **Absolute layout**.



Para cambiar el título de la ventana, selecciona el **JFrame** e ir a la propiedad **title**, asignar el texto **CRUD ALUMNOS** y verifica el nombre se allá puesto en el **JFrame**.



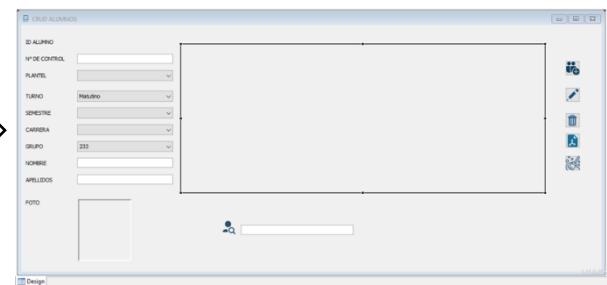
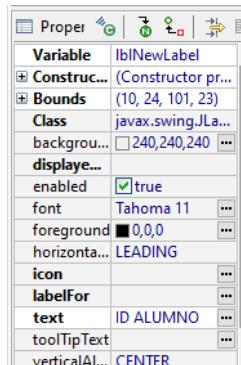
Da clic y arrastra los componentes dentro del **JFrame**, en total 14 **JLabel**, 4 **JTextField**, 5 **JComboBox** y 5 **JButton**, organiza los componentes (ver imagen).



Cambia el nombre de **variable** y **text** en las propiedades de cada componente.

Components

- contentPane
- ↳ lblNewLabel - "ID ALUMNO"
- ↳ lbldNEControl - "N° DE CO"
- ↳ lblPlante - "PLANTEL"
- ↳ lblTurno - "TURNO"
- ↳ lblSemestre - "SEMESTRE"
- ↳ lblCarrera - "CARRERA"
- ↳ lblGrupo - "GRUPO"
- ↳ lblNombre - "NOMBRE"
- ↳ lblApellidos - "APELIDOS"
- ↳ lblAlumno - ""
- ↳ txtNombre
- ↳ txtApellidos
- ↳ cboPlante
- ↳ cboTurno
- ↳ cboSemestre
- ↳ cboCarrera
- ↳ cboGrupo
- scrollPane
- ↳ btnEditar - ""
- ↳ btnEliminar - ""
- ↳ btnPdf - ""
- ↳ lblFoto - ""
- ↳ lblNewLabel_1 - ""
- ↳ lblNewLabel_2 - "FOTO"

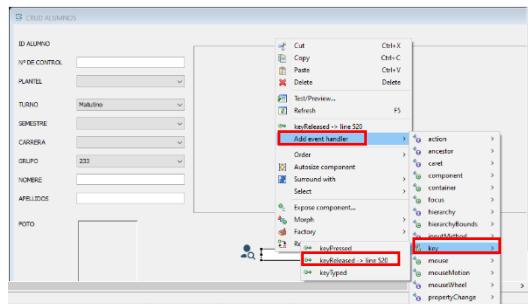


VALIDAR LONGITUD DE CAMPOS

Da clic derecho sobre el campo de texto (JTextField) ir a Add event handler → key → keyReleased

```
txtBuscar = new JTextField();
txtBuscar.addActionListener(new KeyAdapter() {
    @Override
    public void keyReleased(KeyEvent e) {
        refrescarTabla2(txtBuscar.getText().toString());
    }
});
```

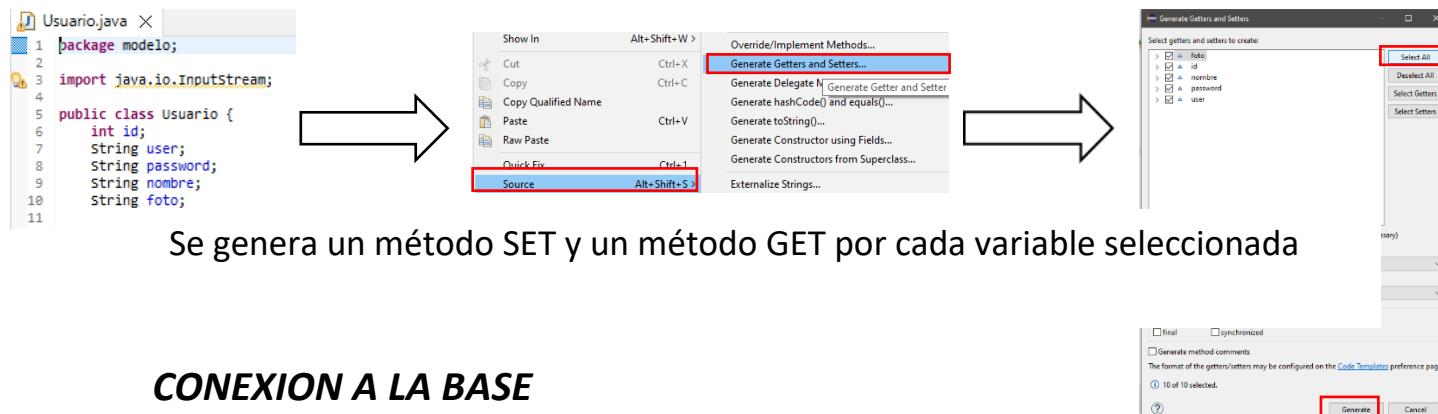
proceso para cada campo de según corresponde el nombre longitud de acuerdo a la estructura.



Repite este texto, y la

METODOS GETTERS AND SETTERS

Crear el Método Getters and Setters se ponen debajo de la última variable declarada.



Se genera un método SET y un método GET por cada variable seleccionada

CONEXION A LA BASE

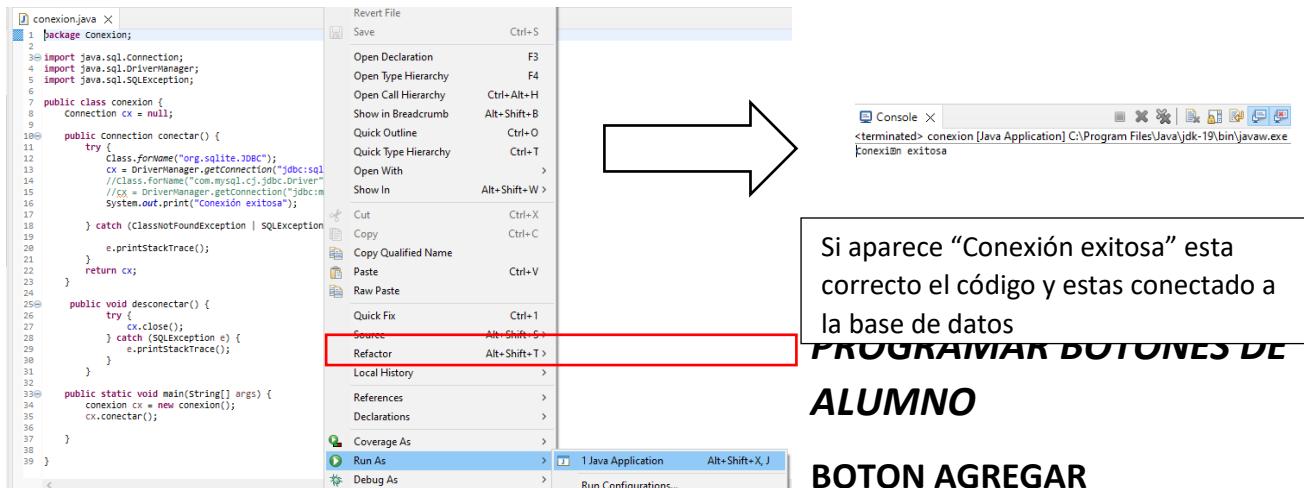
Con la clase Conexión anterior, vamos agregar el siguiente código:

```
package Conexion;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class conexion {
    Connection cx = null;
    public Connection conectar() {
        try {
            Class.forName("org.sqlite.JDBC");
            cx = DriverManager.getConnection("jdbc:sqlite:fourprogramming.db");
            //cx = DriverManager.getConnection("jdbc:mysql://localhost:3306/sistema","root","");
            System.out.print("Conexion exitosa");
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
        }
        return cx;
    }
    public void desconectar() {
        try {
            cx.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
public static void main(String[] args) {
    conexion cx = new conexion();
    cx.conectar();
}
```

Librerías agregadas

Método para conectar a la base de datos

Prueba la conexión a la base de datos, da clic derecho en el código Conexión y selecciona Run As→Java Application.

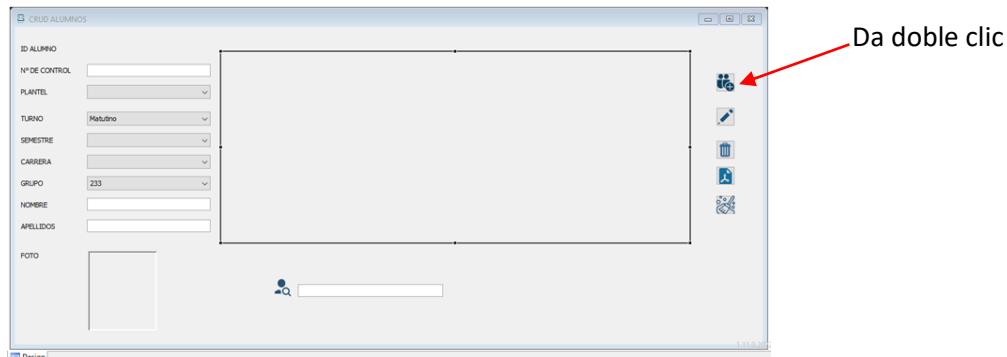


Si aparece "Conexión exitosa" esta correcto el código y estas conectado a la base de datos

PROGRAMAR DUTONES DE ALUMNO

BOTON AGREGAR

Agregar el evento **ActionPerformed** al botón Insertar que se encuentra en **ApplicationWindow->vAlumno**, dando doble clic sobre el botón.



Y así quedaría el código

```

456 btngregar = new JButton("");
457 btngregar.setIcon(new ImageIcon(vAlumno.class.getResource("/img/iconos-añadir-grupo-de-usuarios-hombre-hombre-30.png")));
458 btngregar.addActionListener(new ActionListener() {
459     public void actionPerformed(ActionEvent e) {
460         try {
461             if (txtnControl.getText().equals("") || txtNombre.getText().equals("") ||
462                 || txtApellido.getText().equals("") || cbоТurno.getSelectedItem() == null) {
463                 JOptionPane.showMessageDialog(null, "CAMPOS VACIOS");
464                 return;
465             }
466             Alumno user = new Alumno();
467             user.setnumerocontrol(Integer.parseInt(txtnControl.getText()));
468             user.setplantel(cbоПlantel.getSelectedItem());
469             user.setturno("+" + cbоТurno.getSelectedItem());
470             user.setsemestre("+" + cbоСemestre.getSelectedIndex());
471             user.setcarrera("+" + cbоСarrera.getSelectedIndex());
472             user.setid(Integer.parseInt(cbоАlumno.getSelectedItem().toString()));
473             user.setnombre(txtnNombre.getText());
474             user.setapellidos(txtApellido.getText());
475             user.setfoto(lbfoto.getText());
476             user.setFoto(lbfoto.getText());
477             if (dbo.insertarAlumno(user)) {
478                 ac.addUserToTable();
479                 limpiar();
480                 JOptionPane.showMessageDialog(null, "SE AGREGO CORRECTAMENTE");
481             } else {
482                 JOptionPane.showMessageDialog(null, "ERROR");
483             }
484         } catch (Exception ex) {
485             ex.printStackTrace();
486             JOptionPane.showMessageDialog(null, "ERROR");
487         }
488     }
489 });

```

Debes de verificar que se hallan agregado todas las librerías.

```
vAlumno.java ×

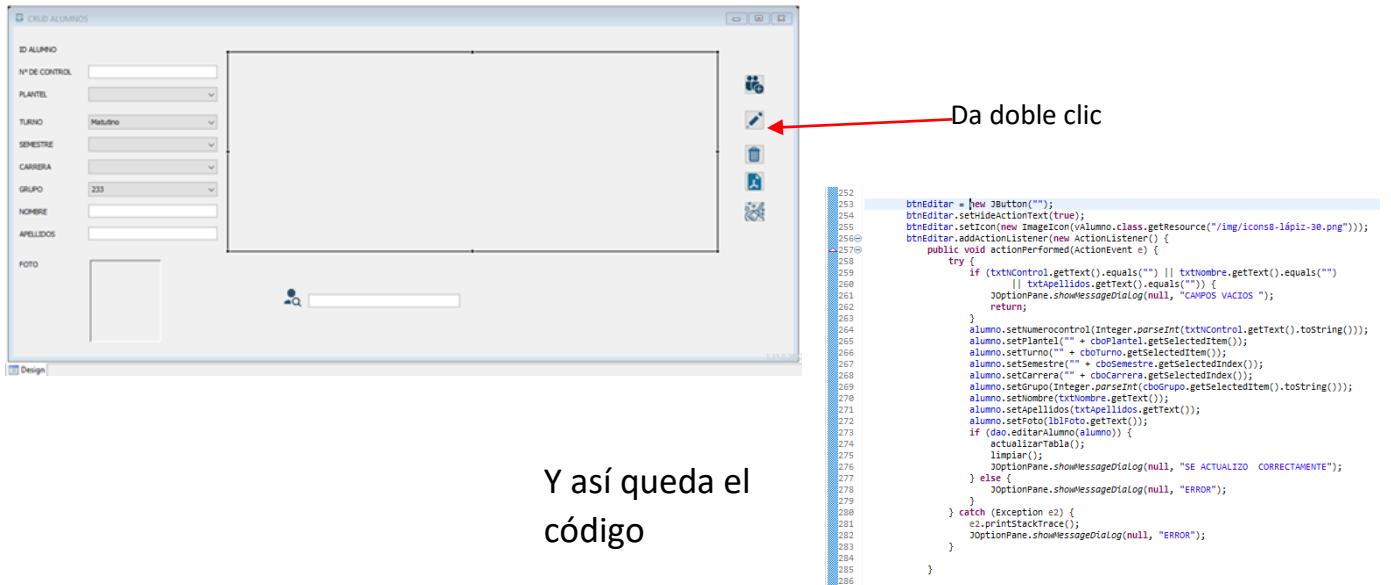
2
3 import java.awt.Desktop;
4 import java.awt.EventQueue;
5
6 import javax.swing.JFrame;
7 import javax.swing.JPanel;
8 import javax.swing.border.EmptyBorder;
9 import javax.swing.JLabel;
10 import javax.swing.JOptionPane;
11 import javax.swing.JTextField;
12 import javax.swing.JComboBox;
13 import javax.swing.JPasswordField;
14 import javax.swing.JButton;
15 import java.awt.Toolkit;
16 import javax.swing.JTable;
17 import javax.swing.table.DefaultTableModel;
18 import javax.swing.text.TableView;
19
20 import com.itextpdf.text.BaseColor;
21 import com.itextpdf.text.Chunk;
22 import com.itextpdf.text.Document;
23 import com.itextpdf.text.DocumentException;
24 import com.itextpdf.text.Element;
25 import com.itextpdf.text.Font;
26 import com.itextpdf.text.Image;
27 import com.itextpdf.text.pdf.PdfFormXObject;
28 import com.itextpdf.text.Phrase;
29 import com.itextpdf.text.log.Logger;
30 import com.itextpdf.text.pdf.AcroFields.Item;
31 import com.itextpdf.text.pdf.PdfCell;
32 import com.itextpdf.text.pdf.PdfTable;
33 import com.itextpdf.text.pdf.PdfWriter;
34 import com.itextpdf.text.pdf.security.TSAClientBouncyCastle;
35
36 import dao.DaoAlumno;
37 import dao.DaoUsuario;
38 import modelo.Alumno;
39
40 import modelo.Usuario;
41
42 import java.awt.event.ActionListener;
43 import java.io.File;
44 import java.io.FileInputStream;
45 import java.io.FileNotFoundException;
46 import java.io.FileOutputStream;
47 import java.io.IOException;
48 import java.net.URL;
49 import java.util.ArrayList;
50 import java.util.List;
51 import java.util.concurrent Phaser;
52 import java.awt.event.ActionEvent;
53 import javax.swing.DefaultComboBoxModel;
54 import java.awt.event.MouseAdapter;
55 import java.awt.event.MouseEvent;
56 import java.awt.event.KeyAdapter;
57 import java.awt.event.KeyEvent;
58 import javax.swing.ImageIcon;
59 import javax.swing.SwingConstantsConstants;
60 import javax.swing.border.EtchedBorder;
61
62 public class vAlumno extends JFrame {
63
64     private JPanel contentPane;
65     private JLabel lblIdAlumno;
66     private JTextField txtIdControl;
67     private JTextField txtNombre;
68     private JTextField txtApellidos;
69     private JComboBox cbxPantel;
70     private JComboBox cbxTurno;
71     private JComboBox cbxSemestre;
```

Agrega el método limpiar para borrar el contenido de los campos del formulario, se agrega antes de la última llave “}” de cierra de la clase Alumno.

```
107  
108     public void limpia() {  
109         lblIdTurno.setText("");  
110         txtNombre.setText("");  
111         cboPlantel.setSelectedItem("");  
112         cboTurno.setSelectedItem("");  
113         cboSemestre.setSelectedItem("");  
114         cboCarrera.setSelectedItem("");  
115         cboGrupo.setSelectedItem("");  
116         txtNombre.setText("");  
117         txtApellidos.setText("");  
118         lblFoto.setText("");  
119     }  
120 }
```

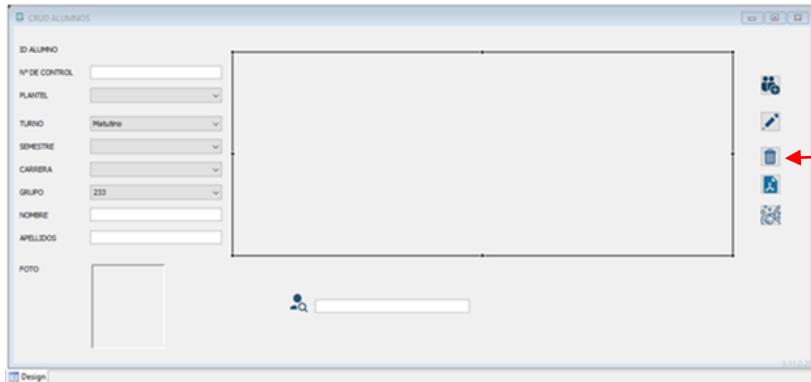
BOTON EDITAR

Agregar el evento **ActionPerformed** al botón Insertar que se encuentra en **ApplicationWindow->vAlumno**, dando doble clic sobre el botón.

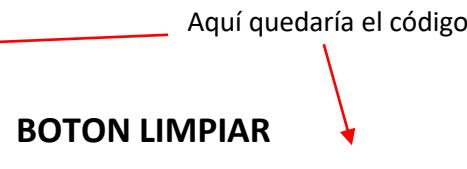


BOTON FLUMINAR

Volvemos hacer el mismo procedimiento que hicimos con Botón agregar y Editar.



Agregar el evento **ActionPerformed** al botón Insertar que se encuentra en **ApplicationWindow->vAlumno**, dando doble clic sobre el botón.



CRUD ALUMNOS

ID ALUMNO	<input type="text"/>
Nº DE CONTROL	<input type="text"/>
PLANTEL	<input type="text"/>
TURNO	<input type="text"/> Mañana
SEMESTRE	<input type="text"/>
CARRERA	<input type="text"/>
GRUPO	<input type="text"/> 233
NOMBRE	<input type="text"/>
APELLIDOS	<input type="text"/>
PORTO	<input type="file"/>
	 <input type="text"/>



 Aquí iría el código

```

btnEditar.setBounds(110, 127, 30, 30);
contentPane.add(btnEditar);

btnEliminar = new JButton("");
btmEliminar.setIcon(new ImageIcon(vAlumno.class.getResource("/img/icons8-eliminar-30.png")));
btmEliminar.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        try {

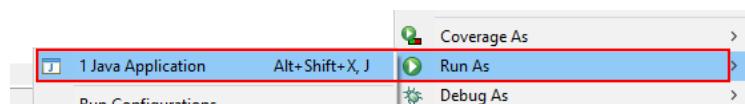
            int opcion = JOptionPane.showConfirmDialog(null, "¿ESTA SEGURO DE ELIMINAR ESTE PRODUCTO?", "ELIMINAR PRODUCTO", JOptionPane.YES_NO_OPTION);
            if (opcion == 0) {
                if (dao.EliminarAlumno(lista.get(fila).getIdAlumno())) {
                    actualizarTabla();
                    limpiar();
                    JOptionPane.showMessageDialog(null, "SE ELIMINO CORRECTAMENTE");
                } else {
                    JOptionPane.showMessageDialog(null, "ERROR");
                }
            }
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(null, "ERROR");
        }
    }
});

//-----AGREGAR AL GRUPO-----//
btngregar = new JButton("");
btngregar.setIcon(new ImageIcon(vAlumno.class.getResource("/img/icon8-agregar-grupo-de-usuarios-hombre-hombre-30.png")));
btngregar.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        try {
            if (txtNombre.getText().equals("") || txtNombre.getText().equals("")){
                if (txtApellido.getText().equals("")) {
                    JOptionPane.showMessageDialog(null, "DEBES SELECCIONAR");
                    return;
                }
                Alumno user = new Alumno();
                user.setNombre(Integer.parseInt(txtNombre.getText().toString()));
                user.setApellido(" " + txtApellido.getText());
                user.setSexo(" " + cbSexo.getSelectedItem());
                user.setEdad(" " + cbEdad.getSelectedItem());
                user.setEstatura(" " + cbEstatura.getSelectedItem());
                user.setGrado(" " + cbGrado.getSelectedItem());
                user.setImagen(" " + cbImagen.getSelectedItem());
                user.setNombre(txtNombre.getText());
                user.setApellido(txtApellido.getText());
                user.setSexo(cbSexo.getSelectedItem());
                user.setEdad(cbEdad.getSelectedItem());
                user.setEstatura(cbEstatura.getSelectedItem());
                user.setGrado(cbGrado.getSelectedItem());
                user.setImagen(cbImagen.getSelectedItem());
                if (dao.InsertarAlumno(user)) {
                    actualizarTabla();
                    limpiar();
                    JOptionPane.showMessageDialog(null, "SE AGREGO CORRECTAMENTE");
                } else {
                    JOptionPane.showMessageDialog(null, "ERROR");
                }
            }
        } catch (Exception ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(null, "ERROR");
        }
    }
});

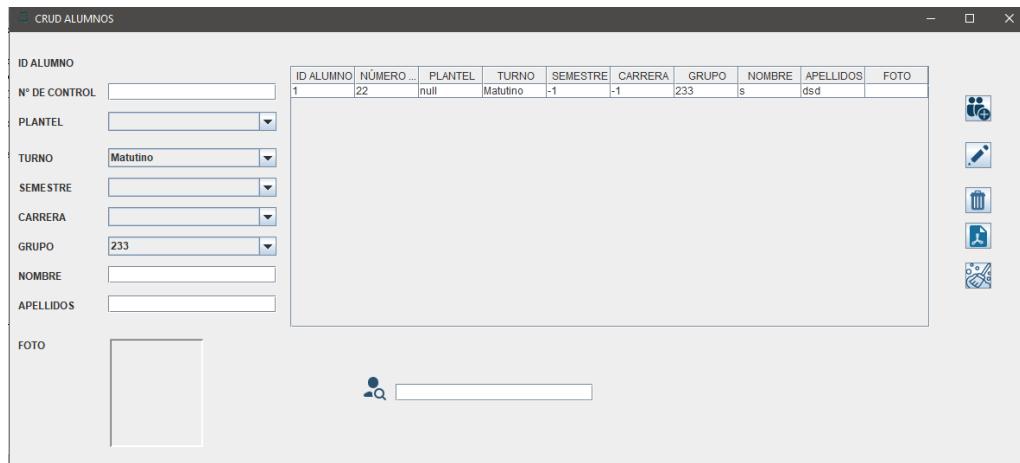
```

PROBAR EL PROGRAMA

Da clic derecho en el código vAlumno y selecciona **Run As → Java Application**.



Debe aparecer así la pantalla



Rellena todos los campos y prueba el Botón Agregar, si ya funciona prueba los otros botones.

Si te marca ERROR regresa al código y chécalo.

DAO USUARIO

En el Package de dao vamos a crear una clase llamada “daoUsuario”, y quedaría así:

```

daoUsuario.java X
1 package dao;
2
3 import java.io.FileInputStream;
4
5 public class daousuario {
6     conexi&on cx = null;
7     FileInputStream fis;
8     int longitudBytes;
9
10    public daousuario() {
11        cx = new conexi&on();
12    }
13
14    public boolean insertarusuario(usuario user) {
15        PreparedStatement ps = null;
16        try {
17            ps = cx.conectar().prepareStatement("INSERT INTO usuario VALUES(null,?,?)");
18            ps.setString(1, user.getuser());
19            ps.setString(2, convertirSHA256(user.getPassword()));
20            ps.executeUpdate();
21            return true;
22        } catch (SQLException e) {
23            e.printStackTrace();
24            return false;
25        }
26    }
27
28    public ArrayList<usuario> buscar(String palabra) {
29        ArrayList<usuario> lista = new ArrayList<usuario>();
30        try {
31            String sql = "SELECT * FROM usuario WHERE "
32            + "(id LIKE ?) OR "
33            + "(user LIKE ?) OR "
34            + "(password LIKE ?) OR "
35            + "(nombre LIKE ?);";
36
37            PreparedStatement ps = cx.conectar().prepareStatement(sql);
38            ps.setString(1, palabra + "%");
39            ps.setString(2, palabra + "%");
40            ps.setString(3, palabra + "%");
41            ps.setString(4, palabra + "%");
42            ps.execute();
43            ResultSet rs = ps.executeQuery();
44            while(rs.next()) {
45                usuario u = new usuario();
46                u.setId(rs.getInt("id"));
47                u.setUser(rs.getString("user"));
48                u.setPassword(rs.getString("password"));
49                u.setNombre(rs.getString("nombre"));
50                lista.add(u);
51            }
52            ps.close();
53            ps = null;
54            cx.desconectar();
55        } catch (SQLException ex) {
56            ex.printStackTrace();
57            System.out.println("Error en BUSCAR");
58        }
59        return lista;
60    }
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165 }

public boolean loginusuario(usuario user) {
    PreparedStatement ps = null;
    ResultSet rs = null;
    try {
        ps = cx.conectar().prepareStatement("SELECT *FROM usuario WHERE user=? AND password=?");
        ps.setString(1, user.getuser());
        ps.setString(2, convertirSHA256(user.getPassword()));
        rs = ps.executeQuery();
        if (rs.next()) {
            return true;
        } else {
            return false;
        }
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    }
}

public boolean Eliminarusuario(int Id) {
    PreparedStatement ps = null;
    try {
        ps = cx.conectar().prepareStatement("DELETE FROM usuario WHERE id=?");
        ps.setInt(1, Id);
        ps.executeUpdate();
        return true;
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    }
}

private String convertirSHA256(String password) {
    MessageDigest md = null;
    try {
        md = MessageDigest.getInstance("SHA-256");
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
        return null;
    }
    byte[] hash = md.digest(password.getBytes());
    StringBuffer sb = new StringBuffer();
    for (byte b : hash) {
        sb.append(String.format("%02x", b));
    }
    return sb.toString();
}

```

MODELO DE USUARIO

Vamos a crear una clase en el Package de Modelo la cuenta se llamará “Usuario”, quedaría algo así:

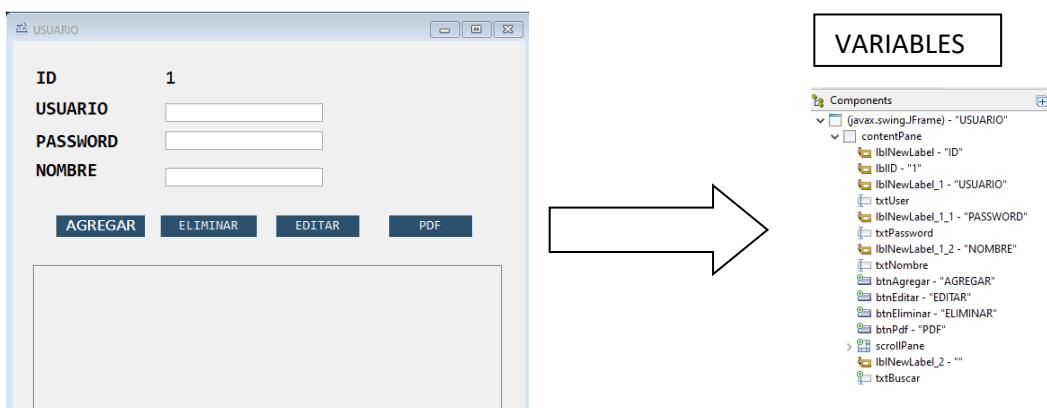
```
1 Usuario.java X
2 package modelo;
3 import java.io.InputStream;
4 public class Usuario {
5     int id;
6     String user;
7     String password;
8     String nombre;
9     String foto;
10}
11 public Usuario() {
12}
13
14    public int getId() {
15        return id;
16    }
17
18    public void setId(int id) {
19        this.id = id;
20    }
21
22    public String getUser() {
23        return user;
24    }
25
26    public void setUser(String user) {
27        this.user = user;
28    }
29
30    public String getPassword() {
31        return password;
32    }
33
34    public void setPassword(String password) {
35        this.password = password;
36    }
37
38    public String getNombre() {
39
40    }
41
42    public String getNombre() {
43
44    }
45    public void setNombre(String nombre) {
46        this.nombre = nombre;
47    }
48    public String getFoto() {
49        return foto;
50    }
51
52    public void setFoto(String foto) {
53        this.foto = foto;
54    }
55
56}
57
```

INTERFAZ DE USUARIO

Vamos a crear un JFrame en el Package de Vistas el JFrame y se llamará “vUsuario”.

Utilizaremos 6 JLabel, 4 JTextField, 4 JButton, un scrollPane, dentro del scrollPane vamos agregar un JTable.

Ya con los nombres se vería algo así:



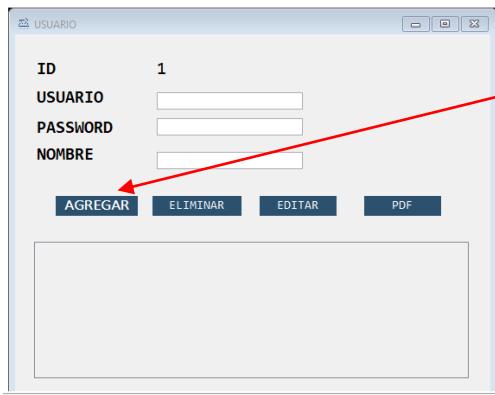
PROGRAMAR BOTONES DE SUSUARIO

BOTON AGREGAR

Agregar el evento **ActionPerformed** al botón Insertar que se encuentra en **ApplicationWindow->vUSUARIO**, dando doble clic sobre el botón.

Doble clic aquí en el botón

El código que debe ir



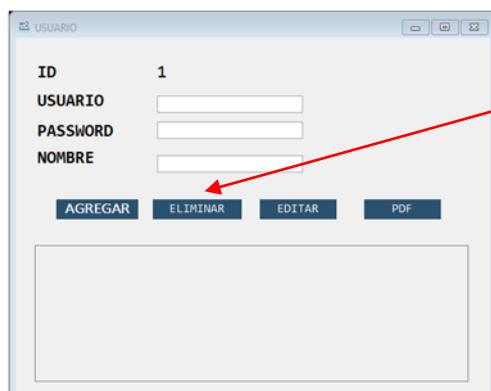
```
143 btnAgregar = new JButton("AGREGAR");
144 btnAgregar.setVerticalAlignment(SwingConstants.TOP);
145 btnAgregar.setForeground(Color.WHITE);
146 btnAgregar.setBorder(null);
147 btnAgregar.setRolloverPainted(false);
148 btnAgregar.setFont(new Font("Times New Roman", Font.PLAIN, 14));
149 btnAgregar.setBackground(new Color(49, 81, 111));
150 btnAgregar.addActionListener(new ActionListener() {
151     public void actionPerformed(ActionEvent e) {
152         try {
153             if (txtUser.getText().equals("") || txtPassword.getText().equals(""))
154                 JOptionPane.showInputDialog(null, "CAMPOS VACIOS ");
155             else {
156                 User user = new User();
157                 user.setUser(txtnombre.getText());
158                 user.setPassword(txtpassword.getText());
159                 user.setNombre(txtnombre.getText());
160                 if (dao.insertarUsuario(user)) {
161                     actualizarTabla();
162                     limpiar();
163                     JOptionPane.showMessageDialog(null, "SE AGREGO CORRECTAMENTE");
164                 } else {
165                     JOptionPane.showMessageDialog(null, "ERROR");
166                 }
167             }
168         } catch (Exception ex) {
169             JOptionPane.showMessageDialog(null, "ERROR");
170         }
171     }
172 });
173 }
```

BOTON ELIMINAR

Volvemos hacer el mismo procedimiento que hicimos con Botón Agregar.

Doble clic aquí en el botón

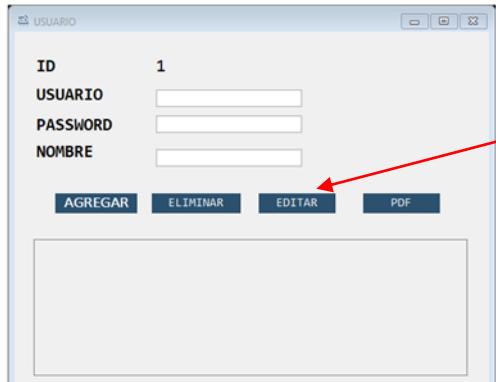
El código que debe ir



```
214 btndelete = new JButton("ELIMINAR");
215 btndelete.setVerticalAlignment(SwingConstants.BOTTOM);
216 btndelete.setForeground(Color.WHITE);
217 btndelete.setBorder(null);
218 btndelete.setRolloverPainted(false);
219 btndelete.setFont(new Font("Times New Roman", Font.PLAIN, 14));
220 btndelete.setBackground(new Color(49, 81, 111));
221 btndelete.addActionListener(new ActionListener() {
222     public void actionPerformed(ActionEvent e) {
223         try {
224             int opcion = JOptionPane.showConfirmDialog(null, "ESTA SEGURO DE ELIMINAR ESTE USUARIO?", "ELIMINAR USUARIO", JOptionPane.YES_NO_OPTION);
225             if (opcion == 0) {
226                 if (dao.eliminarUsuario(lista.get(file).getId())) {
227                     actualizarTabla();
228                     limpiar();
229                     JOptionPane.showMessageDialog(null, "SE ELIMINO CORRECTAMENTE");
230                 } else {
231                     JOptionPane.showMessageDialog(null, "ERROR");
232                 }
233             }
234         } catch (Exception ex) {
235             JOptionPane.showMessageDialog(null, "ERROR");
236         }
237     }
238 });
239 }
```

BOTON EDITAR

Vamos a dar doble clic en el botón Editar y vamos a repetir el mismo procedimiento que hicimos el Botón Agregar.



Doble clic aquí

El código que debe ir aquí

```

btnEditar = new JButton("EDITAR");
btnEditar.setVerticalAlignment(SwingConstants.BOTTOM);
btnEditar.setForeground(color.WHITE);
btnEditar.setBorder(null);
btnEditar.setBorderPainted(false);
btnEditar.setFont(new Font("Consolas", Font.PLAIN, 14));
btnEditar.setBackground(new Color(43, 81, 111));
btnEditar.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        try {
            if (txtUser.getText().equals("") || txtPassword.getText().equals("") ||
                || txtNombre.getText().equals(""))
                JOptionPane.showMessageDialog(null, "CAMPOS VACIOS ");
            return;
        }
        usuario.setUser(txtUser.getText());
        usuario.setPassword(txtPassword.getText());
        usuario.setNombre(txtNombre.getText());
        if (dao.editarUsuario(usuario)) {
            actualizarTabla();
            limpiar();
            JOptionPane.showMessageDialog(null, "SE ACTUALIZO CORRECTAMENTE");
        } else {
            JOptionPane.showMessageDialog(null, "ERROR");
        }
    } catch (Exception e2) {
        JOptionPane.showMessageDialog(null, "ERROR");
    }
}
});
```

BOTON PDF

Tienes que dar doble clic en el botón PDF y vamos a repetir el mismo procedimiento que hemos hecho con los botones anteriores.

```
 245     btmrf = new JButton("Frm");
246     btmrf.setVerticalAlignment(SwingConstants.BOTTOM);
247     btmrf.setForeground(Color.WHITE);
248     btmrf.setBorder(null);
249     btmrf.setRolloverPainted(false);
250     btmrf.setFont(new Font("Calibri Light", Font.PLAIN, 14));
251     btmrf.setBackground(new Color(43, 81, 111));
252     btmrf.addActionListener(new ActionListener() {
253         public void actionPerformed(ActionEvent e) {
254             try {
255                 c1.setHorizontalAlignment(Element.ALIGN_CENTER);
256                 c2.setHorizontalAlignment(Element.ALIGN_CENTER);
257                 c3.setHorizontalAlignment(Element.ALIGN_CENTER);
258                 c4.setHorizontalAlignment(Element.ALIGN_CENTER);
259                 c1.setBackgroundColor(BaseColor.GRAY);
260                 c2.setBackgroundColor(BaseColor.LIGHT_GRAY);
261                 c3.setBackgroundColor(BaseColor.LIGHT_GRAY);
262                 c4.setBackgroundColor(BaseColor.LIGHT_GRAY);
263                 tabla.addCell(c1);
264                 tabla.addCell(c2);
265                 tabla.addCell(c3);
266                 tabla.addCell(c4);
267 
268                 for (Usuario u : lista) {
269                     tabla.addCell(" " + u.getId());
270                     tabla.addCell(u.getUsuario());
271                     tabla.addCell(u.getPassword());
272                     tabla.addCell(u.getNombre());
273 
274                 }
275 
276                 doc.add(tabla);
277                 Paragraph p1 = new Paragraph(10);
278                 p1.add(chunk.NEWLINE);
279                 p1.add("NÚMERO DE REGISTRO " + lista.size());
280                 p1.add(chunk.NEWLINE);
281                 p1.setAlignment(Element.ALIGN_RIGHT);
282                 doc.add(p1);
283                 doc.close();
284                 archivo.close();
285                 Desktop.getDesktop().open(file);
286             } catch (FileNotFoundException e1) {
287                 JOptionPane.showMessageDialog(null, "ERROR AL CREAR ARCHIVO");
288             } catch (DocumentException e1) {
289                 JOptionPane.showMessageDialog(null, "ERROR AL CREAR DOCUMENTO PDF");
290             } catch (IOException e1) {
291                 JOptionPane.showMessageDialog(null, "ERROR AL CREAR IO");
292             }
293         }
294     });
295 }
```

```
328         JOptionPane.showMessageDialog(null, "ERROR AL CREAR ID");
329     } catch (URISyntaxException e) {
330         // TODO Auto-generated catch block
331         e.printStackTrace();
332     }
333 }
334 });
335 btmPf.setBorder(new BevelBorder(BevelBorder.RAISED, null, null, null));
336 btmPf.setFont(new Font("Consolas", Font.PLAIN, 14));
337 btmPf.setBounds(484, 187, 89, 23);
338 contentPane.add(btmPf);
339 scrollPane = new JScrollPane();
340 contentPane.add(scrollPane);
341 scrollPane.addMouseListener(new MouseAdapter() {
342     @Override
343     public void mouseClicked(MouseEvent e) {
344     }
345 });
346 scrollPane.setBounds(22, 240, 503, 158);
347 contentPane.add(scrollPane);
348 tbliusarios = new JTable();
349 tbliusarios.addMouseListener(new MouseAdapter() {
350     @Override
351     public void mouseClicked(MouseEvent e) {
352         fila = tbliusarios.getSelectedRow();
353         filas = tbliusarios.getSelectedRow();
354         usuario = lista.get(fila);
355         lId.setText(usuario.getId());
356         txtUser.setText(usuario.getUsuario());
357         txtPassword.setText(usuario.getPassword());
358         txthombre.setText(usuario.getNombre());
359     }
360 });
361 });
362 }
```

DAO SEMESTRE

En el Package de dao vamos a crear una clase llamada “daoSemestre” y quedaría así:

```
vSemestre.java  daoSemestre.java  Semestre.java
1 package dao;
2
3 import java.sql.PreparedStatement;
4
5 public class daoSemestre {
6     conexion cx = null;
7     public daoSemestre() {
8         cx = new conexion();
9     }
10
11    public boolean insertarSemestre(Semestre user) {
12        PreparedStatement ps = null;
13        try {
14            ps = cx.conectar().prepareStatement("INSERT INTO semestre VALUES(null,?)");
15            ps.setString(1, user.getSemestre());
16            ps.executeUpdate();
17            return true;
18        } catch (SQLException e) {
19            e.printStackTrace();
20            return false;
21        }
22    }
23
24
25
26    public ArrayList<Semestre> fetchSemestres() {
27        ArrayList<Semestre> lista = new ArrayList<Semestre>();
28        PreparedStatement ps = null;
29        ResultSet rs = null;
30        try {
31            ps = cx.conectar().prepareStatement("SELECT *FROM semestre");
32            rs = ps.executeQuery();
33            while (rs.next()) {
34                Semestre u = new Semestre();
35                u.setIdSemestre(rs.getInt("idsemestre"));
36                u.setSemestre(rs.getString("semestre"));
37                lista.add(u);
38            }
39        } catch (SQLException e) {
40            // TODO Auto-generated catch block
41            e.printStackTrace();
42        }
43    }
44
45
46 }
```

```
46
47     }
48     return lista;
49 }
50
51
52
53
54    public boolean eliminarSemestre(int id) {
55        PreparedStatement ps = null;
56        try {
57            ps = cx.conectar().prepareStatement("DELETE FROM semestre WHERE idsemestre=?");
58            ps.setInt(1, id);
59            ps.executeUpdate();
60            return true;
61        } catch (SQLException e) {
62            e.printStackTrace();
63            return false;
64        }
65    }
66
67
68    public boolean editarSemestre(Semestre user) {
69        PreparedStatement ps = null;
70        try {
71            ps = cx.conectar().prepareStatement("UPDATE semestre SET semestre=? WHERE idSemestre=?");
72            ps.setString(1, user.getSemestre());
73            ps.setInt(2, user.getIdSemestre());
74            ps.executeUpdate();
75            return true;
76        } catch (SQLException e) {
77            e.printStackTrace();
78            return false;
79        }
80    }
81
82
83
84
85 }
```

MODELO SEMESTRE

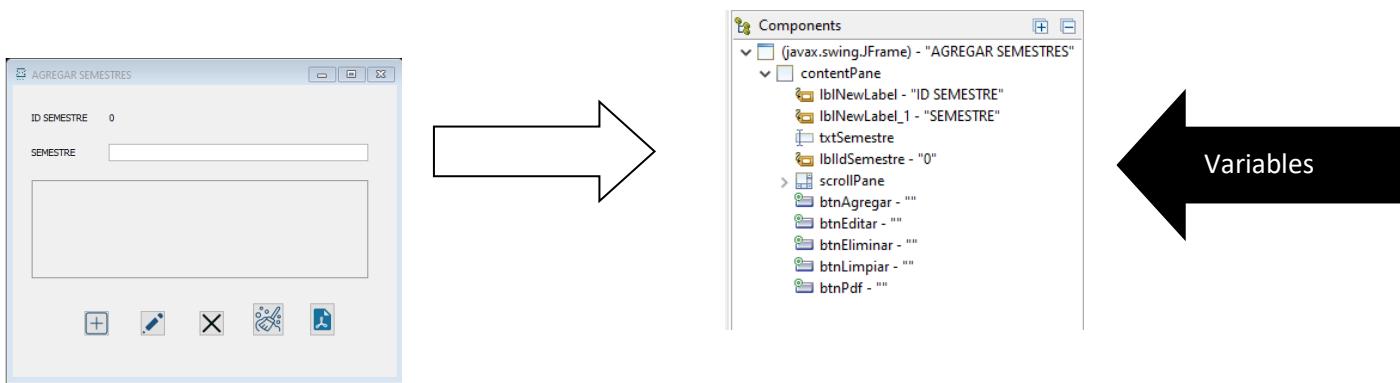
Vamos a crear una clase en el Package de Modelo la cuenta se llamará “Semestre” quedaría algo así:

```
vSemestre.java  Semestre.java
1 package modelo;
2
3 public class Semestre {
4     int idsemestre;
5     String semestre;
6
7     public Semestre() {
8
9     }
10
11    public int getIdSemestre() {
12        return idsemestre;
13    }
14
15    public void setIdSemestre(int idsemestre) {
16        this.idsemestre = idsemestre;
17    }
18
19    public String getSemestre() {
20        return semestre;
21    }
22
23    public void setSemestre(String semestre) {
24        this.semestre = semestre;
25    }
26
27 }
28
```

INTERFAZ DE SEMESTRE

Vamos a crear un JFrame en el Package de Vistas el JFrame y se llamará "vSemestre".

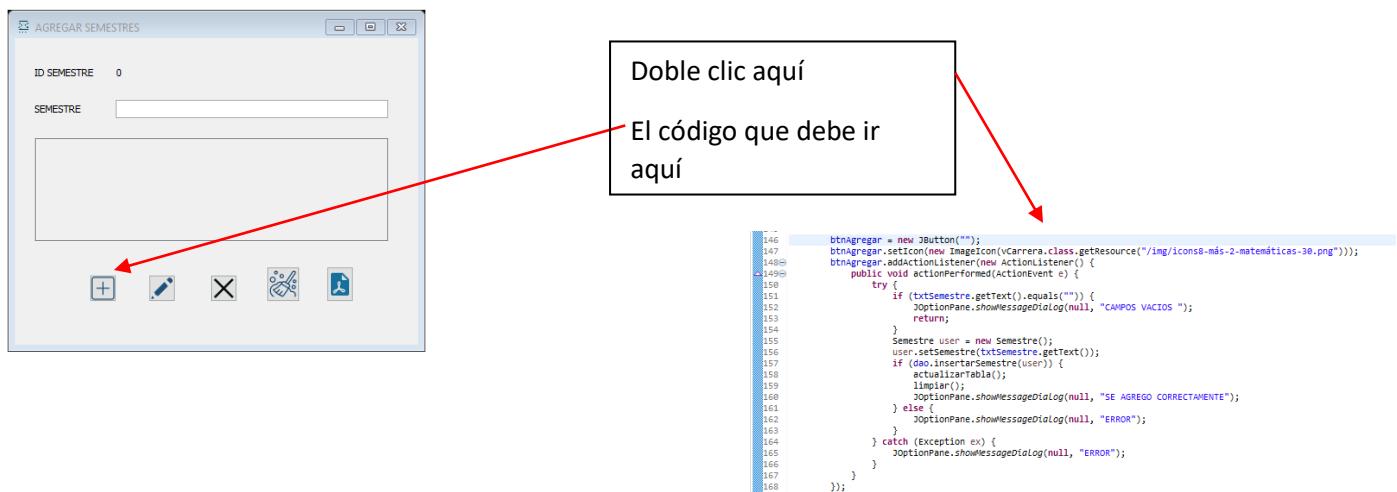
Utilizaremos 3 JLabel, 1 JTextField, 5 JButton, un scrollPane, dentro del scrollPane vamos a agregar un JTable.



PROGRAMAR LOS BOTONES

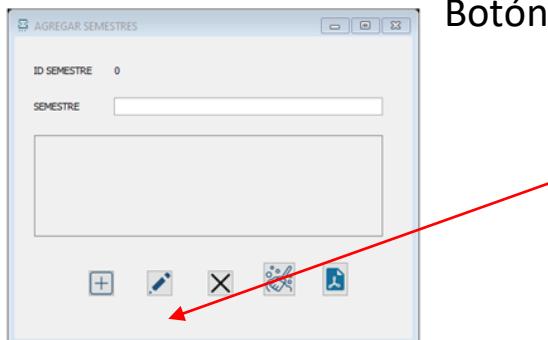
BOTON DE AGREGAR

Agregar el evento **ActionPerformed** al botón Insertar que se encuentra en **ApplicationWindow->vUSUARIO**, dando doble clic sobre el botón.



BOTON EDITAR

Vamos a dar doble clic en el botón Editar y vamos a repetir el mismo procedimiento que



Botón

Doble clicaquí
El código que debe ir
aquí

hicimos el
Aregar.

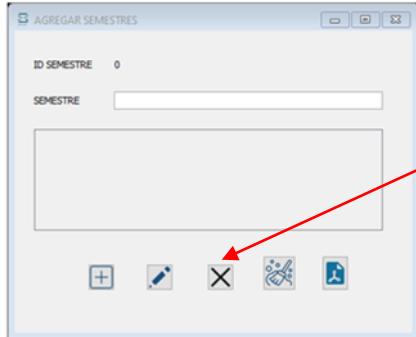
```

172 btnEditar = new JButton("");
173 btnEditar.addActionListener(new ActionListener() {
174     public void actionPerformed(ActionEvent e) {
175         try {
176             if (txtSemestre.getText().equals("")) {
177                 JOptionPane.showMessageDialog(null, "CAMPOS VACIOS ");
178                 return;
179             }
180             txtSemestre.setSemestre(txtSemestre.getText());
181             if (do_editarSemestre(semestre)) {
182                 actualizarTabla();
183                 limpiar();
184                 JOptionPane.showMessageDialog(null, "SE ACTUALIZO CORRECTAMENTE");
185             } else {
186                 JOptionPane.showMessageDialog(null, "ERROR");
187             }
188         } catch (Exception e2) {
189             JOptionPane.showMessageDialog(null, "ERROR");
190         }
191     }
192 });

```

BOTON ELIMINAR

Vamos a dar doble clic en el botón Eliminar y vamos a repetir el mismo procedimiento que hicimos el Botón Agregar y Botón Editar.



Doble clicaquí
El código que debe ir
aquí

```

169 btnAgregar.setBounds(82, 260, 30, 30);
170 contentPane.add(btnAgregar);
171
172 btnEditar = new JButton("");
173 btnEditar.addActionListener(new ActionListener() {
174     public void actionPerformed(ActionEvent e) {
175         try {
176             if (txtSemestre.getText().equals("")) {
177                 JOptionPane.showMessageDialog(null, "CAMPOS VACIOS ");
178                 return;
179             }
180             semestre.setSemestre(txtSemestre.getText());
181             if (do_editarSemestre(semestre)) {
182                 actualizarTabla();
183                 limpiar();
184                 JOptionPane.showMessageDialog(null, "SE ACTUALIZO CORRECTAMENTE");
185             } else {
186                 JOptionPane.showMessageDialog(null, "ERROR");
187             }
188         } catch (Exception e2) {
189             JOptionPane.showMessageDialog(null, "ERROR");
190         }
191     }
192 });

```

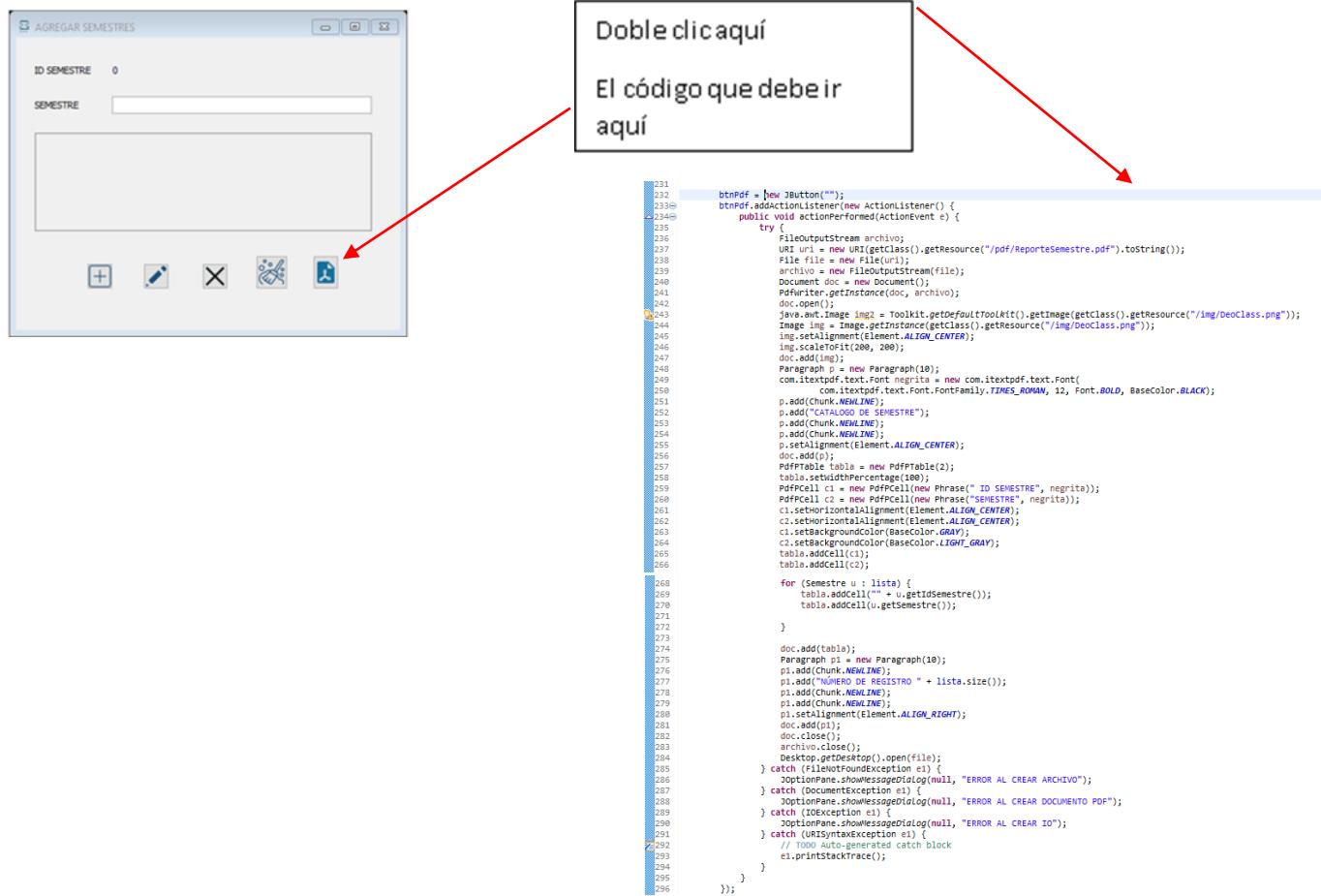
BOTON DE LIMPIAR

Agregar el evento **ActionPerformed** al botón Insertar que se encuentra en **ApplicationWindow->vSemestre**, dando doble clic sobre el botón.



BOTON DE PDF

Tienes que dar doble clic en el botón PDF y vamos a repetir el mismo procedimiento que hemos hecho con los botones anteriores.



DAO PROMEDIO

En el Package de dao vamos a crear una clase llamada “daoPromedio” y quedaría así:

MODELO DE PROMEDIO

Vamos a crear una clase en el Package de Modelo la cuenta se llamará “Promedio” quedaría algo así:

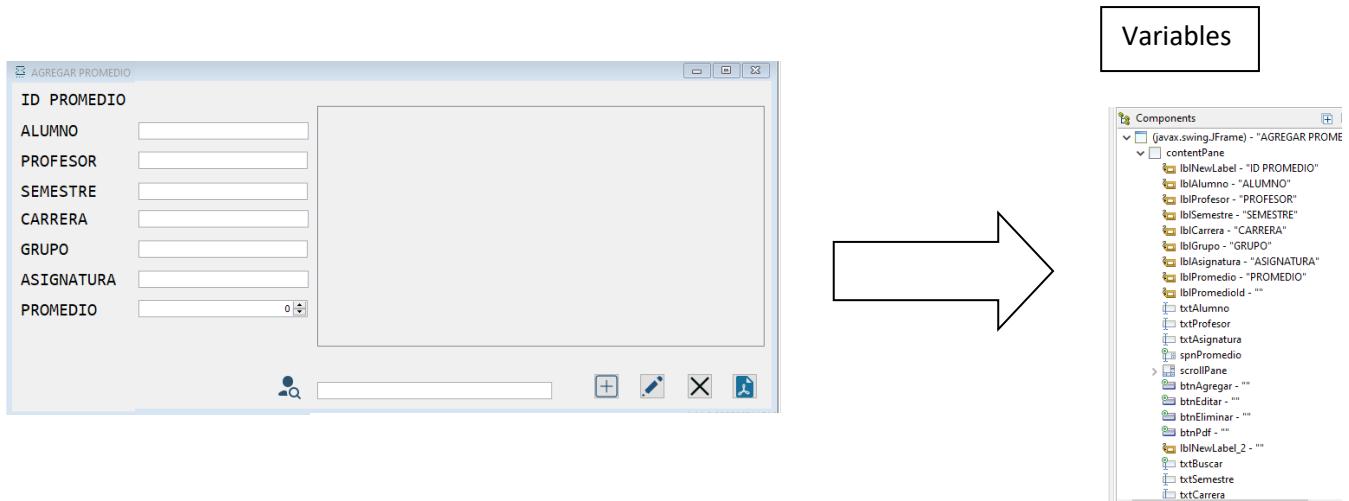
```
ProyectoJava X vPromedio.java

1 package modelo;
2
3 public class Promedio {
4     int idPromedio;
5     String alumno;
6     String profesor;
7     String semestre;
8     String carrera;
9     int grupo;
10    String asignaturas;
11    Double promedio;
12
13
14    public Promedio() {
15
16    }
17
18
19    public int getIdPromedio() {
20        return idPromedio;
21    }
22
23
24    public void setIdPromedio(int idPromedio) {
25        this.idPromedio = idPromedio;
26    }
27
28
29    public String getAlumno() {
30        return alumno;
31    }
32
33
34    public void setAlumno(String alumno) {
35        this.alumno = alumno;
36    }
37
38
39    public String getProfesor() {
40        return profesor;
41
42
43
44    public void setProfesor(String profesor) {
45        this.profesor = profesor;
46    }
47
48
49    public String getSemestre() {
50        return semestre;
51    }
52
53
54    public void setSemestre(String semestre) {
55        this.semestre = semestre;
56    }
57
58
59    public String getCarrera() {
60        return carrera;
61    }
62
63
64    public void setCarrera(String carrera) {
65        this.carrera = carrera;
66    }
67
68
69    public int getGrupo() {
70        return grupo;
71    }
72
73
74    public void setGrupo(int grupo) {
75        this.grupo = grupo;
76    }
77
78
79    public String.getAsignaturas() {
80        return asignaturas;
81
82
83
84    public void setAsignaturas(String asignaturas) {
85        this.asignaturas = asignaturas;
86    }
87
88
89    public Double getPromedio() {
90        return promedio;
91    }
92
93
94    public void setPromedio(Double promedio) {
95        this.promedio = promedio;
96    }
97
98
99 }
```

INTERFAZ DE PROMEDIO

Vamos a crear un JFrame en el Package de Vistas el JFrame y se llamará "vPromedio".

Utilizaremos 10 JLabel, 6 JTextField, 4 JButton, un JSpinner, un scrollPane, dentro del scrollPane vamos a agregar un JTable.



PROGRAMAR LOS BOTONES

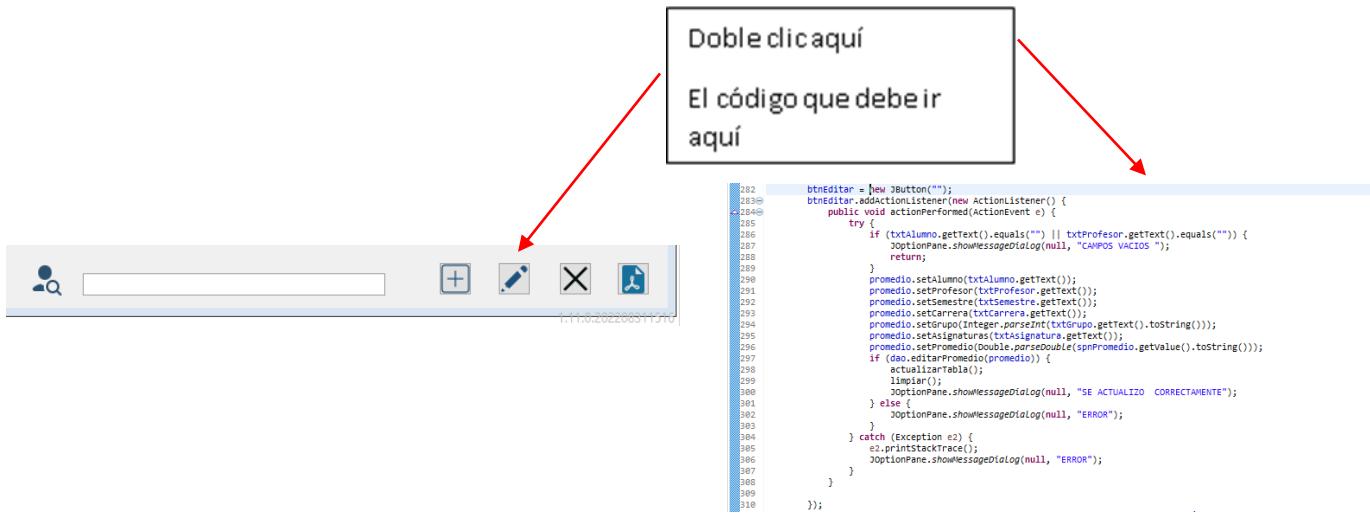
BOTON AGREGAR

Agregar el evento **ActionPerformed** al botón Insertar que se encuentra en **ApplicationWindow->vPromedio**, dando doble clic sobre el botón.



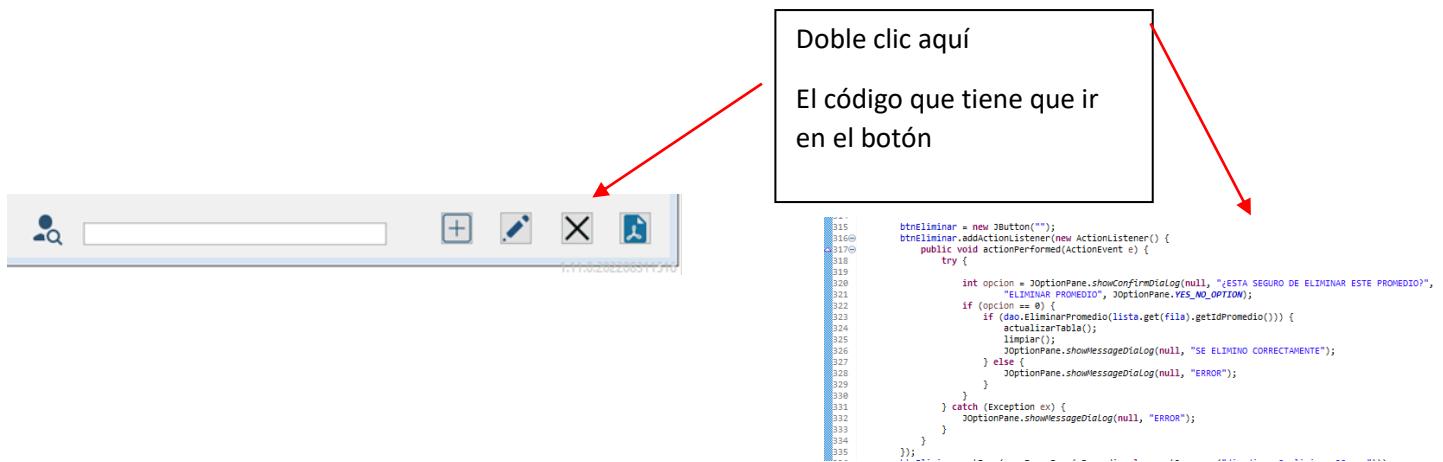
BOTON EDITAR

Vamos a dar doble clic en el botón Editar y vamos a repetir el mismo procedimiento que hicimos el Botón Agregar.



BOTON ELIMINAR

Vamos a dar doble clic en el botón Editar y vamos a repetir el mismo procedimiento que hicimos el Botón Agregar.



BOTON PDF

Tienes que dar doble clic en el botón PDF y vamos a repetir el mismo procedimiento que hemos hecho con los botones anteriores.



Doble clic aquí

El código que tiene que ir en el botón

```
c3.setHorizontalAlignment(Element.ALIGN_CENTER);
c4.setHorizontalAlignment(Element.ALIGN_CENTER);
c5.setHorizontalAlignment(Element.ALIGN_CENTER);
c6.setHorizontalAlignment(Element.ALIGN_CENTER);
c7.setHorizontalAlignment(Element.ALIGN_CENTER);
c8.setHorizontalAlignment(Element.ALIGN_CENTER);
c1.setBackGroundColor(BaseColor.GRAY);
c2.setBackGroundColor(BaseColor.LIGHT_GRAY);
c3.setBackGroundColor(BaseColor.LIGHT_GRAY);
c4.setBackGroundColor(BaseColor.LIGHT_GRAY);
c5.setBackGroundColor(BaseColor.LIGHT_GRAY);
c6.setBackGroundColor(BaseColor.LIGHT_GRAY);
c7.setBackGroundColor(BaseColor.LIGHT_GRAY);
c8.setBackGroundColor(BaseColor.LIGHT_GRAY);
tabla.addCell(c1);
tabla.addCell(c2);
tabla.addCell(c3);
tabla.addCell(c4);
tabla.addCell(c5);
tabla.addCell(c6);
tabla.addCell(c7);
tabla.addCell(c8);

for (Promedio u : lista) {
    tabla.addCell(" "+u.getNombre());
    tabla.addCell(" "+u.getApellido());
    tabla.addCell(u.getProfesor());
    tabla.addCell(u.getSemestre());
    tabla.addCell(u.getCarerra());
    tabla.addCell(" "+u.getGrupo());
    tabla.addCell(u.getAsignaturas());
    tabla.addCell(" "+u.getPromedio());
}

doc.add(tabla);

public void visualizarTabla() {
    doc = new Document();
    while (modelo.getRowCount() > 0) {
        modelo.removeRow();
    }
    doc.setPageSize(new PageSize());
    lista = doc.createTable();
    for (Promedio u : lista) {
        Object[] o = new Object[8];
        o[0] = u.getPromedio();
        o[1] = u.getAlumno();
        o[2] = u.getApellido();
        o[3] = u.getProfesor();
        o[4] = u.getCarerra();
        o[5] = u.getGrupo();
        o[6] = u.getAsignaturas();
        o[7] = u.getPromedio();
        modelo.addRow(o);
    }
    tbIPromedios.setModel(modelo);
}
}

Paragraph p1 = new Paragraph(10);
p1.add(Chunk.NEWLINE);
p1.add("NOMBRE DE REGISTRO " + lista.size());
p1.add(Chunk.NEWLINE);
p1.add(Chunk.NEWLINE);
p1.setAlignment(Element.ALIGN_RIGHT);
doc.add(p1);
doc.close();
archive.close();
Desktop.getDesktop().open(file);
} catch (FileNotFoundException e1) {
    JOptionPane.showMessageDialog(null, "ERROR AL CREAR ARCHIVO");
} catch (IOException e1) {
    JOptionPane.showMessageDialog(null, "ERROR AL CREAR ARCHIVO");
} catch (URISyntaxException e1) {
    // TODO Auto-generated catch block
    e1.printStackTrace();
}
}
```

```
331
332     btnDnf = new JButton("");
333     btnDnf.addActionListener(new ActionListener() {
334         public void actionPerformed(ActionEvent e) {
335             try {
336                 File newOutputstream_archivo;
337                 URL uri = new URI(getClass().getResource("/pdf/ReportePromedio.pdf").toString());
338                 File file = new File(uri);
339                 archivo = new FileOutputStream(file);
340                 Document doc = new Document();
341                 PdfWriter writer = PdfWriter.getInstance(doc, archivo);
342                 doc.open();
343                 java.awt.Image img = Toolkit.getDefaultToolkit().getImage(getClass().getResource("/img/DeoClass.png"));
344                 Image imgc = Image.getContentsOfFile(getClass().getResource("/img/DeoClass.png"));
345                 imgc.setAlignment(Element.ALIGN_CENTER);
346                 imgc.scaleToFit(200, 200);
347                 doc.add(imgc);
348                 Paragraph p = new Paragraph("10");
349                 com.itextpdf.text.Font negrita = new com.itextpdf.text.Font(
350                     com.itextpdf.text.Font.FontFamily.TIMES_ROMAN, 12, Font.BOLD, BaseColor.BLACK);
351                 p.setFont(negrita);
352                 p.add("PROMEDIO");
353                 p.add("NUEVLINE");
354                 p.add("PROMEDIO");
355                 p.setAlignment(Element.ALIGN_CENTER);
356                 doc.add(p);
357                 PdfTable tabla = new PdfPTable(8);
358                 tabla.setWidthPercentage(100);
359                 PdfPCell c1 = new PdfPCell(new Phrase(" ID PROMEDIO", negrita));
360                 PdfPCell c2 = new PdfPCell(new Phrase(" ALUMNO", negrita));
361                 PdfPCell c3 = new PdfPCell(new Phrase(" PROFESOR", negrita));
362                 PdfPCell c4 = new PdfPCell(new Phrase(" SEMESTRE", negrita));
363                 PdfPCell c5 = new PdfPCell(new Phrase(" CARRERA", negrita));
364                 PdfPCell c6 = new PdfPCell(new Phrase(" ASIGNATURA", negrita));
365                 PdfPCell c7 = new PdfPCell(new Phrase(" PROMEDIO", negrita));
366                 PdfPCell c8 = new PdfPCell(new Phrase(" ", negrita));
367                 c1.setHorizontalAlignment(Element.ALIGN_CENTER);
368                 c2.setHorizontalAlignment(Element.ALIGN_CENTER);
```

Y vamos a agregar el siguiente código:

```
496+ public void cargarTablaZ(String palabra) {
497+     while ((modelo.getRowCount()) > 0)
498+         modelo.removeRow(0);
499+
500+     lista = dao.buscar(palabra);
501+     for (Promedio p : lista) {
502+         Object item[] = new Object[8];
503+         item[0] = getNombrePromedio();
504+         item[1] = getNombre();
505+         item[2] = getProfesion();
506+         item[3] = getEstimaciones();
507+         item[4] = getCaracter();
508+         item[5] = getGrupo();
509+         item[6] = getSignaturas();
510+         item[7] = getPromedio();
511+         modelo.addRow(item);
512+     }
513+     tbIPromedios.setModel(modelo);
514+
515+ }
```

Estos códigos de arriba se ponen hasta lo último del código oh arriba.

DAO PROFESOR

En el Package de dao vamos a crear una clase llamada “daoProfesor” y quedaría así:

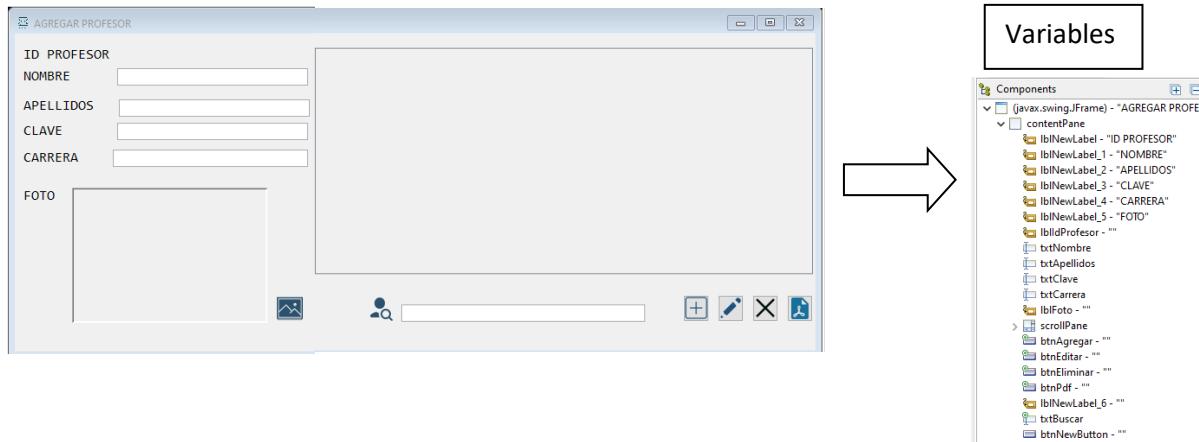
```

1 package dao;
2
3 import java.io.FileInputStream;
4
5 public class daoProfesor {
6     Conexion cx = null;
7
8     public daoProfesor() {
9         cx = new Conexion();
10    }
11
12     public boolean insertarProfesor(Profesor user) {
13         PreparedStatement ps = null;
14         try {
15             ps = cx.conectar().prepareStatement("INSERT INTO profesor VALUES(null,?, ?, ?, ?)");
16             ps.setString(1, user.getNombre());
17             ps.setString(2, user.getApellidos());
18             ps.setInt(3, user.getClave());
19             ps.setString(4, user.getCarrera());
20             ps.setString(5, user.getFoto());
21             ps.executeUpdate();
22             return true;
23         } catch (SQLException e) {
24             e.printStackTrace();
25             return false;
26         }
27     }
28
29     public ArrayList<Profesor> buscar(String palabra) {
30         ArrayList<Profesor> lista2 = new ArrayList<Profesor>();
31         try {
32             String sql = "SELECT * FROM profesor WHERE "
33                 + "(nombre LIKE ? OR "
34                 + "(apellidos LIKE ? OR "
35                 + "(clave LIKE ?) OR "
36                 + "(carrera LIKE ?) OR "
37                 + "(foto LIKE ?);";
38
39             PreparedStatement ps = cx.conectar().prepareStatement(sql);
40
41             ps.setString(1, "%" + palabra + "%");
42             ps.setString(2, "%" + palabra + "%");
43             ps.setString(3, "%" + palabra + "%");
44             ps.setString(4, "%" + palabra + "%");
45             ps.setString(5, "%" + palabra + "%");
46             //System.out.println("CONSULTA " + ps.toString());
47             ResultSet rs = ps.executeQuery();
48             while (rs.next()) {
49                 Profesor u = new Profesor();
50                 u.setIdProfesor(rs.getInt("idProfesor"));
51                 u.setNombre(rs.getString("nombre"));
52                 u.setApellido(rs.getString("apellidos"));
53                 u.setClave(rs.getInt("clave"));
54                 u.setCarrera(rs.getString("carrera"));
55                 u.setFoto(rs.getString("foto"));
56                 lista2.add(u);
57             }
58         } catch (SQLException e) {
59             e.printStackTrace();
60         }
61         cx.desconectar();
62     }
63     public ArrayList<Profesor> fetcProfesors() {
64         ArrayList<Profesor> lista = new ArrayList<Profesor>();
65         PreparedStatement ps = null;
66         ResultSet rs = null;
67         try {
68             ps = cx.conectar().prepareStatement("SELECT *FROM profesor");
69             rs = ps.executeQuery();
70             while (rs.next()) {
71                 Profesor u = new Profesor();
72                 u.setIdProfesor(rs.getInt("idProfesor"));
73                 u.setNombre(rs.getString("nombre"));
74             }
75         } catch (SQLException e) {
76             e.printStackTrace();
77         }
78     }
79     public boolean EliminarProfesor(int Id) {
80         PreparedStatement ps = null;
81         try {
82             ps = cx.conectar().prepareStatement("DELETE FROM profesor WHERE idProfesor=?");
83             ps.setInt(1, Id);
84             ps.executeUpdate();
85             return true;
86         } catch (SQLException e) {
87             e.printStackTrace();
88             return false;
89         }
90     }
91     public boolean editarProfesor(Profesor user) {
92         PreparedStatement ps = null;
93         try {
94             ps = cx.conectar().prepareStatement("UPDATE profesor SET nombre=?,apellidos=?,clave=?,carrera=?,foto=? WHERE idProfesor=?");
95             ps.setString(1, user.getNombre());
96             ps.setString(2, user.getApellidos());
97             ps.setInt(3, user.getIdProfesor());
98             ps.setString(4, user.getCarrera());
99             ps.setString(5, user.getFoto());
100            ps.setInt(6, user.getIdProfesor());
101            ps.executeUpdate();
102            return true;
103        } catch (SQLException e) {
104            e.printStackTrace();
105            return false;
106        }
107    }
108
109    public boolean editarProfesor(Profesor user) {
110        PreparedStatement ps = null;
111        try {
112            ps = cx.conectar().prepareStatement("UPDATE profesor SET nombre=?,apellidos=?,clave=?,carrera=?,foto=? WHERE idProfesor=?");
113            ps.setString(1, user.getNombre());
114            ps.setString(2, user.getApellidos());
115            ps.setInt(3, user.getIdProfesor());
116            ps.setString(4, user.getCarrera());
117            ps.setString(5, user.getFoto());
118            ps.setInt(6, user.getIdProfesor());
119            ps.executeUpdate();
120            return true;
121        } catch (SQLException e) {
122            e.printStackTrace();
123            return false;
124        }
125    }
126
127    public void setApellido(String apellido) {
128        this.apellido = apellido;
129    }
130
131    public void setNombre(String nombre) {
132        this.nombre = nombre;
133    }
134
135    public void setClave(int clave) {
136        this.clave = clave;
137    }
138
139    public String getCarrera() {
140        return carrera;
141    }
142
143    public void setCarrera(String carrera) {
144        this.carrera = carrera;
145    }
146
147    public String getFoto() {
148        return foto;
149    }
150
151    public void setFoto(String foto) {
152        this.foto = foto;
153    }
154
155    public int getIdProfesor() {
156        return idProfesor;
157    }
158
159    public void setIdProfesor(int idProfesor) {
160        this.idProfesor = idProfesor;
161    }
162
163    public String getNombre() {
164        return nombre;
165    }
166
167    public void setNombre(String nombre) {
168        this.nombre = nombre;
169    }
170
171    public String getApellidos() {
172        return apellido;
173    }
174
175    public void setApellidos(String apellidos) {
176        this.apellido = apellidos;
177    }
178
179    public int getClave() {
180        return clave;
181    }
182
183    public void setClave(int clave) {
184        this.clave = clave;
185    }
186
187    public void setCarrera(String carrera) {
188        this.carrera = carrera;
189    }
190
191    public void setFoto(String foto) {
192        this.foto = foto;
193    }
194
195    public void setApellido(String apellido) {
196        this.apellido = apellido;
197    }
198
199    public void setNombre(String nombre) {
200        this.nombre = nombre;
201    }
202
203    public void setIdProfesor(int idProfesor) {
204        this.idProfesor = idProfesor;
205    }
206
207    public void setClave(int clave) {
208        this.clave = clave;
209    }
210
211    public void setCarrera(String carrera) {
212        this.carrera = carrera;
213    }
214
215    public void setFoto(String foto) {
216        this.foto = foto;
217    }
218
219    public void setApellido(String apellido) {
220        this.apellido = apellido;
221    }
222
223    public void setNombre(String nombre) {
224        this.nombre = nombre;
225    }
226
227    public void setIdProfesor(int idProfesor) {
228        this.idProfesor = idProfesor;
229    }
230
231    public void setClave(int clave) {
232        this.clave = clave;
233    }
234
235    public void setCarrera(String carrera) {
236        this.carrera = carrera;
237    }
238
239    public void setFoto(String foto) {
240        this.foto = foto;
241    }
242
243    public void setApellido(String apellido) {
244        this.apellido = apellido;
245    }
246
247    public void setNombre(String nombre) {
248        this.nombre = nombre;
249    }
250
251    public void setIdProfesor(int idProfesor) {
252        this.idProfesor = idProfesor;
253    }
254
255    public void setClave(int clave) {
256        this.clave = clave;
257    }
258
259    public void setCarrera(String carrera) {
260        this.carrera = carrera;
261    }
262
263    public void setFoto(String foto) {
264        this.foto = foto;
265    }
266
267    public void setApellido(String apellido) {
268        this.apellido = apellido;
269    }
270
271    public void setNombre(String nombre) {
272        this.nombre = nombre;
273    }
274
275    public void setIdProfesor(int idProfesor) {
276        this.idProfesor = idProfesor;
277    }
278
279    public void setClave(int clave) {
280        this.clave = clave;
281    }
282
283    public void setCarrera(String carrera) {
284        this.carrera = carrera;
285    }
286
287    public void setFoto(String foto) {
288        this.foto = foto;
289    }
290
291    public void setApellido(String apellido) {
292        this.apellido = apellido;
293    }
294
295    public void setNombre(String nombre) {
296        this.nombre = nombre;
297    }
298
299    public void setIdProfesor(int idProfesor) {
300        this.idProfesor = idProfesor;
301    }
302
303    public void setClave(int clave) {
304        this.clave = clave;
305    }
306
307    public void setCarrera(String carrera) {
308        this.carrera = carrera;
309    }
310
311    public void setFoto(String foto) {
312        this.foto = foto;
313    }
314
315    public void setApellido(String apellido) {
316        this.apellido = apellido;
317    }
318
319    public void setNombre(String nombre) {
320        this.nombre = nombre;
321    }
322
323    public void setIdProfesor(int idProfesor) {
324        this.idProfesor = idProfesor;
325    }
326
327    public void setClave(int clave) {
328        this.clave = clave;
329    }
330
331    public void setCarrera(String carrera) {
332        this.carrera = carrera;
333    }
334
335    public void setFoto(String foto) {
336        this.foto = foto;
337    }
338
339    public void setApellido(String apellido) {
340        this.apellido = apellido;
341    }
342
343    public void setNombre(String nombre) {
344        this.nombre = nombre;
345    }
346
347    public void setIdProfesor(int idProfesor) {
348        this.idProfesor = idProfesor;
349    }
350
351    public void setClave(int clave) {
352        this.clave = clave;
353    }
354
355    public void setCarrera(String carrera) {
356        this.carrera = carrera;
357    }
358
359    public void setFoto(String foto) {
360        this.foto = foto;
361    }
362
363    public void setApellido(String apellido) {
364        this.apellido = apellido;
365    }
366
367    public void setNombre(String nombre) {
368        this.nombre = nombre;
369    }
370
371    public void setIdProfesor(int idProfesor) {
372        this.idProfesor = idProfesor;
373    }
374
375    public void setClave(int clave) {
376        this.clave = clave;
377    }
378
379    public void setCarrera(String carrera) {
380        this.carrera = carrera;
381    }
382
383    public void setFoto(String foto) {
384        this.foto = foto;
385    }
386
387    public void setApellido(String apellido) {
388        this.apellido = apellido;
389    }
390
391    public void setNombre(String nombre) {
392        this.nombre = nombre;
393    }
394
395    public void setIdProfesor(int idProfesor) {
396        this.idProfesor = idProfesor;
397    }
398
399    public void setClave(int clave) {
400        this.clave = clave;
401    }
402
403    public void setCarrera(String carrera) {
404        this.carrera = carrera;
405    }
406
407    public void setFoto(String foto) {
408        this.foto = foto;
409    }
410
411    public void setApellido(String apellido) {
412        this.apellido = apellido;
413    }
414
415    public void setNombre(String nombre) {
416        this.nombre = nombre;
417    }
418
419    public void setIdProfesor(int idProfesor) {
420        this.idProfesor = idProfesor;
421    }
422
423    public void setClave(int clave) {
424        this.clave = clave;
425    }
426
427    public void setCarrera(String carrera) {
428        this.carrera = carrera;
429    }
430
431    public void setFoto(String foto) {
432        this.foto = foto;
433    }
434
435    public void setApellido(String apellido) {
436        this.apellido = apellido;
437    }
438
439    public void setNombre(String nombre) {
440        this.nombre = nombre;
441    }
442
443    public void setIdProfesor(int idProfesor) {
444        this.idProfesor = idProfesor;
445    }
446
447    public void setClave(int clave) {
448        this.clave = clave;
449    }
450
451    public void setCarrera(String carrera) {
452        this.carrera = carrera;
453    }
454
455    public void setFoto(String foto) {
456        this.foto = foto;
457    }
458
459    public void setApellido(String apellido) {
460        this.apellido = apellido;
461    }
462
463    public void setNombre(String nombre) {
464        this.nombre = nombre;
465    }
466
467    public void setIdProfesor(int idProfesor) {
468        this.idProfesor = idProfesor;
469    }
470
471    public void setClave(int clave) {
472        this.clave = clave;
473    }
474
475    public void setCarrera(String carrera) {
476        this.carrera = carrera;
477    }
478
479    public void setFoto(String foto) {
480        this.foto = foto;
481    }
482
483    public void setApellido(String apellido) {
484        this.apellido = apellido;
485    }
486
487    public void setNombre(String nombre) {
488        this.nombre = nombre;
489    }
490
491    public void setIdProfesor(int idProfesor) {
492        this.idProfesor = idProfesor;
493    }
494
495    public void setClave(int clave) {
496        this.clave = clave;
497    }
498
499    public void setCarrera(String carrera) {
500        this.carrera = carrera;
501    }
502
503    public void setFoto(String foto) {
504        this.foto = foto;
505    }
506
507    public void setApellido(String apellido) {
508        this.apellido = apellido;
509    }
510
511    public void setNombre(String nombre) {
512        this.nombre = nombre;
513    }
514
515    public void setIdProfesor(int idProfesor) {
516        this.idProfesor = idProfesor;
517    }
518
519    public void setClave(int clave) {
520        this.clave = clave;
521    }
522
523    public void setCarrera(String carrera) {
524        this.carrera = carrera;
525    }
526
527    public void setFoto(String foto) {
528        this.foto = foto;
529    }
530
531    public void setApellido(String apellido) {
532        this.apellido = apellido;
533    }
534
535    public void setNombre(String nombre) {
536        this.nombre = nombre;
537    }
538
539    public void setIdProfesor(int idProfesor) {
540        this.idProfesor = idProfesor;
541    }
542
543    public void setClave(int clave) {
544        this.clave = clave;
545    }
546
547    public void setCarrera(String carrera) {
548        this.carrera = carrera;
549    }
550
551    public void setFoto(String foto) {
552        this.foto = foto;
553    }
554
555    public void setApellido(String apellido) {
556        this.apellido = apellido;
557    }
558
559    public void setNombre(String nombre) {
560        this.nombre = nombre;
561    }
562
563    public void setIdProfesor(int idProfesor) {
564        this.idProfesor = idProfesor;
565    }
566
567    public void setClave(int clave) {
568        this.clave = clave;
569    }
570
571    public void setCarrera(String carrera) {
572        this.carrera = carrera;
573    }
574
575    public void setFoto(String foto) {
576        this.foto = foto;
577    }
578
579    public void setApellido(String apellido) {
580        this.apellido = apellido;
581    }
582
583    public void setNombre(String nombre) {
584        this.nombre = nombre;
585    }
586
587    public void setIdProfesor(int idProfesor) {
588        this.idProfesor = idProfesor;
589    }
590
591    public void setClave(int clave) {
592        this.clave = clave;
593    }
594
595    public void setCarrera(String carrera) {
596        this.carrera = carrera;
597    }
598
599    public void setFoto(String foto) {
600        this.foto = foto;
601    }
602
603    public void setApellido(String apellido) {
604        this.apellido = apellido;
605    }
606
607    public void setNombre(String nombre) {
608        this.nombre = nombre;
609    }
610
611    public void setIdProfesor(int idProfesor) {
612        this.idProfesor = idProfesor;
613    }
614
615    public void setClave(int clave) {
616        this.clave = clave;
617    }
618
619    public void setCarrera(String carrera) {
620        this.carrera = carrera;
621    }
622
623    public void setFoto(String foto) {
624        this.foto = foto;
625    }
626
627    public void setApellido(String apellido) {
628        this.apellido = apellido;
629    }
630
631    public void setNombre(String nombre) {
632        this.nombre = nombre;
633    }
634
635    public void setIdProfesor(int idProfesor) {
636        this.idProfesor = idProfesor;
637    }
638
639    public void setClave(int clave) {
640        this.clave = clave;
641    }
642
643    public void setCarrera(String carrera) {
644        this.carrera = carrera;
645    }
646
647    public void setFoto(String foto) {
648        this.foto = foto;
649    }
650
651    public void setApellido(String apellido) {
652        this.apellido = apellido;
653    }
654
655    public void setNombre(String nombre) {
656        this.nombre = nombre;
657    }
658
659    public void setIdProfesor(int idProfesor) {
660        this.idProfesor = idProfesor;
661    }
662
663    public void setClave(int clave) {
664        this.clave = clave;
665    }
666
667    public void setCarrera(String carrera) {
668        this.carrera = carrera;
669    }
670
671    public void setFoto(String foto) {
672        this.foto = foto;
673    }
674
675    public void setApellido(String apellido) {
676        this.apellido = apellido;
677    }
678
679    public void setNombre(String nombre) {
680        this.nombre = nombre;
681    }
682
683    public void setIdProfesor(int idProfesor) {
684        this.idProfesor = idProfesor;
685    }
686
687    public void setClave(int clave) {
688        this.clave = clave;
689    }
690
691    public void setCarrera(String carrera) {
692        this.carrera = carrera;
693    }
694
695    public void setFoto(String foto) {
696        this.foto = foto;
697    }
698
699    public void setApellido(String apellido) {
700        this.apellido = apellido;
701    }
702
703    public void setNombre(String nombre) {
704        this.nombre = nombre;
705    }
706
707    public void setIdProfesor(int idProfesor) {
708        this.idProfesor = idProfesor;
709    }
710
711    public void setClave(int clave) {
712        this.clave = clave;
713    }
714
715    public void setCarrera(String carrera) {
716        this.carrera = carrera;
717    }
718
719    public void setFoto(String foto) {
720        this.foto = foto;
721    }
722
723    public void setApellido(String apellido) {
724        this.apellido = apellido;
725    }
726
727    public void setNombre(String nombre) {
728        this.nombre = nombre;
729    }
730
731    public void setIdProfesor(int idProfesor) {
732        this.idProfesor = idProfesor;
733    }
734
735    public void setClave(int clave) {
736        this.clave = clave;
737    }
738
739    public void setCarrera(String carrera) {
740        this.carrera = carrera;
741    }
742
743    public void setFoto(String foto) {
744        this.foto = foto;
745    }
746
747    public void setApellido(String apellido) {
748        this.apellido = apellido;
749    }
750
751    public void setNombre(String nombre) {
752        this.nombre = nombre;
753    }
754
755    public void setIdProfesor(int idProfesor) {
756        this.idProfesor = idProfesor;
757    }
758
759    public void setClave(int clave) {
760        this.clave = clave;
761    }
762
763    public void setCarrera(String carrera) {
764        this.carrera = carrera;
765    }
766
767    public void setFoto(String foto) {
768        this.foto = foto;
769    }
770
771    public void setApellido(String apellido) {
772        this.apellido = apellido;
773    }
774
775    public void setNombre(String nombre) {
776        this.nombre = nombre;
777    }
778
779    public void setIdProfesor(int idProfesor) {
780        this.idProfesor = idProfesor;
781    }
782
783    public void setClave(int clave) {
784        this.clave = clave;
785    }
786
787    public void setCarrera(String carrera) {
788        this.carrera = carrera;
789    }
790
791    public void setFoto(String foto) {
792        this.foto = foto;
793    }
794
795    public void setApellido(String apellido) {
796        this.apellido = apellido;
797    }
798
799    public void setNombre(String nombre) {
800        this.nombre = nombre;
801    }
802
803    public void setIdProfesor(int idProfesor) {
804        this.idProfesor = idProfesor;
805    }
806
807    public void setClave(int clave) {
808        this.clave = clave;
809    }
810
811    public void setCarrera(String carrera) {
812        this.carrera = carrera;
813    }
814
815    public void setFoto(String foto) {
816        this.foto = foto;
817    }
818
819    public void setApellido(String apellido) {
820        this.apellido = apellido;
821    }
822
823    public void setNombre(String nombre) {
824        this.nombre = nombre;
825    }
826
827    public void setIdProfesor(int idProfesor) {
828        this.idProfesor = idProfesor;
829    }
830
831    public void setClave(int clave) {
832        this.clave = clave;
833    }
834
835    public void setCarrera(String carrera) {
836        this.carrera = carrera;
837    }
838
839    public void setFoto(String foto) {
840        this.foto = foto;
841    }
842
843    public void setApellido(String apellido) {
844        this.apellido = apellido;
845    }
846
847    public void setNombre(String nombre) {
848        this.nombre = nombre;
849    }
850
851    public void setIdProfesor(int idProfesor) {
852        this.idProfesor = idProfesor;
853    }
854
855    public void setClave(int clave) {
856        this.clave = clave;
857    }
858
859    public void setCarrera(String carrera) {
860        this.carrera = carrera;
861    }
862
863    public void setFoto(String foto) {
864        this.foto = foto;
865    }
866
867    public void setApellido(String apellido) {
868        this.apellido = apellido;
869    }
870
871    public void setNombre(String nombre) {
872        this.nombre = nombre;
873    }
874
875    public void setIdProfesor(int idProfesor) {
876        this.idProfesor = idProfesor;
877    }
878
879    public void setClave(int clave) {
880        this.clave = clave;
881    }
882
883    public void setCarrera(String carrera) {
884        this.carrera = carrera;
885    }
886
887    public void setFoto(String foto) {
888        this.foto = foto;
889    }
890
891    public void setApellido(String apellido) {
892        this.apellido = apellido;
893    }
894
895    public void setNombre(String nombre) {
896        this.nombre = nombre;
897    }
898
899    public void setIdProfesor(int idProfesor) {
900        this.idProfesor = idProfesor;
901    }
902
903    public void setClave(int clave) {
904        this.clave = clave;
905    }
906
907    public void setCarrera(String carrera) {
908        this.carrera = carrera;
909    }
910
911    public void setFoto(String foto) {
912        this.foto = foto;
913    }
914
915    public void setApellido(String apellido) {
916        this.apellido = apellido;
917    }
918
919    public void setNombre(String nombre) {
920        this.nombre = nombre;
921    }
922
923    public void setIdProfesor(int idProfesor) {
924        this.idProfesor = idProfesor;
925    }
926
927    public void setClave(int clave) {
928        this.clave = clave;
929    }
930
931    public void setCarrera(String carrera) {
932        this.carrera = carrera;
933    }
934
935    public void setFoto(String foto) {
936        this.foto = foto;
937    }
938
939    public void setApellido(String apellido) {
940        this.apellido = apellido;
941    }
942
943    public void setNombre(String nombre) {
944        this.nombre = nombre;
945    }
946
947    public void setIdProfesor(int idProfesor) {
948        this.idProfesor = idProfesor;
949    }
950
951    public void setClave(int clave) {
952        this.clave = clave;
953    }
954
955    public void setCarrera(String carrera) {
956        this.carrera = carrera;
957    }
958
959    public void setFoto(String foto) {
960        this.foto = foto;
961    }
962
963    public void setApellido(String apellido) {
964        this.apellido = apellido;
965    }
966
967    public void setNombre(String nombre) {
968        this.nombre = nombre;
969    }
970
971    public void setIdProfesor(int idProfesor) {
972        this.idProfesor = idProfesor;
973    }
974
975    public void setClave(int clave) {
976        this.clave = clave;
977    }
978
979    public void setCarrera(String carrera) {
980        this.carrera = carrera;
981    }
982
983    public void setFoto(String foto) {
984        this.foto = foto;
985    }
986
987    public void setApellido(String apellido) {
988        this.apellido = apellido;
989    }
990
991    public void setNombre(String nombre) {
992        this.nombre = nombre;
993    }
994
995    public void setIdProfesor(int idProfesor) {
996        this.idProfesor = idProfesor;
997    }
998
999    public void setClave(int clave) {
1000        this.clave = clave;
1001    }
1002
1003    public void setCarrera(String carrera) {
1004        this.carrera = carrera;
1005    }
1006
1007    public void setFoto(String foto) {
1008        this.foto = foto;
1009    }
1010
1011    public void setApellido(String apellido) {
1012        this.apellido = apellido;
1013    }
1014
1015    public void setNombre(String nombre) {
1016        this.nombre = nombre;
1017    }
1018
1019    public void setIdProfesor(int idProfesor) {
1020        this.idProfesor = idProfesor;
1021    }
1022
1023    public void setClave(int clave) {
1024        this.clave = clave;
1025    }
1026
1027    public void setCarrera(String carrera) {
1028        this.carrera = carrera;
1029    }
1030
1031    public void setFoto(String foto) {
1032        this.foto = foto;
1033    }
1034
1035    public void setApellido(String apellido) {
1036        this.apellido = apellido;
1037    }
1038
1039    public void setNombre(String nombre) {
1040        this.nombre = nombre;
1041    }
1042
1043    public void setIdProfesor(int idProfesor) {
1044        this.idProfesor = idProfesor;
1045    }
1046
1047    public void setClave(int clave) {
1048        this.clave = clave;
1049    }
1050
1051    public void setCarrera(String carrera) {
1052        this.carrera = carrera;
1053    }
1054
1055    public void setFoto(String foto) {
1056        this.foto = foto;
1057    }
1058
1059    public void setApellido(String apellido) {
1060        this.apellido = apellido;
1061    }
1062
1063    public void setNombre(String nombre) {
1064        this.nombre = nombre;
1065    }
1066
1067    public void setIdProfesor(int idProfesor) {
1068        this.idProfesor = idProfesor;
1069    }
1070
1071    public void setClave(int clave) {
1072        this.clave = clave;
1073    }
1074
1075    public void setCarrera(String carrera) {
1076        this.carrera = carrera;
1077    }
1078
1079    public void setFoto(String foto) {
1080        this.foto = foto;
1081    }
1082
1083    public void setApellido(String apellido) {
1084        this.apellido = apellido;
1085    }
1086
1087    public void setNombre(String nombre) {
1088        this.nombre = nombre;
1089    }
1090
1091    public void setIdProfesor(int idProfesor) {
1092        this.idProfesor = idProfesor;
1093    }
1094
1095    public void setClave(int clave) {
1096        this.clave = clave;
1097    }
1098
1099    public void setCarrera(String carrera) {
1100        this.carrera = carrera;
1101    }
1102
1103    public void setFoto(String foto) {
1104        this.foto = foto;
1105    }
1106
1107    public void setApellido(String apellido) {
1108        this.apellido = apellido;
1109    }
1110
1111    public void setNombre(String nombre) {
1112        this.nombre = nombre;
1113    }
1114
1115    public void setIdProfesor(int idProfesor) {
1116        this.idProfesor = idProfesor;
1117    }
1118
1119    public void setClave(int clave) {
1120        this.clave = clave;
1121    }
1122
1123    public void setCarrera(String carrera) {
1124        this.carrera = carrera;
1125    }
1126
1127    public void setFoto(String foto) {
1128        this.foto = foto;
1129    }
1130
1131    public void setApellido(String apellido) {
1132        this.apellido = apellido;
1133    }
1134
1135    public void setNombre(String nombre) {
1136        this.nombre = nombre;
1137    }
1138
1139    public void setIdProfesor(int idProfesor) {
1140        this.idProfesor = idProfesor;
1141    }
1142
1143    public void setClave(int clave) {
1144        this.clave = clave;
1145    }
1146
1147    public void setCarrera(String carrera) {
1148        this.carrera = carrera;
1149    }
1150
1151    public void setFoto(String foto) {
1152        this.foto = foto;
1153    }
1154
1155    public void setApellido(String apellido) {
1156        this.apellido = apellido;
1157    }
1158
1159    public void setNombre(String nombre) {
1160        this.nombre = nombre;
1161    }
1162
1163    public void setIdProfesor(int idProfesor) {
1164        this.idProfesor = idProfesor;
1165    }
1166
1167    public void setClave(int clave) {
1168        this.clave = clave;
1169    }
1170
1171    public void setCarrera(String carrera) {
1172        this.carrera = carrera;
1173    }
1174
1175    public void setFoto(String foto) {
1176        this.foto = foto;
1177    }
1178
1179    public void setApellido(String apellido) {
1180        this.apellido = apellido;
1181    }
1182
1183    public void setNombre(String nombre) {
1184        this.nombre = nombre;
1185    }
1186
1187    public void setIdProfesor(int idProfesor) {
1188        this.idProfesor = idProfesor;
1189    }
1190
1191    public void setClave(int clave) {
1192        this.clave = clave;
1193    }
1194
1195    public void setCarrera(String carrera) {
1196        this.carrera = carrera;
1197    }
1198
1199    public void setFoto(String foto) {
1200        this.foto = foto;
1201    }
1202
1203    public void setApellido(String apellido) {
1204        this.apellido = apellido;
1205    }
1206
1207    public void setNombre(String nombre) {
1208        this.nombre = nombre;
1209    }
1210
1211    public void setIdProfesor(int idProfesor) {
1212        this.idProfesor = idProfesor;
1213    }
1214
1215    public void setClave(int clave) {
1216        this.clave = clave;
1217    }
1218
1219    public void setCarrera(String carrera) {
1220        this.carrera = carrera;
1221    }
1222
1223    public void setFoto(String foto) {
1224        this.foto = foto;
1225    }
1226
1227    public void setApellido(String apellido) {
1228        this.apellido = apellido;
1229    }
1230
1231    public void setNombre(String nombre) {
1232        this.nombre = nombre;
1233    }
1234
1235    public void setIdProfesor(int idProfesor) {
1236        this.idProfesor = idProfesor;
1237    }
1238
1239    public void setClave(int clave) {
1240        this.clave = clave;
1241    }
1242
1243    public void setCarrera(String carrera) {
1244        this.carrera = carrera;
1245    }
1246
1247    public void setFoto(String foto) {
1248        this.foto = foto;
1249    }
1250
1251    public void setApellido(String apellido) {
1252        this.apellido = apellido;
1253    }
1254
1255    public void setNombre(String nombre) {
1256        this.nombre = nombre;
1257    }
1258
1259    public void setIdProfesor(int idProfesor) {
1260        this.idProfesor = idProfesor;
1261    }
1262
1263    public void setClave(int clave) {
1264        this.clave = clave;
1265    }
1266
1267    public void setCarrera(String carrera) {
1268        this.carrera = carrera;
1269    }
1270
1271    public void setFoto(String foto) {
1272        this.foto = foto;
1273    }
1274
1275    public void setApellido(String apellido) {
1276        this.apellido = apellido;
1277    }
1278
1279    public void setNombre(String nombre) {
1280        this.nombre = nombre;
1281    }
1282
1283    public void setIdProfesor(int idProfesor) {
1284        this.idProfesor = idProfesor;
1285    }
1286
1287    public void setClave(int clave) {
1288        this.clave = clave;
1289    }
1290
1291    public void setCarrera(String carrera) {
1292        this.carrera = carrera;
1293    }
1294
1295    public void setFoto(String foto) {
1296        this.foto = foto;
1297    }
1298
1299    public void setApellido(String apellido) {
1300        this.apellido = apellido;
1301    }
1302
1303    public void setNombre(String nombre) {
1304        this.nombre = nombre;
1305    }
1306
1307    public void setIdProfesor(int idProfesor) {
1308        this.idProfesor = idProfesor;
1309    }
1310
1311    public void setClave(int clave) {
1312        this.clave = clave;
1313    }
1314
1315    public void setCarrera(String carrera) {
1316        this.carrera = carrera;
1317    }
1318
1319    public void setFoto(String foto) {
1320        this.foto = foto;
1321    }
1322
1323    public void setApellido(String apellido) {
1324        this.apellido = apellido;
1325    }
1326
1327    public void setNombre(String nombre) {
1328        this.nombre = nombre;
1329    }
1330
1331    public void setIdProfesor(int idProfesor) {
1332        this.idProfesor = idProfesor;
1333    }
1334
1335    public void setClave(int clave) {
1336        this.clave = clave;
1337    }
1338
1339    public void setCarrera(String carrera) {
1340        this.carrera = carrera;
1341    }
1342
1343    public void setFoto(String foto) {
1344        this.foto = foto;
1345    }
1346
1347    public void setApellido(String
```

INTERFAZ DE PROFESOR

Vamos a crear un JFrame en el Package de Vistas el JFrame y se llamará "vProfesor".

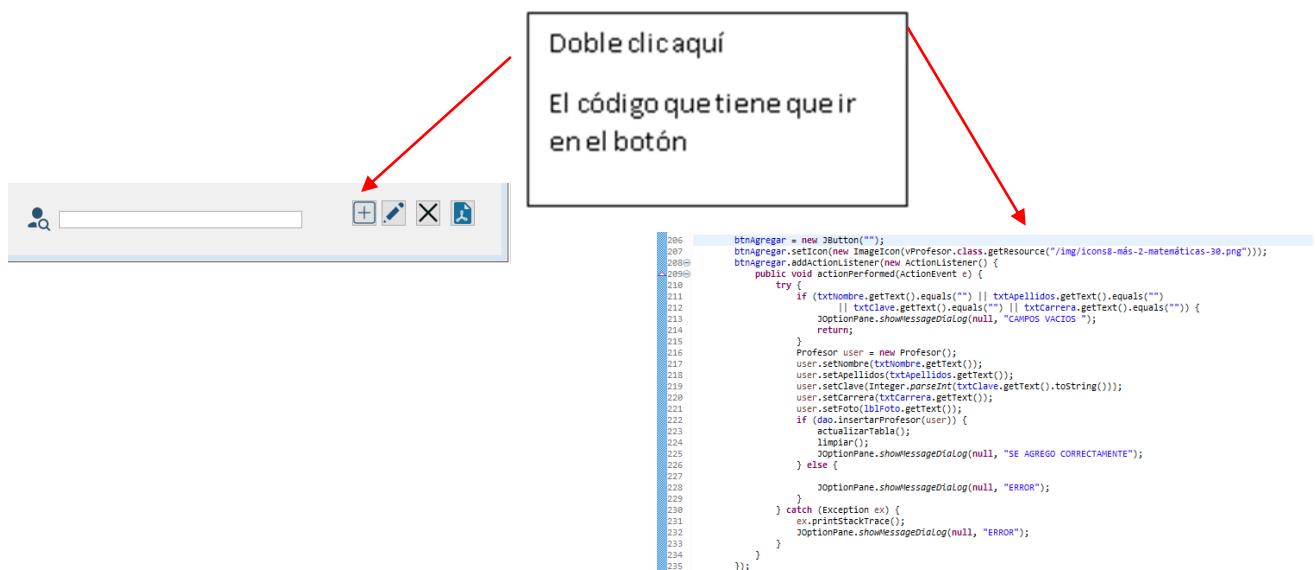
Utilizaremos 9 JLabel, 5 JTextField, 4 JButton, un JSpinner, un scrollPane, dentro del scrollPane vamos a agregar un JTable.



PROGRAMAR LOS BOTONES

BOTÓN AGREGAR

Agregar el evento **ActionPerformed** al botón Insertar que se encuentra en ApplicationWindow->vProfesor, dando doble clic sobre el botón.



BOTON EDITAR

Vamos a dar doble clic en el botón Editar y vamos a repetir el mismo procedimiento que hicimos el Botón Agregar.



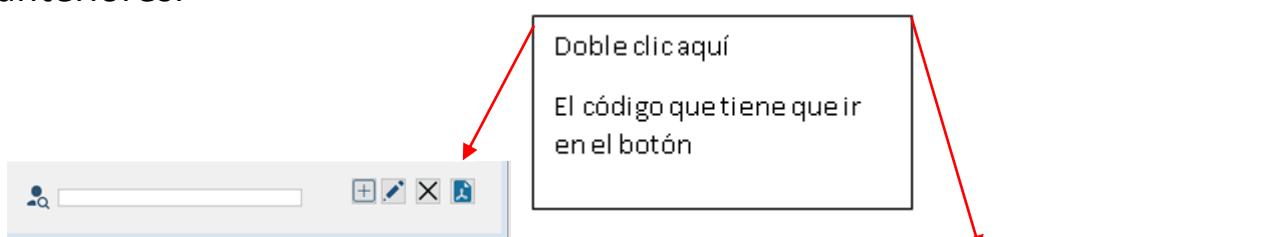
BOTON ELIMINAR

Vamos a dar doble clic en el botón Eliminar y vamos a repetir el mismo procedimiento que hicimos el Botón Agregar.



BOTON PDF

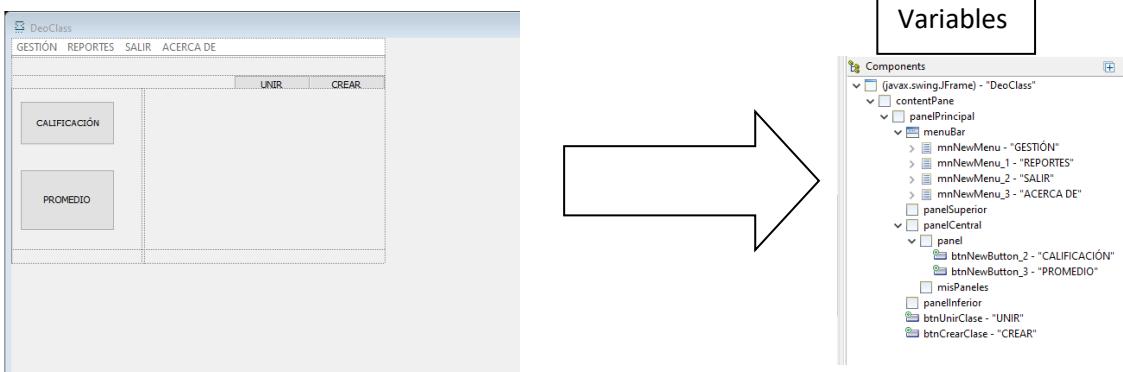
Tienes que dar doble clic en el botón PDF y vamos a repetir el mismo procedimiento que hemos hecho con los botones anteriores.



```
334     c5.setHorizontalAlignment(Element.ALIGN_CENTER);
335     c6.setHorizontalAlignment(Element.ALIGN_CENTER);
336     c1.setBackgroundColor(BaseColor.LIGHT_GRAY);
337     c2.setBackgroundColor(BaseColor.LIGHT_GRAY);
338     c3.setBackgroundColor(BaseColor.LIGHT_GRAY);
339     c4.setBackgroundColor(BaseColor.LIGHT_GRAY);
340     c5.setBackgroundColor(BaseColor.LIGHT_GRAY);
341     c6.setBackgroundColor(BaseColor.LIGHT_GRAY);
342 
343     tabla.addCell(c1);
344     tabla.addCell(c2);
345     tabla.addCell(c3);
346     tabla.addCell(c4);
347     tabla.addCell(c5);
348     tabla.addCell(c6);
349 
350     for (Profesor u : lista) {
351         tabla.addCell("+" + u.getIdProfesor());
352         tabla.addCell(u.getNombre());
353         tabla.addCell(u.getApellidos());
354         tabla.addCell("+" + u.getClave());
355         tabla.addCell(u.getCarera());
356         tabla.addCell(u.getFoto());
357     }
358 
359     doc.add(tabla);
360     Paragraph p1 = new Paragraph(10);
361     p1.add(chunk.NEWLINE);
362     p1.add("NUMERO DE REGISTRO " + lista.size());
363     p1.add(chunk.NEWLINE);
364     p1.add(chunk.NEWLINE);
365     p1.setAlignment(Element.ALIGN_RIGHT);
366 
367     doc.add(p1);
368     doc.close();
369     archivo.close();
370 
371     Desktop.getDesktop().open(file);
372     } catch (FileNotFoundException e1) {
373         JOptionPane.showMessageDialog(null, "ERROR AL CREAR ARCHIVO");
374     } catch (DocumentException e1) {
375         JOptionPane.showMessageDialog(null, "ERROR AL CREAR DOCUMENTO PDF");
376     } catch (IOException e1) {
377         JOptionPane.showMessageDialog(null, "ERROR AL CREAR IO");
378     } catch (URISyntaxException e1) {
379         // TODO Auto-generated catch block
380         e1.printStackTrace();
381     }
382 }
383 }
```

INTERFAZ DE PRINCIPAL

Vamos a crear un JFrame en el Package de Vistas el JFrame y se llamará "vUsuario".



PROGRAMAR LOS BOTONES

BOTON CALIFICACION

```
248 JButton btnNuevoButton_2 = new JButton("CALIFICACIÓN");
249 btnNuevoButton_2.setFont(new Font("Tahoma", Font.PLAIN, 10));
250 btnNuevoButton_2.addActionListener(new ActionListener() {
251     public void actionPerformed(ActionEvent e) {
252         VCalif call = new VCalif();
253         call.setVisible(true);
254     }
255 }
```

BOTON PROMEDIO

```
251 JButton btnNewButton_3 = new JButton("PROMEDIO");
252 btnNewButton_3.addActionListener(new ActionListener() {
253     public void actionPerformed(ActionEvent e) {
254         vPromedio pro = new vPromedio();
255         pro.setVisible(true);
256     }
257});
```

BOTON UNIR

```
269 JButton btnUnirClase = new JButton("UNIR");
270 btnUnirClase.addActionListener(new ActionListener() {
271     public void actionPerformed(ActionEvent e) {
272         Unir unir = new Unir();
273         unir.setVisible(true);
274     }
275});
```

BOTON CREAR

```
279 JButton btnCrearClase = new JButton("CREAR");
280 btnCrearClase.addActionListener(new ActionListener() {
281     public void actionPerformed(ActionEvent e) {
282         vClase clase = new vClase(idUsuario);
283         clase.setVisible(true);
284     }
285});
```

DAO PLANTEL

En el Package de dao vamos a crear una clase llamada “daoPlantele” y quedaría así:

```
1 package dao;
2
3 import java.sql.PreparedStatement;
4
5 public class daoPlantele {
6     Alumno a;
7     ArrayList<Plantel> lista;
8     conexion cx = null;
9
10    public daoPlantele() {
11        cx = new conexion();
12        lista = new ArrayList<Plantel>();
13        a = new Alumno();
14    }
15
16    public boolean insertarPlantel(Plantel a) {
17        PreparedStatement ps = null;
18        try {
19            ps = cx.conectar().prepareStatement("INSERT INTO plantel VALUES(null,?)");
20            ps.setString(1, a.getPlantel());
21            ps.executeUpdate();
22            return true;
23        } catch (SQLException e) {
24            e.printStackTrace();
25            return false;
26        } finally {
27            try {
28                ps.close();
29                ps = null;
30            } catch (SQLException e) {
31                System.out.println("ERROR AL CERRAR EDITAR USUARIO");
32            }
33        }
34    }
35
36    public ArrayList<Plantel> fetchPlantels() {
37        ArrayList<Plantel> lista = new ArrayList<Plantel>();
38        PreparedStatement ps = null;
39        ResultSet rs = null;
40        try {
41            ps = cx.conectar().prepareStatement("SELECT *FROM plantel");
42            ps.executeQuery();
43            while (rs.next()) {
44                Plantel p = new Plantel();
45                p.setIdPlantel(rs.getInt("idPlantel"));
46                u.setPlantel(rs.getString("plantel"));
47                lista.add(u);
48            }
49        } catch (SQLException e) {
50            // TODO Auto-generated catch block
51            e.printStackTrace();
52        } finally {
53            try {
54                ps.close();
55                ps = null;
56            } catch (SQLException e) {
57                System.out.println("ERROR AL CERRAR EDITAR USUARIO");
58            }
59        }
60    }
61
62    public boolean editarPlantel(Plantel user) {
63        PreparedStatement ps = null;
64        try {
65            ps = cx.conectar().prepareStatement("UPDATE plantel SET plantel=? WHERE idPlantel=?");
66            ps.setInt(1, user.getIdPlantel());
67            ps.setString(2, user.getPlantel());
68            ps.executeUpdate();
69        } catch (SQLException e) {
70            e.printStackTrace();
71        } finally {
72            try {
73                ps.close();
74                ps = null;
75            } catch (SQLException e) {
76                System.out.println("ERROR AL CERRAR EDITAR USUARIO");
77            }
78        }
79    }
80
81    public boolean EliminarPlantel(int Id) {
82        PreparedStatement ps = null;
83        try {
84            ps = cx.conectar().prepareStatement("DELETE FROM plantel WHERE idPlantel=?");
85            ps.setInt(1, Id);
86            ps.executeUpdate();
87        } catch (SQLException e) {
88            e.printStackTrace();
89        }
90    }
91}
```

MODELO PLANTEL

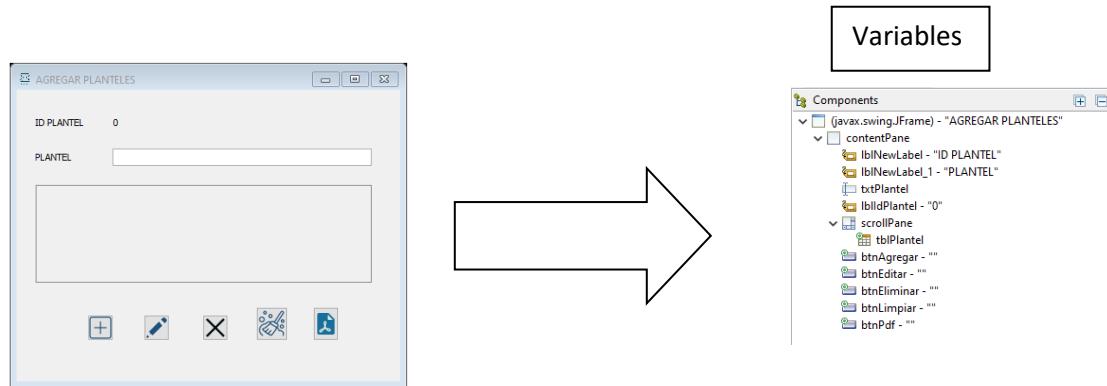
Vamos a crear una clase en el Package de Modelo la cuenta se llamará “Plantel” quedaría algo así:

```
1 package modelo;
2
3 public class Plantel {
4     int idPlantel;
5     String plantel;
6
7     public Plantel() {
8     }
9     public int getIdPlantel() {
10         return idPlantel;
11     }
12     public void setIdPlantel(int idPlantel) {
13         this.idPlantel = idPlantel;
14     }
15     public String getplantel() {
16         return plantel;
17     }
18     public void setPlantel(String plantel) {
19         plantel = plantel;
20     }
21
22 }
23
```

INTERFAZ DE PLANTEL

Vamos a crear un JFrame en el Package de Vistas el JFrame y se llamará “vPlantel”.

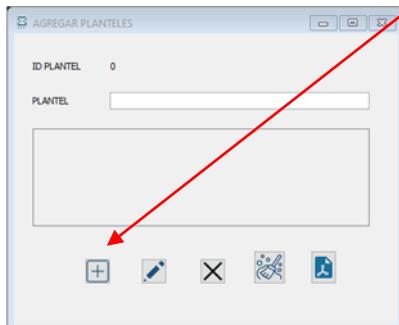
Utilizaremos 3 JLabel, 1 JTextField, 4 JButton, un JSpinner, un scrollPane, dentro del scrollPane vamos a agregar un JTable.



PROGRAMAR LOS BOTONES

BOTON AGEGAR

Agregar el evento **ActionPerformed** al botón Insertar que se encuentra en **ApplicationWindow->vPlantel**, dando doble clic sobre el botón.



Doble clic aquí

El código que tiene que ir en el botón

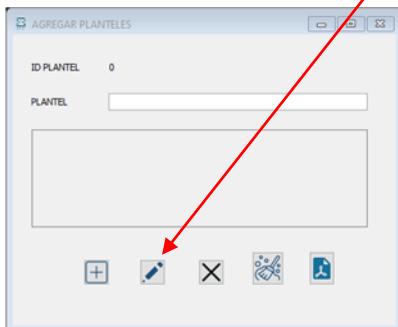
```

144 btngregar = new JButton("");
145 btngregar.setIcon(new ImageIcon(Carrera.class.getResource("/img/iconos-más-2-matemáticas-30.png")));
146 btngregar.addActionListener(new ActionListener() {
147     public void actionPerformed(ActionEvent e) {
148         try {
149             if (txtPlantele.getText().equals("")) {
150                 JOptionPane.showMessageDialog(null, "CAMPOS VACIOS ");
151                 return;
152             }
153             Plantel user = new Plantel();
154             user.setPlantel(txtPlantele.getText());
155             if (dao.insertarPlantel(user)) {
156                 actualizarTabla();
157                 limpiar();
158                 JOptionPane.showMessageDialog(null, "SE AGREGO CORRECTAMENTE");
159             } else {
160                 JOptionPane.showMessageDialog(null, "ERROR");
161             }
162         } catch (Exception ex) {
163             JOptionPane.showMessageDialog(null, "ERROR");
164         }
165     }
166 });

```

BOTON EDITAR

Vamos a dar doble clic en el botón Editar y vamos a repetir el mismo procedimiento que hicimos el Botón Agregar.



Doble clic aquí

El código que debe ir aquí

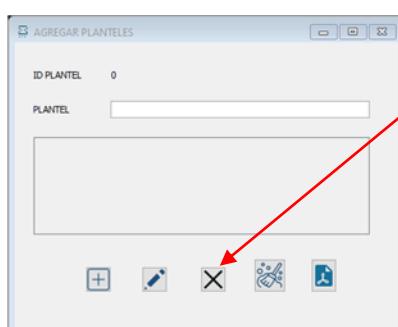
```

178 btndesar = new JButton("");
179 btndesar.addActionListener(new ActionListener() {
180     public void actionPerformed(ActionEvent e) {
181         try {
182             if (txtPlantele.getText().equals("")) {
183                 JOptionPane.showMessageDialog(null, "CAMPOS VACIOS ");
184                 return;
185             }
186             plantel.setPlantel(txtPlantele.getText());
187             if (dao.actualizarPlantel(plantel)) {
188                 actualizarTabla();
189                 limpiar();
190                 JOptionPane.showMessageDialog(null, "SE ACTUALIZO CORRECTAMENTE");
191             } else {
192                 JOptionPane.showMessageDialog(null, "ERROR");
193             }
194         } catch (Exception ex) {
195             JOptionPane.showMessageDialog(null, "ERROR");
196         }
197     }
198 });

```

BOTON ELIMINAR

Vamos a dar doble clic en el botón Eliminar y vamos a repetir el mismo procedimiento que hicimos el Botón Agregar.



Doble clic aquí

El código que tiene que ir en el botón

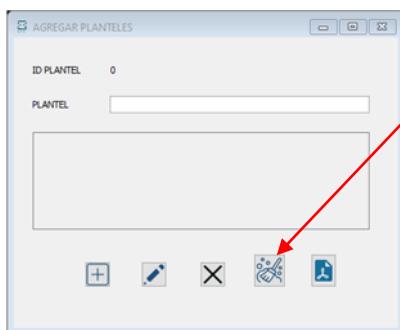
```

200 btndelete = new JButton("");
201 btndelete.addActionListener(new ActionListener() {
202     public void actionPerformed(ActionEvent e) {
203         try {
204             int opcion = JOptionPane.showConfirmDialog(null, "ESTA SEGURO DE ELIMINAR EL PLANTEL?", "ELIMINAR PLANTEL", JOptionPane.YES_NO_OPTION);
205             if (opcion == 1) {
206                 if (dao.eliminarPlantel([lista.get(Fila).getPlantel()])) {
207                     actualizarTabla();
208                     limpiar();
209                     JOptionPane.showMessageDialog(null, "SE ELIMINO CORRECTAMENTE");
210                 } else {
211                     JOptionPane.showMessageDialog(null, "ERROR");
212                 }
213             }
214         } catch (Exception ex) {
215             JOptionPane.showMessageDialog(null, "ERROR");
216         }
217     }
218 });

```

BOTON LIMPIAR

Vamos a dar doble clic en el botón Editar y vamos a repetir el mismo procedimiento que hicimos el Botón Agregar.

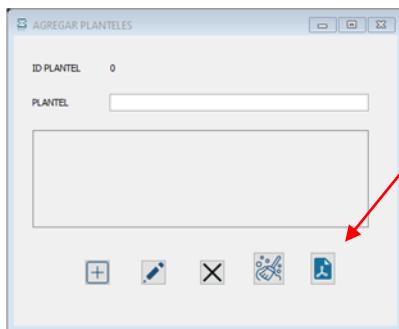


Doble clic aquí
El código que tiene que ir en el botón

```
83
84 public void limpiar() {
85     lblIdPlantel.setText("");
86     txtPlantel.setText("");
87 }
88
```

BOTON PDF

Tienes que dar doble clic en el botón PDF y vamos a repetir el mismo procedimiento que hemos hecho con los botones anteriores.



Doble clic aquí
El código que tiene que ir en el botón

```
231 btnPdf.addActionListener(new ActionListener() {
232     public void actionPerformed(ActionEvent e) {
233         try {
234             FileOutputStream archivo;
235             URI uri = new URI(getClass().getgetResource("/pdf/ReportePlanteles.pdf").toString());
236             File file = new File(uri);
237             PdfWriter.getInstance(doc, new FileOutputStream(file));
238             Document doc = new Document();
239             PdfWriter.getInstance(doc, archivo);
240             doc.open();
241             java.awt.Image img2 = Toolkit.getDefaultToolkit().getImage(getClass().getgetResource("/img/DeoClass.png"));
242             Image img = Image.getInstance(img2);
243             img.setAlignment(Element.ALIGN_CENTER);
244             img.scaleToFit(200, 200);
245             doc.add(img);
246             Paragraph p = new Paragraph(10);
247             com.itextpdf.text.Font negrita = new com.itextpdf.text.Font(
248                 com.itextpdf.text.FontFamily.TIMES_ROMAN, 12, Font.BOLD, BaseColor.BLACK);
249             p.addChunk(new Chunk("CATALOGO DE PLANTELES"));
250             p.addChunk(new Chunk("NEWLINE"));
251             p.addChunk(new Chunk("REGISTRO " + lista.size()));
252             p.setAlign(Element.ALIGN_CENTER);
253             p.setAlignment(Element.ALIGN_CENTER);
254             doc.add(p);
255             PdfPTable tabla = new PdfPTable(2);
256             tabla.setWidthPercentage(100);
257             PdfPCell c1 = new PdfPCell(new Phrase("ID PLANTEL", negrita));
258             PdfPCell c2 = new PdfPCell(new Phrase("PLANTEL", negrita));
259             c1.setHorizontalAlignment(Element.ALIGN_CENTER);
260             c2.setHorizontalAlignment(Element.ALIGN_CENTER);
261             c1.setBackgroundColor(BaseColor.GRAY);
262             c2.setBackgroundColor(BaseColor.LIGHT_GRAY);
263             tabla.addCell(c1);
264             tabla.addCell(c2);
265             for (Plantel u : lista) {
266                 tabla.addCell(" " + u.getIdPlantel());
267                 tabla.addCell(u.getPlantel());
268             }
269         } catch (FileNotFoundException e1) {
270             JOptionPane.showMessageDialog(null, "ERROR AL CREAR ARCHIVO");
271         } catch (IOException e1) {
272             JOptionPane.showMessageDialog(null, "ERROR AL CREAR DOCUMENTO PDF");
273         } catch (IOException e1) {
274             JOptionPane.showMessageDialog(null, "ERROR AL CREAR ID");
275         } catch (URISyntaxException e1) {
276             JOptionPane.showMessageDialog(null, "ERROR DE SINTAXIS");
277         } catch (SQLException e1) {
278             e1.printStackTrace();
279         }
280     }
281 }
282 );
283
284 
```

DAO NOVEDADES

En el Package de dao vamos a crear una clase llamada “daoNovedades” y quedaría así:

```
1 package dao;
2
3 import java.sql.PreparedStatement;
4
5 public class daoNovedades {
6     Conexion cx = null;
7     public daoNovedades() {
8         cx = new Conexion();
9     }
10    public boolean insertarnovedad(Novedades user) {
11        PreparedStatement ps = null;
12        try {
13            ps = cx.conectar().prepareStatement("INSERT INTO novedades VALUES(null,?)");
14            ps.executeUpdate();
15            return true;
16        } catch (SQLException e) {
17            e.printStackTrace();
18            return false;
19        } finally {
20            try {
21                ps.close();
22                ps = null;
23            } catch (SQLException e) {
24                System.out.println("ERROR AL CERRAR EDITAR USUARIO");
25                e.printStackTrace();
26            }
27        }
28    }
29    public ArrayList<Novedades> fetchnovedades() {
30        ArrayList<Novedades> lista = new ArrayList<Novedades>();
31        PreparedStatement ps = null;
32        ResultSet rs = null;
33        try {
34            ps = cx.conectar().prepareStatement("SELECT *FROM novedades");
35            rs = ps.executeQuery();
36            while (rs.next()) {
37                Novedades u = new Novedades();
38                u.setidnovedades(rs.getInt("idnovedad"));
39                u.setnovedades(rs.getString("Novedad"));
40                lista.add(u);
41            }
42        } catch (SQLException e) {
43            // TODO Auto-generated catch block
44            e.printStackTrace();
45        } finally {
46            try {
47                ps.close();
48                ps = null;
49                rs.close();
50                rs = null;
51            } catch (SQLException e) {
52                System.out.println("ERROR AL CERRAR EDITAR USUARIO");
53                e.printStackTrace();
54            }
55        }
56    }
57    public boolean EliminarNovedades(int Id) {
58        PreparedStatement ps = null;
59        try {
60            ps = cx.conectar().prepareStatement("DELETE FROM novedades WHERE idnovedad=?");
61            ps.setInt(1, Id);
62            ps.executeUpdate();
63            return true;
64        } catch (SQLException e) {
65            e.printStackTrace();
66        } finally {
67            try {
68                ps.close();
69                ps = null;
70            } catch (SQLException e) {
71                System.out.println("ERROR AL CERRAR EDITAR USUARIO");
72                e.printStackTrace();
73            }
74        }
75    }
76}
77
78
79
80
81
82    return true;
83 } catch (SQLException e) {
84     e.printStackTrace();
85     return false;
86 } finally {
87     try {
88         ps.close();
89         ps = null;
90         cx.desconectar();
91     } catch (SQLException e) {
92         System.out.println("ERROR AL CERRAR EDITAR USUARIO");
93         e.printStackTrace();
94     }
95 }
96
97
98
99    public boolean editarNovedades(Novedades user) {
100        PreparedStatement ps = null;
101        try {
102            ps = cx.conectar().prepareStatement("UPDATE novedades SET Novedad=? WHERE idNovedad=?");
103            ps.setString(1, user.getnovedades());
104            ps.setInt(2, user.getidnovedades());
105            ps.executeUpdate();
106            return true;
107        } catch (SQLException e) {
108            e.printStackTrace();
109            return false;
110        } finally {
111            try {
112                ps.close();
113                ps = null;
114                cx.desconectar();
115            } catch (SQLException e) {
116                System.out.println("ERROR AL CERRAR EDITAR USUARIO");
117                e.printStackTrace();
118            }
119        }
120 }
```

MODELO NOVEDADES

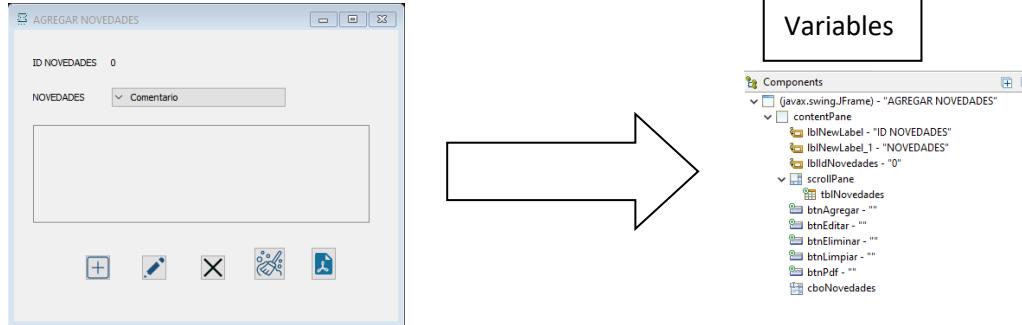
Vamos a crear una clase en el Package de Modelo la cuenta se llamará “Novedades” quedaría algo así:

```
1 package modelo;
2
3 public class Novedades {
4     int idnovedades;
5     String novedades;
6
7     public Novedades() {
8
9     }
10    public int getIdnovedades() {
11        return idnovedades;
12    }
13    public void setIdnovedades(int idnovedades) {
14        this.idnovedades = idnovedades;
15    }
16    public String getnovedades() {
17        return novedades;
18    }
19    public void setnovedades(String novedades) {
20        this.novedades = novedades;
21    }
22
23}
```

INTERFAZ DE NOVEDADES

Vamos a crear un JFrame en el Package de Vistas el JFrame y se llamará “vNovedades”.

Utilizaremos 3 JLabel, 1 JComboBox, 4 JButton, un JSpinner, un scrollPane, dentro del scrollPane vamos a agregar un JTable



PROGRAMAR LOS BOTONES

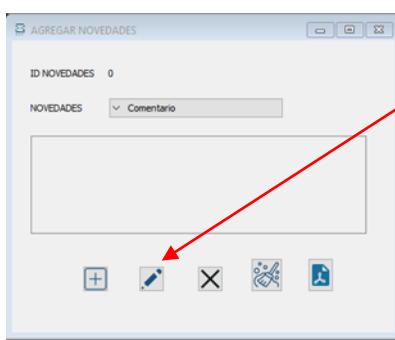
BOTON AGREGAR

Agregar el evento **ActionPerformed** al botón Insertar que se encuentra en **ApplicationWindow->vNovedades**, dando doble clic sobre el botón.



BOTON EDITAR

Vamos a dar doble clic en el botón Editar y vamos a repetir el mismo procedimiento que hicimos el Botón Agregar.



Doble clicaquí
El código que tiene que ir en el botón

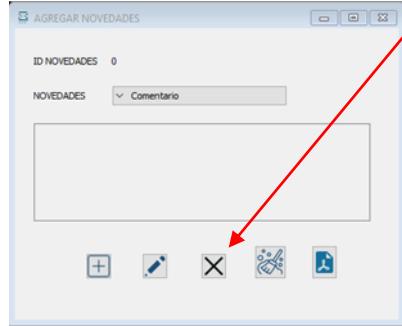
```

172     btnEditar = new JButton("");
173     btnEditar.addActionListener(new ActionListener() {
174         public void actionPerformed(ActionEvent e) {
175             try {
176                 novedades.setnovedades(""+cbNovedades.getSelectedItem());
177                 if (dao.editarnovedades(novedades)) {
178                     actualizarTabla();
179                     limpiar();
180                     JOptionPane.showMessageDialog(null, "SE ACTUALIZO CORRECTAMENTE");
181                 } else {
182                     JOptionPane.showMessageDialog(null, "ERROR");
183                 }
184             } catch (Exception e2) {
185                 JOptionPane.showMessageDialog(null, "ERROR");
186             }
187         }
188     });
189 });

```

BOTON ELIMINAR

Vamos a dar doble clic en el botón Editar y vamos a repetir el mismo procedimiento que hicimos el Botón Agregar.



Doble clic aquí
El código que tiene que ir en el botón

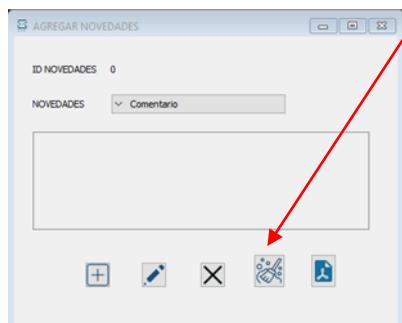
```

194     btnEliminar = new JButton("");
195     btnEliminar.addActionListener(new ActionListener() {
196         public void actionPerformed(ActionEvent e) {
197             try {
198                 int opcion = JOptionPane.showConfirmDialog(null, "ESTA SEGURO DE ELIMINAR LA NOVEDAD?", "ELIMINAR NOVEDAD", JOptionPane.YES_NO_OPTION);
199                 if (opcion == 0) {
200                     if (dao.eliminarNovedades(Lista.get(fila).getIdNovedades())) {
201                         actualizarTabla();
202                         limpiar();
203                         JOptionPane.showMessageDialog(null, "SE ELIMINO CORRECTAMENTE");
204                     } else {
205                         JOptionPane.showMessageDialog(null, "ERROR");
206                     }
207                 }
208             } catch (Exception ex) {
209                 JOptionPane.showMessageDialog(null, "ERROR");
210             }
211         }
212     });
213 });

```

BOTON LIMPIAR

Vamos a dar doble clic en el botón Editar y vamos a repetir el mismo procedimiento que hicimos el Botón Agregar.



Doble clic aquí
El código que tiene que ir en el botón

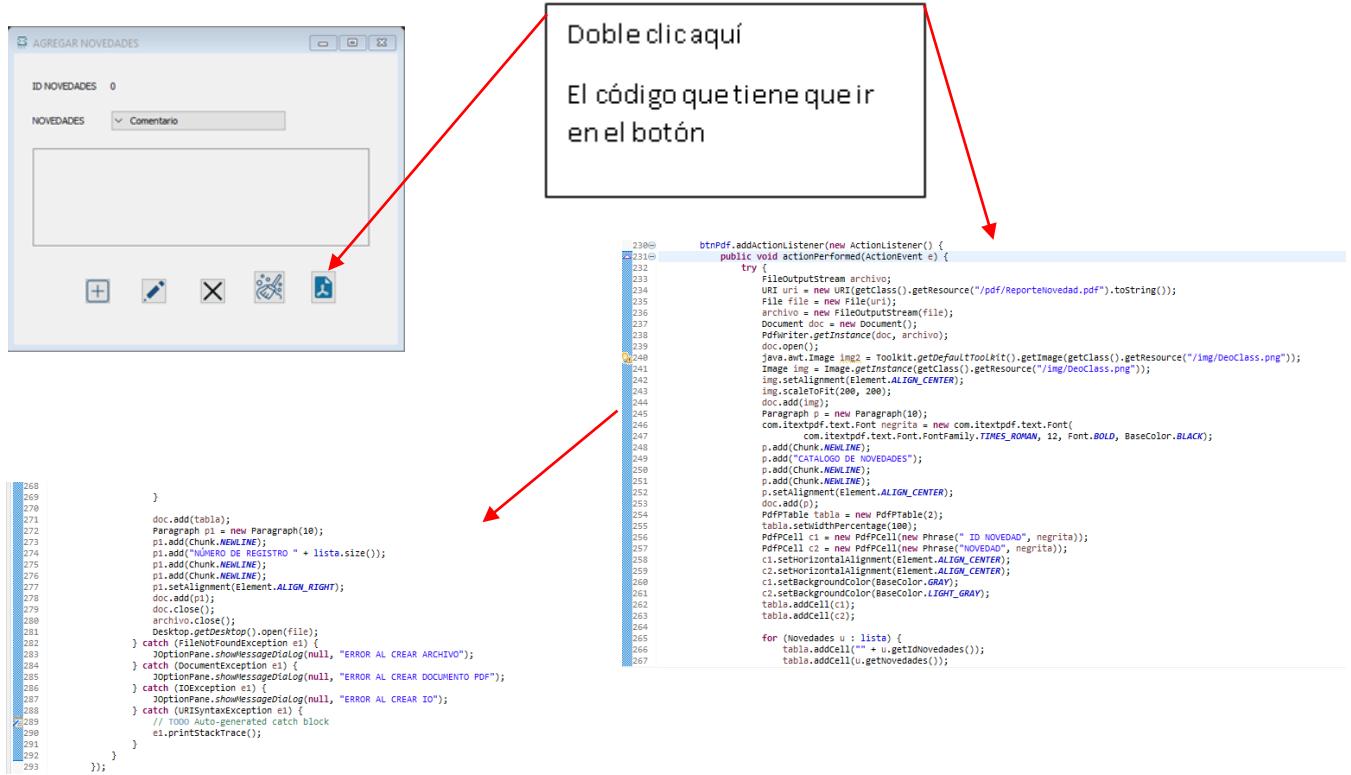
```

93     public void limpiar() {
94         lblIdNovedades.setText("");
95         cboNovedades.setSelectedItem("");
96     }

```

BOTON PDF

Tienes que dar doble clic en el botón PDF y vamos a repetir el mismo procedimiento que hemos hecho con los botones anteriores.



DAO MATERIAS

En el Package de dao vamos a crear una clase llamada "daoMaterias" y quedaría así:

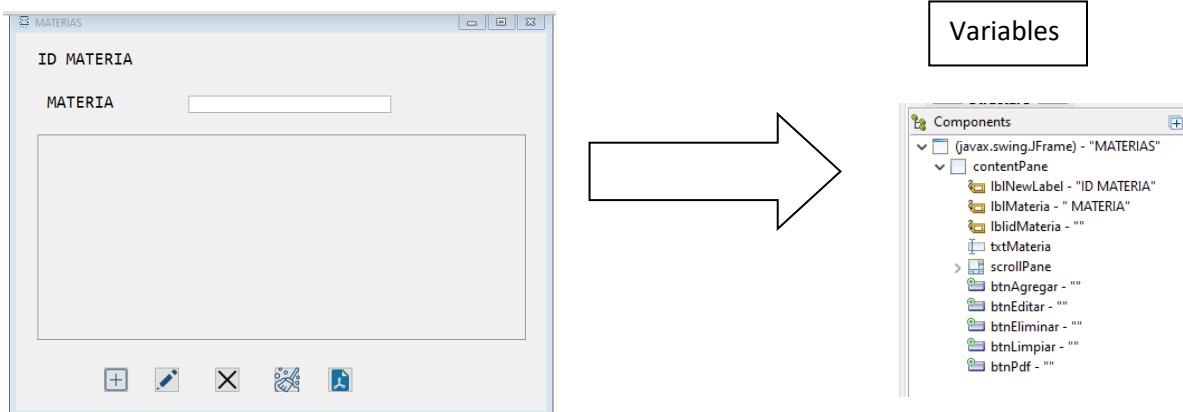
```

1 package dao;
2
3 import java.sql.PreparedStatement;
4
5 public class daomateria {
6     conexion cx = null;
7     public daomateria() {
8         cx = new conexion();
9     }
10
11    public boolean insertarMateria(Materias user) {
12        PreparedStatement ps = null;
13        try {
14            ps = cx.conectar().prepareStatement("INSERT INTO materias VALUES(null,?)");
15            ps.setString(1, user.getMateria());
16            ps.executeUpdate();
17            return true;
18        } catch (SQLException e) {
19            e.printStackTrace();
20            return false;
21        } finally {
22            try {
23                ps.close();
24                ps = null;
25            } catch (SQLException e) {
26                System.out.println("ERROR AL CERRAR EDITAR USUARIO");
27                e.printStackTrace();
28            }
29        }
30    }
31
32    public ArrayList<Materias> fetchMaterias() {
33        ArrayList<Materias> lista = new ArrayList<Materias>();
34        PreparedStatement ps = null;
35        ResultSet rs = null;
36        try {
37            ps = cx.conectar().prepareStatement("SELECT *FROM materias");
38            rs = ps.executeQuery();
39            while (rs.next()) {
40                Materias u = new Materias();
41                u.setIdmateria(rs.getInt("idMateria"));
42                u.setMateria(rs.getString("Materia"));
43                lista.add(u);
44            }
45        } catch (SQLException e) {
46            // This auto-generated catch block
47            e.printStackTrace();
48        } finally {
49            try {
50                ps.close();
51                ps = null;
52                rs.close();
53            } catch (SQLException e) {
54                System.out.println("ERROR AL CERRAR EDITAR USUARIO");
55                e.printStackTrace();
56            }
57        }
58    }
59
60    public boolean EliminarMateria(int ID) {
61        PreparedStatement ps = null;
62        try {
63            ps = cx.conectar().prepareStatement("DELETE FROM materias WHERE idMateria=?");
64            ps.setInt(1, ID);
65            ps.executeUpdate();
66            return true;
67        } catch (SQLException e) {
68            e.printStackTrace();
69        }
70    }
71
72
73    public boolean editarMateria(Materias user) {
74        PreparedStatement ps = null;
75        try {
76            ps = cx.conectar().prepareStatement("UPDATE materias SET materia=? WHERE idMateria=?");
77            ps.setString(1, user.getMateria());
78            ps.setInt(2, user.getIdMateria());
79            ps.executeUpdate();
80            return true;
81        } catch (SQLException e) {
82            System.out.println("ERROR AL CERRAR EDITAR USUARIO");
83            e.printStackTrace();
84        }
85    }
86
87    public boolean cerrarMateria(Materias user) {
88        PreparedStatement ps = null;
89        try {
90            ps = cx.conectar().prepareStatement("UPDATE materias SET materia=? WHERE idMateria=?");
91            ps.setString(1, user.getMateria());
92            ps.setInt(2, user.getIdMateria());
93            ps.executeUpdate();
94            return true;
95        } catch (SQLException e) {
96            System.out.println("ERROR AL CERRAR EDITAR USUARIO");
97            e.printStackTrace();
98        }
99    }
100
101    public boolean cerrarMateria() {
102        PreparedStatement ps = null;
103        try {
104            ps = cx.conectar().prepareStatement("UPDATE materias SET materia=? WHERE idMateria=?");
105            ps.setString(1, "CERRADA");
106            ps.setInt(2, 1);
107            ps.executeUpdate();
108            return true;
109        } catch (SQLException e) {
110            System.out.println("ERROR AL CERRAR EDITAR USUARIO");
111            e.printStackTrace();
112        }
113    }
114
115    public boolean cerrarMateria(int id) {
116        PreparedStatement ps = null;
117        try {
118            ps = cx.conectar().prepareStatement("UPDATE materias SET materia=? WHERE idMateria=?");
119            ps.setString(1, "CERRADA");
120            ps.setInt(2, id);
121            ps.executeUpdate();
122            return true;
123        } catch (SQLException e) {
124            System.out.println("ERROR AL CERRAR EDITAR USUARIO");
125            e.printStackTrace();
126        }
127    }
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
617
618
619
619
620
621
622
623
623
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523

```

Vamos a crear un JFrame en el Package de Vistas el JFrame y se llamará "vMateria".

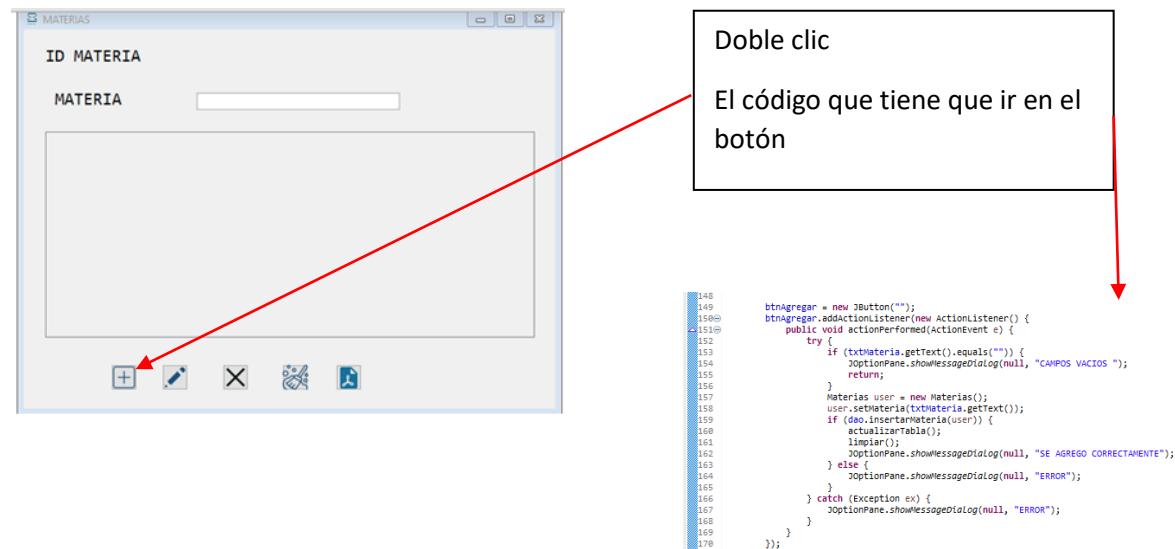
Utilizaremos 3 JLabel, 1 JComboBox, 4 JButton, un JSpinner, un scrollPane, dentro del scrollPane vamos a agregar un JTable



PROGRAMAR BOTONES

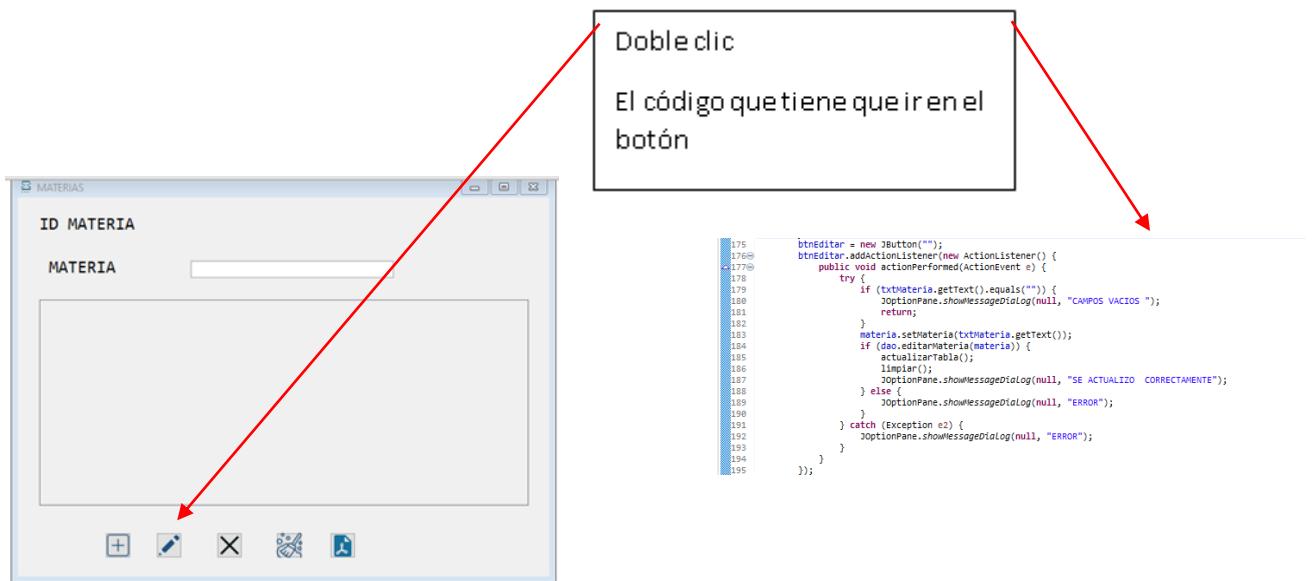
BOTON AGREGAR

Agregar el evento **ActionPerformed** al botón Insertar que se encuentra en **ApplicationWindow->vMaterias**, dando doble clic sobre el botón.



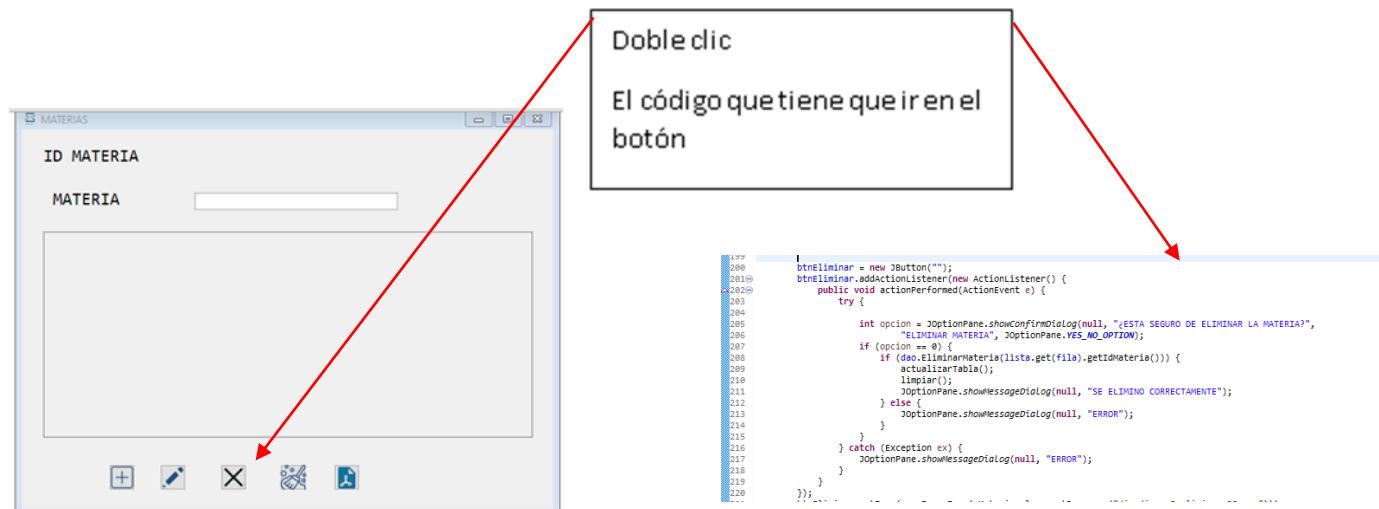
BOTON EDITAR

Vamos a dar doble clic en el botón Editar y vamos a repetir el mismo procedimiento que hicimos el Botón Agregar.



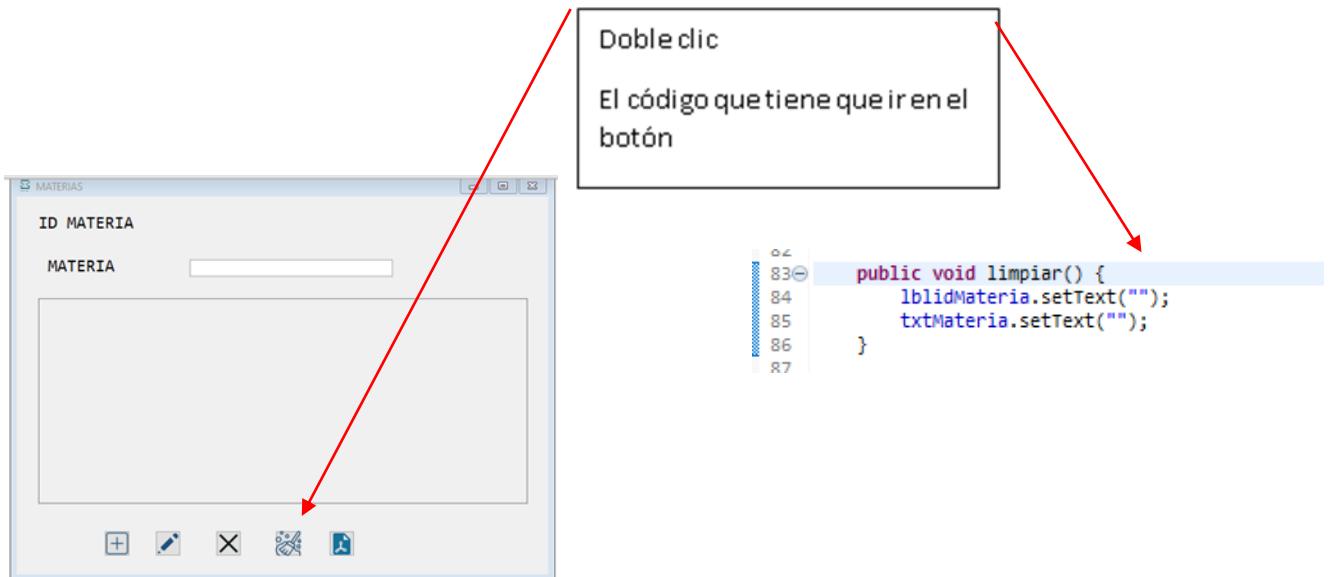
BOTON ELIMINAR

Vamos a dar doble clic en el botón Eliminar y vamos a repetir el mismo procedimiento que hicimos el Botón Agregar.



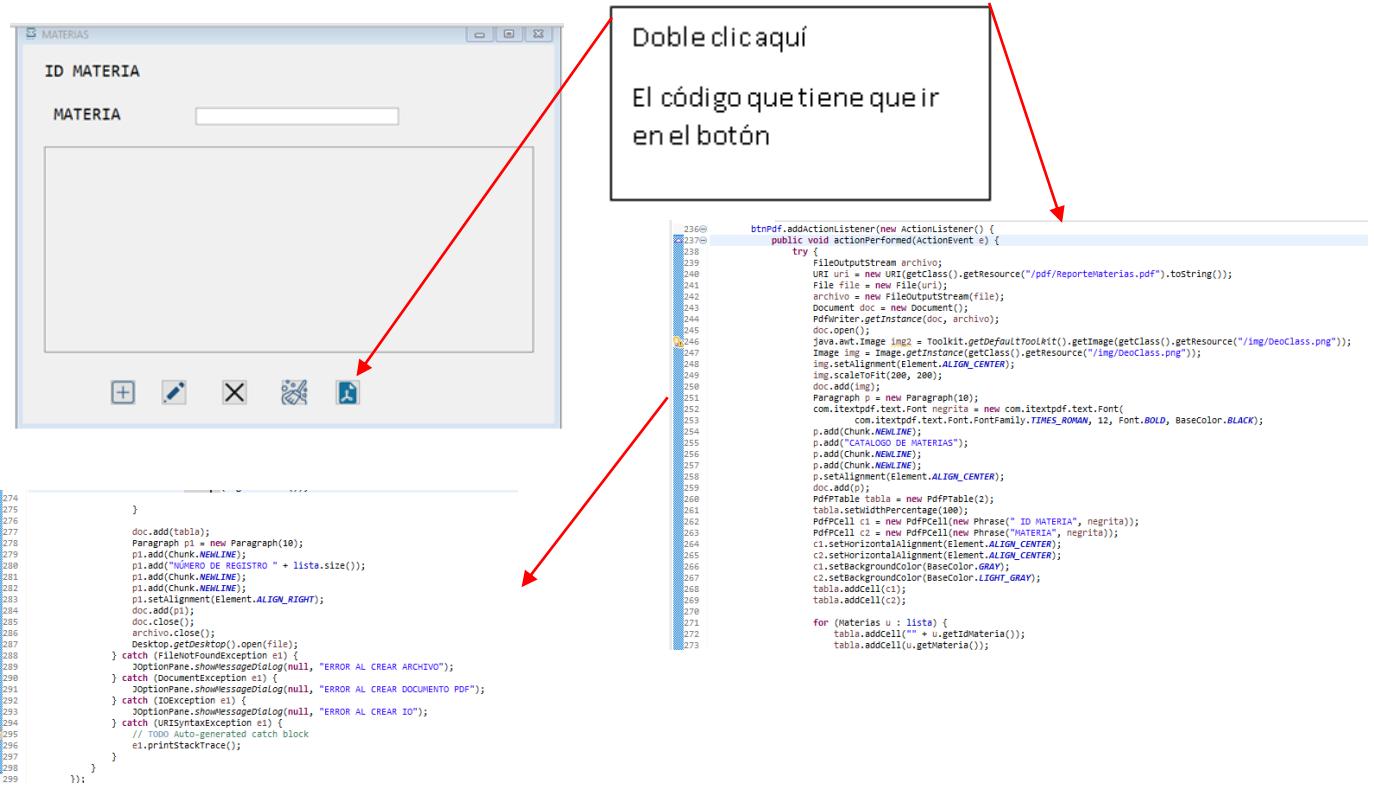
BOTON LIMPIAR

Vamos a dar doble clic en el botón Eliminar y vamos a repetir el mismo procedimiento que hicimos el Botón Agregar.



BOTON PDF

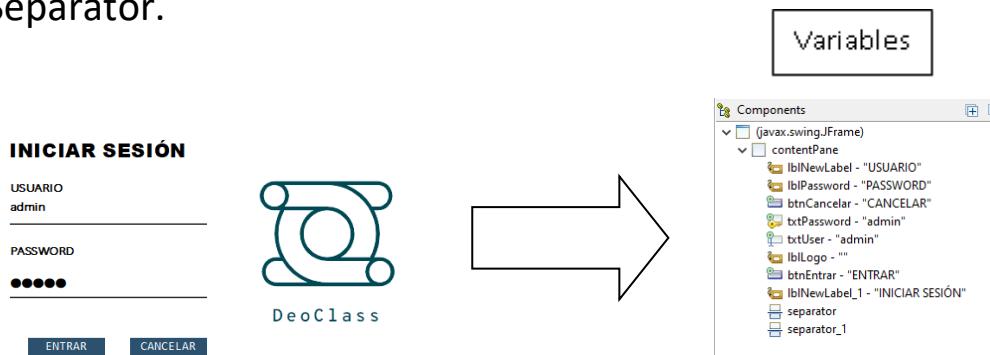
Tienes que dar doble clic en el botón PDF y vamos a repetir el mismo procedimiento que hemos hecho con los botones anteriores.



INTERFAZ LOGIN

Vamos a crear un JFrame en el Package de Vistas el JFrame y se llamará "vLogin".

Utilizaremos 4 JLabel, 1 JTextField, 2 JButton, Vamos a usar 2 JSeparator.



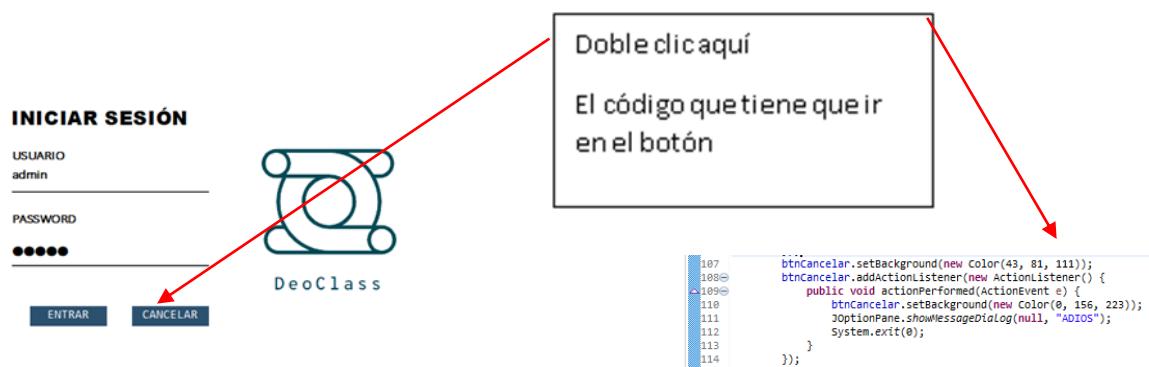
BOTON ENTRAR

Vamos a dar doble clic sobre el botón y vamos a poner el código que se dará a continuación.



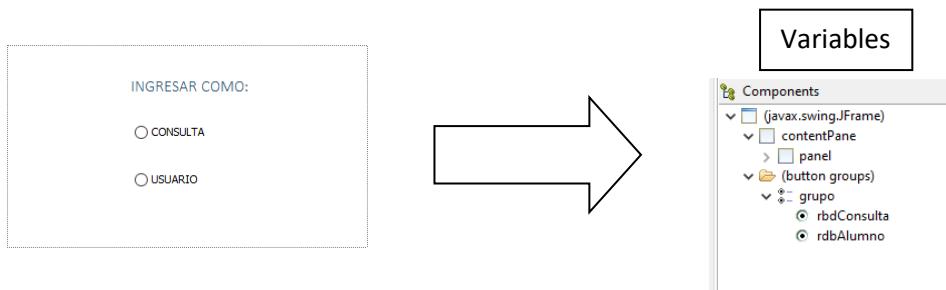
BOTON CANCELAR

Vamos a dar doble clic sobre el botón y vamos a poner el código que se dará a continuación.



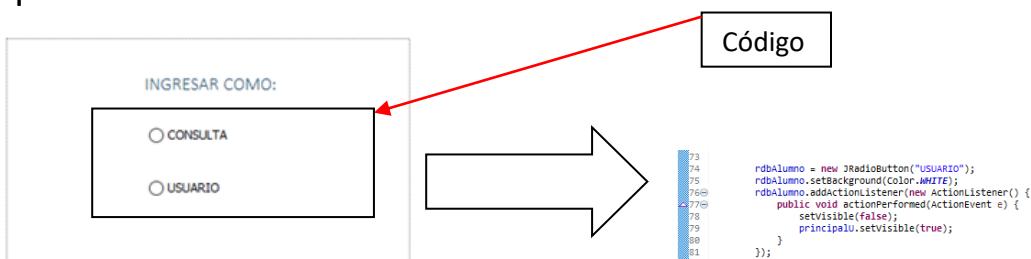
INTERFAZ DE INGRESO

Vamos a crear un JFrame en el Package de Vistas el JFrame y se llamará “vIngreso”.



PROGRAMAR LOS BUTTON GROUPS

Vamos a dar doble clic sobre el botón y vamos a poner el código que se dará a continuación.



DAO GRUPO

En el Package de dao vamos a crear una clase llamada “daoGrupo” y quedaría así:

```
1 package dao;
2
3 import java.sql.PreparedStatement;
4
5 public class dao {
6     conex = null;
7     public daoGrupo() {
8         cx = new conexion();
9     }
10
11     public boolean insertarGrupo(Grupo user) {
12         PreparedStatement ps = null;
13         try {
14             ps = cx.conectar().prepareStatement("INSERT INTO grupo VALUES(null,?)");
15             ps.setInt(1, user.getIdGrupo());
16             ps.executeUpdate();
17             return true;
18         } catch (SQLException e) {
19             e.printStackTrace();
20             return false;
21         } finally {
22             try {
23                 ps.close();
24                 ps = null;
25             } catch (SQLException e) {
26                 System.out.println("ERROR AL CERRAR EDITAR USUARIO");
27                 e.printStackTrace();
28             }
29         }
30     }
31
32     public ArrayList<Grupo> fetchGrupos() {
33         ArrayList<Grupo> lista = new ArrayList<Grupo>();
34         PreparedStatement ps = null;
35         ResultSet rs = null;
36         try {
37             ps = cx.conectar().prepareStatement("SELECT *FROM grupo");
38             rs = ps.executeQuery();
39             while (rs.next()) {
40                 Grupo u = new Grupo();
41                 u.setIdGrupo(rs.getInt("idGrupo"));
42                 u.setGrupo(rs.getString("grupo"));
43                 lista.add(u);
44             }
45         } catch (SQLException e) {
46             // TODO Auto-generated catch block
47             e.printStackTrace();
48         } finally {
49             try {
50                 ps.close();
51                 ps = null;
52                 rs.close();
53                 rs = null;
54             } catch (SQLException e) {
55                 System.out.println("ERROR AL CERRAR EDITAR USUARIO");
56                 e.printStackTrace();
57             }
58         }
59     }
60
61     public boolean EliminarGrupo(int ID) {
62         PreparedStatement ps = null;
63         try {
64             ps = cx.conectar().prepareStatement("DELETE FROM grupo WHERE idGrupo=?");
65             ps.setInt(1, ID);
66             ps.executeUpdate();
67             return true;
68         } catch (SQLException e) {
69             e.printStackTrace();
70             return false;
71         } finally {
72             try {
73                 ps.close();
74                 ps = null;
75             } catch (SQLException e) {
76                 System.out.println("ERROR AL CERRAR EDITAR USUARIO");
77                 e.printStackTrace();
78             }
79         }
80     }
81
82     public void editarGrupo(Grupo user) {
83         PreparedStatement ps = null;
84         try {
85             ps = cx.conectar().prepareStatement("UPDATE grupo SET grupo=? WHERE idGrupo=?");
86             ps.setInt(1, user.getIdGrupo());
87             ps.setInt(2, user.getGrupo());
88             ps.executeUpdate();
89             cx.desconectar();
90         } catch (SQLException e) {
91             System.out.println("ERROR AL CERRAR EDITAR USUARIO");
92             e.printStackTrace();
93         }
94     }
95
96     public boolean editarGrupo(Grupo user) {
97         PreparedStatement ps = null;
98         try {
99             ps = cx.conectar().prepareStatement("UPDATE grupo SET grupo=? WHERE idGrupo=?");
100            ps.setInt(1, user.getIdGrupo());
101            ps.setInt(2, user.getGrupo());
102            ps.executeUpdate();
103            return true;
104        } catch (SQLException e) {
105            e.printStackTrace();
106            return false;
107        } finally {
108            try {
109                ps.close();
110                ps = null;
111                cx.desconectar();
112            } catch (SQLException e) {
113                System.out.println("ERROR AL CERRAR EDITAR USUARIO");
114                e.printStackTrace();
115            }
116        }
117    }
118 }
```

MODELO GRUPO

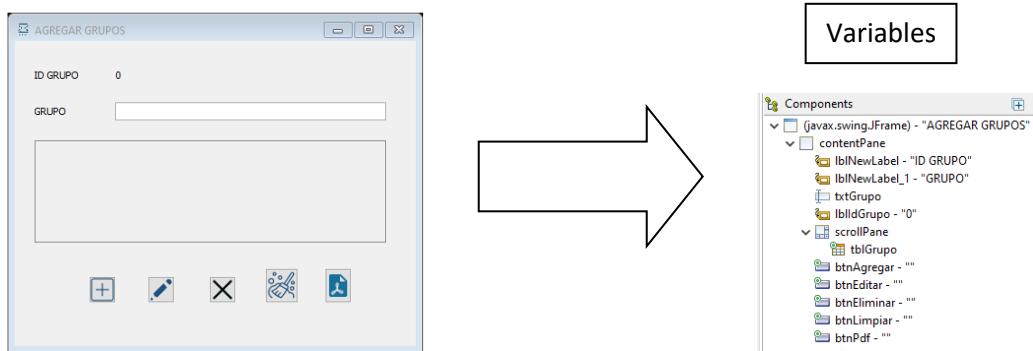
Vamos a crear una clase en el Package de Modelo la cuenta se llamará “Materia” quedaría algo así:

```
1 package modelz;
2
3 public class Grupo {
4     int idGrupo;
5     int Grupo;
6
7     public Grupo() {
8
9     }
10
11     public int getIdGrupo() {
12         return idGrupo;
13     }
14
15     public void setIdGrupo(int idGrupo) {
16         this.idGrupo = idGrupo;
17     }
18
19     public int getGrupo() {
20         return Grupo;
21     }
22
23     public void setGrupo(int grupo) {
24         Grupo = grupo;
25     }
26
27 }
```

INTERFAZ DE GRUPO

Vamos a crear un JFrame en el Package de Vistas el JFrame y se llamará "vGrupo".

Utilizaremos 3 JLabel, 1 JTextField, 5 JButton, un JSpinner, un scrollPane, dentro del scrollPane vamos a agregar un JTable



PROGRAMAR LOS BOTONES

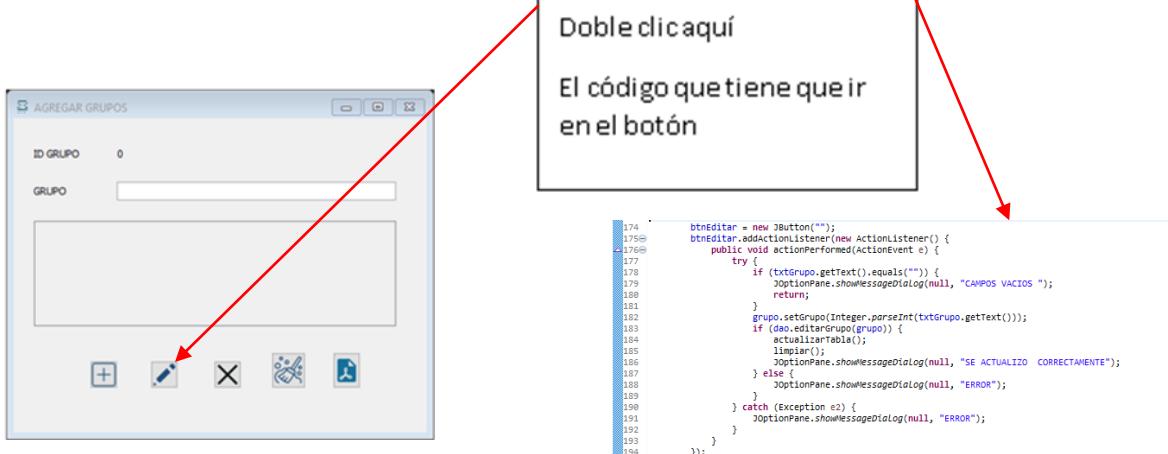
BOTON AGREGAR

Agregar el evento **ActionPerformed** al botón Insertar que se encuentra en **ApplicationWindow->vGrupo**, dando doble clic sobre el botón.



BOTON EDITAR

Vamos a dar doble clic en el botón Editar y vamos a repetir el mismo procedimiento que hicimos el Botón Agregar.



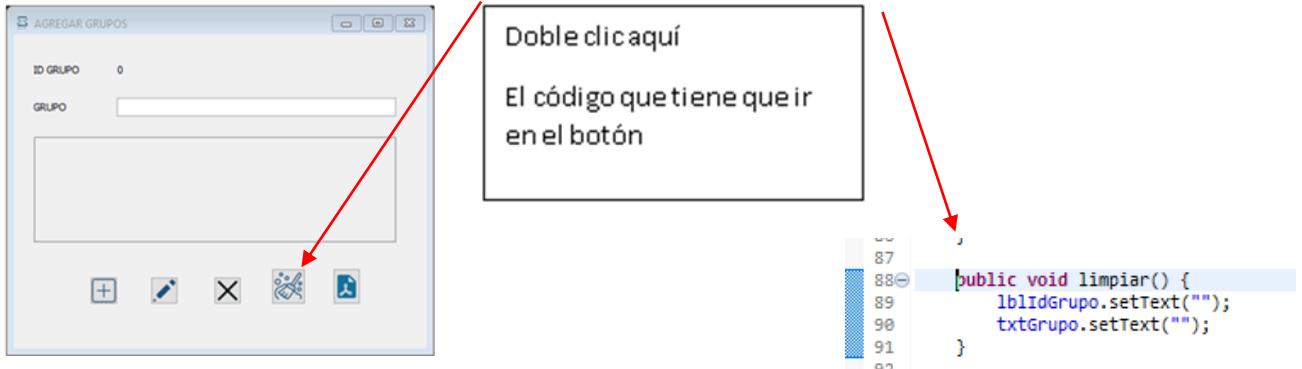
BOTON ELIMINAR

Vamos a dar doble clic en el botón Eliminar y vamos a repetir el mismo procedimiento que hicimos el Botón Agregar.



BOTON LIMPIAR

Vamos a dar doble clic en el botón Limpiar y vamos a repetir el mismo procedimiento que hicimos el Botón Agregar.



BOTON PDF

Tienes que dar doble clic en el botón PDF y vamos a repetir el mismo procedimiento que hemos hecho con los botones anteriores.



DAO FOTO

En el Package de dao vamos a crear una clase llamada “daoFoto” y quedaría así:

```
1 package dao;
2
3 import java.security.MessageDigest[];
4
5 public class DaoFoto {
6     Conexion cx = null;
7     public DaoFoto() {
8         cx = new Conexion();
9     }
10
11    public boolean insertarFoto(Foto user) {
12        PreparedStatement ps = null;
13        try {
14            ps = cx.conectar().prepareStatement("INSERT INTO foto VALUES(null,?)");
15            ps.setString(1, user.getFoto());
16            ps.executeUpdate();
17            return true;
18        } catch (SQLException e) {
19            e.printStackTrace();
20            return false;
21        } finally {
22            try {
23                ps.close();
24                ps = null;
25            } catch (SQLException e) {
26                System.out.println("ERROR AL CERRAR EDITAR USUARIO");
27                e.printStackTrace();
28            }
29        }
30    }
31
32    public ArrayList<Foto> getFotos() {
33        ArrayList<Foto> lista = new ArrayList<Foto>();
34        PreparedStatement ps = null;
35        ResultSet rs = null;
36        try {
37            ps = cx.conectar().prepareStatement("SELECT *FROM foto");
38            rs = ps.executeQuery();
39
40            while (rs.next()) {
41                Foto u = new Foto();
42                u.setIdFoto(rs.getInt("idFoto"));
43                u.setFoto(rs.getString("foto"));
44                lista.add(u);
45            }
46        } catch (SQLException e) {
47            // TODO Auto-generated catch block
48            e.printStackTrace();
49        } finally {
50            try {
51                ps.close();
52                ps = null;
53            } catch (SQLException e) {
54                System.out.println("ERROR AL CERRAR EDITAR USUARIO");
55                e.printStackTrace();
56            }
57        }
58    }
59
60    public boolean eliminarFoto(int id) {
61        PreparedStatement ps = null;
62        try {
63            ps = cx.conectar().prepareStatement("DELETE FROM foto WHERE idFoto=?");
64            ps.setInt(1, id);
65            ps.executeUpdate();
66            return true;
67        } catch (SQLException e) {
68            e.printStackTrace();
69            return false;
70        } finally {
71            try {
72                ps.close();
73                ps = null;
74            } catch (SQLException e) {
75                System.out.println("ERROR AL CERRAR EDITAR USUARIO");
76                e.printStackTrace();
77            }
78        }
79    }
80
81    public boolean editarFoto(Foto user) {
82        PreparedStatement ps = null;
83        try {
84            ps = cx.conectar().prepareStatement("UPDATE foto SET foto=? WHERE idFoto=?");
85            ps.setString(1, user.getFoto());
86            ps.setInt(2, user.getIdFoto());
87            ps.executeUpdate();
88            return true;
89        } catch (SQLException e) {
90            e.printStackTrace();
91            return false;
92        } finally {
93            try {
94                ps.close();
95                ps = null;
96            } catch (SQLException e) {
97                System.out.println("ERROR AL CERRAR EDITAR USUARIO");
98                e.printStackTrace();
99            }
100        }
101    }
102
103    public boolean setFoto(Foto user) {
104        PreparedStatement ps = null;
105        try {
106            ps = cx.conectar().prepareStatement("UPDATE foto SET foto=? WHERE idFoto=?");
107            ps.setString(1, user.getFoto());
108            ps.setInt(2, user.getIdFoto());
109            ps.executeUpdate();
110            return true;
111        } catch (SQLException e) {
112            e.printStackTrace();
113            return false;
114        } finally {
115            try {
116                ps.close();
117                ps = null;
118            } catch (SQLException e) {
119                System.out.println("ERROR AL CERRAR EDITAR USUARIO");
120                e.printStackTrace();
121            }
122        }
123    }
124}
```

MODELO FOTO

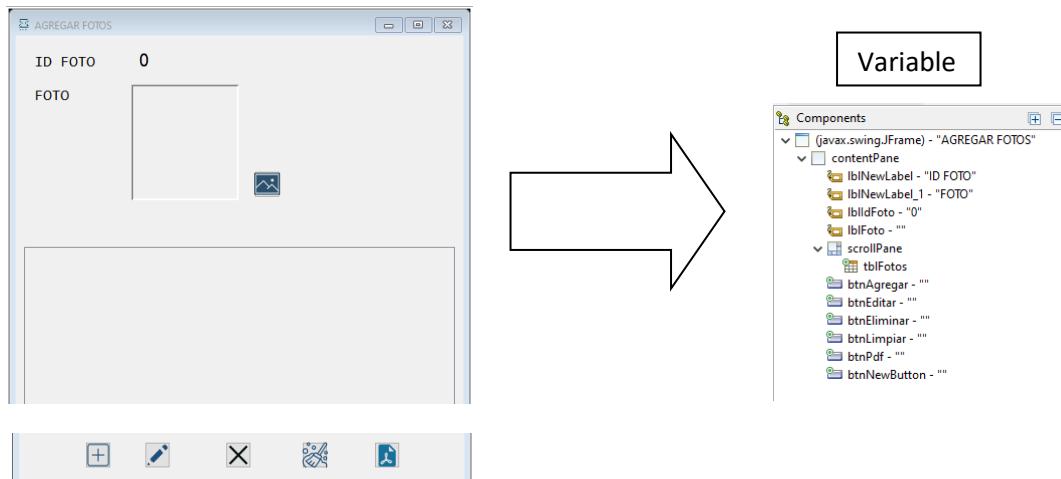
Vamos a crear una clase en el Package de Modelo la cuenta se llamará “Foto” quedaría algo así:

```
1 package modelo;
2
3 public class Foto {
4     int idFoto;
5     String foto;
6
7     public Foto() {
8
9     }
10
11    public int getIdFoto() {
12        return idFoto;
13    }
14
15    public void setIdFoto(int idFoto) {
16        this.idFoto = idFoto;
17    }
18
19    public String getFoto() {
20        return foto;
21    }
22
23    public void setFoto(String foto) {
24        this.foto = foto;
25    }
26
27 }
```

INTERFAZ DE FOTO

Vamos a crear un JFrame en el Package de Vistas el JFrame y se llamará "vFoto".

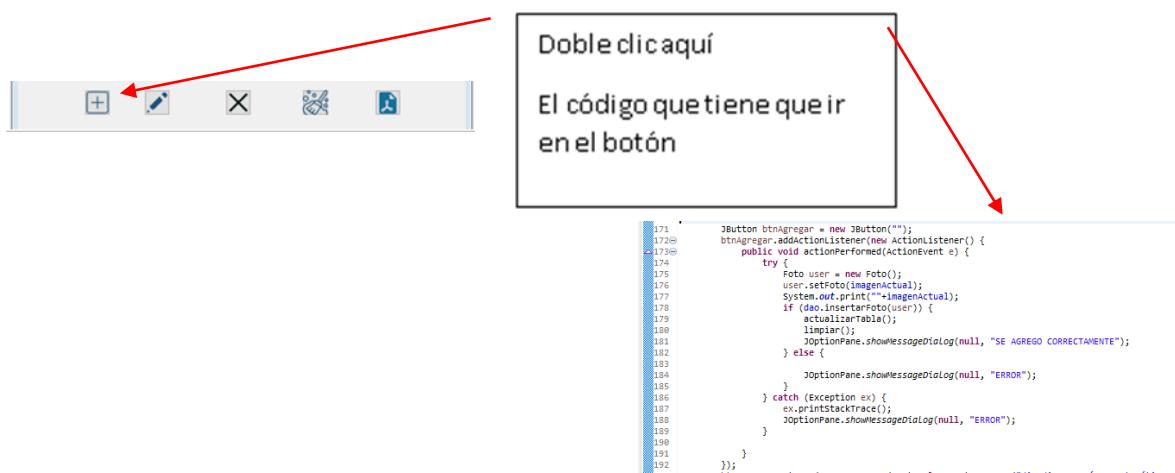
Utilizaremos 4 JLabel, 6 JButton, un JSpinner, un scrollPane, dentro del scrollPane vamos a agregar un JTable



PROGRAMAR BOTONES

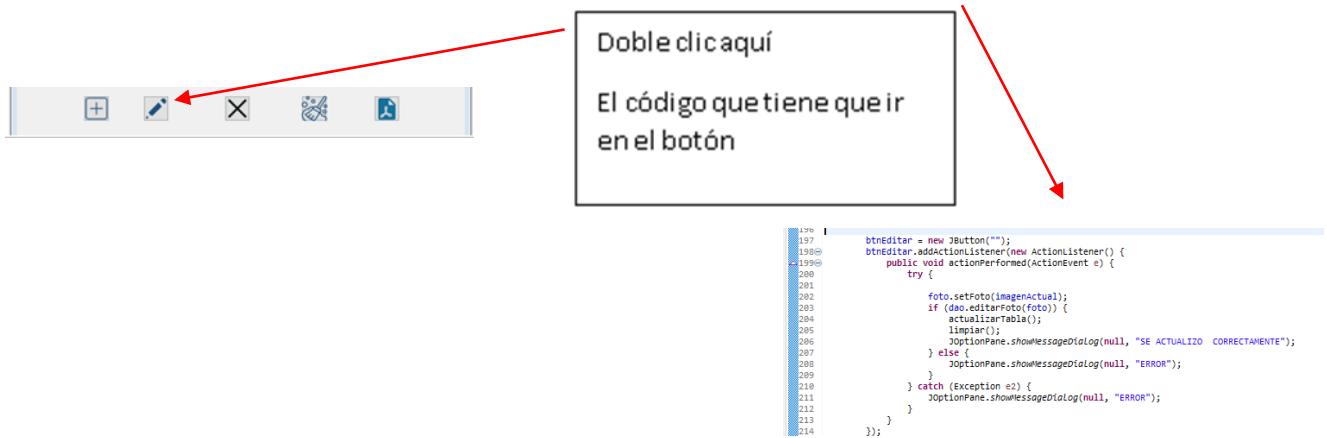
BOTÓN AGREGAR

Agregar el evento **ActionPerformed** al botón Insertar que se encuentra en **ApplicationWindow->vFoto**, dando doble clic sobre el botón.



BOTON EDITAR

Vamos a dar doble clic en el botón Editar y vamos a repetir el mismo procedimiento que hicimos el Botón Agregar.



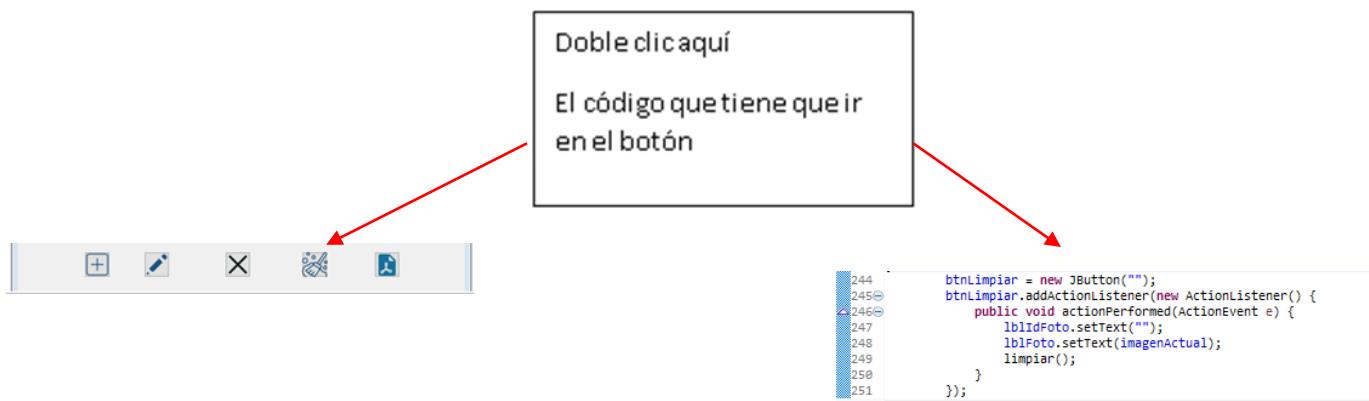
BOTON ELIMINAR

Vamos a dar doble clic en el botón Eliminar y vamos a repetir el mismo procedimiento que hicimos el Botón Agregar.



BOTON LIMPIAR

Vamos a dar doble clic en el botón Limpiar y vamos a repetir el mismo procedimiento que hicimos el Botón Agregar.



BOTON PDF

Tienes que dar doble clic en el botón PDF y vamos a repetir el mismo procedimiento que hemos hecho con los botones anteriores.



DAO ENTREGA

En el Package de dao vamos a crear una clase llamada “daoEntrega” y quedaría así:

```

1 package dao;
2
3 import java.sql.PreparedStatement;
4
5 public class daoenrega {
6     conexion cx = null;
7
8     public daoenrega() {
9         cx = new conexion();
10    }
11
12    public boolean insertarEntrega(Entrega user) {
13        PreparedStatement ps = null;
14        try {
15            ps = cx.conectar().prepareStatement("INSERT INTO entrega VALUES(null,?, ?, ?)");
16            ps.setString(1, user.getClase());
17            ps.setString(2, user.getActividad());
18            ps.setString(3, user.getFecha());
19            ps.setString(4, user.getEntrega());
20            ps.executeUpdate();
21        } catch (SQLException e) {
22            e.printStackTrace();
23        } finally {
24            try {
25                ps.close();
26                ps = null;
27            } catch (SQLException e) {
28                System.out.println("ERROR AL CERRAR EDITAR USUARIO");
29                e.printStackTrace();
30            }
31        }
32    }
33
34    public ArrayList<Entrega> fetcEntregas() {
35        ArrayList<Entrega> lista = new ArrayList<Entrega>();
36        PreparedStatement ps = null;
37
38        ResultSet rs = null;
39        try {
40            ps = cx.conectar().prepareStatement("SELECT *FROM entrega");
41            rs = ps.executeQuery();
42            while (rs.next()) {
43                Entrega user = new Entrega();
44                user.setIdEntrega(rs.getInt("idEntrega"));
45                user.setClase(rs.getString("clase"));
46                user.setActividad(rs.getString("actividad"));
47                user.setFecha(rs.getString("fecha"));
48                user.setEntrega(rs.getString("entrega"));
49                lista.add(user);
50            }
51        } catch (SQLException e) {
52            // TODO Auto-generated catch block
53            e.printStackTrace();
54        } finally {
55            try {
56                ps.close();
57                ps = null;
58            } catch (SQLException e) {
59                cx.desconectar();
60            }
61        }
62    }
63
64    public boolean EliminarEntrega(int id) {
65        PreparedStatement ps = null;
66        try {
67            ps = cx.conectar().prepareStatement("DELETE FROM entrega WHERE idEntrega=?");
68            ps.setInt(1, id);
69            ps.executeUpdate();
70        } catch (SQLException e) {
71            e.printStackTrace();
72        }
73    }
74
75    public boolean editarEntrega(Entrega user) {
76        PreparedStatement ps = null;
77        try {
78            ps = cx.conectar().prepareStatement("UPDATE entrega SET clase=?,actividad=?,fecha=? WHERE idEntrega=?");
79            ps.setString(1, user.getClase());
80            ps.setString(2, user.getActividad());
81            ps.setString(3, user.getFecha());
82            ps.setInt(4, user.getIdEntrega());
83            ps.executeUpdate();
84        } catch (SQLException e) {
85            System.out.println("ERROR AL CERRAR EDITAR USUARIO");
86            e.printStackTrace();
87        }
88    }
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125

```

MODELO DE ENTREGA

Vamos a crear una clase en el Package de Modelo la cuenta se llamará “Entrega” quedaría algo así:

```

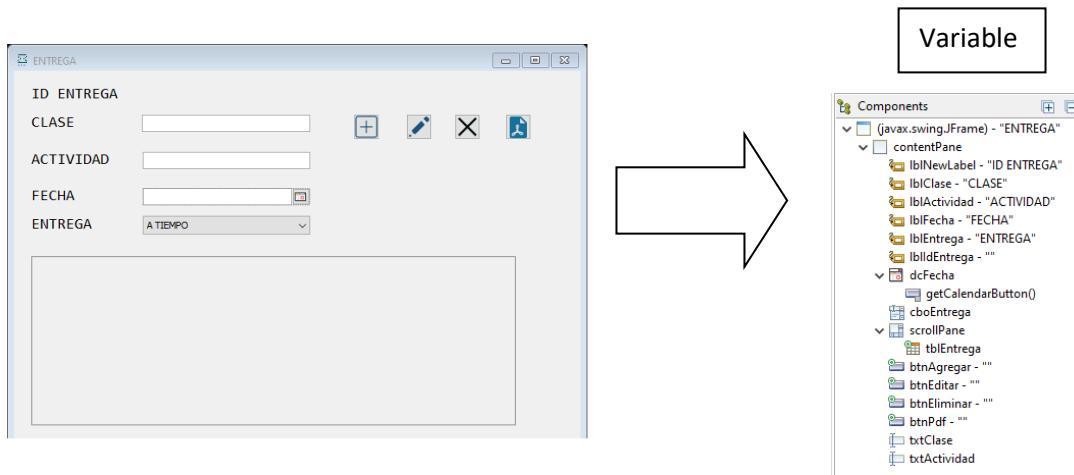
1 package modelo;
2
3 public class Entrega {
4     int idEntrega;
5     String clase;
6     String actividad;
7     String fecha;
8     String entrega;
9
10
11    public Entrega() {
12    }
13
14
15    public int getIdEntrega() {
16        return idEntrega;
17    }
18
19
20    public void setIdEntrega(int idEntrega) {
21        this.idEntrega = idEntrega;
22    }
23
24
25    public String getClass() {
26        return clase;
27    }
28
29
30    public void setClass(String clase) {
31        this.clase = clase;
32    }
33
34
35    public String getActividad() {
36        return actividad;
37    }
38
39
40
41
42
43
44
45
46    public void setActividad(String actividad) {
47        this.actividad = actividad;
48    }
49
50
51    public String getFecha() {
52        return fecha;
53    }
54
55
56    public void setFecha(String fecha) {
57        this.fecha = fecha;
58    }
59
60
61    public String getEntrega() {
62        return entrega;
63    }
64
65
66
67    public void setEntrega(String entrega) {
68        this.entrega = entrega;
69    }
70
71
72
73
74
75
76
77
78
79
7
80
81
82
83
84
85
86
87
88
89
8
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
10
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125

```

INTERFAZ DE ENTREGA

Vamos a crear un JFrame en el Package de Vistas el JFrame y se llamará "vEntrega".

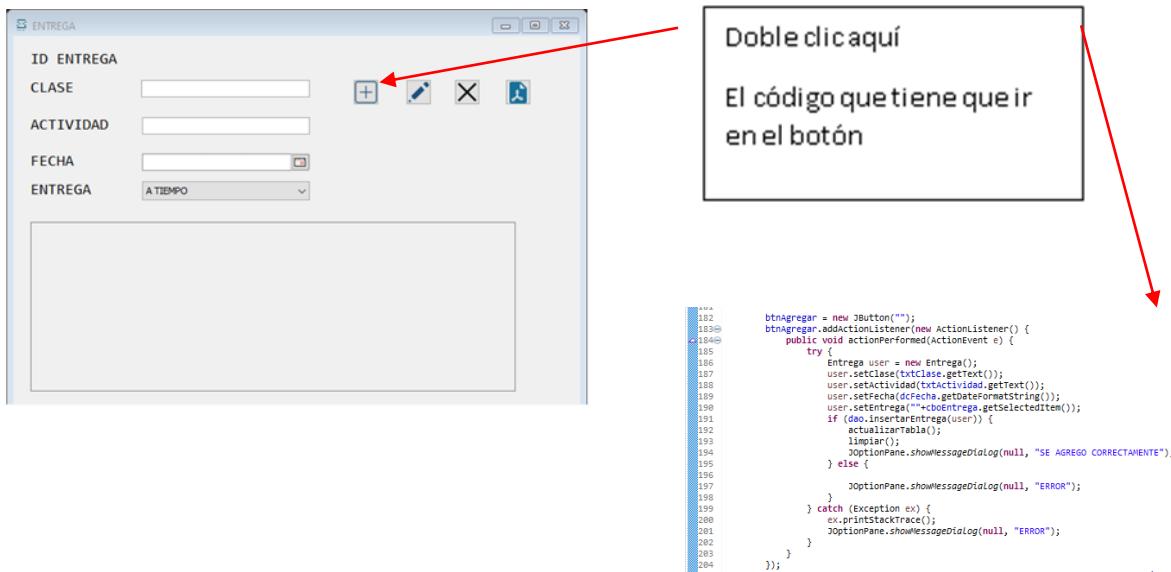
Utilizaremos 6 JLabel, 4 JButton, 2 JTextField, 1 JComboBox, un scrollPane, dentro del scrollPane vamos a agregar un JTable



PROGRAMAR LOS BOTONES

BOTON AGREGAR

Agregar el evento **ActionPerformed** al botón Insertar que se encuentra en ApplicationWindow->vEntrega, dando doble clic sobre el botón.



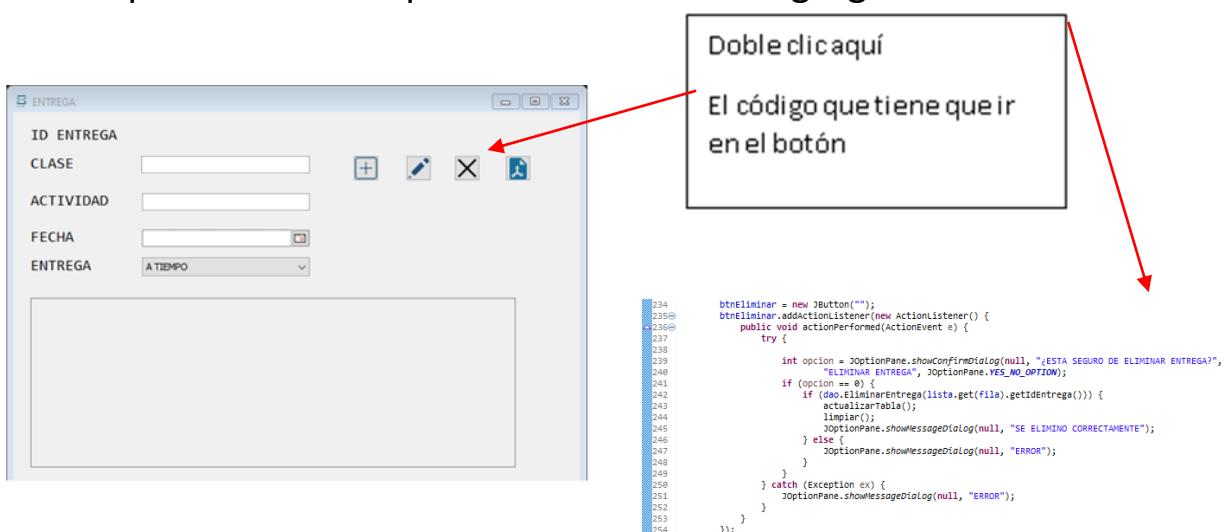
BOTON EDITAR

Vamos a dar doble clic en el botón Editar y vamos a repetir el mismo procedimiento que hicimos el Botón Agregar.



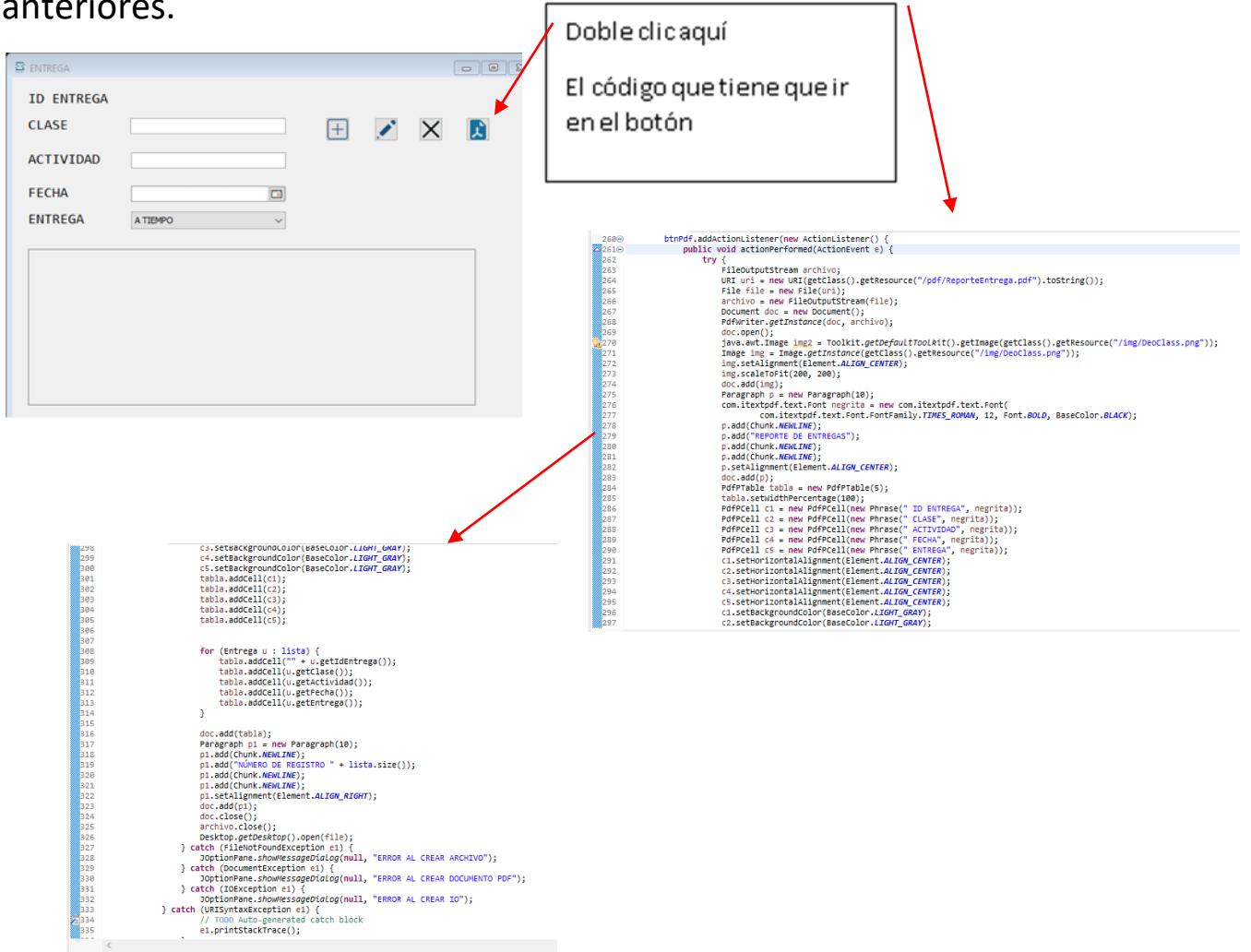
BOTON ELIMINAR

Vamos a dar doble clic en el botón Eliminar y vamos a repetir el mismo procedimiento que hicimos el Botón Agregar.



BOTON PDF

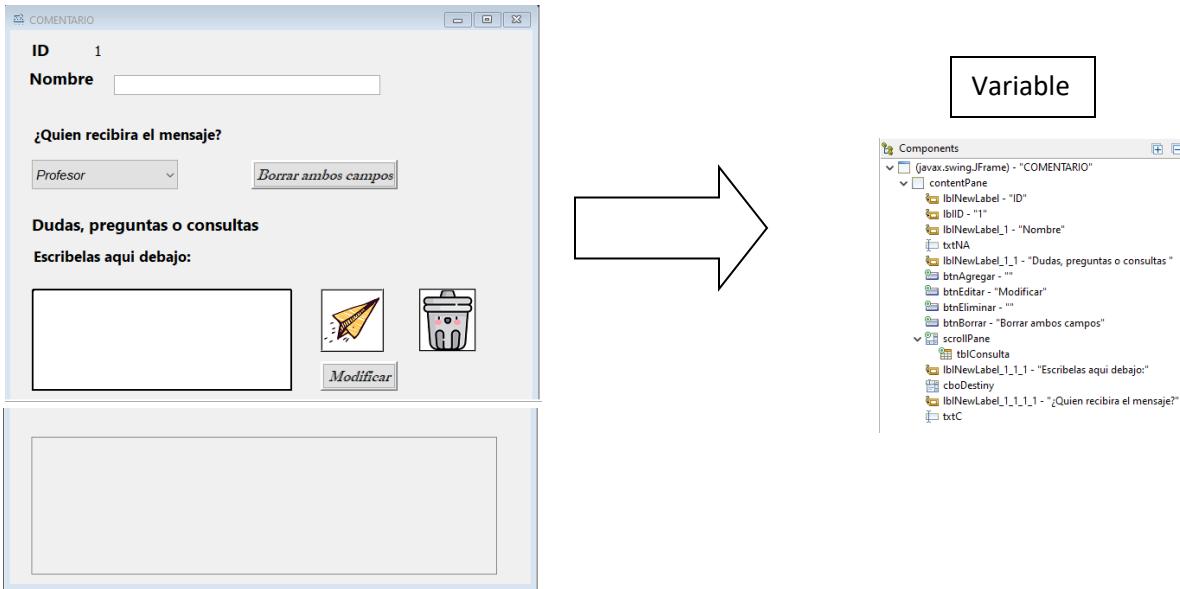
Tienes que dar doble clic en el botón PDF y vamos a repetir el mismo procedimiento que hemos hecho con los botones anteriores.



INTERFAZ DE COMENTARIO

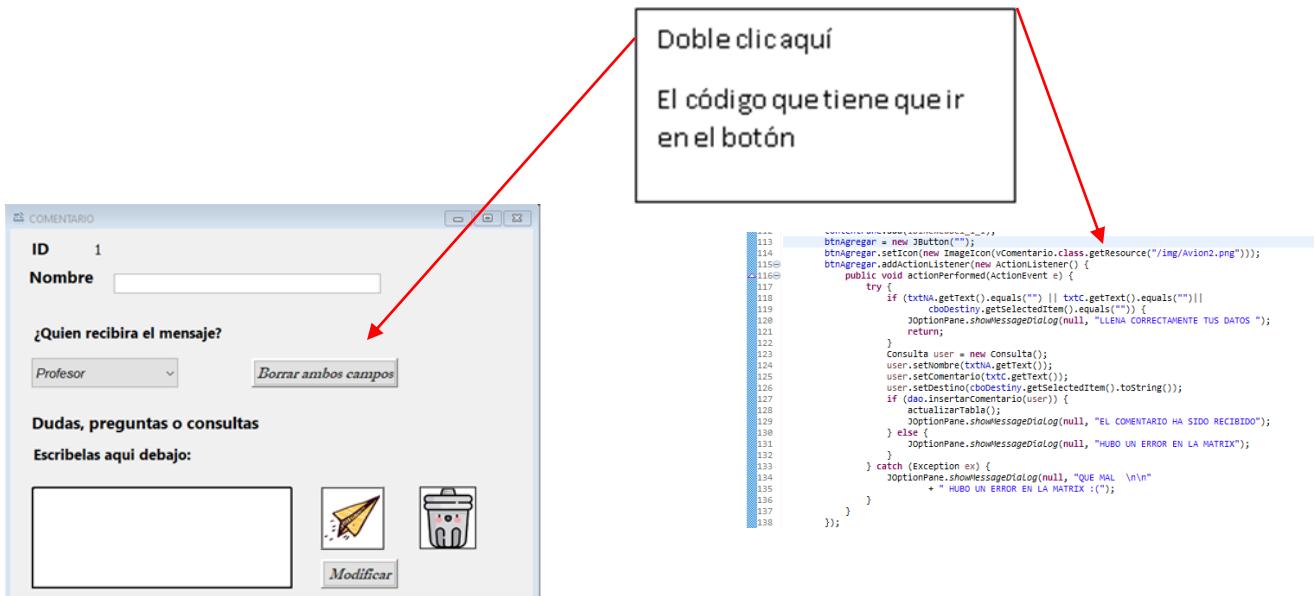
Vamos a crear un JFrame en el Package de Vistas el JFrame y se llamará "vComentario".

Utilizaremos 6 JLabel, 4 JButton, 2 JTextField, 1 JComboBox, un scrollPane, dentro del scrollPane vamos agregar un JTable



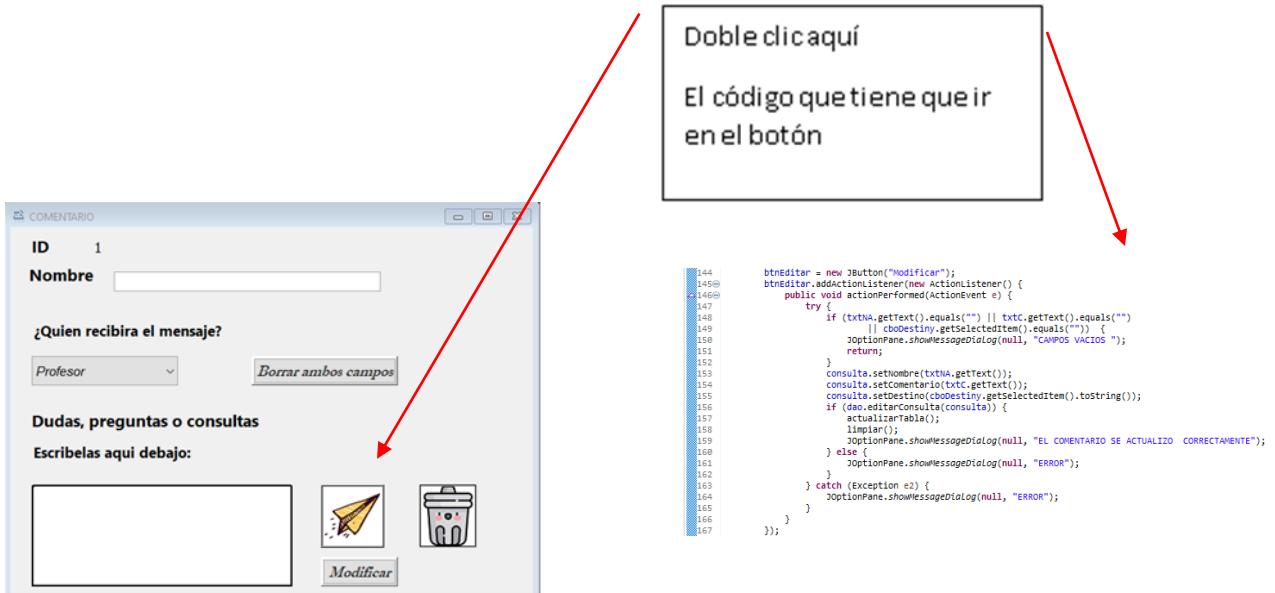
PROGRAMAR BOTONES

Agregar el evento **ActionPerformed** al botón Insertar que se encuentra en **ApplicationWindow->vComentario**, dando doble clic sobre el botón.



BOTON MODIFICAR

Vamos a dar doble clic en el botón Modificar y vamos a repetir el mismo procedimiento que hicimos el Botón Agregar.



BOTON ELIMINAR

Vamos a dar doble clic en el botón Eliminar y vamos a repetir el mismo procedimiento que hicimos el Botón Agregar.

Doble clic aquí
El código que tiene que ir en el botón