UNIVERSIDAD PERUANA LOS ANDES

FACULTAD DE INGENIERÍA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

PRACTICA N°2

ASIGNATURA: ARQUITECTURA DE SOFTWARE

DOCENTE: FERNÁNDEZ BEJARANO RAUL ENRIQUE

INTEGRANTE: AQUINO CANO, MOISES ISRAEL

CICLO: VI

HUANCAYO-2024
PERÚ

PROYECTO 2: SISTEMA DE REGISTRO DE ALUMNOS

Enunciado del Proyecto: El proyecto "Sistema de registro de alumnos" es una aplicación de escritorio que permite administrar un registro de estudiantes. La aplicación permite agregar, modificar, eliminar, y buscar estudiantes, además de visualizar una lista completa de ellos. El registro incluye información relevante como el código del alumno, nombre, apellido, DNI, teléfono y fecha de nacimiento. El sistema está diseñado utilizando Java Swing para la interfaz gráfica de usuario y sigue el patrón Modelo-Vista-Controlador (MVC) para mantener una separación clara entre la lógica de la aplicación, la interfaz gráfica y la gestión de datos.

El sistema mejora la versión anterior al incluir un modelo de datos más estructurado (`AlumnoArray`) y controladores para gestionar de manera eficiente las interacciones entre la vista y los datos. También se ha implementado la validación de datos para garantizar la integridad de la información ingresada.

REQUERIMIENTOS FUNCIONALES

Agregar Alumno:

- El sistema debe permitir ingresar la información de un nuevo alumno, incluyendo: código, nombre, apellido, DNI, teléfono y fecha de nacimiento.

Modificar Alumno:

- El sistema debe permitir modificar los datos de un alumno existente.
- El usuario debe poder seleccionar un alumno de la tabla, cargar sus datos y actualizar la información.

Eliminar Alumno:

- El sistema debe permitir eliminar un alumno seleccionado de la lista, mostrando una confirmación antes de realizar la acción.

Buscar Alumno:

- El sistema debe permitir buscar alumnos por nombre o apellido en tiempo real. Los resultados deben mostrarse dinámicamente en la tabla.

Visualizar Lista de Alumnos:

- El sistema debe mostrar una tabla con todos los alumnos registrados, con las siguientes columnas: Código, Nombre, Apellido, DNI, Teléfono y Fecha de Nacimiento.

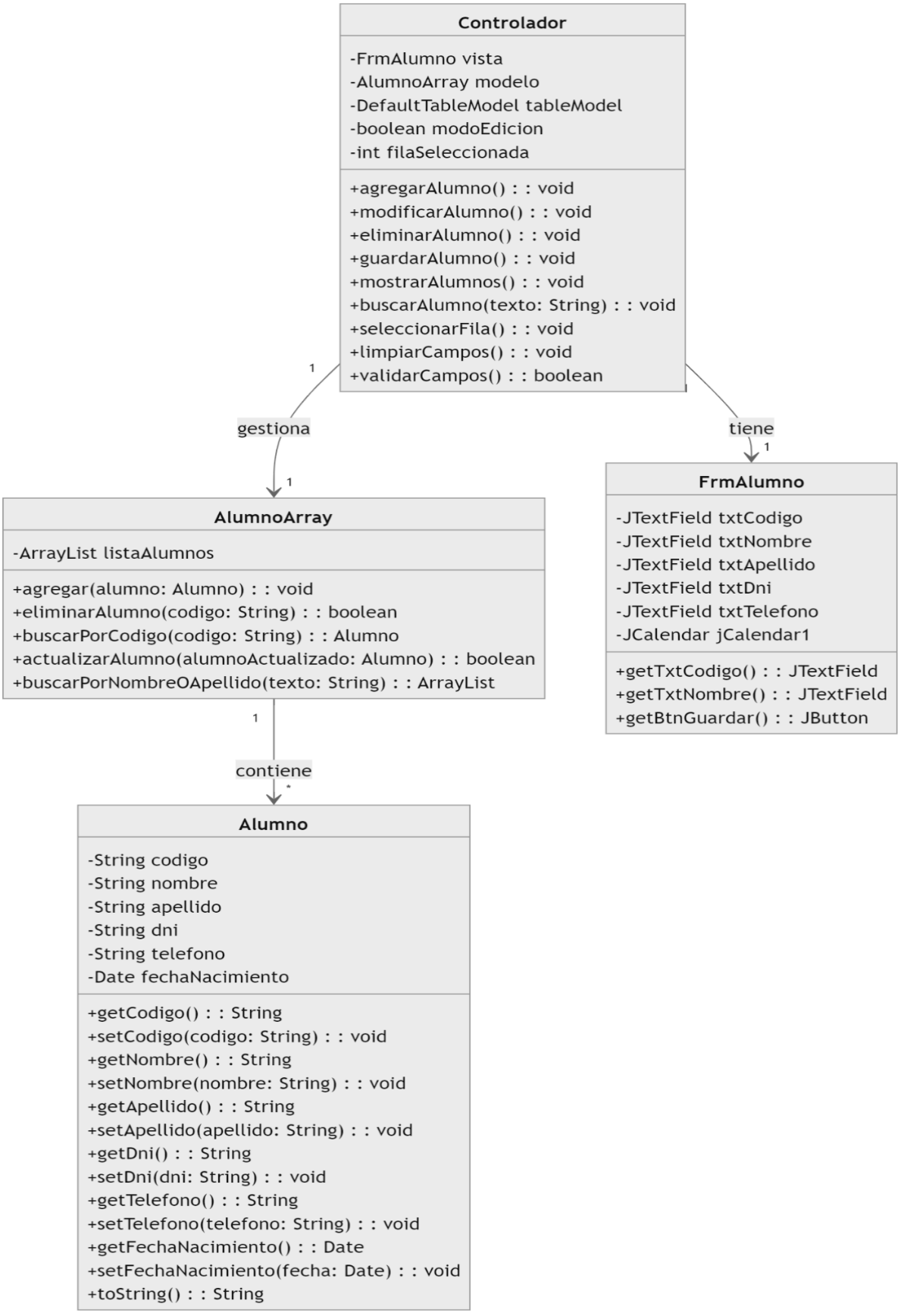
Validación de Datos:

- El sistema debe validar que todos los campos requeridos estén completos antes de agregar o modificar un alumno.
- El sistema debe validar que el DNI contenga solo 8 dígitos y el teléfono contenga 9 dígitos.

Guardar Cambios:

- El sistema debe permitir guardar los cambios realizados en los datos de los alumnos, ya sea al agregar un nuevo alumno o modificar uno existente.

DIAGRAMA DE CLASE


Controlador
-FrmAlumno vista
-AlumnoArray modelo
-DefaultTableModel tableModel
-boolean modoEdicion
-int filaSeleccionada
+agregarAlumno() : : void
+modificarAlumno() : : void
+eliminarAlumno() : : void
+guardarAlumno() : : void
+mostrarAlumnos() : : void
+buscarAlumno(texto: String) : : void
+seleccionarFila() : : void
+limpiarCampos() : : void
+validarCampos() : : boolean
1
gestiona
1
tiene
1
AlumnoArray
-ArrayList listaAlumnos
+agregar(alumno: Alumno) : : void
+eliminarAlumno(codigo: String) : : boolean
+buscarPorCodigo(codigo: String) : : Alumno
+actualizarAlumno(alumnoActualizado: Alumno) : : boolean
+buscarPorNombreOApellido(texto: String) : : ArrayList
FrmAlumno
-JTextField txtCodigo
-JTextField txtNombre
-JTextField txtApellido
-JTextField txtDni
-JTextField txtTelefono
-JCalendar jCalendar1
+getTxtCodigo() : : JTextField
+getTxtNombre() : : JTextField
+getBtnGuardar() : : JButton
1
contiene
*
Alumno
-String codigo
-String nombre
-String apellido
-String dni
-String telefono
-Date fechaNacimiento
+getCodigo() : : String
+setCodigo(codigo: String) : : void
+getNombre() : : String
+setNombre(nombre: String) : : void
+getApellido() : : String
+setApellido(apellido: String) : : void
+getDni() : : String
+setDni(dni: String) : : void
+getTelefono() : : String
+setTelefono(telefono: String) : : void
+getFechaNacimiento() : : Date
+setFechaNacimiento(fecha: Date) : : void
+toString() : : String

CODIGO DEL PROGRAMA

CLASE ALUMNO

La clase `Alumno` representa la estructura básica para almacenar los datos de un estudiante en un sistema de gestión de alumnos. Incluye información como el código del alumno, nombre, apellido, DNI, teléfono y fecha de nacimiento.

```java
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package MODELO;

import java.util.Date;

/**
 *
 * @author USER 17
 */
public class Alumno {
    private String codigo;
    private String nombre;
    private String apellido;
    private String dni;
    private String telefono;
    private Date fechaNacimiento;

    // Constructor vacío
    public Alumno() {
    }

    // Constructor con parámetros
    public Alumno(String codigo, String nombre, String apellido, String dni, String telefono, Date fechaNacimiento) {
        this.codigo = codigo;
        this.nombre = nombre;
        this.apellido = apellido;
        this.dni = dni;
        this.telefono = telefono;
```

```
        this.fechaNacimiento = fechaNacimiento;
    }

    // Getters y Setters
    public String getCodigo() {
        return codigo;
    }

    public void setCodigo(String codigo) {
        this.codigo = codigo;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getApellido() {
        return apellido;
    }

    public void setApellido(String apellido) {
        this.apellido = apellido;
    }

    public String getDni() {
        return dni;
    }

    public void setDni(String dni) {
        this.dni = dni;
    }

    public String getTelefono() {
        return telefono;
    }

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

```
    public Date getFechaNacimiento() {
        return fechaNacimiento;
    }

    public void setFechaNacimiento(Date fechaNacimiento) {
        this.fechaNacimiento = fechaNacimiento;
    }

    @Override
    public String toString() {
        return "Alumno{" + "codigo=" + codigo + ", nombre=" + nombre + ", apellido=" +
apellido +
                ", dni=" + dni + ", telefono=" + telefono + ", fechaNacimiento=" +
fechaNacimiento + '}';
    }
}
```

CLASE ALUMNOARRAY

La clase `AlumnoArray` gestiona una colección de objetos de tipo `Alumno` utilizando un `ArrayList`. Proporciona métodos para agregar, eliminar, buscar y actualizar los alumnos en la lista. Los alumnos se pueden buscar por código o por coincidencia en nombre o apellido.

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this
template
 */
package MODELO;

import java.util.ArrayList;

/**
 * Clase que gestiona un array de objetos Alumno
 */
public class AlumnoArray {

    private ArrayList<Alumno> listaAlumnos;
```

```
    // Constructor que inicializa el ArrayList
    public AlumnoArray() {
        listaAlumnos = new ArrayList<>();
    }

    // Método para agregar un Alumno al ArrayList
    public void agregar(Alumno alumno) {
        listaAlumnos.add(alumno);
    }

    public boolean eliminarAlumno(String codigo) {
    for (int i = 0; i < listaAlumnos.size(); i++) {
        if (listaAlumnos.get(i).getCodigo().equals(codigo)) {
            listaAlumnos.remove(i);
            return true; // Alumno eliminado correctamente
        }
    }
    return false; // No se encontró el alumno
}


    // Método para buscar un Alumno por código
    public Alumno buscarPorCodigo(String codigo) {
        for (Alumno alumno : listaAlumnos) {
            if (alumno.getCodigo().equals(codigo)) {
                return alumno;
            }
        }
        return null; // No se encontró el alumno
    }

    // Método para obtener la lista completa de alumnos
    public ArrayList<Alumno> getListaAlumnos() {
        return listaAlumnos;
    }

    // Método para actualizar un Alumno por código
    public boolean actualizarAlumno(Alumno alumnoActualizado) {
    for (int i = 0; i < listaAlumnos.size(); i++) {
        Alumno alumno = listaAlumnos.get(i);
        if (alumno.getCodigo().equals(alumnoActualizado.getCodigo())) {
            listaAlumnos.set(i, alumnoActualizado);
            return true;  // Alumno actualizado correctamente
```

```
        }
    }
    return false;  // No se encontró el alumno con ese código
}


    // Método para buscar alumnos por nombre o apellido
    public ArrayList<Alumno> buscarPorNombreOApellido(String texto) {
        ArrayList<Alumno> resultados = new ArrayList<>();
        texto = texto.toLowerCase(); // Convertir a minúsculas para evitar problemas
con mayúsculas
        for (Alumno alumno : listaAlumnos) {
            if (alumno.getNombre().toLowerCase().contains(texto) ||
                alumno.getApellido().toLowerCase().contains(texto)) {
                resultados.add(alumno);
            }
        }
        return resultados;
    }
}
```

JFRAME

La clase `FrmAlumno` es la interfaz gráfica de usuario (GUI) que permite gestionar un registro de alumnos. Está vinculada a un controlador y a un modelo que maneja los datos.

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit
this template
 */
package VISTA;

import javax.swing.table.DefaultTableModel;
import MODELO.AlumnoArray;
import CONTROLADOR.Controlador;
```

```java
/**
 *
 * @author ADMIN
 */
public class FrmAlumno extends javax.swing.JFrame {

    private DefaultTableModel modelo;
    private AlumnoArray gestor;
    private Controlador controlador;

    // Getters de botones y campos de texto para el controlador
    public javax.swing.JButton getBtnGuardar() {
        return btnGuardar;
    }

    public javax.swing.JButton getBtnEliminar() {
        return btnEliminar;
    }

    public javax.swing.JButton getBtnModificar() {
        return btnModificar;
    }

    public javax.swing.JButton getBtnAgregar() {
        return btnAgregar;
    }

    public javax.swing.JTextField getTxtCodigo() {
        return txtCodigo;
    }

    public javax.swing.JTextField getTxtNombre() {
        return txtNombre;
    }

    public javax.swing.JTextField getTxtApellido() {
        return txtApellido;
    }

    public javax.swing.JTextField getTxtDni() {
        return txtDni;
    }
```

```java
public javax.swing.JTextField getTxtTelefono() {
    return txtTelefono;
}

public com.toedter.calendar.JCalendar getJCalendar() {
    return jCalendar1;
}

public javax.swing.JTextField getTxtBuscar() {
    return txtBuscar;
}

public javax.swing.JTable getVistaRegistro() {
    return VistaRegistro;
}

/**
 * Creates new form FrmAlumno
 */
public FrmAlumno() {
    initComponents();
    setLocationRelativeTo(null); // Centrar la ventana

    // Inicializar el modelo de la tabla
    modelo = new DefaultTableModel();
    modelo.addColumn("Código");
    modelo.addColumn("Nombre");
    modelo.addColumn("Apellido");
    modelo.addColumn("DNI");
    modelo.addColumn("Teléfono");
    modelo.addColumn("Fecha de Nacimiento");

    // Asignar el modelo a la tabla
    VistaRegistro.setModel(modelo);

    // Crear el gestor y controlador con el modelo ya inicializado
    gestor = new AlumnoArray();
    controlador = new Controlador(this, gestor);
}
```

CLASE CONTROLADOR

Este código pertenece al controlador de una aplicación de gestión de alumnos. El controlador es responsable de manejar la interacción entre la interfaz gráfica de usuario (FrmAlumno) y el modelo de datos (AlumnoArray).

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */

/**
 *
 * @author ADMIN
 */
package CONTROLADOR;

import VISTA.FrmAlumno;
import MODELO.Alumno;
import MODELO.AlumnoArray;

import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import java.util.Date;
import java.util.ArrayList;

public class Controlador {

    private FrmAlumno vista;
    private AlumnoArray modelo;
    private DefaultTableModel tableModel;
    private boolean modoEdicion = false;
    private int filaSeleccionada = -1; // Para guardar el índice de la fila seleccionada


    // Constructor que acepta los parámetros FrmAlumno y AlumnoArray
    public Controlador(FrmAlumno vista, AlumnoArray modelo) {
        this.vista = vista;
        this.modelo = modelo;
        this.tableModel = (DefaultTableModel) this.vista.getVistaRegistro().getModel();
```

```
        // Asignar los listeners a los botones
        this.vista.getBtnGuardar().addActionListener(e -> guardarAlumno());
        this.vista.getBtnEliminar().addActionListener(e -> eliminarAlumno());
        this.vista.getBtnModificar().addActionListener(e -> modificarAlumno());
        this.vista.getBtnAgregar().addActionListener(e -> agregarAlumno());
        this.vista.getTxtBuscar().addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyReleased(java.awt.event.KeyEvent evt) {
                buscarAlumno(vista.getTxtBuscar().getText());
            }
        });

        // Evento para seleccionar una fila y cargar los datos
        this.vista.getVistaRegistro().addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                seleccionarFila();
            }
        });

        mostrarAlumnos();
    }

    // Método para agregar un nuevo alumno
    private void agregarAlumno() {
        if (!validarCampos()) {
            JOptionPane.showMessageDialog(vista, "Por favor, complete todos los
campos.");
            return;
        }

        Alumno nuevoAlumno = new Alumno(
                vista.getTxtCodigo().getText(),
                vista.getTxtNombre().getText(),
                vista.getTxtApellido().getText(),
                vista.getTxtDni().getText(),
                vista.getTxtTelefono().getText(),
                vista.getJCalendar().getDate()
        );

        modelo.agregar(nuevoAlumno);
        mostrarAlumnos();
        limpiarCampos();
```

```
        JOptionPane.showMessageDialog(vista, "Alumno agregado correctamente.");
    }


    // Método para modificar un alumno
    private void modificarAlumno() {
        int selectedRow = vista.getVistaRegistro().getSelectedRow();
        if (selectedRow == -1) {
            JOptionPane.showMessageDialog(vista, "Seleccione un alumno de la
tabla.");
            return;
        }

        Alumno alumno = modelo.getListaAlumnos().get(selectedRow);
        alumno.setCodigo(vista.getTxtCodigo().getText());
        alumno.setNombre(vista.getTxtNombre().getText());
        alumno.setApellido(vista.getTxtApellido().getText());
        alumno.setDni(vista.getTxtDni().getText());
        alumno.setTelefono(vista.getTxtTelefono().getText());
        alumno.setFechaNacimiento(vista.getJCalendar().getDate());

        mostrarAlumnos();
        limpiarCampos();
        JOptionPane.showMessageDialog(vista, "Alumno modificado correctamente.");
    }

    // Método para mostrar los alumnos en la tabla
    private void mostrarAlumnos() {
        tableModel.setRowCount(0);
        for (Alumno alumno : modelo.getListaAlumnos()) {
            tableModel.addRow(new Object[]{
                    alumno.getCodigo(),
                    alumno.getNombre(),
                    alumno.getApellido(),
                    alumno.getDni(),
                    alumno.getTelefono(),
                    alumno.getFechaNacimiento()
            });
        }
    }

    // Método para buscar alumnos por nombre o apellido
    private void buscarAlumno(String texto) {
        tableModel.setRowCount(0);
```

```
        ArrayList<Alumno> resultados = modelo.buscarPorNombreOApellido(texto);
        for (Alumno alumno : resultados) {
            tableModel.addRow(new Object[]{
                    alumno.getCodigo(),
                    alumno.getNombre(),
                    alumno.getApellido(),
                    alumno.getDni(),
                    alumno.getTelefono(),
                    alumno.getFechaNacimiento()
            });
        }
    }

    private void seleccionarFila() {
    filaSeleccionada = vista.getVistaRegistro().getSelectedRow();
    if (filaSeleccionada != -1) {
        // Obtenemos los datos directamente de la tabla
        vista.getTxtCodigo().setText((String) tableModel.getValueAt(filaSeleccionada,
0));
        vista.getTxtNombre().setText((String) tableModel.getValueAt(filaSeleccionada,
1));
        vista.getTxtApellido().setText((String) tableModel.getValueAt(filaSeleccionada,
2));
        vista.getTxtDni().setText((String) tableModel.getValueAt(filaSeleccionada, 3));
        vista.getTxtTelefono().setText((String) tableModel.getValueAt(filaSeleccionada,
4));
        vista.getJCalendar().setDate((Date) tableModel.getValueAt(filaSeleccionada,
5));

        // Activamos modo edición
        modoEdicion = true;
    }
}


    private void eliminarAlumno() {
    int filaSeleccionada = vista.getVistaRegistro().getSelectedRow();

    // Verificamos si hay una fila seleccionada
    if (filaSeleccionada == -1) {
        JOptionPane.showMessageDialog(vista, "Seleccione un alumno de la tabla
para eliminar.");
        return;
    }
```

```
    // Mostrar diálogo de confirmación
    int confirmacion = JOptionPane.showConfirmDialog(vista,
        "¿Está seguro de que desea eliminar al alumno seleccionado?",
        "Confirmar eliminación",
        JOptionPane.YES_NO_OPTION);

    // Si el usuario confirma la eliminación
    if (confirmacion == JOptionPane.YES_OPTION) {
        // Eliminar el alumno de la lista y del modelo
        modelo.getListaAlumnos().remove(filaSeleccionada); // Elimina del ArrayList
        tableModel.removeRow(filaSeleccionada); // Elimina la fila de la tabla

        JOptionPane.showMessageDialog(vista, "Alumno eliminado correctamente.");
        limpiarCampos(); // Limpiar los campos después de la eliminación
    }
}


    private void guardarAlumno() {
    if (!validarCampos()) {
        JOptionPane.showMessageDialog(vista, "Por favor, complete todos los
campos.");
        return;
    }

    // Si estamos en modo edición, actualizamos la fila seleccionada
    if (modoEdicion && filaSeleccionada != -1) {
        // Modificamos los datos directamente en la tabla
        tableModel.setValueAt(vista.getTxtCodigo().getText(), filaSeleccionada, 0);
        tableModel.setValueAt(vista.getTxtNombre().getText(), filaSeleccionada, 1);
        tableModel.setValueAt(vista.getTxtApellido().getText(), filaSeleccionada, 2);
        tableModel.setValueAt(vista.getTxtDni().getText(), filaSeleccionada, 3);
        tableModel.setValueAt(vista.getTxtTelefono().getText(), filaSeleccionada, 4);
        tableModel.setValueAt(vista.getJCalendar().getDate(), filaSeleccionada, 5);

        // Actualizamos el alumno en la lista
        Alumno alumnoModificado = modelo.getListaAlumnos().get(filaSeleccionada);
        alumnoModificado.setCodigo(vista.getTxtCodigo().getText());
        alumnoModificado.setNombre(vista.getTxtNombre().getText());
        alumnoModificado.setApellido(vista.getTxtApellido().getText());
        alumnoModificado.setDni(vista.getTxtDni().getText());
        alumnoModificado.setTelefono(vista.getTxtTelefono().getText());
        alumnoModificado.setFechaNacimiento(vista.getJCalendar().getDate());
```

```
            // Salimos del modo edición
            modoEdicion = false;
            filaSeleccionada = -1;
            JOptionPane.showMessageDialog(vista, "Alumno modificado correctamente.");
        } else {
            // Si no estamos en modo edición, agregamos un nuevo alumno
            Alumno nuevoAlumno = new Alumno(
                    vista.getTxtCodigo().getText(),
                    vista.getTxtNombre().getText(),
                    vista.getTxtApellido().getText(),
                    vista.getTxtDni().getText(),
                    vista.getTxtTelefono().getText(),
                    vista.getJCalendar().getDate()
            );

            modelo.agregar(nuevoAlumno);
            mostrarAlumnos();
            JOptionPane.showMessageDialog(vista, "Alumno guardado correctamente.");
        }

        limpiarCampos();
    }



    // Método para limpiar los campos de texto
    private void limpiarCampos() {
        vista.getTxtCodigo().setText("");
        vista.getTxtNombre().setText("");
        vista.getTxtApellido().setText("");
        vista.getTxtDni().setText("");
        vista.getTxtTelefono().setText("");
        vista.getJCalendar().setDate(null);
    }

    // Método para validar que todos los campos están llenos
    private boolean validarCampos() {
        return !(vista.getTxtCodigo().getText().isEmpty() ||
                vista.getTxtNombre().getText().isEmpty() ||
                vista.getTxtApellido().getText().isEmpty() ||
                vista.getTxtDni().getText().isEmpty() ||
                vista.getTxtTelefono().getText().isEmpty() ||
                vista.getJCalendar().getDate() == null);
```

```
    }
}
```

MAIN

Este código es responsable de establecer el estilo visual de la aplicación y de lanzar el formulario principal (`FrmAlumno`), donde se gestionan las funciones de registro de alumnos.

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 */

package com.mycompany.semana_2;


import VISTA.FrmAlumno;
import com.jtattoo.plaf.aluminium.AluminiumLookAndFeel;
import javax.swing.UIManager;


/**
 *
 * @author ADMIN
 */
public class SEMANA_2 {

    public static void main(String[] args) {

        try {
            // Establecer el tema Aluminium de JTattoo
            UIManager.setLookAndFeel("com.jtattoo.plaf.texture.TextureLookAndFeel");



        } catch (Exception ex) {
            ex.printStackTrace();
        }

        // Iniciar tu aplicación o mostrar la interfaz gráfica
```

```
        java.awt.EventQueue.invokeLater(() -> {
            new FrmAlumno().setVisible(true); // Reemplaza FrmAlumno con el nombre
de tu JFrame o formulario principal
        });
    }
}
```