INSTITUTO TECNOLÓGICO SUPERIOR CENESTUR
Desde 1987
Guía • Turismo • Hotelería • Gastronomía

CARRERA DESARROLLO DE SOFTWARE

MATERIA PROGRAMACION ORIENTADA A OBJETOS

NOMBRE DE LA ACTIVIDAD

PROYECTO FINAL

APELLIDOS, NOMBRES

ANDREA CAISA

JOSSELYN QUIÑONEZ

CORREO ELECTRÓNICO ESTUDIANTE

josselyn.quinonez@cenestur.edu.ec

andrea.caisa@cenestur.edu.ec

PROFESOR:

Yadira Guissela Franco Rocha

Quito, Ecuador

2025

INTRODUCCION:

Llevar un control eficiente de las calificaciones es clave en cualquier institución educativa. Por eso, creamos una aplicación para escritorio en Java Swing que hace mucho más fácil ingresar, buscar, actualizar y eliminar notas escolares. Esta aplicación se conecta a una base de datos confiable, asegurando que la información académica se mantenga segura y siempre esté actualizada. Además, es muy fácil de usar, pensada para profesores y administradores que necesitan una herramienta práctica y confiable.

OJETIVO GENERAL:

Crea una aplicación de escritorio usando Java Swing que te permita gestionar las notas escolares de manera sencilla y eficiente. La app debería permitirte agregar, consultar, actualizar y eliminar notas fácilmente, asegurando que los datos se guarden correctamente y no se pierdan, conectándose a una base de datos para mantener todo seguro y organizado.

OBJETIVOS ESPECIFICOS:

* Diseñar y construir una interfaz gráfica de usuario (GUI) intuitiva y fácil de usar, utilizando componentes de Java Swing, para facilitar el ingreso y la visualización de las notas escolares.

* Implementar la funcionalidad de añadir nuevas calificaciones, incluyendo la asociación con estudiantes y asignaturas, validando la entrada de datos para asegurar su formato correcto.

* Desarrollar la capacidad de consultar y mostrar las notas existentes en la base de datos, permitiendo filtros por estudiante, asignatura o período académico.

* Codificar las funcionalidades de actualización de las calificaciones, permitiendo a los usuarios modificar notas ya registradas.

* Programar la opción de eliminar registros de notas, incorporando una confirmación para prevenir borrados accidentales.

DESARROLLO:

El pilar fundamental para la persistencia de los datos es la clase ConexionBD, ubicada en el paquete ConexionSQL. Esta clase es la responsable de establecer y gestionar la conexión con nuestra base de datos MySQL. Para ello, utiliza la API JDBC (Java Database Connectivity) y configura parámetros esenciales comola URL de la base de datos el usuario (root) y la contraseña. El método estático getConnection() carga dinámicamente el driver de MySQL y crea una conexión que puede ser utilizada por otras partes de la aplicación. Además, incorpora un manejo de excepciones (ClassNotFoundException, SQLException) para asegurar que cualquier problema de conexión sea detectado e informado, contribuyendo a la estabilidad general del sistema. En esencia, ConexionBD actúa como el puente entre la aplicación y la base de datos, garantizando la disponibilidad y la integridad de la información.

Gestión de Datos de Estudiantes (EstudianteDAO). Para interactuar con la información específica de los estudiantes en la base de datos, se implementó la clase EstudianteDAO (Data Access Object), situada en el paquete Usuarios. Esta clase encapsula toda la lógica de acceso a datos para la entidad Estudiante, lo que significa que aísla las operaciones de base de datos del resto de la lógica de negocio de la aplicación. EstudianteDAO proporciona un conjunto de métodos claros y específicos para realizar las operaciones CRUD (Crear, Leer, Actualizar y Eliminar):

* El método crear() permite insertar nuevos registros de estudiantes en la tabla estudiantes.

* leerTodos() recupera una lista completa de todos los estudiantes registrados.

* actualizar() modifica los datos de un estudiante existente.

* eliminar() borra un registro de estudiante de la base de datos.

Todos estos métodos utilizan PreparedStatement para ejecutar consultas SQL de manera segura y eficiente, previniendo vulnerabilidades como la inyección SQL. El manejo de excepciones en cada operación asegura que los errores de la base de datos sean controlados, mejorando la robustez de la aplicación y

manteniendo la separación de responsabilidades, lo que facilita el mantenimiento y la escalabilidad del sistema.

Interfaz de Administración (Administrador)

La clase Administrador, ubicada en el paquete Ventanas, representa la interfaz principal del panel de control para los usuarios con privilegios administrativos. Esta ventana, desarrollada con Java Swing, proporciona un acceso centralizado a las diferentes funcionalidades del sistema de gestión de notas.

El constructor de la clase invoca initComponents() para configurar la interfaz gráfica. La ventana se titula "Panel de Administración", se centra en la pantalla y organiza sus componentes utilizando un GridLayout para una disposición vertical clara de los botones.

Los elementos visuales incluyen un título (JLabel) y múltiples botones (JButton) que representan las diferentes secciones del sistema:

* "Estudiantes": Para la gestión de la información de los alumnos.

* "Materias": Para administrar las asignaturas.

* "Notas": Para el ingreso y consulta de las calificaciones.

* "Inscripciones de alumno a materia": Para relacionar estudiantes con sus respectivas materias.

* "Gestion Usuarios": Para la administración de cuentas de usuario.

* "Cerrar Sesión": Un botón con estilo distintivo que permite al usuario salir de la sesión actual.

El método configurarListeners() asocia eventos de clic a cada botón, abriendo la ventana correspondiente (Estudiantes, FormularioMateria, FormularioNota, Inscripcion, Usuarios) cuando se activan. El botón "Cerrar Sesión" invoca el método cerrarSesion(), el cual, tras una confirmación del usuario, cierra la ventana actual del administrador y retorna a la pantalla de Login. La ejecución del panel se inicia desde el método main(), asegurando que la interfaz se cargue correctamente en el hilo de eventos de Swing..

1. **CODIGO:**

```
package ConexionSQL;


import java.sql.Connection;
import java.sql.DriverManager; //conecta con drive
import java.sql.SQLException;   // maneja errores
public class ConexionBD {
   //url para la conexion de bd
   private static final String URL = "jdbc:mysql://localhost:3306/sistema_de_notas_escolares?useSSL=false&serverTimezone=UTC";
   private static final String USER = "root";//superusuario por defecto
   private static final String PASSWORD = ""; // En XAMPP por defecto
   public static Connection getConnection() {
       Connection connection = null;
       try {
           Class.forName("com.mysql.cj.jdbc.Driver"); // establece conexion
           connection = DriverManager.getConnection(URL, USER, PASSWORD);
       } catch (ClassNotFoundException e) {
           System.out.println("ERROR: No se encontró el driver de MySQL. " + e.getMessage());
       } catch (SQLException e) {
           System.out.println("ERROR: No se pudo conectar a la Base de Datos. " + e.getMessage());
       }
       return connection;
```

```
    }
}
```

2. CODIGO:

```
package Datos;
public class Estudiante {
    //Variables privadas
    private int idEstudiante;
    private String cedula;
    private String nombres;
    private String apellidos;
    //CONTRUCTOR VACIO PARA CREAR SIN DATOS INICIALES
    public Estudiante() {
    }
    // CONTRUCTOR CON PARAMETROS
    public Estudiante(int idEstudiante, String cedula, String nombres, String apellidos) {
        this.idEstudiante = idEstudiante;
        this.cedula = cedula;
        this.nombres = nombres;
        this.apellidos = apellidos;
    }
    // Métodos Getters y Setters para acceder y modificar las variables
    public int getIdEstudiante() {
        return idEstudiante;
    }
```

```
    public void setIdEstudiante(int idEstudiante) {
        this.idEstudiante = idEstudiante;
    }
    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;
    }
}
```

3. CODIGO:

```
package Usuarios;

import ConexionSQL.ConexionBD;  //se crea conexion con la bd

import Datos.Estudiante;

import java.sql.*;

import java.util.ArrayList;

import java.util.List;

// el dao es por dato access object para conectar con la bd

public class EstudianteDAO {

    public boolean crear(Estudiante estudiante) {

        // instrucción SQL con marcadores de posición (?)

        String sql = "INSERT INTO estudiantes (cedula, nombres, apellidos) VALUES (?, ?, ?)";

        try (Connection conn = ConexionBD.getConnection();//se obtiene la conexion a la bd

             PreparedStatement pstmt = conn.prepareStatement(sql)) {    //se prepara la instruccion

            //se reemplaza los ?? con datos del estudiante

            pstmt.setString(1, estudiante.getCedula());

            pstmt.setString(2, estudiante.getNombres());

            pstmt.setString(3, estudiante.getApellidos());

            pstmt.executeUpdate();// se guarda en bd

            return true;


        } catch (SQLException e) {

            System.out.println("Error al crear estudiante: " + e.getMessage());
```

```
            return false;
        }
    }
    // creo una lista para guardar los estudiantes
    public List<Estudiante> leerTodos() {
        List<Estudiante> estudiantes = new ArrayList<>();
        String sql = "SELECT * FROM estudiantes ORDER BY apellidos, nombres";
     //resultados con bd
        try (Connection conn = ConexionBD.getConnection();
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
                estudiantes.add(new Estudiante(
                        rs.getInt("idEstudiante"),
                        rs.getString("cedula"),
                        rs.getString("nombres"),
                        rs.getString("apellidos")
                ));
            }
        } catch (SQLException e) {
            System.out.println("Error al leer estudiantes: " + e.getMessage());
        }
        return estudiantes;
    }
```

```
//para actualizar al estudiante
    public boolean actualizar(Estudiante estudiante) {
        String sql = "UPDATE estudiantes SET cedula = ?, nombres = ?, apellidos = ? WHERE idEstudiante = ?";
        try (Connection conn = ConexionBD.getConnection();
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, estudiante.getCedula());
            pstmt.setString(2, estudiante.getNombres());
            pstmt.setString(3, estudiante.getApellidos());
            pstmt.setInt(4, estudiante.getIdEstudiante());//esto se une al where
           // se guarda actualizacion
            pstmt.executeUpdate();
            return true
        } catch (SQLException e) {
            System.out.println("Error al actualizar estudiante: " + e.getMessage());
            return false;
        }
    }
   //para eliminar estudiante
    public boolean eliminar(int id) {
        String sql = "DELETE FROM estudiantes WHERE idEstudiante = ?";
        try (Connection conn = ConexionBD.getConnection();
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setInt(1, id);
```

```
            //guarda la eliminacion
             pstmt.executeUpdate();
             return true;
        } catch (SQLException e) {
             System.out.println("Error al eliminar estudiante: " + e.getMessage());
             return false;
        }
    }
}
```

4.CODIGO:

```
package Ventanas;
import javax.swing.*;
import java.awt.*
public class Administrador extends JFrame {
    //variables privadas
    private JButton btnEstudiante;
    private JButton btnMateria;
    private JButton btnInscripcion;
    private JButton btnUsuarios;
    private JButton btnCerrarSesion;
    private JButton btnNota;
    //CONSTRUCTOR
    public Administrador() {
        // Llama a los métodos para construir y configurar la ventana
```

```
        initComponents();
        configurarListeners();
    }
    //Crea y posiciona todos los componentes visuales de la ventana.
    private void initComponents() {
        //  ventana principal
        setTitle("Panel de Administración");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // Cierra toda la app al cerrar esta ventana
        setSize(400, 500); // Tamaño de la ventana
        setLocationRelativeTo(null); // Centra la ventana en la pantalla
        setLayout(new GridLayout(7, 1, 10, 10)); // organizar los botones verticalmente
        ((JPanel) getContentPane()).setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));
        // Creación de los componentes ---
        JLabel lblTitulo = new JLabel("Panel de Control", SwingConstants.CENTER);
        lblTitulo.setFont(new Font("Arial", Font.BOLD, 24));
        btnEstudiante = new JButton("Estudiantes");
        btnMateria = new JButton("Materias");
        btnNota = new JButton("Notas");
        btnInscripcion = new JButton("Inscripciones de alumno a materia");
        btnUsuarios = new JButton("Gestion Usuarios");
        btnCerrarSesion = new JButton("Cerrar Sesión");
```

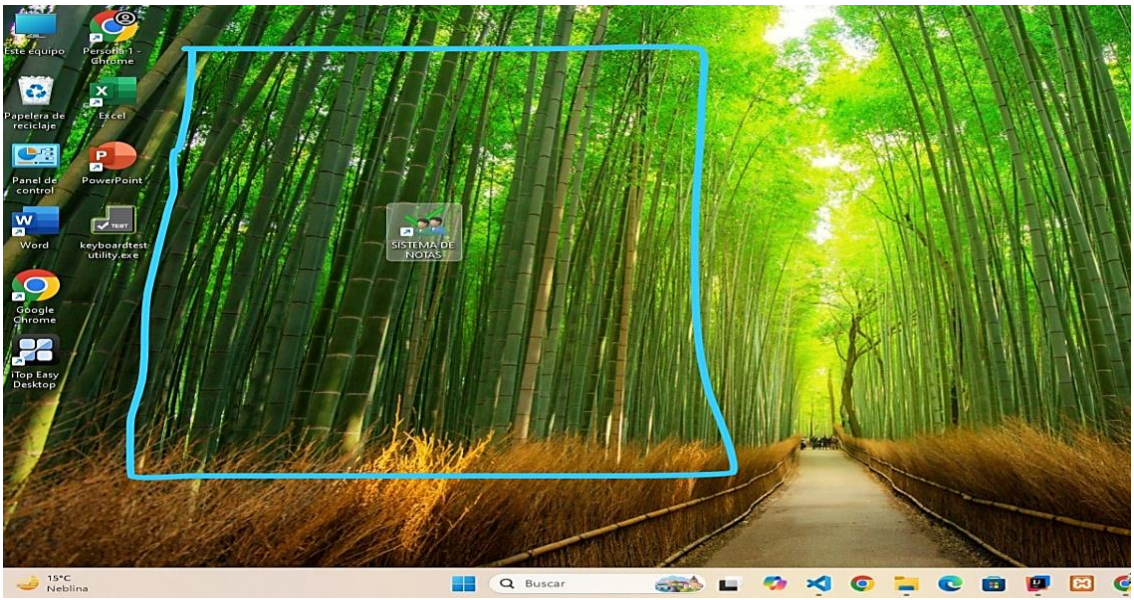
```
        // Estilo para los botones
        Font botonFont = new Font("Arial", Font.PLAIN, 16);
        btnEstudiante.setFont(botonFont);
        btnMateria.setFont(botonFont);
        btnNota.setFont(botonFont);
        btnInscripcion.setFont(botonFont);
        btnUsuarios.setFont(botonFont);
        btnCerrarSesion.setFont(botonFont);
        btnCerrarSesion.setBackground(new Color(220, 53, 69)); // Color rojo
        btnCerrarSesion.setForeground(Color.WHITE);
        // --- Añadir componentes a la ventana ---
        add(lblTitulo);
        add(btnEstudiante);
        add(btnMateria);
        add(btnNota);
        add(btnInscripcion);
        add(btnUsuarios);
        add(btnCerrarSesion);
        // Hace visible la ventana al final
        setVisible(true);
    }
    // si da click aparece esta ventana
    private void configurarListeners() {
        btnEstudiante.addActionListener(e -> new
Estudiantes().setVisible(true));
```

```
        btnMateria.addActionListener(e -> new FormularioMateria().setVisible(true));
        btnNota.addActionListener(e -> new FormularioNota().setVisible(true));
        btnInscripcion.addActionListener(e -> new Inscripcion().setVisible(true));
        btnUsuarios.addActionListener(e -> new Usuarios().setVisible(true));
        btnCerrarSesion.addActionListener(e -> cerrarSesion());
    }
    //Cierra la sesión actual y vuelve a la ventana de Login.
    private void cerrarSesion() {
        int respuesta = JOptionPane.showConfirmDialog(
                this,
                "¿Estás seguro de que quieres cerrar sesión?",
                "Confirmar Cierre",
                JOptionPane.YES_NO_OPTION,
                JOptionPane.QUESTION_MESSAGE
        );
        if (respuesta == JOptionPane.YES_OPTION) {
            new Login(); // Crea y muestra la nueva ventana de Login
            this.dispose();    // Cierra la ventana actual de Administrador
        }
    }
    //para ejeccutar
    public static void main(String[] args) {
        // Esto asegura que la ventana se cree en el hilo correcto de Swing
```

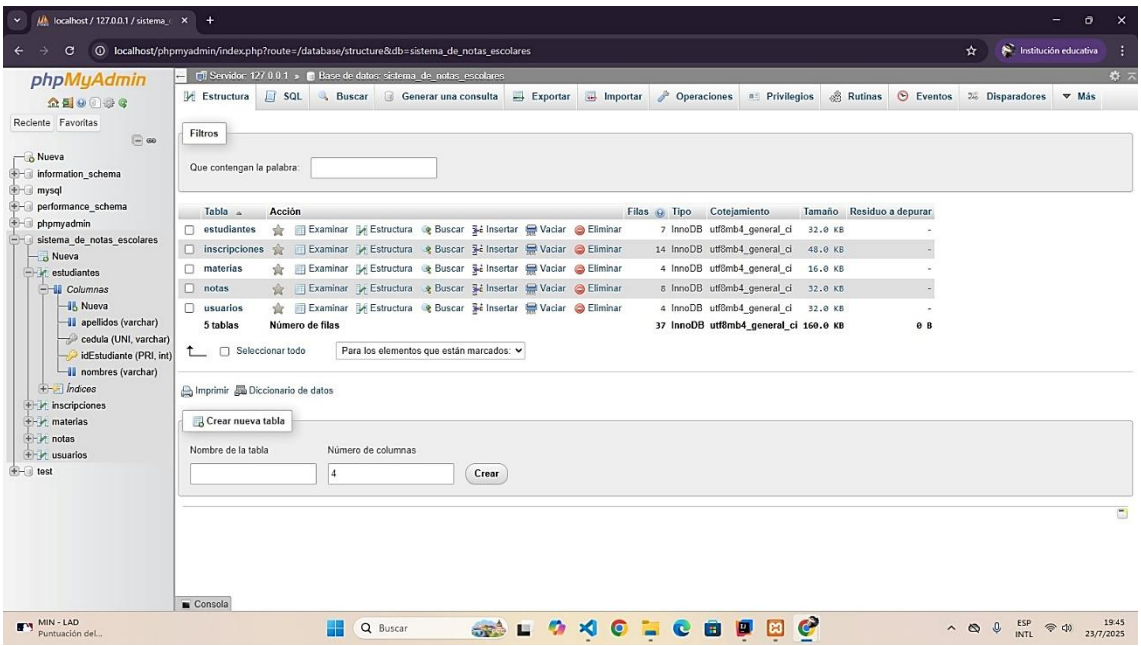
```
        SwingUtilities.invokeLater(() -> new Administrador());
    }
}
```

RESULTADOS:

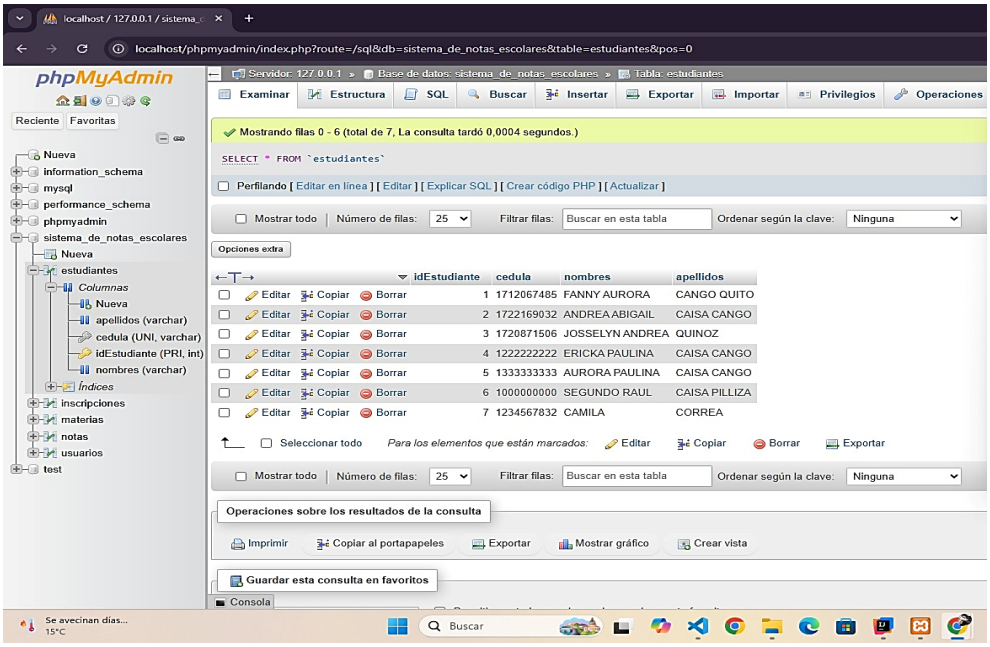
Aplicación ya transformada con el JAR para que se vea como una aplicación de escritorio.



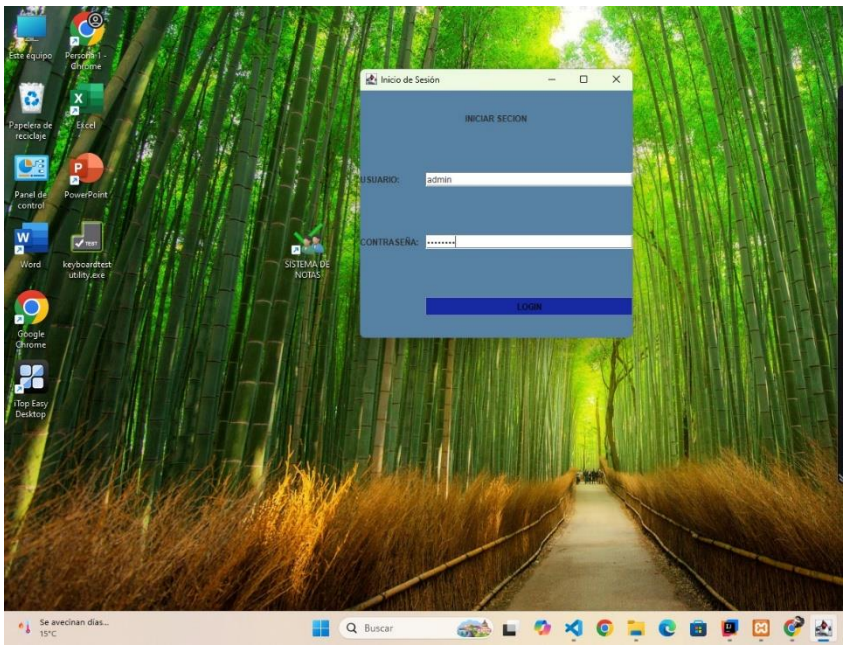
Base de datos.



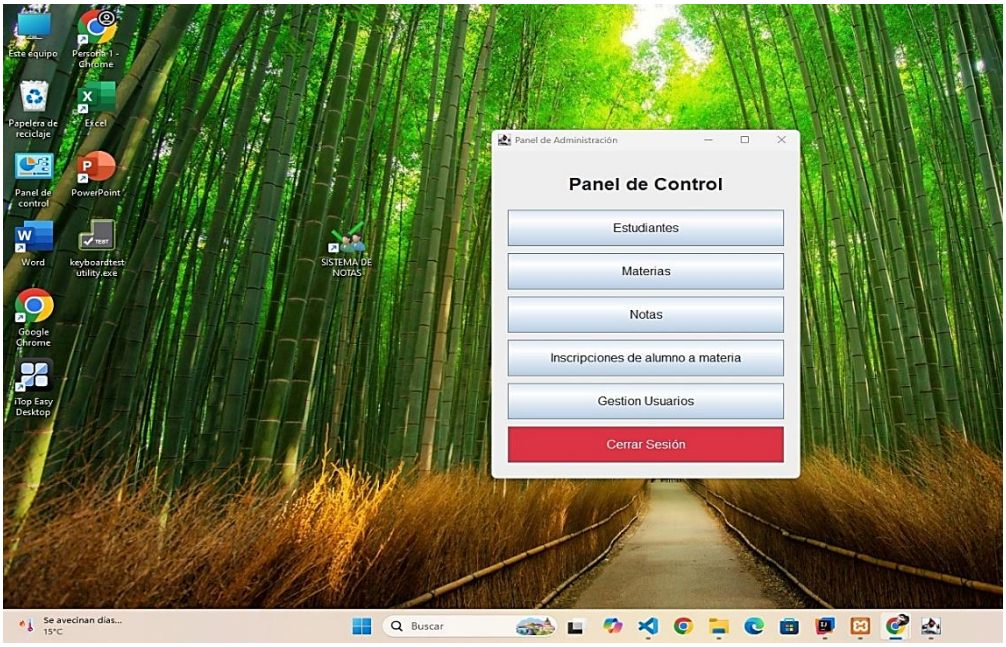
Aquí la información que se guarda.



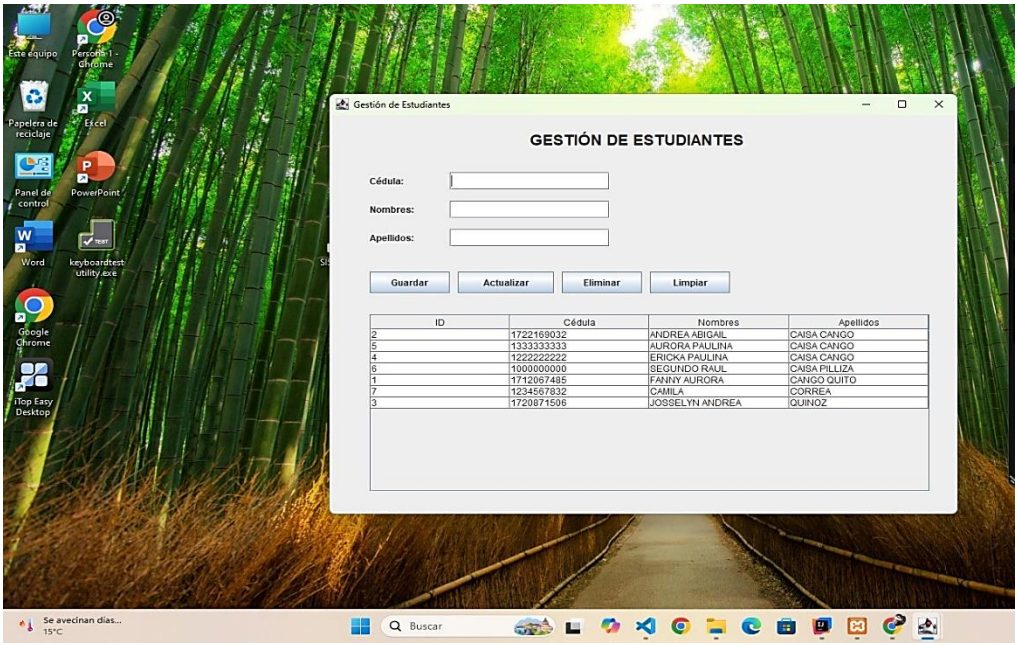
Ventana de login.

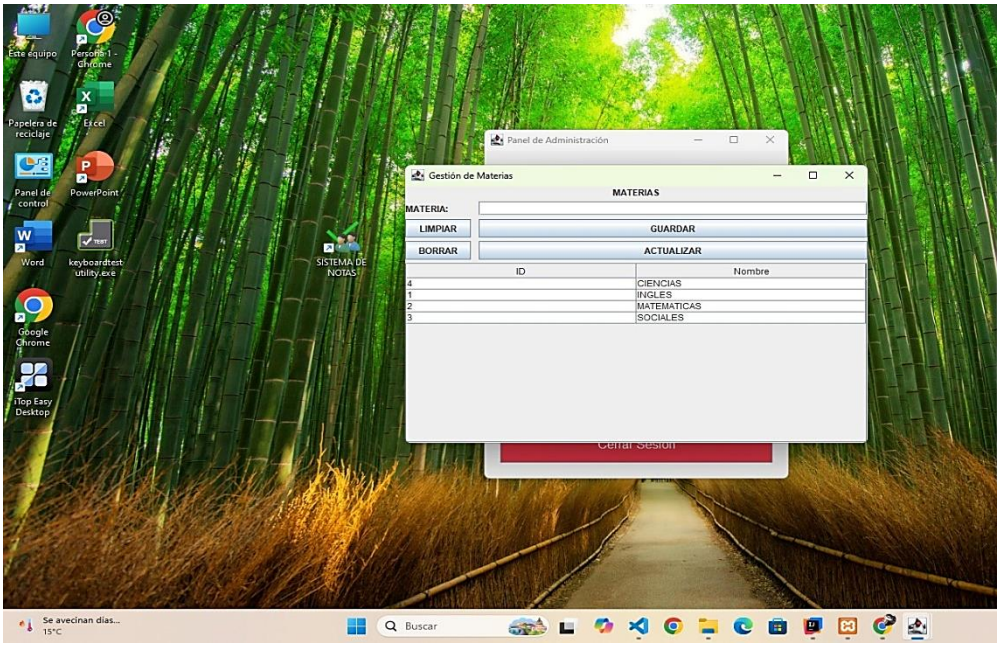


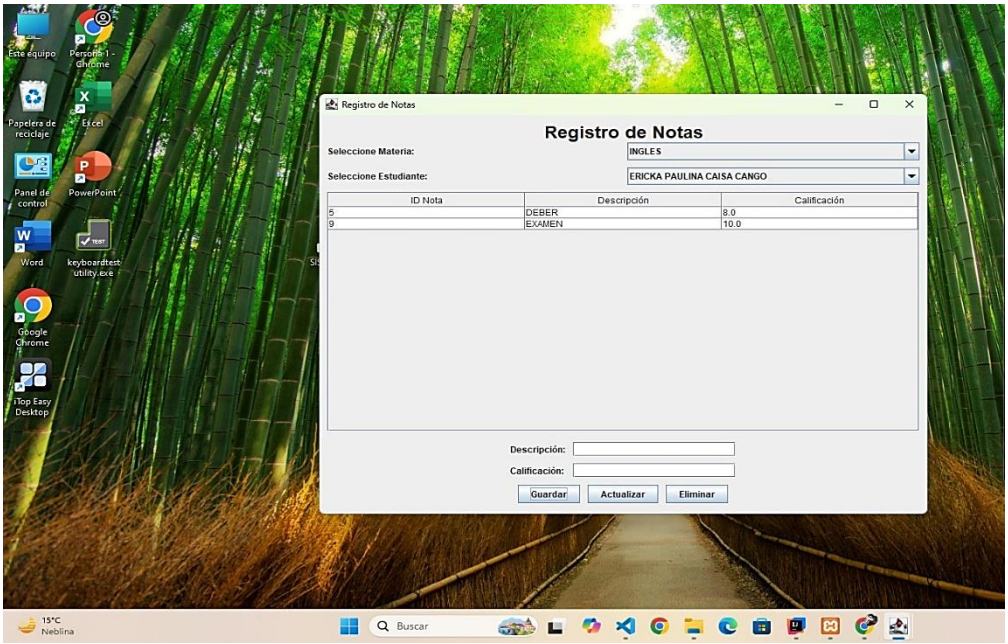
La primera ventana.

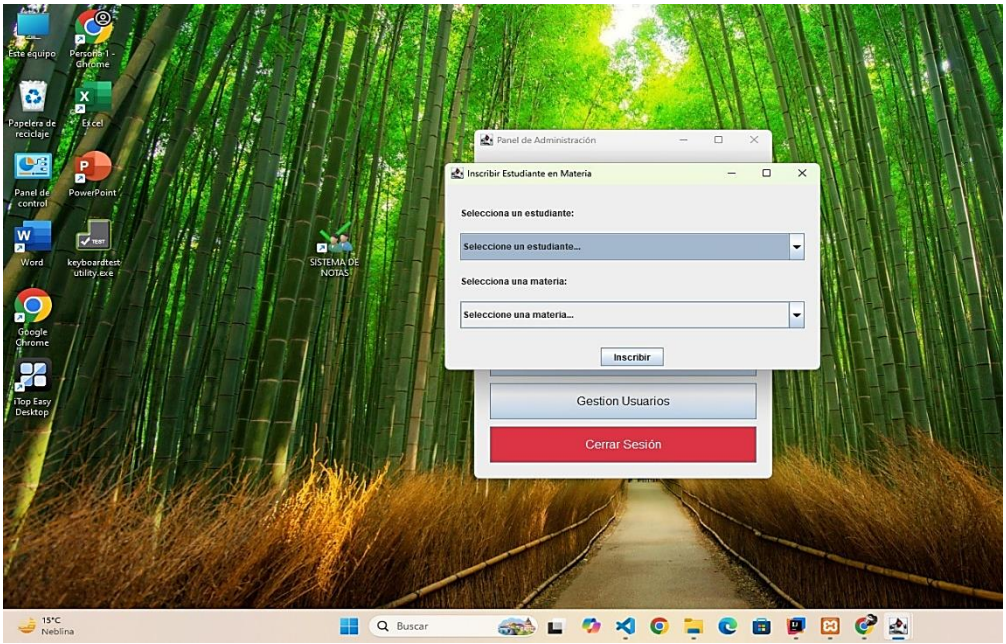


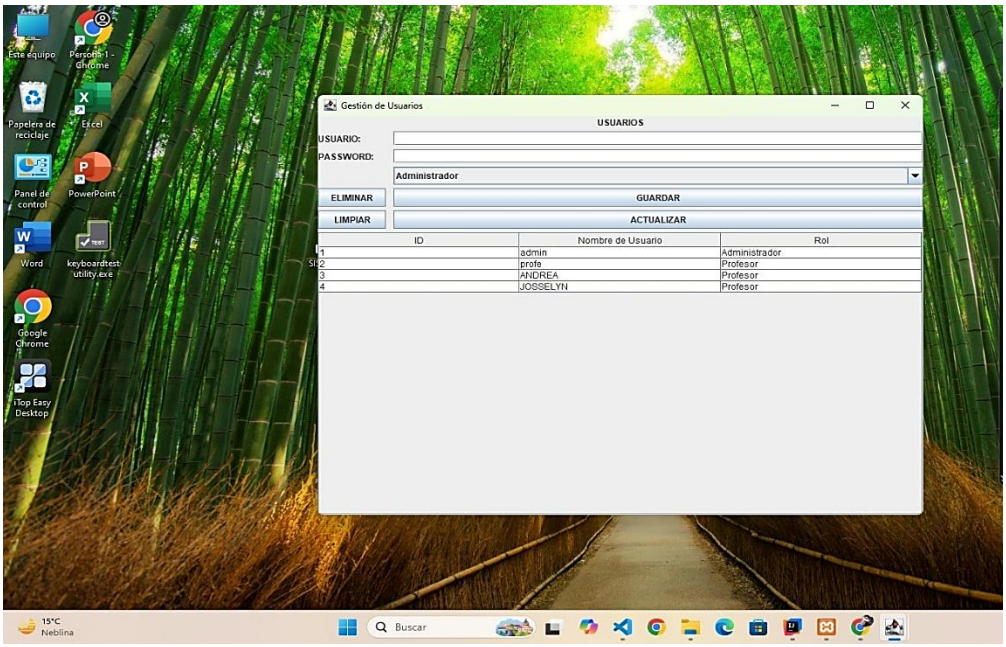
Cada botón del panel de control.



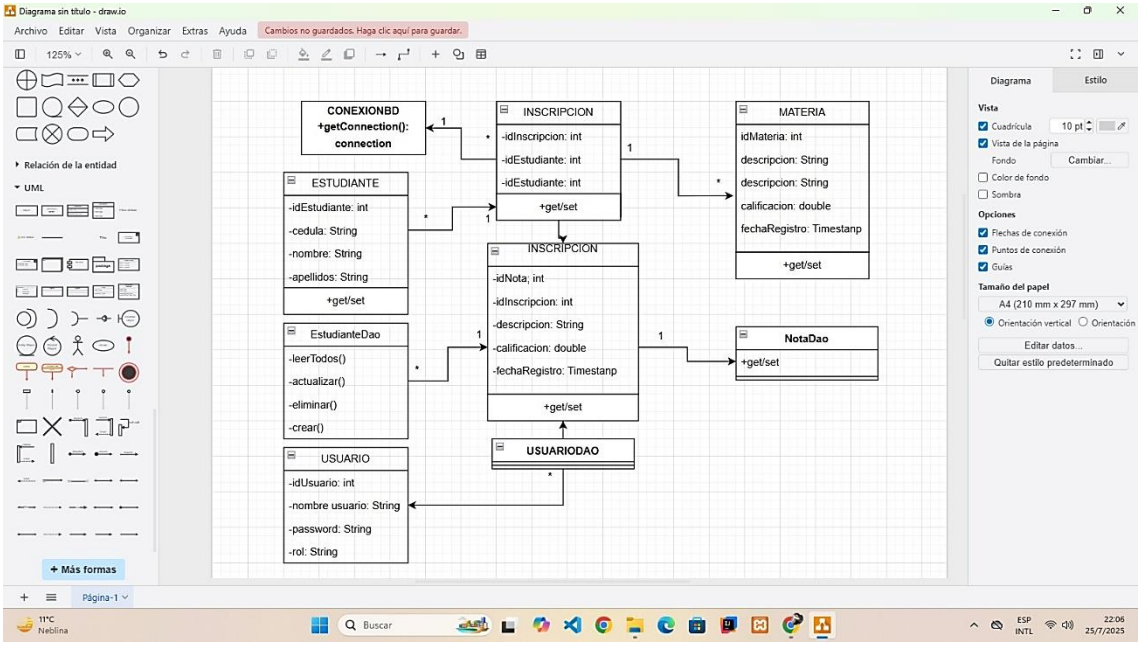

Este equipo
Persona 1 - Chrome
Papelera de reciclaje
Excel
Panel de control
PowerPoint
Word
keyboardtest-utility.exe
SISTEMA DE NOTAS
Google Chrome
iTop Easy Desktop
Panel de Administración
Gestión de Materias
MATERIAS
MATERIA:
LIMPIAR
GUARDAR
BORRAR
ACTUALIZAR
ID
Nombre
4 CIENCIAS
1 INGLES
2 MATEMATICAS
3 SOCIALES
Cerrar Sesión
Se avecinan dias...
15°C
Buscar


Este equipo
Persona 1 - Chrome
Papelera de reciclaje
Excel
Panel de control
PowerPoint
Word
keyboardtest-utility.exe
Google Chrome
iTop Easy Desktop
Registro de Notas
Registro de Notas
Seleccione Materia:
INGLES
Seleccione Estudiante:
ERICKA PAULINA CAISA CANGO
ID Nota
Descripción
Calificación
5 DEBER 8.0
9 EXAMEN 10.0
Descripción:
Calificación:
Guardar
Actualizar
Eliminar
15°C
Neblina
Buscar


Este equipo
Persona 1 - Chrome
Papelera de reciclaje
Excel
Panel de control
PowerPoint
Word
keyboardtest-utility.exe
SISTEMA DE NOTAS
Google Chrome
iTop Easy Desktop
Panel de Administración
Inscribir Estudiante en Materia
Selecciona un estudiante:
Seleccione un estudiante...
Selecciona una materia:
Seleccione una materia...
Inscribir
Gestion Usuarios
Cerrar Sesión
15°C
Neblina
Buscar


Este equipo
Persona 1 - Chrome
Papelera de reciclaje
Excel
Panel de control
PowerPoint
Word
keyboardtest-utility.exe
Google Chrome
iTop Easy Desktop
Gestión de Usuarios
USUARIOS
USUARIO:
PASSWORD:
Administrador
ELIMINAR
GUARDAR
LIMPIAR
ACTUALIZAR
ID
Nombre de Usuario
Rol
1
admin
Administrador
2
profe
Profesor
3
ANDREA
Profesor
4
JOSSELYN
Profesor
15°C
Neblina
Buscar

UML:



CONCLUSIONES:

Crear este sistema para gestionar notas escolares usando Java Swing y MySQL ha sido una experiencia bastante sólida y útil para administrar la información académica. La clase ConexionBD se encarga de la conexión a la base de datos, las clases DAO (como EstudianteDAO) manejan las operaciones básicas de crear, leer, actualizar y eliminar, y las clases en el paquete Ventanas (como Administrador) gestionan cómo se ve la interfaz para el usuario. Este diseño dividido en módulos no solo hace que sea más fácil construir y solucionar problemas, sino que también ayuda a que el sistema pueda crecer y mantenerse en el futuro. La aplicación cumple con el objetivo principal de facilitar la entrada, consulta, actualización y eliminación de notas escolares de una forma sencilla y segura, lo que la convierte en una herramienta muy útil para mejorar los procesos en la educación. Aunque desarrollar software siempre trae desafíos, este sistema sienta una base fuerte para futuras mejoras, como agregar reportes, hacer la autenticación de usuarios más avanzada o incluso crear una versión web.

RECOMENDACIONES:

Para mejorar nuestro sistema, sería muy útil implementar un sistema de autenticación y gestión de roles. Actualmente, la clase Administrador abre

varias ventanas, pero si pudiéramos crear un login seguro y una gestión de usuarios con roles como Administrador, Profesor y Estudiante, lograríamos asignar permisos específicos a cada uno. Esto permitiría controlar quién puede crear, modificar o eliminar notas y registros de estudiantes, haciendo el sistema mucho más seguro y confiable. También, para operaciones que involucren varias acciones en la base de datos, como registrar notas o inscribir estudiantes, sería recomendable usar transacciones. Así, nos aseguramos de que todas esas operaciones se completen correctamente, o que ninguna se aplique si surge algún error, ayudando a mantener los datos siempre consistentes. Por otro lado, es importante hacer validaciones de datos en el servidor, en el nivel del DAO. Aunque la interfaz puede tener sus propias validaciones, agregar esas verificaciones en los objetos DAO ayuda a prevenir que datos inválidos o malformados se guarden en la base de datos, incluso si alguien intenta manipular la aplicación desde fuera. Finalmente, sería genial añadir funciones para generar reportes y gráficos. Por ejemplo, crear reportes dinámicos con las notas, promedios, listas de estudiantes por curso, etc. Podemos usar librerías como JasperReports o JFreeChart para mostrar estos datos en forma de gráficos, lo cual facilita entender mejor el rendimiento académico y ofrece una visión más clara y útil.

ENLACE GITHUB:

https://github.com/AndreaCaisa/PROYECTO.git

ENLACE VIDEO TUTORIAL DEL APLICATIVO:

https://drive.google.com/file/d/1uXS9K9JkgdRHVwX6Oor1aAtgsLURFR9Z/view?usp=sharing

BIBLIOGRAFIA:

Introducción al IDE IntelliJ IDEA APPS JAVA (Parte 2). (n.d.). En CIPSA. Recuperado el 27 de julio de 2025, de Https://cipsa.net/introduccion-ide-intellij-idea-apps-java-parte-2/

Ripoll, J. A. (s. f.). Consultas SQL desde Java. En Ejercicios Programación JAVA. Recuperado de https://www.juanantonioripoll.es/ejercicios-programacion-java/consultas-sql-desde-java.aspx

Línea de Código. (2021, 4 de octubre). Crear una base de datos en Java. Recuperado de https://lineadecodigo.com/java/crear-una-base-de-datos-en-java/