

“Año del Bicentenario, de la consolidación de nuestra Independencia, y de la conmemoración de las heroicas batallas de Junín y Ayacucho”

“UNIVERSIDAD PERUANA LOS ANDES”

FACULTAD DE INGENIERÍA

**ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS Y
COMPUTACIÓN**



Trabajo 5

Estudiante:

- Rojas Pariona Miguel Angel

Docente:

- Fernandez Bejarano Raul Enrique

Asignatura:

- ARQUITECTURA DE SOFTWARE

Sección:

- B1

Ciclo:

- VII

Huancayo - Perú

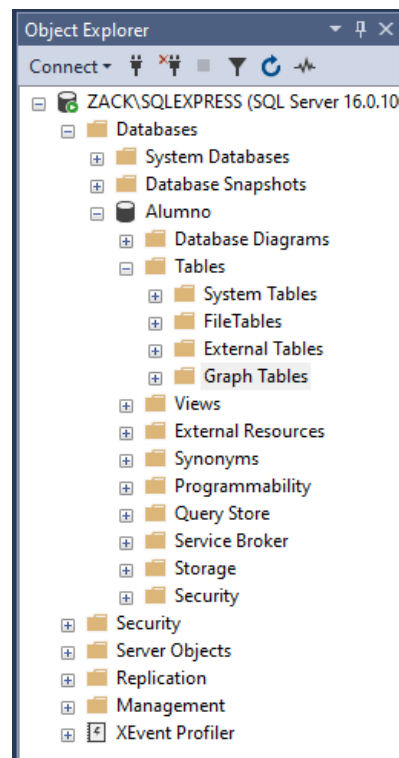
2024

CRUD en Java

Ejercicios

1. CRUD en Java Escritorio MVC – Listar
2. CRUD en Java Escritorio MVC – Agregar
3. CRUD en Java Escritorio MVC – Actualizar
4. CRUD en Java Escritorio MVC – Eliminar

Prueba de Conexión



Connect to Server

SQL Server

Login | Connection Properties | Always Encrypted | Additional Connection Parameters

Server

Server type: Database Engine

Server name: ZACK\SQLEXPRESS

Authentication: SQL Server Authentication

Login: miguel

Password:

☐ Remember password

Connection Security

Encryption: Optional

☐ Trust server certificate

Host name in certificate:

Connect Cancel Help Options <<

Creamos una base de datos llamado Alumno:

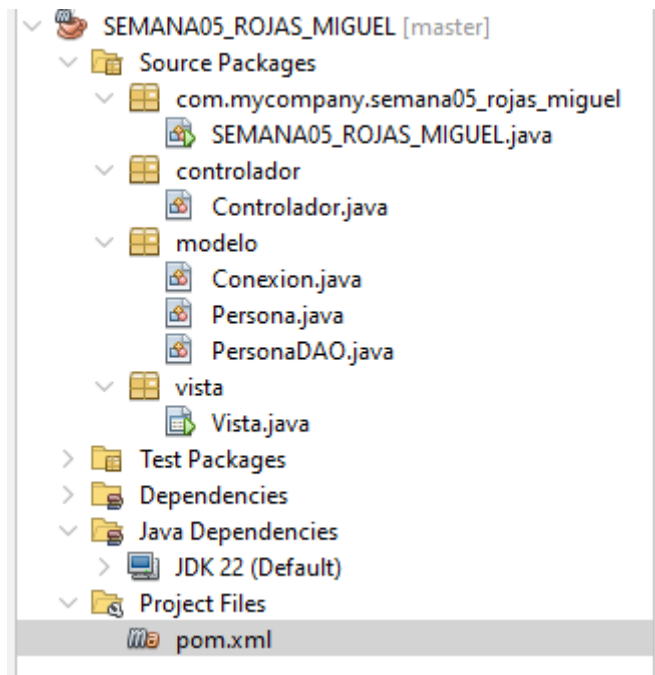
```
SQLQuery1.sql - ZA...umno (miguel (67))* -> X
CREATE DATABASE Alumno;
GO

USE Alumno;
GO

CREATE TABLE persona (
    id INT IDENTITY(1,1) PRIMARY KEY, -- IDENTITY para auto incremento
    nombres VARCHAR(255),
    correo VARCHAR(255),
    telefono VARCHAR(20)
);
GO

SELECT * FROM persona;
GO
```

A Continuación en Netbeans creamos un nuevo proyecto llamado Semana 05



luego nos vamos al pom.xml para poner :

```
    </dependency>
    <!-- https://mvnrepository.com/artifact/com.microsoft.sqlserver/mssql-jdbc -->
] <dependency>
    <groupId>com.microsoft.sqlserver</groupId>
    <artifactId>mssql-jdbc</artifactId>
    <version>12.8.1.jre11</version>
- </dependency>
```

Escribimos el código en modelo que ira la conexión:

package modelo;

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
```

```
/**
 *
 * @author miguel
 */
public class Conexion {
    private Connection conexion = null;
```

```

// Parámetros de conexión
private final String usuario = "miguel";
private final String contraseña = "123";
private final String db = "Alumno";
private final String ip = "localhost";
private final String puerto = "1433";

// Método para obtener la conexión
public Connection getConnection() {
    try {
        String cadena = "jdbc:sqlserver://" + ip + ":" + puerto + ";databaseName=" + db +
";trustServerCertificate=true;";
        conexion = DriverManager.getConnection(cadena, usuario, contraseña);
        System.out.println("Conexión exitosa a la base de datos");
    } catch (SQLException e) {
        Logger.getLogger(Conexion.class.getName()).log(Level.SEVERE, "Error en la
conexión: " + e.toString());
    }
    return conexion;
}

// Método para cerrar la conexión
public void cerrarConexion() {
    if (conexion != null) {
        try {
            conexion.close();
            System.out.println("Conexión cerrada correctamente");
        } catch (SQLException e) {
            Logger.getLogger(Conexion.class.getName()).log(Level.SEVERE, "Error al cerrar
la conexión: " + e.toString());
        }
    }
}
}

```

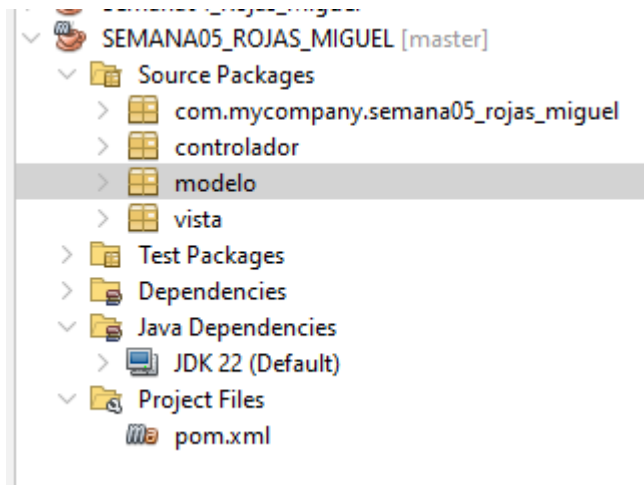
Verificamos que la conexión esté funcionando

```

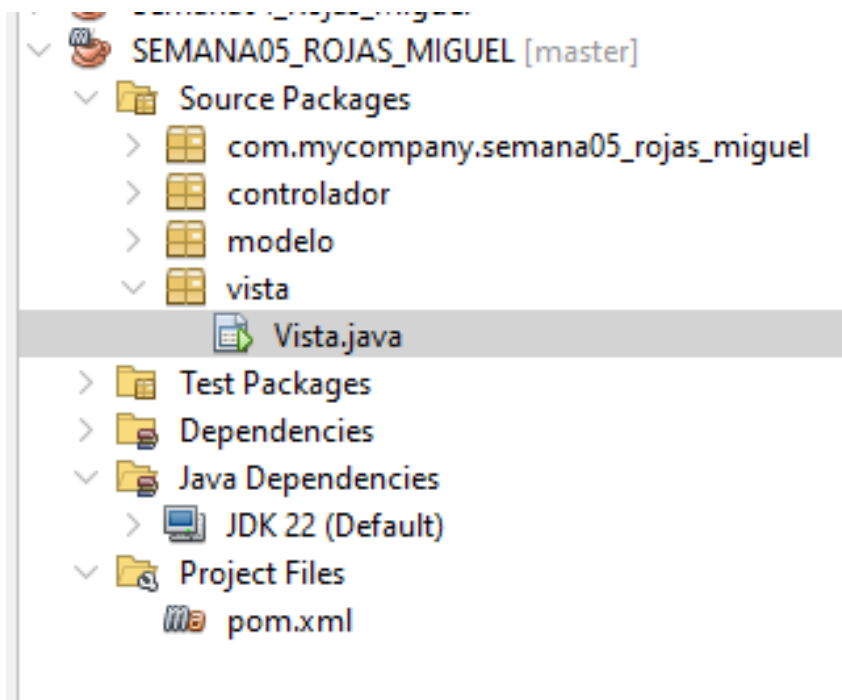
--- exec:3.1.0:exec (default-cli) @ SEMANA05_ROJAS_MIGUEL ---
Conexión exitosa a la base de datos
Conexión exitosa a la base de datos
Conexión exitosa a la base de datos
Conexión exitosa a la base de datos

```

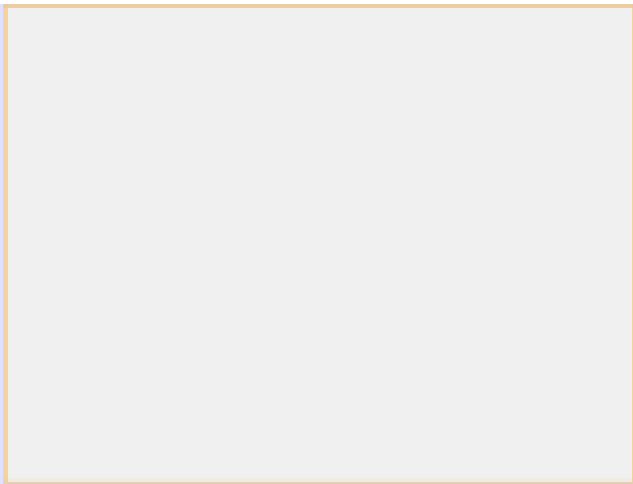
A continuación creamos la estructura de nuestro proyecto, damos clic derecho en Source packages/New/Java Package y crearemos tres paquetes: modelo, controlador, vista.



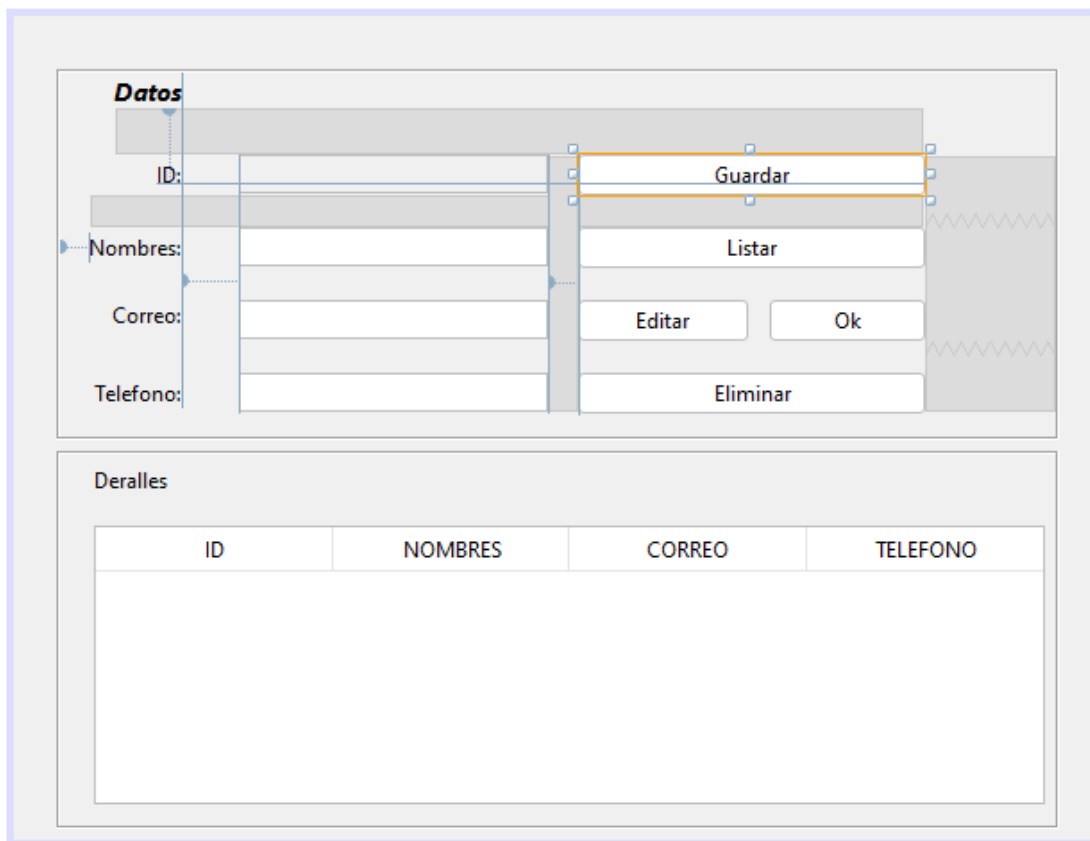
En la parte de vista agregamos un nuevo JFrame Form... llamada Vista y luego finalizar:



A continuación tenemos el formulario:



Diseñamos el formulario, según lo indicado:



ID	NOMBRES	CORREO	TELEFONO

Donde los nombres de las variables de las cajas de texto:

txtId

txtNombres

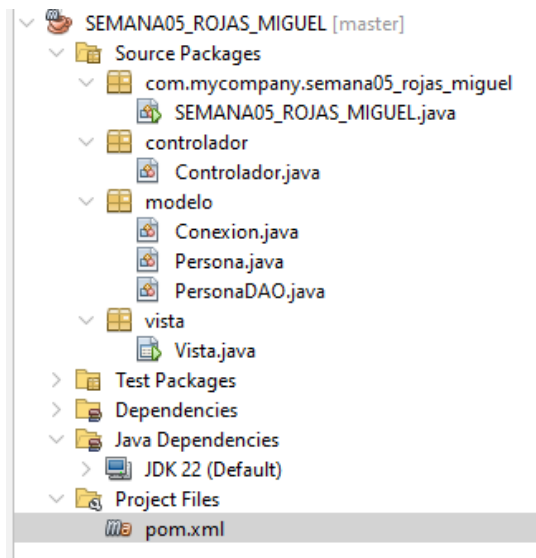
txtCorreo

txtTelefono

btnActualizar

btnEditar

A continuación, ingresamos los códigos:



Controlador.java:

```
package controlador;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.List;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import modelo.Persona;
import modelo.PersonaDAO;
import vista.Vista;

/**
 *
 * @author Miguel
 */
public class Controlador implements ActionListener {
    private PersonaDAO dao = new PersonaDAO();
    private Vista vista;
    private DefaultTableModel modelo;

    public Controlador(Vista v) {
        this.vista = v;
        this.modelo = (DefaultTableModel) vista.getTabla().getModel(); // Acceso a la tabla

        // Registro de los listeners de los botones usando getters
        vista.getBtnListar().addActionListener(this);
        vista.getBtnGuardar().addActionListener(this);
        vista.getBtnEditar().addActionListener(this);
        vista.getBtnActualizar().addActionListener(this);
        vista.getBtnEliminar().addActionListener(this);

        listar(); // Listar al inicializar el controlador
    }
}
```



```

@Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == vista.getBtnListar()) { // Usa el getter
        listar();
    } else if (e.getSource() == vista.getBtnGuardar()) { // Usa el getter
        agregar();
    } else if (e.getSource() == vista.getBtnEditar()) { // Usa el getter
        editar();
    } else if (e.getSource() == vista.getBtnActualizar()) { // Usa el getter
        actualizar();
    } else if (e.getSource() == vista.getBtnEliminar()) { // Usa el getter
        eliminar();
    }
}

public void guardar(String id, String nombres, String correo, String telefono) {
    // Validar campos (opcional, puedes mover la validación a otro método)
    if (nombres.isEmpty() || correo.isEmpty() || telefono.isEmpty()) {
        showMessage("Todos los campos son obligatorios.");
        return;
    }

    // Crear una nueva Persona
    Persona persona = new Persona(nombres, correo, telefono);

    // Llamar al DAO para agregar la persona
    if (dao.agregar(persona) == 1) {
        showMessage("Persona guardada con éxito.");
    } else {
        showMessage("Error al guardar persona.");
    }

    // Listar nuevamente para actualizar la tabla
    listar();
}

public void editar() {
    int fila = vista.getTabla().getSelectedRow(); // Obtener la fila seleccionada
    if (fila == -1) {
        showMessage("Debe seleccionar una fila.");
    } else {
        int id = (int) vista.getTabla().getValueAt(fila, 0);
        String nombres = (String) vista.getTabla().getValueAt(fila, 1);
        String correo = (String) vista.getTabla().getValueAt(fila, 2);
        String telefono = (String) vista.getTabla().getValueAt(fila, 3);

        // Llenar los campos de texto de la vista usando setters
        vista.setId(String.valueOf(id));
        vista.setNombres(nombres);
        vista.setCorreo(correo);
        vista.setTelefono(telefono);
    }
}
}

```

```

public void eliminar() {
    int fila = vista.getTabla().getSelectedRow(); // Obtener la fila seleccionada
    if (fila == -1) {
        showMessage("Debe seleccionar un usuario.");
    } else {
        int id = (int) vista.getTabla().getValueAt(fila, 0); // Obtener el ID de la tabla
        if (dao.delete(id) > 0) { // Si se eliminó correctamente
            showMessage("Usuario eliminado.");
            listar(); // Actualiza la tabla después de eliminar
        } else {
            showMessage("Error al eliminar usuario.");
        }
    }
}

```

```

public void eliminar() {
    int fila = vista.getTabla().getSelectedRow(); // Obtener la fila seleccionada
    if (fila == -1) {
        showMessage("Debe seleccionar un usuario.");
    } else {
        int id = (int) vista.getTabla().getValueAt(fila, 0); // Obtener el ID de la tabla
        if (dao.delete(id) > 0) { // Si se eliminó correctamente
            showMessage("Usuario eliminado.");
            listar(); // Actualiza la tabla después de eliminar
        } else {
            showMessage("Error al eliminar usuario.");
        }
    }
}

```

```

public void actualizar() {
    if (validarCampos()) {
        try {
            int id = Integer.parseInt(vista.getId()); // Obtener ID usando el método getter
            String nom = vista.getNombres(); // Usar el método getter
            String correo = vista.getCorreo(); // Usar el método getter
            String tel = vista.getTelefono(); // Usar el método getter

            Persona p = new Persona(id, nom, correo, tel);
            if (dao.actualizar(p) == 1) {
                showMessage("Usuario actualizado con éxito.");
                vista.limpiarCampos(); // Limpia los campos después de actualizar
            } else {
                showMessage("Error al actualizar usuario.");
            }
            listar(); // Actualizar la tabla después de la operación
        } catch (NumberFormatException ex) {
            showMessage("ID debe ser un número.");
        }
    }
}

```

```

public void agregar() {
    if (validarCampos()) {
        String nom = vista.getNombres(); // Obtener nombres desde la vista
        String correo = vista.getCorreo(); // Obtener correo desde la vista
        String tel = vista.getTelefono(); // Obtener teléfono desde la vista
        Persona p = new Persona(nom, correo, tel); // Crear una nueva instancia de Persona

        // Inserta en la base de datos y muestra un mensaje
        if (dao.agregar(p) == 1) {
            showMessage("Usuario agregado con éxito.");
            vista.limpiarCampos(); // Limpia los campos después de agregar
        } else {
            showMessage("Error al agregar usuario.");
        }
    }
}

public void listar() {
    limpiarTabla(); // Limpia la tabla antes de mostrar nuevos datos
    List<Persona> lista = dao.listar(); // Obtiene la lista de personas del DAO

    // Llenar la tabla con los datos obtenidos
    for (Persona p : lista) {
        modelo.addRow(new Object[]{p.getId(), p.getNombres(), p.getCorreo(), p.getTelefono()});
    }
}

private void limpiarTabla() {
    modelo.setRowCount(0); // Limpiar todas las filas
}

private boolean validarCampos() {
    String nom = vista.getNombres(); // Usar el getter
    String correo = vista.getCorreo(); // Usar el getter
    String tel = vista.getTelefono(); // Usar el getter

    if (nom.isEmpty() || correo.isEmpty() || tel.isEmpty()) {
        showMessage("Todos los campos son obligatorios.");
        return false;
    }

    if (!correo.matches("^[\\w-\\.]+@[\\w-]+\\.+[\\w-]{2,4}$")) {
        showMessage("Formato de correo no válido.");
        return false;
    }

    return true;
}

private void showMessage(String message) {
    JOptionPane.showMessageDialog(vista, message);
}

```

Conexion.java:

```
package modelo;
```

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
```

```
/**
 *
 * @author miguel
 */
```

```
public class Conexion {
    private Connection conexion = null;

    // Parámetros de conexión
    private final String usuario = "miguel";
    private final String contraseña = "123";
    private final String db = "Alumno";
    private final String ip = "localhost";
    private final String puerto = "1433";
```

```
// Método para obtener la conexión
```

```
public Connection getConnection() {
    try {
        String cadena = "jdbc:sqlserver://" + ip + ":" + puerto + ";databaseName=" + db
            + ";trustServerCertificate=true;";
        conexion = DriverManager.getConnection(cadena, usuario, contraseña);
        System.out.println("Conexión exitosa a la base de datos");
    } catch (SQLException e) {
        Logger.getLogger(Conexion.class.getName()).log(Level.SEVERE,
            "Error en la conexión: " + e.toString());
    }
    return conexion;
}
```

```
// Método para cerrar la conexión
```

```
public void cerrarConexion() {
    if (conexion != null) {
        try {
            conexion.close();
            System.out.println("Conexión cerrada correctamente");
        } catch (SQLException e) {
            Logger.getLogger(Conexion.class.getName()).log(Level.SEVERE,
                "Error al cerrar la conexión: " + e.toString());
        }
    }
}
```

Persona.java:

```
package modelo;

/**
 *
 * @author miguel
 */
public class Persona {
    private int id;
    private String nombres; // Cambiado de 'nom' a 'nombres'
    private String correo;
    private String telefono; // Cambiado de 'tel' a 'telefono'

    // Constructor sin parámetros
    public Persona() {}

    // Constructor para crear una nueva Persona (sin ID)
    public Persona(String nombres, String correo, String telefono) {
        this.nombres = nombres;
        this.correo = correo;
        this.telefono = telefono;
    }

    // Constructor para editar una Persona existente (con ID)
    public Persona(int id, String nombres, String correo, String telefono) {
        this.id = id;
        this.nombres = nombres;
        this.correo = correo;
        this.telefono = telefono;
    }

    // Getters y Setters
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getNombres() {
        return nombres;
    }

    public void setNombres(String nombres) {
        this.nombres = nombres;
    }

    public String getCorreo() {
        return correo;
    }

    public void setCorreo(String correo) {
        this.correo = correo;
    }

    public String getTelefono() {
        return telefono;
    }

    public void setTelefono(String telefono) {
        this.telefono = telefono;
    }
}
```

PersonaDAO.java:

```
package modelo;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author miguel
 */
public class PersonaDAO {
    private Conexion conexion = new Conexion();

    // Método para agregar una nueva persona a la base de datos
    public int agregar(Persona persona) {
        String sql = "INSERT INTO persona (nombres, correo, telefono) VALUES (?, ?, ?)";
        try (Connection conn = conexion.getConnection();
            PreparedStatement ps = conn.prepareStatement(sql)) {
            ps.setString(1, persona.getNombres());
            ps.setString(2, persona.getCorreo());
            ps.setString(3, persona.getTelefono());
            return ps.executeUpdate(); // Retorna 1 si la inserción fue exitosa
        } catch (Exception e) {
            System.out.println("Error al agregar persona: " + e.getMessage());
            return 0; // Retorna 0 si ocurre un error
        }
    }

    // Método para actualizar una persona existente
    public int actualizar(Persona persona) {
        String sql = "UPDATE persona SET nombres=?, correo=?, telefono=? WHERE id=?";
        try (Connection conn = conexion.getConnection();
            PreparedStatement ps = conn.prepareStatement(sql)) {
            ps.setString(1, persona.getNombres());
            ps.setString(2, persona.getCorreo());
            ps.setString(3, persona.getTelefono());
            ps.setInt(4, persona.getId());
            return ps.executeUpdate(); // Retorna 1 si la actualización fue exitosa
        } catch (Exception e) {
            System.out.println("Error al actualizar persona: " + e.getMessage());
            return 0; // Retorna 0 si ocurre un error
        }
    }
}
```

```

// Método para eliminar una persona de la base de datos
public int delete(int id) {
    String sql = "DELETE FROM persona WHERE id = ?";
    try (Connection conn = conexion.getConnection();
        PreparedStatement ps = conn.prepareStatement(sql)) {
        ps.setInt(1, id);
        return ps.executeUpdate(); // Retorna el número de filas afectadas
    } catch (Exception e) {
        System.out.println("Error al eliminar persona: " + e.getMessage());
        return 0; // Retorna 0 si ocurre un error
    }
}

// Método para listar todas las personas de la base de datos
public List<Persona> listar() {
    List<Persona> lista = new ArrayList<>();
    String sql = "SELECT * FROM persona";
    try (Connection conn = conexion.getConnection();
        PreparedStatement ps = conn.prepareStatement(sql);
        ResultSet rs = ps.executeQuery()) {
        while (rs.next()) {
            Persona p = new Persona();
            p.setId(rs.getInt("id"));
            p.setNombres(rs.getString("nombres"));
            p.setCorreo(rs.getString("correo"));
            p.setTelefono(rs.getString("telefono"));
            lista.add(p);
        }
    } catch (Exception e) {
        System.out.println("Error al listar personas: " + e.getMessage());
    }
    return lista;
}
}

```

Vista.java:

```

package vista;
import controlador.Controlador;

/**
 * @author miguel
 */
public class Vista extends javax.swing.JFrame {
    private Controlador controlador;

    /**
     * Creates new form Vista
     */
    public Vista() {
        initComponents();
        controlador = new Controlador(this);
    }
}

```

```
// Métodos get para obtener los valores de los campos de texto
[-] public javax.swing.JButton getBtnListar() {
    |     return btnListar;
    | }

[-] public javax.swing.JButton getBtnGuardar() {
    |     return btnGuardar;
    | }

[-] public javax.swing.JButton getBtnEditar() {
    |     return btnEditar;
    | }

[-] public javax.swing.JButton getBtnActualizar() {
    |     return btnActualizar;
    | }

[-] public javax.swing.JButton getBtnEliminar() {
    |     return btnEliminar;
    | }

public void setId(String id) {
    |     txtId.setText(id);
    | }

public void setNombres(String nombres) {
    |     txtNombres.setText(nombres);
    | }

public void setCorreo(String correo) {
    |     txtCorreo.setText(correo);
    | }

public void setTelefono(String telefono) {
    |     txtTelefono.setText(telefono);
    | }

    public String getId() {
    |         return txtId.getText();
    |     }

    public String getNombres() {
    |         return txtNombres.getText();
    |     }

    public String getCorreo() {
    |         return txtCorreo.getText();
    |     }
}
```



```

3 public javax.swing.JTable getTabla() {
    return tabla;
- }

3 public String getTelefono() {
    return txtTelefono.getText();
- }

3 public void limpiarCampos() {
    txtId.setText("");
    txtNombres.setText("");
    txtCorreo.setText("");
    txtTelefono.setText("");
- }

    // Método para mostrar datos en la tabla
3 public void mostrarDatosEnTabla(Object[][] data) {
    tabla.setModel(new javax.swing.table.DefaultTableModel(
        data,
        new String[] { "ID", "NOMBRES", "CORREO", "TELEFONO" }
    ));
- }

```

```

private void txtIdActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void txtNombresActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void txtCorreoActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void txtTelefonoActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void btnGuardarActionPerformed(java.awt.event.ActionEvent evt) {
    controlador.guardar(getId(), getNombres(), getCorreo(), getTelefono());
}

private void btnListarActionPerformed(java.awt.event.ActionEvent evt) {
    controlador.listar();
}

private void btnEditarActionPerformed(java.awt.event.ActionEvent evt) {
    controlador.editar();
}

private void btnActualizarActionPerformed(java.awt.event.ActionEvent evt) {
    controlador.actualizar();
}

private void btnEliminarActionPerformed(java.awt.event.ActionEvent evt) {
    controlador.eliminar();
}

```

```

] /**
-  * @param args the command line arguments
-  */
] public static void main(String args[]) {
-     /* Set the Nimbus look and feel */
-     Look and feel setting code (optional)

-     /* Create and display the form */
-     java.awt.EventQueue.invokeLater(new Runnable() {
-     public void run() {
-         new Vista().setVisible(true);
-     }
-     });
- }

```

```

// Variables declaration - do not modify
private javax.swing.JButton btnActualizar;
private javax.swing.JButton btnEditar;
private javax.swing.JButton btnEliminar;
private javax.swing.JButton btnGuardar;
private javax.swing.JButton btnListar;
private com.jtattoo.plaf.hifi.HiFiLookAndFeel hiFiLookAndFeel1;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JPanel jPanel5;
private javax.swing.JPanel jPanel6;
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JScrollPane jScrollPane3;
private javax.swing.JTable jTable1;
private javax.swing.JTable tabla;
private javax.swing.JTextField txtCorreo;
private javax.swing.JTextField txtId;
private javax.swing.JTextField txtNombres;
private javax.swing.JTextField txtTelefono;
// End of variables declaration

```

```

}

```

A continuación ejecutamos y presionamos el botón Listar:

The diagram shows a web form with two main sections: **Datos** and **Deralles**.

Datos Section:

- Labels: ID:, Nombres:, Correo:, Telefono:
- Input fields: Four horizontal text boxes corresponding to the labels.
- Buttons: A vertical stack of buttons on the right: Guardar, Listar, Editar, Ok, and Eliminar.
- Visual cues: Blue dashed lines connect labels to input fields. Orange dashed lines highlight the Listar button.

Deralles Section:

- Table with 4 columns: ID, NOMBRES, CORREO, TELEFONO.
- The table body is currently empty.

The screenshot shows the web application in a browser window. The **Datos** section has input fields for ID, Nombres, Correo, and Telefono, with buttons Guardar, Listar, Editar, Ok, and Eliminar. The **Deralles** section displays a table with data.

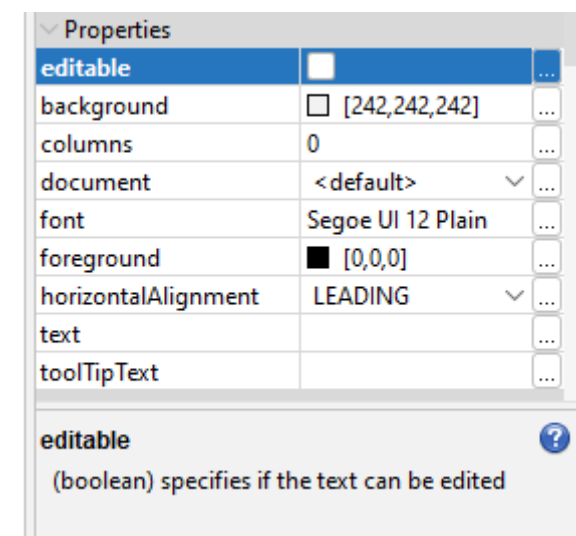
ID	NOMBRES	CORREO	TELEFONO
6	miguel	zdaker98@gmail.com	929376207
7	zack	q035@gmail.com	928382
8	zacker	ash@gmail.com	12345456

	id	nombres	correo	telefono
1	6	miguel	zdaker98@gmail.com	929376207
2	7	zack	q035@gmail.com	928382
3	8	zacker	ash@gmail.com	12345456

2. CRUD en Java Escritorio MVC – Agregar

El campo Id no debe ser editable, para ello, vamos Vista, en la parte de diseño, le damos clic derecho, propiedades y desactivamos la opción editable:

The screenshot shows a Java Swing window titled "Datos". It contains a form with four text input fields: "ID:", "Nombres:", "Correo:", and "Telefono:". To the right of these fields are five buttons: "Guardar", "Listar", "Editar", "Ok", and "Eliminar". A red rectangular box is drawn around the "ID:" label and its corresponding text input field, indicating that this field's properties are being modified.



Ejecutamos, y agregamos un nuevo usuario, y luego guardar:

Datos

ID:

Nombres:

Correo:

Telefono:

Guardar

Listar

Editar **Ok**

Eliminar

Deralles

ID	NOMBRES	CORREO	TELEFONO
6	miguel	zdaker98@gmail.com	929376207
7	zack	q035@gmail.com	928382
8	zacker	ash@gmail.com	12345456

Datos

ID:

Nombres:

Correo:

Telefono:

Guardar

Listar

Editar **Ok**

Eliminar

Deralles

ID	NOMBRES	CORREO	TELEFONO
6	miguel	zdaker98@gmail.com	929376207
7	zack	q035@gmail.com	928382
8	zacker	ash@gmail.com	12345456

Message **X**

Usuario agregado con éxito.

OK

A continuación presionamos en el botón listar, y observamos que se ha agregado sin problemas al nuevo usuario:

Datos

ID:

Nombres:

Correo:

Telefono:

Deralles

ID	NOMBRES	CORREO	TELEFONO
6	miguel	zdaker98@gmail.com	929376207
7	zack	q035@gmail.com	928382
8	zacker	ash@gmail.com	12345456
9	bell	bell@gmail.com	98766454

3. CRUD en Java Escritorio MVC – Actualizar

The screenshot shows a Java desktop application window titled "Datos". It contains several input fields and buttons. A message dialog box is displayed in the center, indicating an error: "Debe seleccionar una fila." (You must select a row). The dialog has an "OK" button. Below the dialog is a table with the following data:

ID	NOMBRES	CORREO	TELEFONO
6	miguel	zdaker98@gmail.com	929376207
7	zack	q035@gmail.com	928382
8	zacker	ash@gmail.com	12345456
9	bell	bell@gmail.com	98766454

Editamos:

The screenshot shows the same Java desktop application window, but now it is in the "Edit" mode. The "ID" field is set to "9". The "Nombres" field is highlighted with a red rectangle and contains the text "Angel". The "Correo" field is set to "bell@gmail.com" and has "Editar" and "Ok" buttons next to it. The "Telefono" field is set to "98766454" and has an "Eliminar" button next to it. The table below the form remains the same as in the previous screenshot.

ID	NOMBRES	CORREO	TELEFONO
6	miguel	zdaker98@gmail.com	929376207
7	zack	q035@gmail.com	928382
8	zacker	ash@gmail.com	12345456
9	bell	bell@gmail.com	98766454

Ahora actualizamos dando clic en el botón ok:

The screenshot shows a Java Swing window titled "Datos" with a dark theme. It contains a form with four input fields: "ID:" with the value "9", "Nombres:" with the value "Angel", "Correo:" with the value "bell@g...", and "Telefono:" with the value "987664...". There are buttons "Guardar" and "Listar" next to the "ID:" and "Nombres:" fields respectively. A "Message" dialog box is open in the center, displaying an information icon and the text "Usuario actualizado con éxito." with an "OK" button. Below the form, there is a section titled "Deralles" containing a table with four columns: "ID", "NOMBRES", "CORREO", and "TELEFONO". The table has four rows of data.

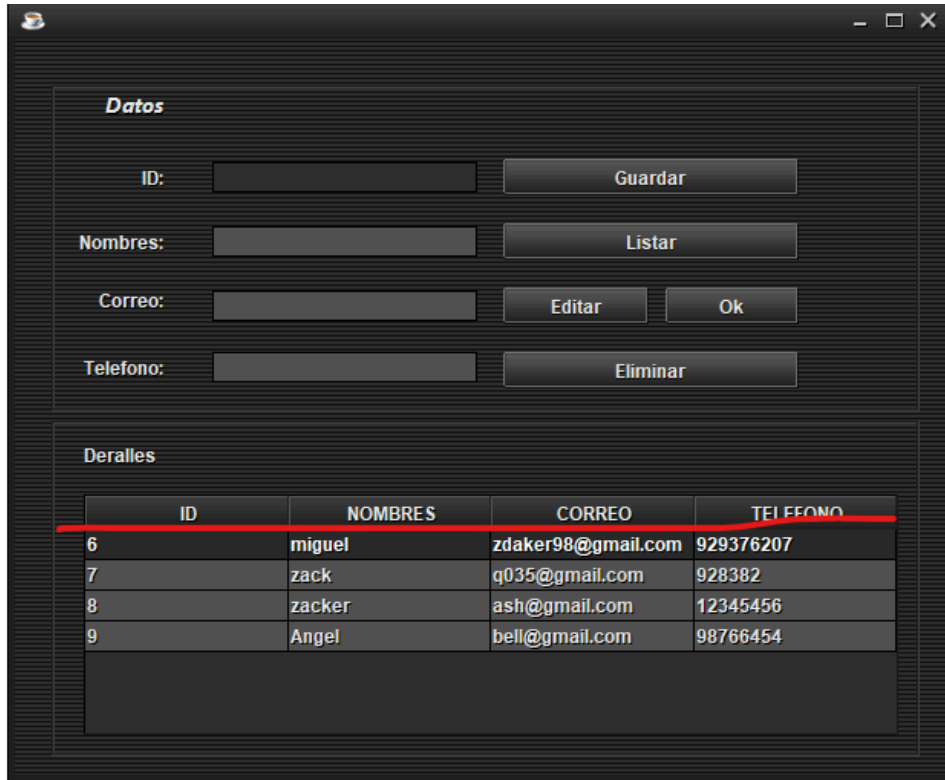
ID	NOMBRES	CORREO	TELEFONO
6	miguel	zdaker98@gmail.com	929376207
7	zack	q035@gmail.com	928382
8	zacker	ash@gmail.com	12345456
9	bell	bell@gmail.com	98766454

Listamos nuevamente y observamos que se ya se ha actualizó:

ID	NOMBRES	CORREO	TELEFONO
6	miguel	zdaker98@gmail.com	929376207
7	zack	q035@gmail.com	928382
8	zacker	ash@gmail.com	12345456
9	Angel	bell@gmail.com	98766454

4. CRUD en Java Escritorio MVC – Eliminar

A continuación ejecutamos, seleccionamos (El usuario de la fila 6) y finalmente eliminamos:



Datos

ID:

Nombres:

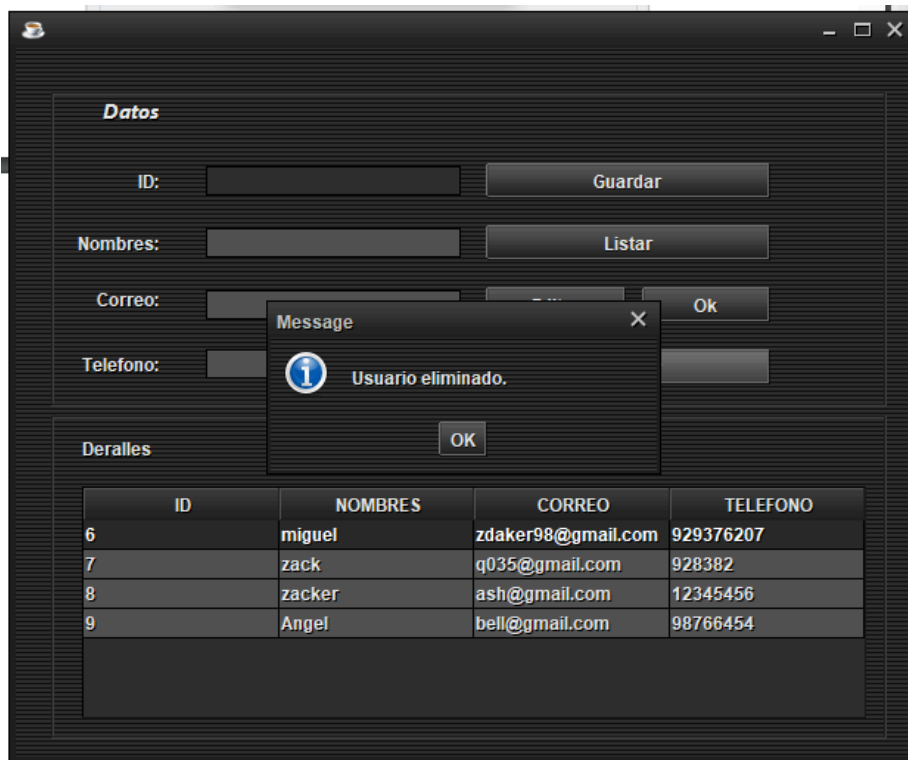
Correo:

Telefono:

Deralles

ID	NOMBRES	CORREO	TELEFONO
6	miguel	zdaker98@gmail.com	929376207
7	zack	q035@gmail.com	928382
8	zacker	ash@gmail.com	12345456
9	Angel	bell@gmail.com	98766454

Aparece el mensaje Usuario eliminado:



Datos

ID:

Nombres:


Correo:

Telefono:

Deralles

ID	NOMBRES	CORREO	TELEFONO
6	miguel	zdaker98@gmail.com	929376207
7	zack	q035@gmail.com	928382
8	zacker	ash@gmail.com	12345456
9	Angel	bell@gmail.com	98766454

Mensaje

 Usuario eliminado.

Le damos aceptar y se actualiza con el usuario eliminado (fila 6):

Deralles			
ID	NOMBRES	CORREO	TELEFONO
7	zack	q035@gmail.com	928382
8	zacker	ash@gmail.com	12345456
9	Angel	bell@gmail.com	98766454