SEMINARIO DE PRÁCTICA

UNIVERSIDAD SIGLO 21

La educación evoluciona

CATEDRA - E - INF275 - EDH - PR - TUTORIAS

CUATRIMESTRALES 2/24

Trabajo Práctico N° 4

“Situación Estación Meteorológica Convencional del INTA”

Alumno:

Leandro Ariel GONZALEZ - VINF02351

Profesor:

Titular Experto Ing. (Msc) Hugo Fernando FRIAS

Nota: TP4 Solo, las correcciones del código observadas en el TP3 se realizaron para hacerlo más eficiente

Contenido

Titulo

Desarrollo de un Sistema de Registro de datos agrometeorológicos de estaciones con instrumentos convencionales del **INTA**, Unidad **Colonia Benítez**.

Introducción

Para esta etapa del prototipado conectaremos con la base de datos, haremos apertura de conexiones y su respectivo manejo de excepciones, se hará dos módulos más el módulo de usuario.-

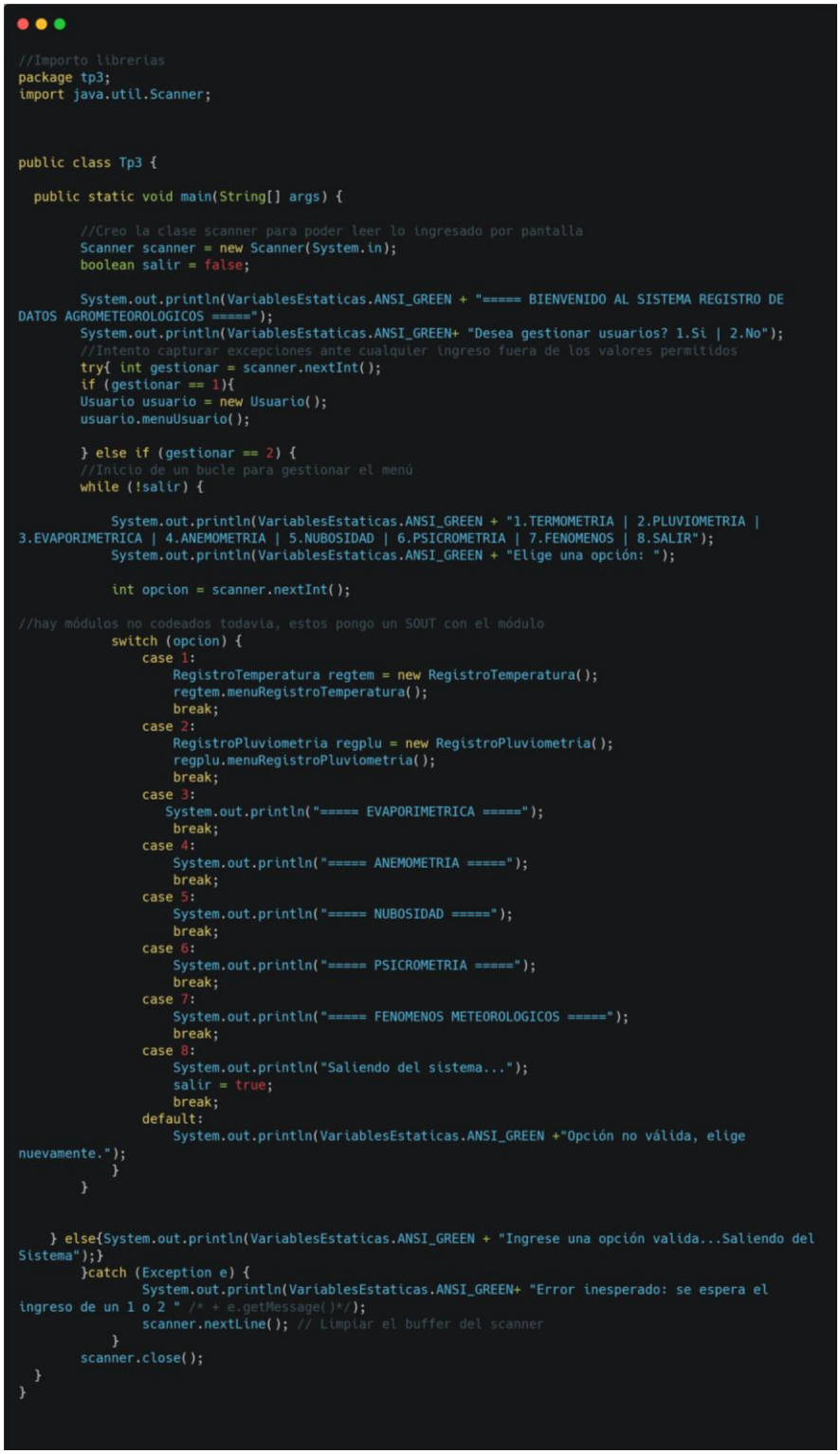
Explicación del Código

Solo se realizó el código de algunos módulos para los cuales usaremos como modelo de la función del software, se usó métodos para manejo de datos **ABMCL**, métodos para menú, métodos para captura de datos.-

Las clases:

- RegistroTemperatura
- RegistroPluviometria
- Pluviometria
- Termometria
- Usuarios

Código Main:



```java
//Importo librerias
package tp3;
import java.util.Scanner;


public class Tp3 {

  public static void main(String[] args) {

        //Creo la clase scanner para poder leer lo ingresado por pantalla
        Scanner scanner = new Scanner(System.in);
        boolean salir = false;

        System.out.println(VariablesEstaticas.ANSI_GREEN + "===== BIENVENIDO AL SISTEMA REGISTRO DE
DATOS AGROMETEOROLOGICOS =====");
        System.out.println(VariablesEstaticas.ANSI_GREEN+ "Desea gestionar usuarios? 1.Si | 2.No");
        //Intento capturar excepciones ante cualquier ingreso fuera de los valores permitidos
        try{ int gestionar = scanner.nextInt();
        if (gestionar == 1){
        Usuario usuario = new Usuario();
        usuario.menuUsuario();

        } else if (gestionar == 2) {
        //Inicio de un bucle para gestionar el menú
        while (!salir) {

            System.out.println(VariablesEstaticas.ANSI_GREEN + "1.TERMOMETRIA | 2.PLUVIOMETRIA |
3.EVAPORIMETRICA | 4.ANEMOMETRIA | 5.NUBOSIDAD | 6.PSICROMETRIA | 7.FENOMENOS | 8.SALIR");
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Elige una opción: ");

            int opcion = scanner.nextInt();

//hay módulos no codeados todavia, estos pongo un SOUT con el módulo
            switch (opcion) {
                case 1:
                    RegistroTemperatura regtem = new RegistroTemperatura();
                    regtem.menuRegistroTemperatura();
                    break;
                case 2:
                    RegistroPluviometria regplu = new RegistroPluviometria();
                    regplu.menuRegistroPluviometria();
                    break;
                case 3:
                   System.out.println("===== EVAPORIMETRICA =====");
                    break;
                case 4:
                    System.out.println("===== ANEMOMETRIA =====");
                    break;
                case 5:
                    System.out.println("===== NUBOSIDAD =====");
                    break;
                case 6:
                    System.out.println("===== PSICROMETRIA =====");
                    break;
                case 7:
                    System.out.println("===== FENOMENOS METEOROLOGICOS =====");
                    break;
                case 8:
                    System.out.println("Saliendo del sistema...");
                    salir = true;
                    break;
                default:
                    System.out.println(VariablesEstaticas.ANSI_GREEN +"Opción no válida, elige
nuevamente.");
                }
        }


    } else{System.out.println(VariablesEstaticas.ANSI_GREEN + "Ingrese una opción valida...Saliendo del
Sistema");}
        }catch (Exception e) {
                System.out.println(VariablesEstaticas.ANSI_GREEN+ "Error inesperado: se espera el
ingreso de un 1 o 2 " /* + e.getMessage()*/);
                scanner.nextLine(); // Limpiar el buffer del scanner
            }
        scanner.close();
  }
}
```

Imagen N° 1: Código Main – *fuente*: Elaboración propia

Se ordeno el código, se lo optimizo conforme la versión anterior, se creó una clase soporte para variables estáticas. -

Inicio del Sistema

```
====== BIENVENIDO AL SISTEMA REGISTRO DE DATOS AGROMETEOROLOGICOS ======
Desea gestionar usuarios? 1.Si | 2.No
```

Imagen N° 2: Inicio del sistema – *fuente*: Elaboración propia

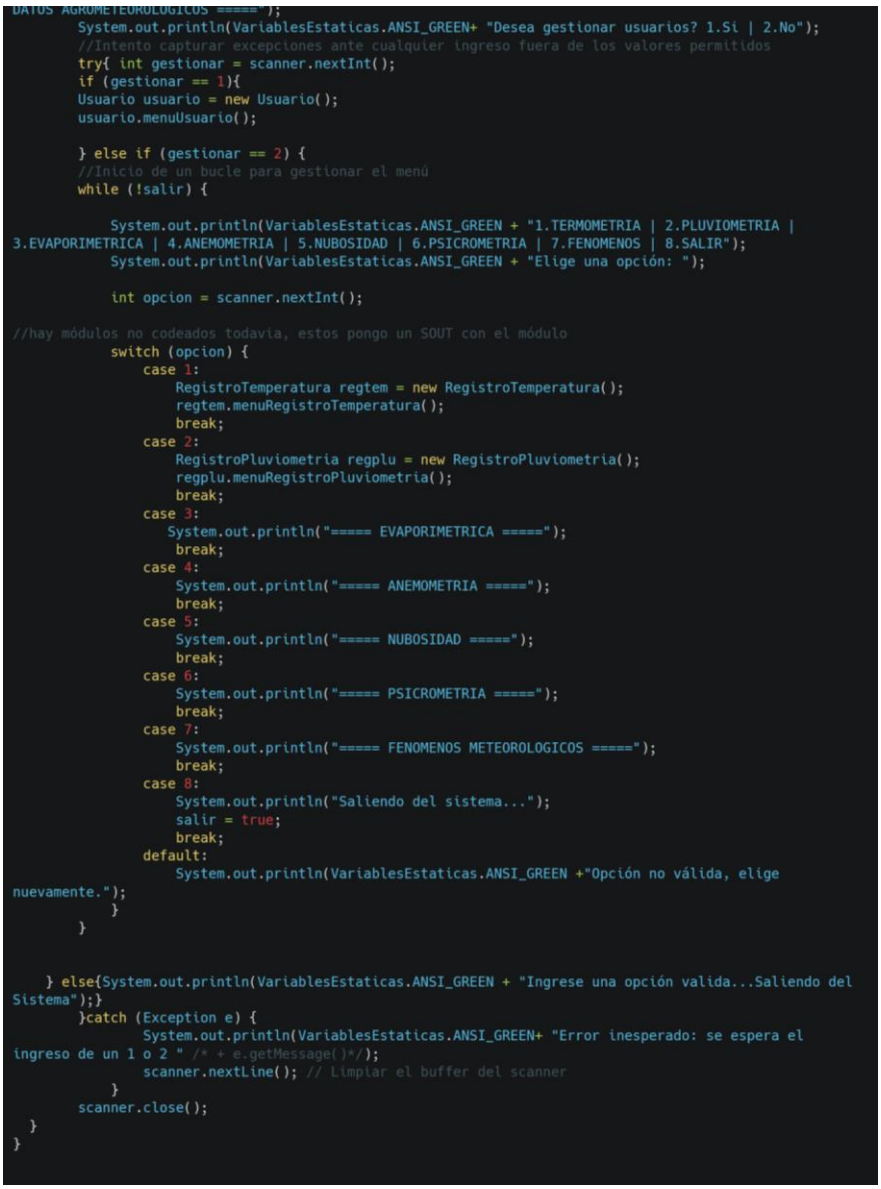
En la primera parte se podrá elegir si ingresar a la parte de *administración de usuario* o la carga de datos, para esto usamos una estructura condicional del estilo **IF – ELSE,** el cual permitirá llamar un método dentro de la clase usuario para gestionarlo, esto solamente se ve en la pantalla del admin, como no se hizo el login, ahora se muestra como si estuviéramos en el módulo del administrador, no la de usuario u observador. -

Menú Principal

```
===== BIENVENIDO AL SISTEMA REGISTRO DE DATOS AGROMETEOROLOGICOS =====
Desea gestionar usuarios? 1.Si | 2.No
2
1.TERMOMETRIA | 2.EVAPORIMETRICA | 3.PLUVIOMETRIA | 4.ANEMOMETRIA | 5.NUBOSIDAD | 6.PSICROMETRIA | 7.FENOMENOS | 8.SALIR
Elige una opción:
```

Imagen N° 3: Menú principal del sistema – *fuente*: Elaboración propia

Una vez pasada la etapa del inicio del sistema, nos encontramos con la parte *menú principal*, donde podemos gestionar los módulos del software, para esto usamos una estructura cíclica **WHILE**, también utilizamos una estructura condicional **SWITCH – CASE. -**



```
DATOS AGROMETEOROLOGICOS =====");
        System.out.println(VariablesEstaticas.ANSI_GREEN+ "Desea gestionar usuarios? 1.Si | 2.No");
        //Intento capturar excepciones ante cualquier ingreso fuera de los valores permitidos
        try{ int gestionar = scanner.nextInt();
        if (gestionar == 1){
        Usuario usuario = new Usuario();
        usuario.menuUsuario();

        } else if (gestionar == 2) {
        //Inicio de un bucle para gestionar el menú
        while (!salir) {

            System.out.println(VariablesEstaticas.ANSI_GREEN + "1.TERMOMETRIA | 2.PLUVIOMETRIA |
3.EVAPORIMETRICA | 4.ANEMOMETRIA | 5.NUBOSIDAD | 6.PSICROMETRIA | 7.FENOMENOS | 8.SALIR");
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Elige una opción: ");

            int opcion = scanner.nextInt();

//hay módulos no codeados todavia, estos pongo un SOUT con el módulo
            switch (opcion) {
                case 1:
                    RegistroTemperatura regtem = new RegistroTemperatura();
                    regtem.menuRegistroTemperatura();
                    break;
                case 2:
                    RegistroPluviometria regplu = new RegistroPluviometria();
                    regplu.menuRegistroPluviometria();
                    break;
                case 3:
                   System.out.println("===== EVAPORIMETRICA =====");
                    break;
                case 4:
                    System.out.println("===== ANEMOMETRIA =====");
                    break;
                case 5:
                    System.out.println("===== NUBOSIDAD =====");
                    break;
                case 6:
                    System.out.println("===== PSICROMETRIA =====");
                    break;
                case 7:
                    System.out.println("===== FENOMENOS METEOROLOGICOS =====");
                    break;
                case 8:
                    System.out.println("Saliendo del sistema...");
                    salir = true;
                    break;
                default:
                    System.out.println(VariablesEstaticas.ANSI_GREEN +"Opción no válida, elige
nuevamente.");
                }
        }


    } else{System.out.println(VariablesEstaticas.ANSI_GREEN + "Ingrese una opción valida...Saliendo del
Sistema");}
        }catch (Exception e) {
                System.out.println(VariablesEstaticas.ANSI_GREEN+ "Error inesperado: se espera el
ingreso de un 1 o 2 " /* + e.getMessage()*/);
                scanner.nextLine(); // Limpiar el buffer del scanner
            }
        scanner.close();
  }
}
```

Imagen N° 4: Código fuente del Switch Case, Menú Principal – *fuente*: Elaboración propia

Como vimos usamos para el manejo de excepciones un **TRY - CATCH** al bloque donde capturamos el ingreso por teclado, en el case 1 y case 2, instanciamos un objeto para poder invocar el método.
Nota: usamos una clase y dentro las variables estáticas para poner las letras en verde para aparentar que es una pantalla de fósforo o monitor monocromo. –

Módulo Termometría

Código Registro Temperatura:

Esta parte creamos una consola con **While** y un **Switch Case** y mediante una clase, ConexionDB.java, hacemos el CRUD o el **ABMCL (Alta – Baja - Modificación – Consulta - Listar)** a la base de datos.

```java
public void menuRegistroTemperatura() {
        Scanner scanner = new Scanner(System.in);
        int opcion;

        try (Connection con = ConexionBD.obtenerConexion()) {
```

Imagen N° 5: Código del módulo termometría, registro de temperatura, conexión a la base de datos – *fuente*: Elaboración propia

Hacemos enfoque de la conexión dentro del menú en las clases, así no cerramos y abrimos por cada operación hacia los datos, cerrado el menú, se cierra la conexión automáticamente, esto debido que nos encontramos con uno error que no podíamos mantenernos trabajando dentro de un mismo modulo, porque cada método cerraba la conexión con el manejo de excepciones.-

Imagen N° 6: Código completo, Registro de Temperatura – *fuente*: Elaboración propia

```java
// Agregar un nuevo registro de temperatura
    public void agregarRegistro(Connection con) {
        String query = "INSERT INTO registrotemp (fecha, hora, observacion, grados, Usuarios_legajo,
Termometria_tipo_termometro) VALUES (?, ?, ?, ?, ?, ?)";

        try (PreparedStatement stmt = con.prepareStatement(query)) {
            stmt.setString(1, this.fecha);
            stmt.setString(2, this.hora);
            stmt.setString(3, this.observacion);
            stmt.setDouble(4, this.grados);
            stmt.setInt(5, this.usuariosLegajo);
            stmt.setInt(6, this.tipoTermometro);

            int rowsInserted = stmt.executeUpdate();
            System.out.println(rowsInserted > 0 ? VariablesEstaticas.ANSI_GREEN + "Registro de
temperatura agregado exitosamente!" : "Error al agregar registro.");
        } catch (SQLException e) {
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Error al agregar registro de
temperatura: " + e.getMessage());
        }
    }

    // Listar todos los registros
    public void listarRegistros(Connection con) {
        String query = "SELECT * FROM registrotemp";

        try (PreparedStatement stmt = con.prepareStatement(query);
             ResultSet rs = stmt.executeQuery()) {

            System.out.println(VariablesEstaticas.ANSI_GREEN + "Registros de temperatura:");
            while (rs.next()) {
                System.out.println(VariablesEstaticas.ANSI_GREEN + "Fecha: " + rs.getString("fecha") +
                                   ", Hora: " + rs.getString("hora") +
                                   ", Observación: " + rs.getString("observacion") +
                                   ", Grados: " + rs.getDouble("grados") +
                                   ", Usuario Legajo: " + rs.getInt("Usuarios_legajo") +
                                   ", Tipo Termómetro: " + rs.getInt("Termometria_tipo_termometro"));
            }
        } catch (SQLException e) {
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Error al listar registros: " +
e.getMessage());
        }
    }

    // Eliminar un registro por fecha y hora
    public void eliminarRegistro(Connection con, String fecha, String hora) {
        String query = "DELETE FROM registrotemp WHERE fecha = ? AND hora = ?";

        try (PreparedStatement stmt = con.prepareStatement(query)) {
            stmt.setString(1, fecha);
            stmt.setString(2, hora);
            int rowsDeleted = stmt.executeUpdate();
            System.out.println(rowsDeleted > 0 ? VariablesEstaticas.ANSI_GREEN + "Registro eliminado
exitosamente!" : "Registro no encontrado.");
        } catch (SQLException e) {
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Error al eliminar el registro: " +
e.getMessage());
        }
    }

    // Actualizar un registro por fecha y hora
    public void actualizarRegistro(Connection con, String fecha, String hora) {
        Scanner scanner = new Scanner(System.in);
        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese la nueva observación: ");
        this.observacion = scanner.nextLine();

        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese los nuevos grados: ");
        this.grados = scanner.nextDouble();

        String query = "UPDATE registrotemp SET observacion = ?, grados = ? WHERE fecha = ? AND hora =
?";

        try (PreparedStatement stmt = con.prepareStatement(query)) {
            stmt.setString(1, this.observacion);
            stmt.setDouble(2, this.grados);
            stmt.setString(3, fecha);
            stmt.setString(4, hora);

            int rowsUpdated = stmt.executeUpdate();
            System.out.println(rowsUpdated > 0 ? VariablesEstaticas.ANSI_GREEN + "Registro actualizado
exitosamente!" : "Registro no encontrado.");
        } catch (SQLException e) {
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Error al actualizar el registro: " +
e.getMessage());
        }
    }

    // Consultar un registro por fecha y hora
    public void consultarRegistro(Connection con, String fecha, String hora) {
        String query = "SELECT * FROM registrotemp WHERE fecha = ? AND hora = ?";

        try (PreparedStatement stmt = con.prepareStatement(query)) {
            stmt.setString(1, fecha);
            stmt.setString(2, hora);

            try (ResultSet rs = stmt.executeQuery()) {
                if (rs.next()) {
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Registro encontrado:");
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Fecha: " +
rs.getString("fecha"));
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Hora: " +
rs.getString("hora"));
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Observación: " +
rs.getString("observacion"));
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Grados: " +
rs.getDouble("grados"));
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Usuario Legajo: " +
rs.getInt("Usuarios_legajo"));
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Tipo Termómetro: " +
rs.getInt("Termometria_tipo_termometro"));
                } else {
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Registro no encontrado.");
                }
            }
        } catch (SQLException e) {
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Error al consultar el registro: " +
e.getMessage());
        }
    }
```

<u>Imagen N° 7</u>: Código métodos **ABMCL** del módulo Termometría, Registro de Temperatura – *fuente*: Elaboración propia.

Al tener inconvenientes con el manejo de los “try-catch”, que cerraban la conexión luego de operar algún módulo, como por ejemplo agregar registro, puse la conexión (invocación a la clase ConexionDB.java) dentro del método menú y esta se cierra luego de salir de él, el código esta mejor, para no crear una clase menú, generé un método para respetar el diagrama de clases realizado anteriormente. -

```java
public void menuRegistroTemperatura() {
        Scanner scanner = new Scanner(System.in);
        int opcion;

        try (Connection con = ConexionBD.obtenerConexion()) {
            do {
                System.out.println(VariablesEstaticas.ANSI_GREEN + "===== Gestión de Registros de
Temperatura =====");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "1. Agregar registro");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "2. Listar registros");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "3. Eliminar registro");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "4. Actualizar registro");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "5. Consultar registro");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "6. Gestionar termómetro");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "7. Salir");
                System.out.print(VariablesEstaticas.ANSI_GREEN + "Seleccione una opción: ");
                opcion = scanner.nextInt();

                switch (opcion) {
                    case 1:
                        capturarDatosRegistro(con);
                        break;
                    case 2:
                        listarRegistros(con);
                        break;
                    case 3:
                        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese la fecha del registro
a eliminar: ");
                        String fechaEliminar = scanner.next();
                        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese la hora del registro
a eliminar: ");
                        String horaEliminar = scanner.next();
                        eliminarRegistro(con, fechaEliminar, horaEliminar);
                        break;
                    case 4:
                        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese la fecha del registro
a actualizar: ");
                        String fechaActualizar = scanner.next();
                        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese la hora del registro
a actualizar: ");
                        String horaActualizar = scanner.next();
                        actualizarRegistro(con, fechaActualizar, horaActualizar);
                        break;
                    case 5:
                        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese la fecha del registro
a consultar: ");
                        String fechaConsultar = scanner.next();
                        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese la hora del registro
a consultar: ");
                        String horaConsultar = scanner.next();
                        consultarRegistro(con, fechaConsultar, horaConsultar);
                        break;
                    case 6:
                        Termometria termometria = new Termometria();
                        termometria.menuTermometria();
                        break;
                    case 7:
                        System.out.println(VariablesEstaticas.ANSI_GREEN + "Saliendo del menú de
registros de temperatura...");
                        break;
                    default:
                        System.out.println(VariablesEstaticas.ANSI_GREEN + "Opción no válida.");
                        break;
                }
            } while (opcion != 7);
        } catch (SQLException e) {
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Error en la conexión con la base de
datos: " + e.getMessage());
        }
    }
```

Imagen N° 8: Código fuente del módulo termometría, Registro de Temperatura, Menú – *fuente*: Elaboración propia

En el código menú esta la gestión de la conexión

```java
package tp3;

// Importa las librerías a usar
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Scanner;

public class Termometria {
    private int tipoTermometro;
    private double totalGrados;
    private String descripcion;

    public Termometria() {}

    // Captura de datos desde la consola
    public void capturarDatosTermometro(Connection con) {
        Scanner scanner = new Scanner(System.in);
        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese el tipo de termómetro: ");
        this.tipoTermometro = scanner.nextInt();
        scanner.nextLine();
        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese la descripción del termómetro: ");
        this.descripcion = scanner.nextLine();
        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese el total de grados: ");
        this.totalGrados = scanner.nextDouble();
        agregarTermometro(con);
    }

    // Agregar un nuevo termómetro
    public void agregarTermometro(Connection con) {
        String query = "INSERT INTO termometria (tipo_termometro, total_grados, descripcion) VALUES (?,
?, ?)";

        try (PreparedStatement stmt = con.prepareStatement(query)) {
            stmt.setInt(1, this.tipoTermometro);
            stmt.setDouble(2, this.totalGrados);
            stmt.setString(3, this.descripcion);

            int rowsInserted = stmt.executeUpdate();
            System.out.println(rowsInserted > 0 ? VariablesEstaticas.ANSI_GREEN + "Termómetro agregado
exitosamente!" : "Error al agregar termómetro.");
        } catch (SQLException e) {
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Error al agregar termómetro: " +
e.getMessage());
        }
    }

    // Listar todos los termómetros
    public void listarTermometros(Connection con) {
        String query = "SELECT * FROM termometria";

        try (PreparedStatement stmt = con.prepareStatement(query);
             ResultSet rs = stmt.executeQuery()) {

            System.out.println(VariablesEstaticas.ANSI_GREEN + "Listado de termómetros:");
            while (rs.next()) {
                System.out.println(VariablesEstaticas.ANSI_GREEN + "Tipo: " +
rs.getInt("tipo_termometro") +
                                   ", Total Grados: " + rs.getDouble("total_grados") +
                                   ", Descripción: " + rs.getString("descripcion"));
            }
        } catch (SQLException e) {
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Error al listar termómetros: " +
e.getMessage());
        }
    }

    // Eliminar un termómetro
    public void eliminarTermometro(int tipoTermometro, Connection con) {
        String query = "DELETE FROM termometria WHERE tipo_termometro = ?";

        try (PreparedStatement stmt = con.prepareStatement(query)) {
            stmt.setInt(1, tipoTermometro);
            int rowsDeleted = stmt.executeUpdate();
            System.out.println(rowsDeleted > 0 ? VariablesEstaticas.ANSI_GREEN + "Termómetro eliminado
exitosamente!" : "Termómetro no encontrado.");
        } catch (SQLException e) {
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Error al eliminar el termómetro: " +
e.getMessage());
        }
    }

    // Actualizar un termómetro
    public void actualizarTermometro(int tipoTermometro, Connection con) {
        Scanner scanner = new Scanner(System.in);
        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese la nueva descripción: ");
        this.descripcion = scanner.nextLine();
        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese el nuevo total de grados: ");
        this.totalGrados = scanner.nextDouble();

        String query = "UPDATE termometria SET descripcion = ?, total_grados = ? WHERE tipo_termometro
= ?";

        try (PreparedStatement stmt = con.prepareStatement(query)) {
            stmt.setString(1, this.descripcion);
            stmt.setDouble(2, this.totalGrados);
            stmt.setInt(3, tipoTermometro);

            int rowsUpdated = stmt.executeUpdate();
            System.out.println(rowsUpdated > 0 ? VariablesEstaticas.ANSI_GREEN + "Termómetro
actualizado exitosamente!" : "Termómetro no encontrado.");
        } catch (SQLException e) {
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Error al actualizar el termómetro: " +
e.getMessage());
        }
    }

    // Consultar Termometro
    public void consultarTermometro(int tipoTermometro, Connection con) {
        String query = "SELECT * FROM termometria WHERE tipo_termometro = ?";

        try (PreparedStatement stmt = con.prepareStatement(query)) {
            stmt.setInt(1, tipoTermometro);

            try (ResultSet rs = stmt.executeQuery()) {
                if (rs.next()) {
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Termómetro encontrado:");
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Tipo: " +
rs.getInt("tipo_termometro"));
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Total Grados: " +
rs.getDouble("total_grados"));
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Descripción: " +
rs.getString("descripcion"));
                } else {
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Termómetro no encontrado.");
                }
            }
        } catch (SQLException e) {
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Error al consultar el termómetro: " +
e.getMessage());
        }
    }

    // Menú de opciones para la gestión de termómetros
    public void menuTermometria() {
        try (Connection con = ConexionBD.obtenerConexion()) {  // Conexión establecida aquí, cerrada
automáticamente al salir del bloque
            Scanner scanner = new Scanner(System.in);
            int opcion;

            do {
                System.out.println(VariablesEstaticas.ANSI_GREEN + "===== Gestión de Termómetros
=====");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "1. Agregar termómetro");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "2. Eliminar termómetro");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "3. Actualizar termómetro");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "4. Consultar termómetro");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "5. Listar termómetros");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "0. Salir");
                System.out.print(VariablesEstaticas.ANSI_GREEN + "Seleccione una opción: ");
                opcion = scanner.nextInt();

                switch (opcion) {
                    case 1:
                        capturarDatosTermometro(con);
                        break;
                    case 2:
                        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese el tipo de termómetro
a eliminar: ");
                        int tipoEliminar = scanner.nextInt();
                        eliminarTermometro(tipoEliminar, con);
                        break;
                    case 3:
                        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese el tipo de termómetro
a actualizar: ");
                        int tipoActualizar = scanner.nextInt();
                        actualizarTermometro(tipoActualizar, con);
                        break;
                    case 4:
                        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese el tipo de termómetro
a consultar: ");
                        int tipoConsulta = scanner.nextInt();
                        consultarTermometro(tipoConsulta, con);
                        break;
                    case 5:
                        listarTermometros(con);
                        break;
                    case 0:
                        System.out.println(VariablesEstaticas.ANSI_GREEN + "Saliendo...");
                        break;
                    default:
                        System.out.println(VariablesEstaticas.ANSI_GREEN + "Opción inválida. Intente de
nuevo.");
                }
            } while (opcion != 0);

        } catch (SQLException e) {
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Error en la conexión a la base de
datos: " + e.getMessage());
        }
    }
}
```

Imagen N° 9: Código del módulo termometría – *fuente*: Elaboración propia

```java
// Captura de datos desde la consola
    public void capturarDatosTermometro(Connection con) {
        Scanner scanner = new Scanner(System.in);
        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese el tipo de termómetro: ");
        this.tipoTermometro = scanner.nextInt();
        scanner.nextLine();
        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese la descripción del termómetro: ");
        this.descripcion = scanner.nextLine();
        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese el total de grados: ");
        this.totalGrados = scanner.nextDouble();
        agregarTermometro(con);
    }

    // Agregar un nuevo termómetro
    public void agregarTermometro(Connection con) {
        String query = "INSERT INTO termometria (tipo_termometro, total_grados, descripcion) VALUES (?,
?, ?)";

        try (PreparedStatement stmt = con.prepareStatement(query)) {
            stmt.setInt(1, this.tipoTermometro);
            stmt.setDouble(2, this.totalGrados);
            stmt.setString(3, this.descripcion);

            int rowsInserted = stmt.executeUpdate();
            System.out.println(rowsInserted > 0 ? VariablesEstaticas.ANSI_GREEN + "Termómetro agregado
exitosamente!" : "Error al agregar termómetro.");
        } catch (SQLException e) {
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Error al agregar termómetro: " +
e.getMessage());
        }
    }

    // Listar todos los termómetros
    public void listarTermometros(Connection con) {
        String query = "SELECT * FROM termometria";

        try (PreparedStatement stmt = con.prepareStatement(query);
             ResultSet rs = stmt.executeQuery()) {

            System.out.println(VariablesEstaticas.ANSI_GREEN + "Listado de termómetros:");
            while (rs.next()) {
                System.out.println(VariablesEstaticas.ANSI_GREEN + "Tipo: " +
rs.getInt("tipo_termometro") +
                                   ", Total Grados: " + rs.getDouble("total_grados") +
                                   ", Descripción: " + rs.getString("descripcion"));
            }
        } catch (SQLException e) {
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Error al listar termómetros: " +
e.getMessage());
        }
    }

    // Eliminar un termómetro
    public void eliminarTermometro(int tipoTermometro, Connection con) {
        String query = "DELETE FROM termometria WHERE tipo_termometro = ?";

        try (PreparedStatement stmt = con.prepareStatement(query)) {
            stmt.setInt(1, tipoTermometro);
            int rowsDeleted = stmt.executeUpdate();
            System.out.println(rowsDeleted > 0 ? VariablesEstaticas.ANSI_GREEN + "Termómetro eliminado
exitosamente!" : "Termómetro no encontrado.");
        } catch (SQLException e) {
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Error al eliminar el termómetro: " +
e.getMessage());
        }
    }

    // Actualizar un termómetro
    public void actualizarTermometro(int tipoTermometro, Connection con) {
        Scanner scanner = new Scanner(System.in);
        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese la nueva descripción: ");
        this.descripcion = scanner.nextLine();
        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese el nuevo total de grados: ");
        this.totalGrados = scanner.nextDouble();

        String query = "UPDATE termometria SET descripcion = ?, total_grados = ? WHERE tipo_termometro
= ?";

        try (PreparedStatement stmt = con.prepareStatement(query)) {
            stmt.setString(1, this.descripcion);
            stmt.setDouble(2, this.totalGrados);
            stmt.setInt(3, tipoTermometro);

            int rowsUpdated = stmt.executeUpdate();
            System.out.println(rowsUpdated > 0 ? VariablesEstaticas.ANSI_GREEN + "Termómetro
actualizado exitosamente!" : "Termómetro no encontrado.");
        } catch (SQLException e) {
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Error al actualizar el termómetro: " +
e.getMessage());
        }
    }

    // Consultar Termómetro
    public void consultarTermometro(int tipoTermometro, Connection con) {
        String query = "SELECT * FROM termometria WHERE tipo_termometro = ?";

        try (PreparedStatement stmt = con.prepareStatement(query)) {
            stmt.setInt(1, tipoTermometro);

            try (ResultSet rs = stmt.executeQuery()) {
                if (rs.next()) {
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Termómetro encontrado:");
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Tipo: " +
rs.getInt("tipo_termometro"));
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Total Grados: " +
rs.getDouble("total_grados"));
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Descripción: " +
rs.getString("descripcion"));
                } else {
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Termómetro no encontrado.");
                }
            }
        } catch (SQLException e) {
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Error al consultar el termómetro: " +
e.getMessage());
        }
    }
```

<u>Imagen N° 10</u>: Código ABMCL Termometría – *fuente*: Elaboración propia

```java
// Menú de opciones para la gestión de termómetros
    public void menuTermometria() {
        try (Connection con = ConexionBD.obtenerConexion()) {  // Conexión establecida aquí, cerrada
automáticamente al salir del bloque
            Scanner scanner = new Scanner(System.in);
            int opcion;

            do {
                System.out.println(VariablesEstaticas.ANSI_GREEN + "===== Gestión de Termómetros
=====");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "1. Agregar termómetro");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "2. Eliminar termómetro");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "3. Actualizar termómetro");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "4. Consultar termómetro");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "5. Listar termómetros");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "6. Salir");
                System.out.print(VariablesEstaticas.ANSI_GREEN + "Seleccione una opción: ");
                opcion = scanner.nextInt();

                switch (opcion) {
                    case 1:
                        capturarDatosTermometro(con);
                        break;
                    case 2:
                        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese el tipo de termómetro
a eliminar: ");
                        int tipoEliminar = scanner.nextInt();
                        eliminarTermometro(tipoEliminar, con);
                        break;
                    case 3:
                        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese el tipo de termómetro
a actualizar: ");
                        int tipoActualizar = scanner.nextInt();
                        actualizarTermometro(tipoActualizar, con);
                        break;
                    case 4:
                        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese el tipo de termómetro
a consultar: ");
                        int tipoConsulta = scanner.nextInt();
                        consultarTermometro(tipoConsulta, con);
                        break;
                    case 5:
                        listarTermometros(con);
                        break;
                    case 6:
                        System.out.println(VariablesEstaticas.ANSI_GREEN + "Saliendo...");
                        break;
                    default:
                        System.out.println(VariablesEstaticas.ANSI_GREEN + "Opción inválida. Intente de
nuevo.");
                }
            } while (opcion != 6);

        } catch (SQLException e) {
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Error en la conexión a la base de
datos: " + e.getMessage());
        }
    }
```

Imagen N° 11: Código fuente del módulo Menú, Termometría – *fuente*: Elaboración propia

Modulo Pluviometría

Código Registro Pluviometría:

Imagen N° 12: Código del módulo Registro de Pluviometría – *fuente*: Elaboración propia

```
    // Método para agregar un nuevo registro de pluviometría
    public void agregarRegistroPluviometria(Connection con) throws SQLException {
        String query = "INSERT INTO registropluviometria (fecha, hora, volumen,
RegistroPluviometriaCol, Usuarios_legajo, Pluviometria_tipo_pluviometro) VALUES (?, ?, ?, ?, ?, ?)";
        try (PreparedStatement stmt = con.prepareStatement(query)) {
            stmt.setString(1, this.fecha);
            stmt.setString(2, this.hora);
            stmt.setDouble(3, this.volumen);
            stmt.setString(4, this.registroPluviometriaCol);
            stmt.setInt(5, this.usuariosLegajo);
            stmt.setInt(6, this.pluviometriaTipoPluviometro);

            int rowsInserted = stmt.executeUpdate();
            System.out.println(rowsInserted > 0 ? VariablesEstaticas.ANSI_GREEN + "Registro de
pluviometría agregado exitosamente!" : "Error al agregar el registro.");
        }
    }

    // Método para listar todos los registros de pluviometría
    public void listarRegistrosPluviometria(Connection con) throws SQLException {
        String query = "SELECT * FROM registropluviometria";
        try (PreparedStatement stmt = con.prepareStatement(query);
             ResultSet rs = stmt.executeQuery()) {

            System.out.println(VariablesEstaticas.ANSI_GREEN + "Listado de registros de
pluviometría:");
            while (rs.next()) {
                System.out.println(VariablesEstaticas.ANSI_GREEN + "Fecha: " + rs.getString("fecha") +
                                   ", Hora: " + rs.getString("hora") +
                                   ", Volumen: " + rs.getDouble("volumen") +
                                   ", Código de Registro: " + rs.getString("RegistroPluviometriaCol") +
                                   ", Legajo de Usuario: " + rs.getInt("Usuarios_legajo") +
                                   ", Tipo de Pluviómetro: " +
rs.getInt("Pluviometria_tipo_pluviometro"));
            }
        }
    }

    // Método para eliminar un registro de pluviometría por su fecha y hora
    public void eliminarRegistroPluviometria(Connection con, String fecha, String hora) throws
SQLException {
        String query = "DELETE FROM registropluviometria WHERE fecha = ? AND hora = ?";
        try (PreparedStatement stmt = con.prepareStatement(query)) {
            stmt.setString(1, fecha);
            stmt.setString(2, hora);

            int rowsDeleted = stmt.executeUpdate();
            System.out.println(rowsDeleted > 0 ? VariablesEstaticas.ANSI_GREEN + "Registro de
pluviometría eliminado exitosamente!" : "Registro de pluviometría no encontrado.");
        }
    }

    // Método para actualizar un registro de pluviometría
    public void actualizarRegistroPluviometria(Connection con, String fecha, String hora) throws
SQLException {
        Scanner scanner = new Scanner(System.in);
        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese el nuevo volumen de agua colectado:
");
        this.volumen = scanner.nextDouble();
        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese el nuevo tipo de pluviómetro: ");
        this.pluviometriaTipoPluviometro = scanner.nextInt();
        scanner.nextLine();
        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese el nuevo código de registro: ");
        this.registroPluviometriaCol = scanner.nextLine();

        String query = "UPDATE registropluviometria SET volumen = ?, RegistroPluviometriaCol = ?,
Pluviometria_tipo_pluviometro = ? WHERE fecha = ? AND hora = ?";
        try (PreparedStatement stmt = con.prepareStatement(query)) {
            stmt.setDouble(1, this.volumen);
            stmt.setString(2, this.registroPluviometriaCol);
            stmt.setInt(3, this.pluviometriaTipoPluviometro);
            stmt.setString(4, fecha);
            stmt.setString(5, hora);

            int rowsUpdated = stmt.executeUpdate();
            System.out.println(rowsUpdated > 0 ? VariablesEstaticas.ANSI_GREEN + "Registro de
pluviometría actualizado exitosamente!" : "Registro de pluviometría no encontrado.");
        }
    }

    // Método para consultar un registro de pluviometría por fecha y hora
    public void consultarRegistroPluviometria(Connection con, String fecha, String hora) throws
SQLException {
        String query = "SELECT * FROM registropluviometria WHERE fecha = ? AND hora = ?";
        try (PreparedStatement stmt = con.prepareStatement(query)) {
            stmt.setString(1, fecha);
            stmt.setString(2, hora);

            try (ResultSet rs = stmt.executeQuery()) {
                if (rs.next()) {
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Registro encontrado:");
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Fecha: " +
rs.getString("fecha"));
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Hora: " +
rs.getString("hora"));
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Volumen: " +
rs.getDouble("volumen"));
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Código de Registro: " +
rs.getString("RegistroPluviometriaCol"));
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Legajo de Usuario: " +
rs.getInt("Usuarios_legajo"));
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Tipo de Pluviómetro: " +
rs.getInt("Pluviometria_tipo_pluviometro"));
                } else {
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Registro de pluviometría no
encontrado.");
                }
            }
        }
```

Imagen N° 13: Código fuente del módulo Registro Pluviometría, métodos manejo de datos – *fuente*: Elaboración propia

```java
    // Menú de opciones para gestionar los registros de pluviometría
    public void menuRegistroPluviometria() {
        try (Connection con = ConexionBD.obtenerConexion()) {
            Scanner scanner = new Scanner(System.in);
            int opcion;

            do {
                System.out.println(VariablesEstaticas.ANSI_GREEN + "===== Gestión de Registros de
Pluviometría =====");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "1. Agregar registro");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "2. Eliminar registro");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "3. Actualizar registro");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "4. Consultar registro");
                System.out.println(VariablesEstaticas.ANSI_GREEN +"5. Listar todos los registros");
                System.out.println(VariablesEstaticas.ANSI_GREEN +"6. Gestionar Pluviómetro");
                System.out.println(VariablesEstaticas.ANSI_GREEN +"7. Salir");
                System.out.print(VariablesEstaticas.ANSI_GREEN +"Seleccione una opción: ");
                opcion = scanner.nextInt();

                switch (opcion) {
                    case 1:
                        capturarDatosRegistroPluviometria(con);
                        break;
                    case 2:
                        scanner.nextLine();
                        System.out.print("Ingrese la fecha del registro a eliminar (AAAA-MM-DD): ");
                        String fechaEliminar = scanner.nextLine();
                        System.out.print("Ingrese la hora del registro a eliminar (HH:MM): ");
                        String horaEliminar = scanner.nextLine();
                        eliminarRegistroPluviometria(con, fechaEliminar, horaEliminar);
                        break;
                    case 3:
                        scanner.nextLine();
                        System.out.print("Ingrese la fecha del registro a actualizar (AAAA-MM-DD): ");
                        String fechaActualizar = scanner.nextLine();
                        System.out.print("Ingrese la hora del registro a actualizar (HH:MM): ");
                        String horaActualizar = scanner.nextLine();
                        actualizarRegistroPluviometria(con, fechaActualizar, horaActualizar);
                        break;
                    case 4:
                        scanner.nextLine();
                        System.out.print("Ingrese la fecha del registro a consultar (AAAA-MM-DD): ");
                        String fechaConsulta = scanner.nextLine();
                        System.out.print("Ingrese la hora del registro a consultar (HH:MM): ");
                        String horaConsulta = scanner.nextLine();
                        consultarRegistroPluviometria(con, fechaConsulta, horaConsulta);
                        break;
                    case 5:
                        listarRegistrosPluviometria(con);
                        break;
                    case 6:
                        Pluviometria pluviometria = new Pluviometria();
                        pluviometria.menuPluviometria();
                        break;
                    case 7:
                        System.out.println(VariablesEstaticas.ANSI_GREEN + "Saliendo del menú de
registros de pluviometría...");
                        break;
                    default:
                        System.out.println("Opción inválida, intente nuevamente.");
                }
            } while (opcion != 7);
        } catch (SQLException e) {
            System.out.println("Error en la gestión de registros de pluviometría: " + e.getMessage());
        }
    }
```

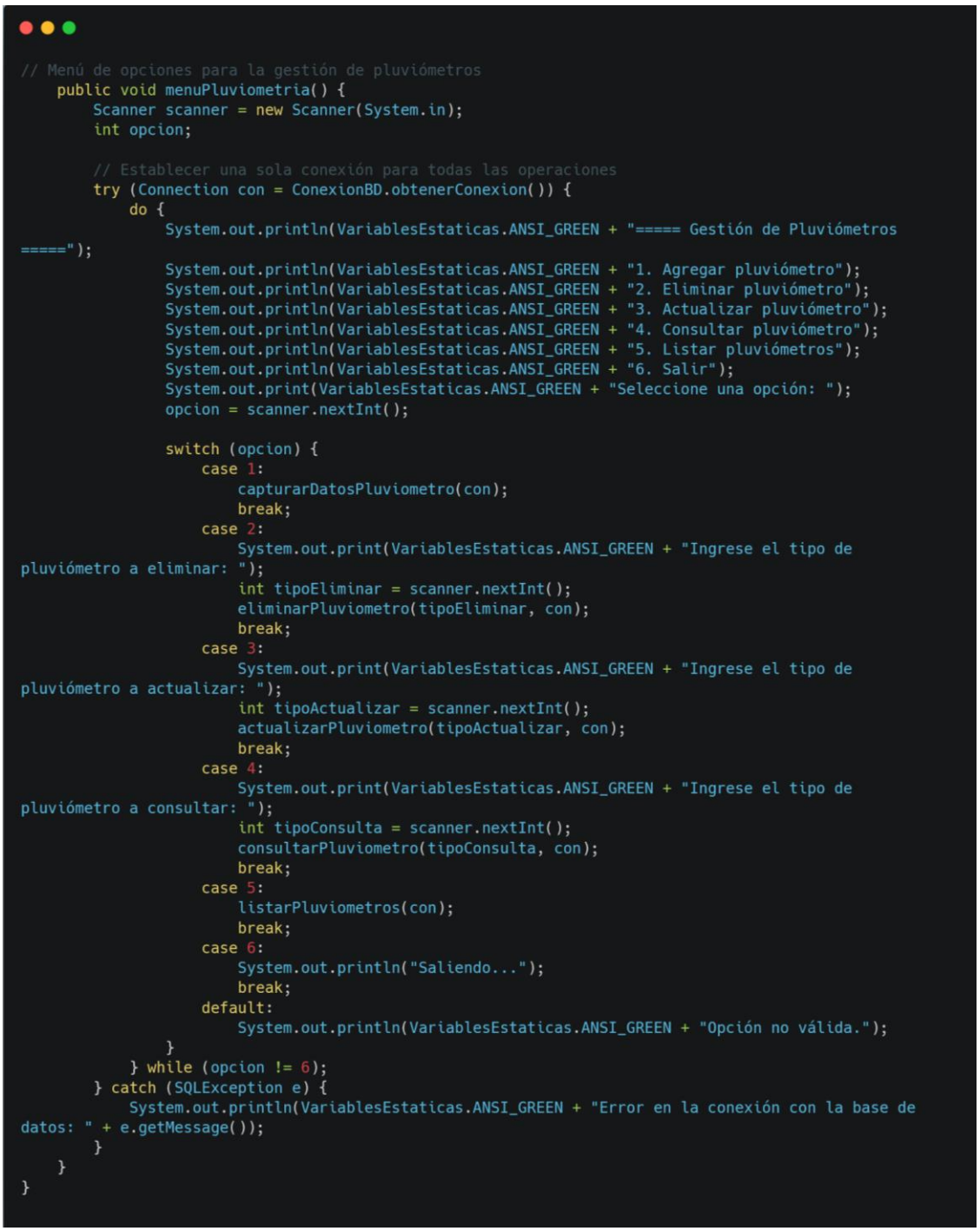
Imagen N° 14: Código fuente del módulo Registro Pluviometría, método menú – *fuente*: Elaboración propia.

```
package tp3;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Scanner;

public class Pluviometria {
    private int tipoPluviometro;
    private double capacidad;
    private String descripcion;

    public Pluviometria() {}

    // Captura de datos desde la consola
    public void capturarDatosPluviometro(Connection con) {
        Scanner scanner = new Scanner(System.in);
        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese el tipo de pluviómetro: ");
        this.tipoPluviometro = scanner.nextInt();
        scanner.nextLine();
        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese la descripción del pluviómetro: ");
        this.descripcion = scanner.nextLine();
        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese la capacidad del pluviómetro: ");
        this.capacidad = scanner.nextDouble();
        agregarPluviometro(con);
    }

    // Agregar un nuevo pluviómetro
    public void agregarPluviometro(Connection con) {
        String query = "INSERT INTO pluviometria (tipo_pluviometro, capacidad, descripcion) VALUES (?,
?, ?)";

        try (PreparedStatement stmt = con.prepareStatement(query)) {
            stmt.setInt(1, this.tipoPluviometro);
            stmt.setDouble(2, this.capacidad);
            stmt.setString(3, this.descripcion);

            int rowsInserted = stmt.executeUpdate();
            System.out.println(rowsInserted > 0 ? VariablesEstaticas.ANSI_GREEN + "Pluviómetro agregado
exitosamente!" : "Error al agregar pluviómetro.");
        } catch (SQLException e) {
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Error al agregar pluviómetro: " +
e.getMessage());
        }
    }

    // Listar todos los pluviómetros
    public void listarPluviometros(Connection con) {
        String query = "SELECT * FROM pluviometria";

        try (PreparedStatement stmt = con.prepareStatement(query);
             ResultSet rs = stmt.executeQuery()) {

            System.out.println(VariablesEstaticas.ANSI_GREEN + "Listado de pluviómetros:");
            while (rs.next()) {
                System.out.println(VariablesEstaticas.ANSI_GREEN + "Tipo: " +
rs.getInt("tipo_pluviometro") +
                                   ", Capacidad: " + rs.getDouble("capacidad") +
                                   ", Descripción: " + rs.getString("descripcion"));
            }
        } catch (SQLException e) {
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Error al listar pluviómetros: " +
e.getMessage());
        }
    }

    // Eliminar un pluviómetro
    public void eliminarPluviometro(int tipoPluviometro, Connection con) {
        String query = "DELETE FROM pluviometria WHERE tipo_pluviometro = ?";

        try (PreparedStatement stmt = con.prepareStatement(query)) {
            stmt.setInt(1, tipoPluviometro);
            int rowsDeleted = stmt.executeUpdate();
            System.out.println(rowsDeleted > 0 ? VariablesEstaticas.ANSI_GREEN + "Pluviómetro eliminado
exitosamente!" : "Pluviómetro no encontrado.");
        } catch (SQLException e) {
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Error al eliminar el pluviómetro: " +
e.getMessage());
        }
    }

    // Actualizar un pluviómetro
    public void actualizarPluviometro(int tipoPluviometro, Connection con) {
        Scanner scanner = new Scanner(System.in);
        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese la nueva descripción: ");
        this.descripcion = scanner.nextLine();
        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese la nueva capacidad: ");
        this.capacidad = scanner.nextDouble();

        String query = "UPDATE pluviometria SET descripcion = ?, capacidad = ? WHERE tipo_pluviometro =
?";

        try (PreparedStatement stmt = con.prepareStatement(query)) {
            stmt.setString(1, this.descripcion);
            stmt.setDouble(2, this.capacidad);
            stmt.setInt(3, tipoPluviometro);

            int rowsUpdated = stmt.executeUpdate();
            System.out.println(rowsUpdated > 0 ? VariablesEstaticas.ANSI_GREEN + "Pluviómetro
actualizado exitosamente!" : "Pluviómetro no encontrado.");
        } catch (SQLException e) {
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Error al actualizar el pluviómetro: " +
e.getMessage());
        }
    }

    // Consultar un pluviómetro específico
    public void consultarPluviometro(int tipoPluviometro, Connection con) {
        String query = "SELECT * FROM pluviometria WHERE tipo_pluviometro = ?";

        try (PreparedStatement stmt = con.prepareStatement(query)) {
            stmt.setInt(1, tipoPluviometro);

            try (ResultSet rs = stmt.executeQuery()) {
                if (rs.next()) {
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Pluviómetro encontrado:");
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Tipo: " +
rs.getInt("tipo_pluviometro"));
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Capacidad: " +
rs.getDouble("capacidad"));
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Descripción: " +
rs.getString("descripcion"));
                } else {
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Pluviómetro no encontrado.");
                }
            }
        } catch (SQLException e) {
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Error al consultar el pluviómetro: " +
e.getMessage());
        }
    }

    // Menú de opciones para la gestión de pluviómetros
    public void menuPluviometria() {
        Scanner scanner = new Scanner(System.in);
        int opcion;

        // Establecer una sola conexión para todas las operaciones
        try (Connection con = ConexionBD.obtenerConexion()) {
            do {
                System.out.println(VariablesEstaticas.ANSI_GREEN + "===== Gestión de Pluviómetros
=====");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "1. Agregar pluviómetro");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "2. Eliminar pluviómetro");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "3. Actualizar pluviómetro");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "4. Consultar pluviómetro");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "5. Listar pluviómetros");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "6. Salir");
                System.out.print(VariablesEstaticas.ANSI_GREEN + "Seleccione una opción: ");
                opcion = scanner.nextInt();

                switch (opcion) {
                    case 1:
                        capturarDatosPluviometro(con);
                        break;
                    case 2:
                        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese el tipo de
pluviómetro a eliminar: ");
                        int tipoEliminar = scanner.nextInt();
                        eliminarPluviometro(tipoEliminar, con);
                        break;
                    case 3:
                        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese el tipo de
pluviómetro a actualizar: ");
                        int tipoActualizar = scanner.nextInt();
                        actualizarPluviometro(tipoActualizar, con);
                        break;
                    case 4:
                        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese el tipo de
pluviómetro a consultar: ");
                        int tipoConsulta = scanner.nextInt();
                        consultarPluviometro(tipoConsulta, con);
                        break;
                    case 5:
                        listarPluviometros(con);
                        break;
                    case 6:
                        System.out.println("Saliendo...");
                        break;
                    default:
                        System.out.println(VariablesEstaticas.ANSI_GREEN + "Opción no válida.");
                }
            } while (opcion != 6);
        } catch (SQLException e) {
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Error en la conexión con la base de
datos: " + e.getMessage());
        }
    }
}
```

Imagen N° 15: Código fuente del módulo Pluviometría – *fuente*: Elaboración propia.

```java
// Agregar un nuevo pluviómetro
    public void agregarPluviometro(Connection con) {
        String query = "INSERT INTO pluviometria (tipo_pluviometro, capacidad, descripcion) VALUES (?,
?, ?)";

        try (PreparedStatement stmt = con.prepareStatement(query)) {
            stmt.setInt(1, this.tipoPluviometro);
            stmt.setDouble(2, this.capacidad);
            stmt.setString(3, this.descripcion);

            int rowsInserted = stmt.executeUpdate();
            System.out.println(rowsInserted > 0 ? VariablesEstaticas.ANSI_GREEN + "Pluviómetro agregado
exitosamente!" : "Error al agregar pluviómetro.");
        } catch (SQLException e) {
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Error al agregar pluviómetro: " +
e.getMessage());
        }
    }

    // Listar todos los pluviómetros
    public void listarPluviometros(Connection con) {
        String query = "SELECT * FROM pluviometria";

        try (PreparedStatement stmt = con.prepareStatement(query);
             ResultSet rs = stmt.executeQuery()) {

            System.out.println(VariablesEstaticas.ANSI_GREEN + "Listado de pluviómetros:");
            while (rs.next()) {
                System.out.println(VariablesEstaticas.ANSI_GREEN + "Tipo: " +
rs.getInt("tipo_pluviometro") +
                                   ", Capacidad: " + rs.getDouble("capacidad") +
                                   ", Descripción: " + rs.getString("descripcion"));
            }
        } catch (SQLException e) {
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Error al listar pluviómetros: " +
e.getMessage());
        }
    }

    // Eliminar un pluviómetro
    public void eliminarPluviometro(int tipoPluviometro, Connection con) {
        String query = "DELETE FROM pluviometria WHERE tipo_pluviometro = ?";

        try (PreparedStatement stmt = con.prepareStatement(query)) {
            stmt.setInt(1, tipoPluviometro);
            int rowsDeleted = stmt.executeUpdate();
            System.out.println(rowsDeleted > 0 ? VariablesEstaticas.ANSI_GREEN + "Pluviómetro eliminado
exitosamente!" : "Pluviómetro no encontrado.");
        } catch (SQLException e) {
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Error al eliminar el pluviómetro: " +
e.getMessage());
        }
    }

    // Actualizar un pluviómetro
    public void actualizarPluviometro(int tipoPluviometro, Connection con) {
        Scanner scanner = new Scanner(System.in);
        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese la nueva descripción: ");
        this.descripcion = scanner.nextLine();
        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese la nueva capacidad: ");
        this.capacidad = scanner.nextDouble();

        String query = "UPDATE pluviometria SET descripcion = ?, capacidad = ? WHERE tipo_pluviometro =
?";

        try (PreparedStatement stmt = con.prepareStatement(query)) {
            stmt.setString(1, this.descripcion);
            stmt.setDouble(2, this.capacidad);
            stmt.setInt(3, tipoPluviometro);

            int rowsUpdated = stmt.executeUpdate();
            System.out.println(rowsUpdated > 0 ? VariablesEstaticas.ANSI_GREEN + "Pluviómetro
actualizado exitosamente!" : "Pluviómetro no encontrado.");
        } catch (SQLException e) {
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Error al actualizar el pluviómetro: " +
e.getMessage());
        }
    }

    // Consultar un pluviómetro específico
    public void consultarPluviometro(int tipoPluviometro, Connection con) {
        String query = "SELECT * FROM pluviometria WHERE tipo_pluviometro = ?";

        try (PreparedStatement stmt = con.prepareStatement(query)) {
            stmt.setInt(1, tipoPluviometro);

            try (ResultSet rs = stmt.executeQuery()) {
                if (rs.next()) {
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Pluviómetro encontrado:");
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Tipo: " +
rs.getInt("tipo_pluviometro"));
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Capacidad: " +
rs.getDouble("capacidad"));
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Descripción: " +
rs.getString("descripcion"));
                } else {
                    System.out.println(VariablesEstaticas.ANSI_GREEN + "Pluviómetro no encontrado.");
                }
            }
        } catch (SQLException e) {
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Error al consultar el pluviómetro: " +
e.getMessage());
        }
    }
```

Imagen N° 16: Código fuente del módulo Pluviometría, manejo de datos **ABMCL** – *fuente*: Elaboración propia.



```java
// Menú de opciones para la gestión de pluviómetros
    public void menuPluviometria() {
        Scanner scanner = new Scanner(System.in);
        int opcion;

        // Establecer una sola conexión para todas las operaciones
        try (Connection con = ConexionBD.obtenerConexion()) {
            do {
                System.out.println(VariablesEstaticas.ANSI_GREEN + "===== Gestión de Pluviómetros
=====");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "1. Agregar pluviómetro");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "2. Eliminar pluviómetro");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "3. Actualizar pluviómetro");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "4. Consultar pluviómetro");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "5. Listar pluviómetros");
                System.out.println(VariablesEstaticas.ANSI_GREEN + "6. Salir");
                System.out.print(VariablesEstaticas.ANSI_GREEN + "Seleccione una opción: ");
                opcion = scanner.nextInt();

                switch (opcion) {
                    case 1:
                        capturarDatosPluviometro(con);
                        break;
                    case 2:
                        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese el tipo de
pluviómetro a eliminar: ");
                        int tipoEliminar = scanner.nextInt();
                        eliminarPluviometro(tipoEliminar, con);
                        break;
                    case 3:
                        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese el tipo de
pluviómetro a actualizar: ");
                        int tipoActualizar = scanner.nextInt();
                        actualizarPluviometro(tipoActualizar, con);
                        break;
                    case 4:
                        System.out.print(VariablesEstaticas.ANSI_GREEN + "Ingrese el tipo de
pluviómetro a consultar: ");
                        int tipoConsulta = scanner.nextInt();
                        consultarPluviometro(tipoConsulta, con);
                        break;
                    case 5:
                        listarPluviometros(con);
                        break;
                    case 6:
                        System.out.println("Saliendo...");
                        break;
                    default:
                        System.out.println(VariablesEstaticas.ANSI_GREEN + "Opción no válida.");
                }
            } while (opcion != 6);
        } catch (SQLException e) {
            System.out.println(VariablesEstaticas.ANSI_GREEN + "Error en la conexión con la base de
datos: " + e.getMessage());
        }
    }
}
```

Imagen N° 17: Código fuente del módulo Pluviometría, Menú – *fuente*: Elaboración propia.

Novedades del Proyecto

Buen día,
Con mucha alegría queremos contarles que esta nuevamente operativo el SIA. Gracias al trabajo principalmente de [redacted] con la colaboración de Patricio y Vane está nuevamente activa la plataforma de carga de los observatorios convencionales de la Red de INTA.

Por este motivo, los invitamos a que comiencen a cargar los datos diarios del mes de noviembre. Les solicitamos que por favor a medida que vayan cargando, corroboren que los datos impacten en el SIGA. En el caso de encontrar dificultades, errores, problemas con la carga de datos, con el usuario o con la contraseña por favor se comuniquen con Vanesa.

Los datos de los observatorios que han sido oportunamente enviados desde el hackeo hasta el mes de octubre 2024 serán actualizados en la base de manera automática a partir de las libretas digitales enviadas por cada observatorio.

Los observatorios que no han enviado libretas digitales desde el hackeo, pero quieren completar los datos desde el hackeo a octubre 2024 en el SIA, por favor comunicarse con Vane.

Muchas gracias a todos por su labor diaria!
Estamos disponibles para cualquier consulta.
Cariños
L[redacted]

Imagen N° 18: Captura de imagen del mail de la coordinación de este área -.

La verdad es una alegría que los muchachos de desarrollo de nuestra institución, luego de un cierto tiempo hayan podido poner en línea el sistema de carga, con los ajustes necesarios como la seguridad, los datos ahí cargados y mostrados cuentan con un alto valor estratégico. Este mail fue de esta semana. –

Se intentará ver futuras necesidades para el desarrollo de soluciones funcionales con lo aprendido, esto fue una buena experiencia. -

Video

https://youtu.be/X03dDCdwE88

GitHub

Principal del Proyecto: https://github.com/leandroariel14/TrabajoS21

Código: https://github.com/leandroariel14/TrabajoS21/tree/main/src

Conclusión personal sobre el proyecto:

Como experiencia personal ha sido un gran desafío para mi tener que dedicar el aprendizaje a un lenguaje totalmente diferente a los que manejaba, si bien la teoría de el paradigma de orientado de objetos la hemos visto en etapas anteriores, ponerlo en práctica ha sido un poco complejo, ha sido un nuevo desafío para mí, el lenguaje estructurado con el cual yo tuve mis pocas experiencias como programador me ha condicionado a pensar de otra forma, para mí esto fue tratar de comprender y ver de otra manera, otra visión, si bien los tiempos son exigentes por cuestiones de agenda, ha sido muy gratificante, me ha costado un poco conectar la base de datos, me hubiera gustado hacer una **GUI**, entiendo que los tiempos exigían una vista al estilo consola, igual fue una muy buena experiencia, he utilizado los recursos disponibles que solo el lenguaje y la filosofía *OPEN SOURCE* permiten . -

Nota: MySQL y JAVA son grandes soluciones para el desarrollo de aplicaciones, con muy bajo costo operativo, mucha documentación, muchas alternativas para el desarrollo, como netbeans, eclipse, workbench o phpmyadmin o paquetes enteros como xampp.-