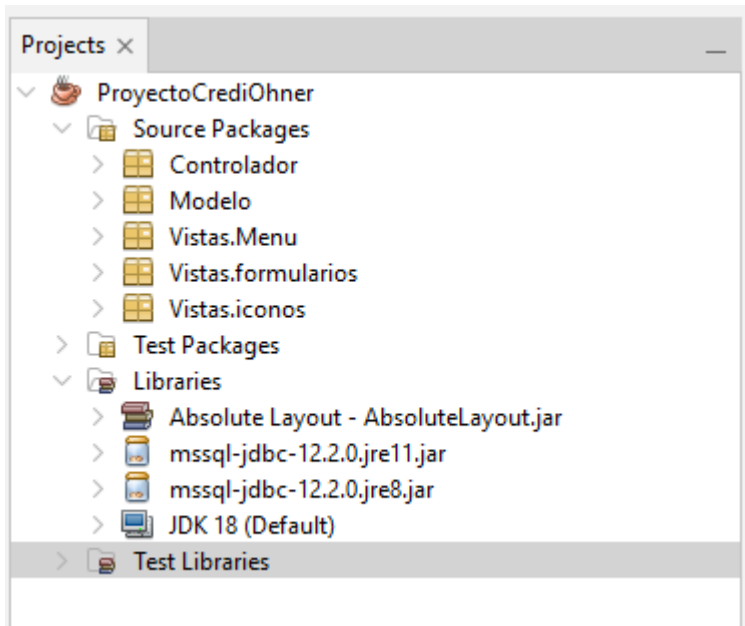


Guía para realizar el CRUD a la base de datos en SQLServer desde Netbeans con Java

Estimados estudiantes, a continuación, se orienta paso a paso las actividades a realizar para lograr la conexión con la base de datos y realizar las 4 operaciones básicas: Leer, Guardar, Actualizar y Eliminar.

Recuerda crear la estructura de paquetes siguiendo la arquitectura MVC.



Operación Leer

1. Crear el procedimiento almacenado que te permita consultar los datos de la tabla. Para esto debes insertar, desde SQLServer, datos de prueba a la tabla que quieres consultar.

```
USE [CREDITOHNER]
GO
CREATE PROCEDURE [dbo].[MostrarClientes]
AS
BEGIN
    select cliente.cedula, nombres, apellidos, sobrenombre, telefono, direccion
    from CLIENTE inner join PERSONA on CLIENTE.cedula=PERSONA.cedula
END
```

2. En el proyecto creado en Netbeans, crea la clase conexión en el paquete **Controlador**

```
package Controlador;

import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.SQLException;

public class Conexion {

    private final String
url="jdbc:sqlserver://localhost:1433;databaseName=CREDITOHNER;"
        + "integratedSecurity=true;" +
        "encrypt=true;trustServerCertificate=true; user=sa; password=123";

    Connection cn;
    public Connection conectar(){
        try {
            cn = DriverManager.getConnection(url);
            System.out.println("Conexion establecida");
            return cn;
        } catch (SQLException e) {
            System.out.println("Error en la conexión: "+e);
        }
        return null;
    }
}
```

3. Crear el POJO correspondiente a la tabla a consultar datos en el paquete modelo.
En el ejemplo que se muestra a continuación utilizo herencia para usar los datos de la case Persona.

POJO Persona

```
public class Persona {
    String cedula;
    String nombres;
    String apellidos;
    String telefono;
    String direccion;

    public Persona(String cedula, String nombres, String apellidos, String
telefono, String direccion) {
        this.cedula = cedula;
        this.nombres = nombres;
        this.apellidos = apellidos;
        this.telefono = telefono;
    }
}
```

```
        this.direccion = direccion;
    }

    public Persona(){

    }

    public String getCedula() {
        return cedula;
    }

    public void setCedula(String cedula) {
        this.cedula = cedula;
    }

    public String getNombres() {
        return nombres;
    }

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

    public String getApellidos() {
        return apellidos;
    }

    public void setApellidos(String apellidos) {
        this.apellidos = apellidos;
    }

    public String getTelefono() {
        return telefono;
    }

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

    public String getDireccion() {
        return direccion;
    }

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

POJO Cliente

```

package Modelo;
public class Cliente extends Persona {

    String sobrenombre;

    public Cliente(String cedula, String nombres, String apellidos,String
sobrenombre,
        String telefono, String direccion) {
        super(cedula, nombres, apellidos, telefono, direccion);
        this.sobrenombre=sobrenombre;
    }

    public Cliente(){
    }

    public String getSobrenombre() {
        return sobrenombre;
    }

    public void setSobrenombre(String sobrenombre) {
        this.sobrenombre = sobrenombre;
    }

}

```

4. Crea la clase CRUD correspondiente a la tabla a consultar, en este ejemplo la clase se nombró **CRUDCliente** y se ubicó en el paquete **Controlador**. En esta clase se crean los métodos que permiten acceder a los procedimientos almacenados que ejecutan las instrucciones SQL Correspondiente. En este caso iniciamos con la operación **Leer**

```

import Modelo.Cliente;
import java.sql.*;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

public class CRUDCliente {

    private final Conexion con = new Conexion();
    private final Connection cn = (Connection) con.conectar();

    public DefaultTableModel mostrarDatos() {
        ResultSet rs;
        DefaultTableModel modelo;
        String[] titulos = {"Cédula", "Nombres", "Apellidos", "Sobrenombre",
"Telefono", "Dirección"};
        String[] registro = new String[6];
    }
}

```

```

        modelo = new DefaultTableModel(null, titulos);

        try {
            CallableStatement cbstc = cn.prepareCall("{call MostrarClientes}");
            rs = cbstc.executeQuery();

            while (rs.next()) {
                registro[0] = rs.getString("cedula");
                registro[1] = rs.getString("nombres");
                registro[2] = rs.getString("apellidos");
                registro[3] = rs.getString("sobrenombre");
                registro[4] = rs.getString("telefono");
                registro[5] = rs.getString("direccion");

                modelo.addRow(registro);
            }
            return modelo;
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, e);
            return null;
        }
    }
}

```

5. Crea in JFrame Form en el paquete vista donde se diseña la interfaz para mostrar los datos en un jTable.



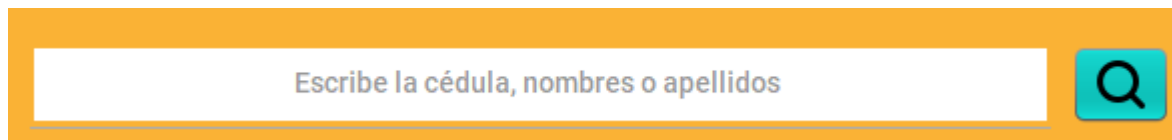
6. El código del formulario ubicar un método llamado mostrarDatos y luego llamarlo desde el constructor del formulario. Con esto, ya puedes verificar si la conexión y la consulta a la base de datos funciona correctamente.

```
public frmGestionarClientes() { //Constructor
    initComponents();
    mostrar(); //llamada al método mostrar()
}

public void mostrar() { //Método mostrar
    try {
        DefaultTableModel modelo;
        CRUDCliente cli = new CRUDCliente(); //objeto de la clase CRUDCliente
        modelo = cli.mostrarDatos();
        tableClientes.setModel(modelo);

    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, e);
    }
}
```

Programación del Botón Buscar



1. Crea el procedimiento almacenado que permita hacer búsquedas de información ya sea por cédula, nombre o apellidos.

```
USE [CREDITOHNER]
GO

CREATE PROCEDURE [dbo].[ConsultarClientes]
    @dato varchar(50)
AS
BEGIN
    select cliente.cedula, nombres, apellidos, sobrenombre, telefono, direccion
    from CLIENTE inner join PERSONA
    on CLIENTE.cedula=PERSONA.cedula
    where cliente.cedula like '%' + RTRIM(@dato) + '%' or nombres like
    '%' + RTRIM(@dato) + '%' or apellidos like '%' + RTRIM(@dato) + '%'
END
```

2. En la clase CRUDCliente del paquete Controlador añadir un método llamado buscarDatos que llama al procedimiento almacenado llamado ConsultarClientes

```

public DefaultTableModel buscarDatos(String dato) {
    ResultSet rs;
    DefaultTableModel modelo;

    String[] titulos = {"Cédula", "Nombres", "Apellidos", "Sobrenombre",
"Telefono", "Dirección"};
    String[] registro = new String[6];

    modelo = new DefaultTableModel(null, titulos);

    try {
        CallableStatement call = cn.prepareCall("{call
ConsultarClientes(?)}")
        call.setString(1, dato);
        rs = call.executeQuery();

        while (rs.next()) {
            registro[0] = rs.getString("cedula");
            registro[1] = rs.getString("nombres");
            registro[2] = rs.getString("apellidos");
            registro[3] = rs.getString("sobrenombre");
            registro[4] = rs.getString("telefono");
            registro[5] = rs.getString("direccion");

            modelo.addRow(registro);
        }
        return modelo;
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, e);
        return null;
    }
}

```

3. En el botón **buscar** agregar el siguiente código

```

try {

    DefaultTableModel modelo;
    CRUDCliente cli = new CRUDCliente();
    modelo = cli.buscarDatos(jTextSearch.getText());
    if (jTextSearch.getText().equals("")){
        JOptionPane.showMessageDialog(null, "Escriba el dato a buscar");
        mostrar();
    } else {
        tableClientes.setModel(modelo);
    }

} catch (Exception e) {
    JOptionPane.showMessageDialog(null, e);
}
}

```

Operación Guardar

1. Crea el procedimiento almacenado que te permita guardar en la tabla correspondiente.

```
USE [CREDITOHNER]
GO

CREATE PROCEDURE [dbo].[CrearCliente]
    @Cedula varchar(16),
    @Nombres varchar(50),
    @Apellidos varchar(50),
    @Sobrenombre varchar(50),
    @Telefono varchar(9),
    @Direccion varchar(100)
AS
BEGIN
INSERT INTO PERSONA(cedula, nombres, apellidos, telefono, direccion)
VALUES (@Cedula, @Nombres, @Apellidos, @Telefono, @Direccion)
INSERT INTO CLIENTE(sobrenombre, cedula)
VALUES ( @Sobrenombre, @Cedula)
```

2. En la clase CRUDCliente ya existente en el paquete Controlador, agrega el **método Guardar** y el **método VerificarDatos** que permita comprobar que no existe en la tabla otro elemento con el mismo ID

```
public void Guardar(Cliente cl) {
    try {
        CallableStatement cbst = cn.prepareCall("{call CrearCliente(?,?,?,?,?,?)");
        cbst.setString(1, cl.getCedula());
        cbst.setString(2, cl.getNombres());
        cbst.setString(3, cl.getApellidos());
        cbst.setString(4, cl.getSobrenombre());
        cbst.setString(5, cl.getTelefono());
        cbst.setString(6, cl.getDireccion());
        cbst.executeUpdate();

    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, e);
    }
}
```

```
public boolean verificarDatos(String dato) {
    ResultSet rs;
```



```

try {
    CallableStatement call = cn.prepareCall("{call ConsultarClientes(?)}")
    call.setString(1, dato);
    rs = call.executeQuery();
    return rs.next();

} catch (SQLException e) {
    JOptionPane.showMessageDialog(null, e);
    return false;
}
}

```

3. Diseña la interfaz para guardar la información de **nuevo Cliente**

Registrar Nuevo Cliente

Cédula

Nombres

Apellidos

Sobrenombre

Teléfono

Dirección

Guardar

4. En el código del formulario creo dos métodos, uno para guardar los datos y otro para limpiar los controles del formulario

Método Guardar

```
public void guardarCliente() {
    CRUDCliente cc = new CRUDCliente();
    Cliente cl = new Cliente(jTextCedula.getText(),
        jTextNombres.getText(),
        jTextApellidos.getText(),
        jTextSobrenombre.getText(),
        jTextTelefono.getText(),
        jTextDireccion.getText());
    cc.Guardar(cl);
}
```

Método Limpiar

```
public void limpiar() {
    jTextCedula.setText("");
    jTextNombres.setText("");
    jTextApellidos.setText("");
    jTextSobrenombre.setText("");
    jTextTelefono.setText("");
    jTextDireccion.setText("");
}
```

En el botón guardar, añade el siguiente código

```
private void jButtonGuardarActionPerformed(java.awt.event.ActionEvent evt) {
    CRUDCliente cl = new CRUDCliente();
    try {
        if ((jTextCedula.getText().equals(""))
            || (jTextNombres.getText().equals(""))
            || (jTextApellidos.getText().equals(""))
            || (jTextTelefono.getText().equals(""))
            || (jTextSobrenombre.getText().equals(""))
            || (jTextDireccion.getText().equals(""))) {
            JOptionPane.showMessageDialog(null, "Tiene datos vacíos");
        } else {
            if (cl.verificarDatos(jTextCedula.getText())) {
                JOptionPane.showMessageDialog(null, "Ya existe cliente con ese
número de Cédula");
            } else {
                guardarCliente();
                limpiar();
                JOptionPane.showMessageDialog(null, "Datos Guardados
Correctamente");
            }
        }
    } catch (HeadlessException e) {
```

```

        JOptionPane.showMessageDialog(null, "Error: " + e);
    }
}

```

Operación Eliminar

1. Crea el procedimiento almacenado que te permita eliminar un dato por su llave primaria

```

USE [CREDITOHNER]
GO
CREATE PROCEDURE [dbo].[EliminarClientes]
    @dato varchar(50)
AS
BEGIN
    delete from cliente where cedula=@dato
    delete from persona where cedula=@dato
END

```

2. En la clase CRUDCliente añade un método llamado Eliminar

```

public void eliminar(String cedula) {
    try {
        CallableStatement cbst = cn.prepareCall("{call EliminarClientes(?)}");
        cbst.setString(1, cedula);
        cbst.executeUpdate();

    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, e);
    }
}

```

3. En el código del formulario donde está la tablaClientes, agrega una variable global que guarde el número de la fila seleccionada en la tabla clientes, la cual se ocupará para eliminar el dato correspondiente al seleccionado en la tabla.


```

public class frmGestionarClientes extends javax.swing.JInternalFrame {

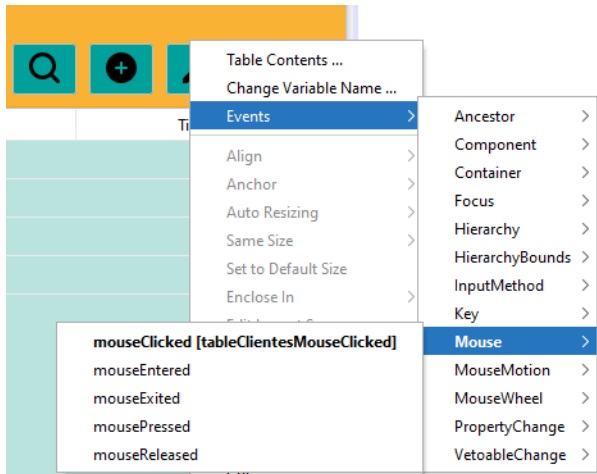
    int datoSeleccionado = -1;

    public frmGestionarClientes() {
        initComponents();
        mostrar();
        botonmostrar.setVisible(false);
    }
}

```



4. En modo diseño del formulario, selecciona la tablaEstudiante, clic derecho>Events>Mouse>mouseClicked y agrega la siguiente programación



```
datoSeleccionado = tableClientes.rowAtPoint(evt.getPoint());
```

5. En el botón eliminar, agregar el siguiente código

```
if (datoSeleccionado >= 0) {
    String dato =
String.valueOf(tableClientes.getValueAt(datoSeleccionado, 0));
    CRUDCliente cli = new CRUDCliente();
    if (JOptionPane.showConfirmDialog(rootPane,
        "Se eliminará el registro, ¿desea continuar?",
        "Eliminar Registro",
        JOptionPane.WARNING_MESSAGE,
        JOptionPane.YES_NO_OPTION)
        == JOptionPane.YES_OPTION) {
        cli.eliminar(dato);
        mostrar();
        JOptionPane.showMessageDialog(null,
            "Dato eliminado correctamente");
    }
} else {
    JOptionPane.showMessageDialog(null,
        "Debe seleccionar un registro de la tabla");
}
```

Operación actualizar

1. Crea el procedimiento almacenado que te permita actualizar datos.

```
USE [CREDITOHNER]
GO

ALTER PROCEDURE [dbo].[ModificarCliente]
    @Cedula varchar(16),
    @Nombres varchar(50),
    @Apellidos varchar(50),
    @Sobrenombre varchar(50),
```

```

        @Telefono varchar(9),
        @Direccion varchar(100)

AS
BEGIN
UPDATE PERSONA SET nombres=@Nombres, apellidos=@Apellidos, telefono=@Telefono,
direccion=@Direccion where cedula=@cedula
UPDATE CLIENTE SET sobrenombre=@sobrenombre where cedula=@cedula
END

```

2. En la clase CRUDCliente, agrega el método actualizar.

```

public void actualizar(Cliente cl) {
    try {
        CallableStatement cbst = cn.prepareCall("{call
ModificarCliente(?,?,?,?,?,?)}");
        cbst.setString(1, cl.getCedula());
        cbst.setString(2, cl.getNombres());
        cbst.setString(3, cl.getApellidos());
        cbst.setString(4, cl.getSobrenombre());
        cbst.setString(5, cl.getTelefono());
        cbst.setString(6, cl.getDireccion());
        cbst.executeUpdate();

    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, e);
    }
}

```

3. En la vista donde seleccionamos el registro a actualizar, añadimos el siguiente código para que me traslade los datos seleccionados al formulario donde se deben editar los datos para luego dar la orden de actualizar.

```

public DefaultTableModel buscarDatos(String dato) {
    ResultSet rs;
    DefaultTableModel modelo;

    String[] titulos = {"Cédula", "Nombres", "Apellidos", "Sobrenombre",
"Telefono", "Dirección"};
    String[] registro = new String[6];

    modelo = new DefaultTableModel(null, titulos);

    try {
        CallableStatement call = cn.prepareCall("{call
ConsultarClientes(?)}");
        call.setString(1, dato);
        rs = call.executeQuery();

        while (rs.next()) {
            registro[0] = rs.getString("cedula");

```

```

        registro[1] = rs.getString("nombres");
        registro[2] = rs.getString("apellidos");
        registro[3] = rs.getString("sobrenombre");
        registro[4] = rs.getString("telefono");
        registro[5] = rs.getString("direccion");

        modelo.addRow(registro);
    }
    return modelo;
} catch (Exception e) {
    JOptionPane.showMessageDialog(null, e);
    return null;
}
}

```

4. En el botón actualizar agregar el siguiente código

```

private void jButtonActualizarActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        if ((jTextNombres.getText().equals(""))
            || (jTextApellidos.getText().equals(""))
            || (jTextTelefono.getText().equals(""))
            || (jTextSobrenombre.getText().equals(""))
            || (jTextDireccion.getText().equals(""))) {
            JOptionPane.showMessageDialog(null, "Tiene datos vacíos");
        } else {
            editarCliente();
            JOptionPane.showMessageDialog(null, "Datos Actualizados
Correctamente");
            dispose();
            frmGestionarClientes.botonmostrar.doClick();
        }
    } catch (HeadlessException e) {
        JOptionPane.showMessageDialog(null, "Error: " + e);
    }
}
}

```