

**“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:**

- Pariona Gozme Maicol Brayan

**Docente:**

- Fernandez Bejarano Raul Enrique

**Asignatura:**

- ARQUITECTURA DE SOFTWARE

**Sección:**

- B1

**Ciclo:**

- VII

**Huancayo - Perú**

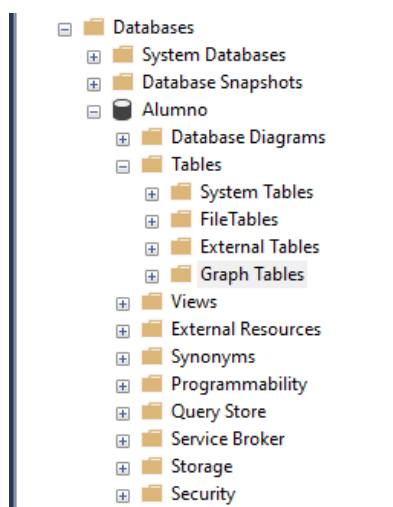
**2024**

# 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



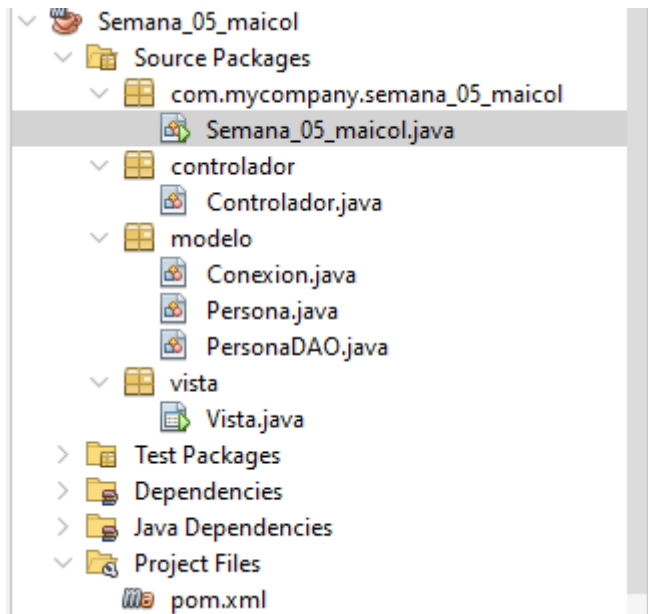
Creamos una base de datos llamado Alumno:

```
CREATE DATABASE Alumno;
GO

CREATE TABLE persona (
    id INT IDENTITY(1,1) PRIMARY KEY,
    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\_maicol**



**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;
```

```
/**
```

```
*
```

```
* @author Maicol
```

```
*/
```

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

```
    // Parámetros de conexión
```

```
    private final String usuario = "maicol";
```

```
    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());
        }
    }
}
}

```

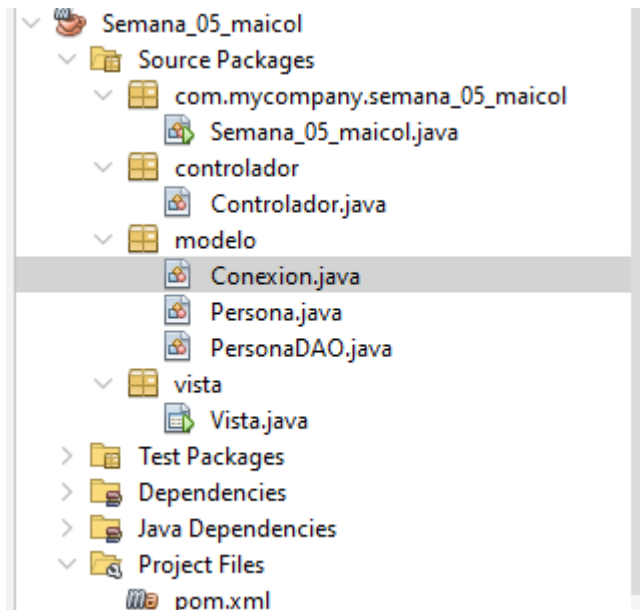
### Verificamos que la conexión esté funcionando

```

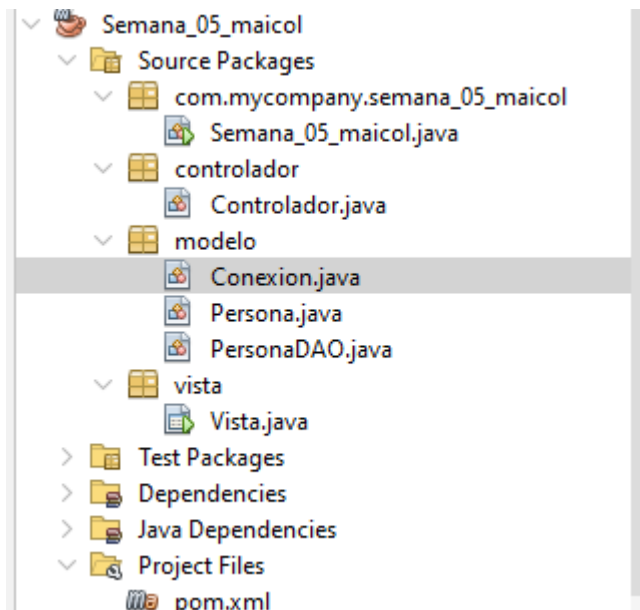
[+] Building Semana_05_maicol 1.0-SNAPSHOT
    from pom.xml
-----[ jar
[+] --- resources:3.3.1:resources (default
    skip non existing resourceDirectory C:
[+] --- compiler:3.11.0:compile (default-c
    Nothing to compile - all classes are u
[+] --- exec:3.1.0:exec (default-cli) @ Se
    Conexion exitosa 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:



Diseñamos el formulario, según lo indicado:

### ***Datos***

ID:

Guardar

Nombres:

Listar

Correo:

Editar

Ok

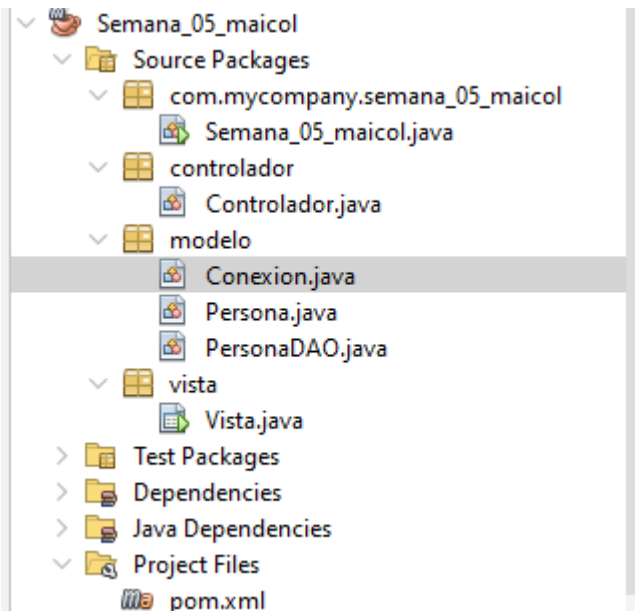
Telefono:

Eliminar

### ***Detalles***

ID	NOMBRES	CORREO	TELEFONO
----	---------	--------	----------

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 Maicol
 */
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 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;

```

```

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

    // Parámetros de conexión
    private final String usuario = "maicol";
    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 Maicol
```

```
*/
```

```
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 Maicol
 */
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 Maicol
 */
public class Vista extends javax.swing.JFrame {
    private Controlador controlador;
    /**
     * Creates new form Vista
     */
    public Vista() {
        initComponents();
        controlador = new Controlador(this);
    }

    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 */
        <!--editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) -->
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and
        feel.
         * For details see
        http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
            javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

            java.util.logging.Logger.getLogger(Vista.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {

            java.util.logging.Logger.getLogger(Vista.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

            java.util.logging.Logger.getLogger(Vista.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

```



```

java.util.logging.Logger.getLogger(Vista.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
    //</editor-fold>

    /* 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 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:

### ***Datos***

ID:

Guardar

Nombres:

Listar

Correo:

Editar

Ok

Telefono:

Eliminar

### ***Detalles***

ID	NOMBRES	CORREO	TELEFONO
----	---------	--------	----------

### ***Datos***

ID:

Guardar

Nombres:

Listar

Correo:

Editar

Ok

Telefono:

Eliminar

### ***Detalles***

ID	NOMBRES	CORREO	TELEFONO
1	maicol	maicol@gmail.com	1234

100 %

Results Messages

	id	nombres	correo	telefono
1	1	maicol	maicol@gmail.com	1234

## 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:

### Datos

ID:

Nombres:

Correo:


Telefono:

### Detalles

ID	NOMBRES	CORREO	TELEFONO

Properties

editable	<input type="checkbox"/>	...
background	<input type="checkbox"/> [242,242,242]	...
columns	0	...
document	<default>	...
font	Segoe UI 12 Plain	...
foreground	<input type="checkbox"/> [0,0,0]	...
horizontalAlignment	LEADING	...
text		...
toolTipText		...

**editable** 

(boolean) specifies if the text can be edited

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

### Datos

ID:

Guardar

Nombres:

brayan

Listar

Correo:

brayan@gmail.com

Editar

Ok

Telefono:

123445

Eliminar

### Detalles

ID	NOMBRES	CORREO	TELEFONO
1	maicol	maicol@gmail.com	1234

### Datos

ID:

Guardar

Nombres:

brayan

Listar

Correo:

brayan

Ok

Telefono:

123445

Message

Usuario agregado con éxito.

OK

### Detalles

ID	NOMBRES	CORREO	TELEFONO
1	maicol	maicol@gmail.com	1234

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

The screenshot shows a Java Swing application window with a dark theme. It is divided into two main sections: 'Datos' (Data) and 'Detalles' (Details).

**Datos Section:**

- ID:** A text input field followed by a 'Guardar' (Save) button.
- Nombres:** A text input field followed by a 'Listar' (List) button.
- Correo:** A text input field followed by 'Editar' (Edit) and 'Ok' buttons.
- Telefono:** A text input field followed by an 'Eliminar' (Delete) button.

**Detalles Section:**

Contains a table with the following data:

ID	NOMBRES	CORREO	TELEFONO
1	maicol	maicol@gmail.com	1234
2	brayan	brayan@gmail.com	123445

### 3. CRUD en Java Escritorio MVC – Actualizar

This screenshot shows the same application window as before, but with an error message dialog box displayed over the 'Detalles' section. The dialog box has a title bar 'Message' and contains the text 'Debe seleccionar una fila.' (You must select a row.) with an 'OK' button.

The background window shows the same 'Datos' and 'Detalles' sections as in the previous image.



Listamos nuevamente y observamos que se ya se ha actualizó:

<i><b>Detalles</b></i>			
ID	NOMBRES	CORREO	TELEFONO
1	gozme	maicol@gmail.com	1234
2	brayan	brayan@gmail.com	123445

#### 4. CRUD en Java Escritorio MVC – Eliminar

A continuación ejecutamos, seleccionamos (El usuario de la fila 2) y finalmente eliminamos:

The screenshot shows a Java desktop application window with a title bar containing a logo and standard window controls. The window is divided into two main sections. The top section, titled 'Datos', contains four rows of input fields and buttons: 'ID:' with a text box and a 'Guardar' button; 'Nombres:' with a text box and a 'Listar' button; 'Correo:' with a text box, an 'Editar' button, and an 'Ok' button; and 'Telefono:' with a text box and an 'Eliminar' button. The bottom section, titled 'Detalles', contains a table with the same structure as the one in the first image, showing two rows of data. The table has columns for ID, NOMBRES, CORREO, and TELEFONO. The first row shows ID 1, NOMBRES gozme, CORREO maicol@gmail.com, and TELEFONO 1234. The second row shows ID 2, NOMBRES brayan, CORREO brayan@gmail.com, and TELEFONO 123445.

<i><b>Datos</b></i>			
ID:	<input type="text"/>	Guardar	
Nombres:	<input type="text"/>	Listar	
Correo:	<input type="text"/>	Editar	Ok
Telefono:	<input type="text"/>	Eliminar	

<i><b>Detalles</b></i>			
ID	NOMBRES	CORREO	TELEFONO
1	gozme	maicol@gmail.com	1234
2	brayan	brayan@gmail.com	123445

Aparece el mensaje Usuario eliminado:

The screenshot shows a web application window titled "Datos". It contains four input fields: "ID:", "Nombres:", "Correo:", and "Telefono:". To the right of the "ID:" field is a "Guardar" button. To the right of the "Nombres:" field is a "Listar" button. To the right of the "Correo:" field is an "Ok" button. A modal dialog box titled "Message" is open in the center, displaying an information icon and the text "Usuario eliminado." with an "OK" button. Below the input fields is a table with the following data:

ID	NOMBRES	CORREO	TELEFONO
1	gozme	maicol@gmail.com	1234
2	brayan	brayan@gmail.com	123445

Le damos aceptar y se actualiza con el usuario eliminado (fila 2):

The screenshot shows a web application window titled "Detalles". It contains a table with the following data:

ID	NOMBRES	CORREO	TELEFONO
1	gozme	maicol@gmail.com	1234