Universidad Nacional del

Altiplano - Puno

**FACULTAD DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA**

**“ADMINISTRADOR DE ESTUDIANTES EN LA UNIVERSIDAD NACIONAL DEL ALTIPLANO – CRUD + INFORME”**

**II Unidad**

Presenta:

**Mayta Quispe Marco Fidel**

Docente:

**DOC. Tito Lipa Jose Panfilo**

**Puno Julio, 2023**

**ADMINISTRADOR DE ESTUDIANTES DE LA UNIVERSIDAD NACIONAL DEL ALTIPLANO**

**CRUD + INFORME**

**JAVA (visual studio code)**

REQUISITOS

# Especificación de Requisitos

## Objetivos del Sistema:

* El objetivo principal del sistema de gestión de estudiantes es facilitar la administración eficiente de los datos de los estudiantes, brindando a los usuarios (administradores, docentes y estudiantes) herramientas para realizar tareas relacionadas con la gestión académica. Los objetivos específicos del sistema son:
* Registrar y mantener información completa y actualizada de los estudiantes, incluyendo datos personales, información de contacto y detalles académicos.
* Permitir a los administradores gestionar y supervisar el proceso de admisión de nuevos estudiantes al establecimiento educativo.

### Funciones Principales:

El sistema de gestión de estudiantes deberá ofrecer las siguientes funciones principales:

* Registro y mantenimiento de datos estudiantiles: Permitir la creación, actualización y eliminación de perfiles de estudiantes con campos para datos personales, información de contacto, matrícula, notas y asistencias.
* Proceso de admisión: Facilitar el registro y seguimiento de solicitudes de admisión, así como la revisión y aprobación por parte del personal autorizado.
* Generación de informes y estadísticas: Permitir la generación de informes académicos y estadísticas sobre el rendimiento estudiantil.

### Restricciones y Limitaciones:

El sistema de gestión de estudiantes estará sujeto a las siguientes restricciones y limitaciones:

* Acceso restringido: El acceso a las funciones del sistema estará protegido mediante autenticación y autorización basada en roles para garantizar la privacidad y seguridad de los datos.
* Escalabilidad: El sistema debe ser capaz de manejar una cantidad creciente de estudiantes y datos sin degradar significativamente el rendimiento.
* Compatibilidad: El sistema deberá ser compatible con los navegadores web modernos y dispositivos comunes para garantizar una amplia accesibilidad.
* Idioma: La interfaz del sistema se proporcionará en un único idioma en esta versión inicial (español), aunque se contempla la posibilidad de expandir a otros idiomas en versiones futuras.

### Casos de Uso:

Actor: Administrador

* Registrar nuevo estudiante: El administrador puede agregar un nuevo estudiante al sistema proporcionando los datos personales y de contacto del estudiante.
* Registrar nuevo estudiante: El administrador puede editar la información de algún estudiante al sistema proporcionando los datos personales y de contacto del estudiante.

Actualizar información personal: El administrador puede actualizar la información de cada uno de los estudiantes en el sistema.

DISEÑO

# Diagramas UML

* Diagrama de Clases: Representa las clases del sistema, sus atributos y relaciones entre ellas. Por ejemplo, tendrías una clase "Estudiante" con atributos como "nombre", "apellido", "correo electrónico", etc., y relaciones con otras clases como "Asignatura" o "Docente".
* Diagrama de Secuencia: Describe las interacciones entre los distintos objetos del sistema a lo largo del tiempo. Por ejemplo, cómo un estudiante se inscribe en una asignatura o cómo un docente registra las calificaciones de un estudiante.
* Diagrama de Actividad: Muestra el flujo de actividades y decisiones en un proceso específico del sistema. Por ejemplo, el proceso de admisión de un estudiante podría ser representado mediante un diagrama de actividad.

# Diseño de Base de Datos

* Esquema de la Base de Datos: Describe la estructura general de la base de datos, incluyendo las tablas y sus relaciones.
  + Tabla "Estudiante":
  + ID (Clave Primaria)
  + Nombre
  + Apellido
  + Correo electrónico
  + Fecha de nacimiento
  + Dirección
  + Teléfono
* Tablas: Define las tablas de la base de datos y sus columnas. Por ejemplo, tendrías una tabla "Estudiantes" con columnas como "ID", "nombre", "apellido", "correo electrónico", etc.
  + Interfaz de usuario gráfica

    Descripción generada automáticamente
* Relaciones: Indica cómo las tablas están relacionadas entre sí, como las claves primarias y foráneas. Por ejemplo, la tabla "Asignaturas" podría tener una clave foránea que hace referencia al ID del estudiante al que está asignada.
  + Interfaz de usuario gráfica, Texto, Aplicación

    Descripción generada automáticamente
* Claves Primarias y Foráneas: Define las claves primarias que identifican de manera única cada registro en una tabla y las claves foráneas que establecen relaciones con otras tablas.
  + Para el presente proyecto se cuenta con una tabla por lo tanto no se tiene llaves foráneas.
  + La llave primaria identificada es id\_estu.

# Diseño de Interfaz de Usuario

* Esquemas de Pantallas: Se proporciona un diseño visual de cada pantalla de la interfaz gráfica, mostrando la ubicación de los elementos como botones, campos de entrada, tablas, etc.
  + Tabla

    Descripción generada automáticamente
  + Interfaz de usuario gráfica, Aplicación

    Descripción generada automáticamente
  + Interfaz de usuario gráfica, Aplicación

    Descripción generada automáticamente
* Ubicación de Botones y Campos: Describe dónde se ubicarán los botones y campos en cada pantalla, para que los usuarios puedan interactuar con el sistema de manera intuitiva.
  + 
  + 
  + 
  + 
  + 
  + 



* Flujos de Navegación:
  + El usuario administrador ingresa al sistema y lo primero que verá será la pantalla donde está la tabla, posteriormente si se desea trasladar a la ventana emergente “crear estudiante” entonces tendrá que hacer click en “crear”, si desea trasladarse a la ventana emergente de editado, entonces tendrá que hacer click en alguno de los botones de editar.

IMPLEMENTACIÓN

# Código fuente

import java.awt.BorderLayout;

import java.awt.Component;

import java.awt.Graphics2D;

import java.awt.GridLayout;

import java.awt.Image;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.image.BufferedImage;

import java.io.ByteArrayInputStream;

import java.io.ByteArrayOutputStream;

import java.io.File;

import java.io.FileOutputStream;

import java.io.IOException;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.sql.Types;

import javax.imageio.ImageIO;

import javax.swing.DefaultCellEditor;

import javax.swing.Icon;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JCheckBox;

import javax.swing.JFileChooser;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.JTextField;

import javax.swing.SwingConstants;

import javax.swing.SwingUtilities;

import javax.swing.UIManager;

import javax.swing.table.DefaultTableCellRenderer;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableCellRenderer;

import com.itextpdf.text.Document;

import com.itextpdf.text.Element;

import com.itextpdf.text.PageSize;

import com.itextpdf.text.Paragraph;

import com.itextpdf.text.pdf.PdfPCell;

import com.itextpdf.text.pdf.PdfPTable;

import com.itextpdf.text.pdf.PdfWriter;

public class InterfazTabla extends JFrame {

    private JTable tabla;

    private DefaultTableModel modelo;

    private JTextField codigoField; // Campo para ingresar el código del estudiante

    // Etiqueta para mostrar una vista previa de la foto seleccionada

    private JLabel lblPhotoPreview;

    public InterfazTabla() {

        setTitle("Sistema de Gestión");

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        setSize(800, 400);

        setLocationRelativeTo(null);

        JPanel panelPrincipal = new JPanel(new BorderLayout());

        // Barra de navegación

        JPanel panelNavegacion = new JPanel();

        JLabel titulo = new JLabel("Sistema de Gestión");

        panelNavegacion.add(titulo);

        panelPrincipal.add(panelNavegacion, BorderLayout.NORTH);

        // Tabla

        String[] columnas = { "Código", "Nombre", "Apellidos", "Facultad", "Carrera", "Foto", "Editar", "Eliminar" };

        modelo = new DefaultTableModel(columnas, 0) {

            @Override

            public Class<?> getColumnClass(int column) {

                if (column == 5) {

                    return ImageIcon.class; // Columna de fotos

                } else if (column == 6 || column == 7) {

                    return JButton.class; // Columnas de botones editar y eliminar

                }

                return super.getColumnClass(column);

            }

        };

        tabla = new JTable(modelo);

        tabla.getColumnModel().getColumn(5).setCellRenderer(new ImageRenderer()); // Mostrar imágenes

        tabla.getColumnModel().getColumn(5).setCellEditor(new ImageEditor()); // Editar imágenes

        tabla.getColumnModel().getColumn(6).setCellRenderer(new ButtonRendererEditar()); // Botón Editar

        tabla.getColumnModel().getColumn(6).setCellEditor(new ButtonEditorEditar()); // Editar botón Editar

        tabla.getColumnModel().getColumn(7).setCellRenderer(new ButtonRendererEliminar()); // Botón Eliminar

        tabla.getColumnModel().getColumn(7).setCellEditor(new ButtonEditorEliminar()); // Editar botón Eliminar

        JScrollPane scrollTabla = new JScrollPane(tabla);

        panelPrincipal.add(scrollTabla, BorderLayout.CENTER);

        // Botón Crear

        JButton btnCrear = new JButton("Crear");

        btnCrear.addActionListener(e -> mostrarDialogoNuevoEstudiante());

        JPanel panelBotones = new JPanel();

        panelBotones.add(btnCrear);

        // Botón Salir

        JButton btnSalir = new JButton("Salir");

        btnSalir.addActionListener(e -> System.exit(0));

        panelBotones.add(btnSalir);

        // Botón Descargar Informe

        JButton btnDescargarInforme = new JButton("Descargar Informe");

        btnDescargarInforme.addActionListener(e -> generarInformePDF());

        panelBotones.add(btnDescargarInforme);

        panelPrincipal.add(panelBotones, BorderLayout.SOUTH);

        getContentPane().add(panelPrincipal);

        // Cargar datos en la tabla al iniciar la interfaz

        cargarDatos();

        // Inicializar lblPhotoPreview para mostrar la foto seleccionada

        lblPhotoPreview = new JLabel();

    }

    // Clase interna para renderizar las imágenes en la tabla

    private class ImageRenderer extends DefaultTableCellRenderer {

        public ImageRenderer() {

            setHorizontalAlignment(SwingConstants.CENTER);

        }

        public Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean hasFocus,

                int row, int column) {

            // Mostrar imagen si existe, o mostrar "No existe Foto" si no hay imagen.

            if (value instanceof ImageIcon) {

                setIcon((ImageIcon) value);

                setText("");

            } else {

                setIcon(null);

                setText("No existe Foto");

            }

            return this;

        }

    }

    // Clase interna para editar las imágenes en la tabla

    private class ImageEditor extends DefaultCellEditor {

        private JButton button;

        private JFileChooser fileChooser;

        private File selectedFile;

        public ImageEditor() {

            super(new JCheckBox());

            button = new JButton();

            button.setOpaque(true);

            button.addActionListener(new ActionListener() {

                @Override

                public void actionPerformed(ActionEvent e) {

                    int row = tabla.getSelectedRow();

                    if (row != -1) {

                        fileChooser = new JFileChooser();

                        int returnValue = fileChooser.showOpenDialog(null);

                        if (returnValue == JFileChooser.APPROVE\_OPTION) {

                            selectedFile = fileChooser.getSelectedFile();

                            ImageIcon imageIcon = new ImageIcon(selectedFile.getAbsolutePath());

                            tabla.setValueAt(imageIcon, row, 5);

                        }

                    }

                }

            });

        }

        public Component getTableCellEditorComponent(JTable table, Object value, boolean isSelected, int row,

                int column) {

            return button;

        }

        public Object getCellEditorValue() {

            return selectedFile != null ? new ImageIcon(selectedFile.getAbsolutePath()) : null;

        }

    }

    // Método para cargar datos desde la base de datos y llenar la tabla

    private void cargarDatos() {

        String url = "jdbc:mysql://localhost/card\_bd1";

        String usuario = "root";

        String contrasena = ""; // Aquí debes agregar tu contraseña si la tienes configurada.

        try {

            Connection conexion = DriverManager.getConnection(url, usuario, contrasena);

            Statement stmt = conexion.createStatement();

            ResultSet rs = stmt.executeQuery("SELECT \* FROM estu\_table1");

            while (rs.next()) {

                String codigo = rs.getString("codigo\_estu");

                String nombre = rs.getString("nombre\_estu");

                String apellidos = rs.getString("apellidos\_estu");

                String facultad = rs.getString("facultad\_estu");

                String carrera = rs.getString("carrera\_estu");

                String fotoPath = rs.getString("foto\_estu");

                ImageIcon fotoIcon;

                if (fotoPath != null) {

                    fotoIcon = new ImageIcon(fotoPath);

                } else {

                    fotoIcon = null;

                }

                // Agregar datos a la tabla

                modelo.addRow(

                        new Object[] { codigo, nombre, apellidos, facultad, carrera, fotoIcon, "Editar", "Eliminar" });

            }

            rs.close();

            stmt.close();

            conexion.close();

        } catch (SQLException e) {

            e.printStackTrace();

        }

    }

    // Clase interna para renderizar el botón Editar en la tabla

    private class ButtonRendererEditar extends JButton implements TableCellRenderer {

        public ButtonRendererEditar() {

            setOpaque(true);

        }

        public Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean hasFocus,

                int row, int column) {

            if (isSelected) {

                setForeground(table.getSelectionForeground());

                setBackground(table.getSelectionBackground());

            } else {

                setForeground(table.getForeground());

                setBackground(UIManager.getColor("Button.background"));

            }

            setText("Editar");

            return this;

        }

    }

    // Clase interna para editar el botón Editar en la tabla

    private class ButtonEditorEditar extends DefaultCellEditor {

        private JButton button;

        public ButtonEditorEditar() {

            super(new JCheckBox());

            button = new JButton();

            button.setOpaque(true);

            button.addActionListener(new ActionListener() {

                @Override

                public void actionPerformed(ActionEvent e) {

                    int row = tabla.getSelectedRow();

                    if (row != -1) {

                        editarEstudiante(row);

                    }

                }

            });

        }

        public Component getTableCellEditorComponent(JTable table, Object value, boolean isSelected, int row,

                int column) {

            return button;

        }

        public Object getCellEditorValue() {

            return "";

        }

    }

    // Clase interna para renderizar el botón Eliminar en la tabla

    private class ButtonRendererEliminar extends JButton implements TableCellRenderer {

        public ButtonRendererEliminar() {

            setOpaque(true);

        }

        public Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean hasFocus,

                int row, int column) {

            if (isSelected) {

                setForeground(table.getSelectionForeground());

                setBackground(table.getSelectionBackground());

            } else {

                setForeground(table.getForeground());

                setBackground(UIManager.getColor("Button.background"));

            }

            setText("Eliminar");

            return this;

        }

    }

    // Clase interna para editar el botón Eliminar en la tabla

    private class ButtonEditorEliminar extends DefaultCellEditor {

        private JButton button;

        public ButtonEditorEliminar() {

            super(new JCheckBox());

            button = new JButton();

            button.setOpaque(true);

            button.addActionListener(new ActionListener() {

                @Override

                public void actionPerformed(ActionEvent e) {

                    int row = tabla.getSelectedRow();

                    if (row != -1) {

                        eliminarEstudiante(row);

                    }

                }

            });

        }

        public Component getTableCellEditorComponent(JTable table, Object value, boolean isSelected, int row,

                int column) {

            return button;

        }

        public Object getCellEditorValue() {

            return "";

        }

    }

    // Método para mostrar el diálogo de edición de estudiante

    private void editarEstudiante(int rowIndex) {

        String codigo = (String) modelo.getValueAt(rowIndex, 0);

        String nombre = (String) modelo.getValueAt(rowIndex, 1);

        String apellidos = (String) modelo.getValueAt(rowIndex, 2);

        String facultad = (String) modelo.getValueAt(rowIndex, 3);

        String carrera = (String) modelo.getValueAt(rowIndex, 4);

        // Crear campos de texto para editar los datos del estudiante

        JTextField nombreField = new JTextField(30);

        nombreField.setText(nombre);

        JTextField apellidosField = new JTextField(30);

        apellidosField.setText(apellidos);

        JTextField facultadField = new JTextField(20);

        facultadField.setText(facultad);

        JTextField carreraField = new JTextField(20);

        carreraField.setText(carrera);

        // Botón para subir una nueva foto

        JButton btnUploadPhoto = new JButton("Subir Foto");

        btnUploadPhoto.addActionListener(e -> {

            JFileChooser fileChooser = new JFileChooser();

            int returnValue = fileChooser.showOpenDialog(null);

            if (returnValue == JFileChooser.APPROVE\_OPTION) {

                File selectedFile = fileChooser.getSelectedFile();

                try {

                    // Leer la imagen seleccionada

                    BufferedImage originalImage = ImageIO.read(selectedFile);

                    // Redimensionar la imagen a 20x20 píxeles

                    BufferedImage resizedImage = new BufferedImage(20, 20, BufferedImage.TYPE\_INT\_ARGB);

                    Graphics2D g = resizedImage.createGraphics();

                    g.drawImage(originalImage, 0, 0, 20, 20, null);

                    g.dispose();

                    // Mostrar la imagen redimensionada en el JLabel

                    lblPhotoPreview.setIcon(new ImageIcon(resizedImage));

                    // Guardar la imagen en la base de datos y actualizar la tabla

                    guardarImagenEnBD(codigo);

                } catch (IOException ex) {

                    ex.printStackTrace();

                }

            }

        });

        JPanel panel = new JPanel(new GridLayout(5, 2));

        panel.add(new JLabel("Nombres:"));

        panel.add(nombreField);

        panel.add(new JLabel("Apellidos:"));

        panel.add(apellidosField);

        panel.add(new JLabel("Facultad:"));

        panel.add(facultadField);

        panel.add(new JLabel("Carrera:"));

        panel.add(carreraField);

        panel.add(btnUploadPhoto);

        panel.add(lblPhotoPreview);

        int result = JOptionPane.showConfirmDialog(null, panel, "Editar estudiante",

                JOptionPane.OK\_CANCEL\_OPTION, JOptionPane.PLAIN\_MESSAGE);

        if (result == JOptionPane.OK\_OPTION) {

            String nuevoNombre = nombreField.getText();

            String nuevoApellidos = apellidosField.getText();

            String nuevaFacultad = facultadField.getText();

            String nuevaCarrera = carreraField.getText();

            if (!nuevoNombre.isEmpty() && !nuevoApellidos.isEmpty() && !nuevaFacultad.isEmpty()

                    && !nuevaCarrera.isEmpty()) {

                // Guardar los datos editados en la base de datos y actualizar la tabla

                actualizarEstudiante(codigo, nuevoNombre, nuevoApellidos, nuevaFacultad, nuevaCarrera, rowIndex);

                // Guardar la imagen en la base de datos y actualizar la tabla con la nueva

                // imagen

                guardarImagenEnBD(codigo);

            } else {

                JOptionPane.showMessageDialog(null, "Por favor, complete todos los campos.", "Error",

                        JOptionPane.ERROR\_MESSAGE);

            }

        }

    }

    // Método para guardar la foto del estudiante en la base de datos

    private void guardarImagenEnBD(String codigoEstudiante) {

        String url = "jdbc:mysql://localhost/card\_bd1";

        String usuario = "root";

        String contrasena = ""; // Aquí debes agregar tu contraseña si la tienes configurada.

        try {

            Connection conexion = DriverManager.getConnection(url, usuario, contrasena);

            PreparedStatement stmt = conexion.prepareStatement(

                    "UPDATE estu\_table1 SET foto\_estu=? WHERE codigo\_estu=?");

            // Obtener la imagen del JLabel lblPhotoPreview

            Icon icon = lblPhotoPreview.getIcon();

            if (icon instanceof ImageIcon) {

                ImageIcon imageIcon = (ImageIcon) icon;

                Image image = imageIcon.getImage();

                BufferedImage bufferedImage = new BufferedImage(image.getWidth(null), image.getHeight(null),

                        BufferedImage.TYPE\_INT\_RGB);

                Graphics2D g2d = bufferedImage.createGraphics();

                g2d.drawImage(image, 0, 0, null);

                g2d.dispose();

                // Guardar la imagen en la base de datos

                ByteArrayOutputStream baos = new ByteArrayOutputStream();

                ImageIO.write(bufferedImage, "png", baos);

                byte[] imageBytes = baos.toByteArray();

                ByteArrayInputStream bais = new ByteArrayInputStream(imageBytes);

                stmt.setBinaryStream(1, bais);

            } else {

                // Si no hay imagen en lblPhotoPreview, almacenar un valor NULL en la base de

                // datos

                stmt.setNull(1, Types.BLOB);

            }

            stmt.setString(2, codigoEstudiante);

            stmt.executeUpdate();

            stmt.close();

            conexion.close();

            // Actualizar la fila en la tabla

            int rowIndex = tabla.getSelectedRow();

            modelo.setValueAt(lblPhotoPreview.getIcon(), rowIndex, 5);

        } catch (SQLException | IOException e) {

            e.printStackTrace();

            JOptionPane.showMessageDialog(null, "Error al guardar la imagen en la base de datos.", "Error",

                    JOptionPane.ERROR\_MESSAGE);

        }

    }

    // Método para actualizar los datos del estudiante en la base de datos y en la

    // tabla

    private void actualizarEstudiante(String codigo, String nuevoNombre, String nuevoApellidos,

            String nuevaFacultad, String nuevaCarrera, int rowIndex) {

        String url = "jdbc:mysql://localhost/card\_bd1";

        String usuario = "root";

        String contrasena = ""; // Aquí debes agregar tu contraseña si la tienes configurada.

        try {

            Connection conexion = DriverManager.getConnection(url, usuario, contrasena);

            PreparedStatement stmt = conexion.prepareStatement(

                    "UPDATE estu\_table1 SET nombre\_estu=?, apellidos\_estu=?, facultad\_estu=?, carrera\_estu=? WHERE codigo\_estu=?");

            stmt.setString(1, nuevoNombre);

            stmt.setString(2, nuevoApellidos);

            stmt.setString(3, nuevaFacultad);

            stmt.setString(4, nuevaCarrera);

            stmt.setString(5, codigo); // Usar el código original para identificar al estudiante a actualizar

            stmt.executeUpdate();

            stmt.close();

            conexion.close();

            // Actualizar la fila en la tabla

            modelo.setValueAt(nuevoNombre, rowIndex, 1);

            modelo.setValueAt(nuevoApellidos, rowIndex, 2);

            modelo.setValueAt(nuevaFacultad, rowIndex, 3);

            modelo.setValueAt(nuevaCarrera, rowIndex, 4);

        } catch (SQLException e) {

            e.printStackTrace();

            JOptionPane.showMessageDialog(null, "Error al actualizar el estudiante.", "Error",

                    JOptionPane.ERROR\_MESSAGE);

        }

    }

    // Método para eliminar un estudiante de la base de datos y de la tabla

    private void eliminarEstudiante(int rowIndex) {

        String codigo = (String) modelo.getValueAt(rowIndex, 0);

        String url = "jdbc:mysql://localhost/card\_bd1";

        String usuario = "root";

        String contrasena = ""; // Aquí debes agregar tu contraseña si la tienes configurada.

        try {

            Connection conexion = DriverManager.getConnection(url, usuario, contrasena);

            PreparedStatement stmt = conexion.prepareStatement("DELETE FROM estu\_table1 WHERE codigo\_estu = ?");

            stmt.setString(1, codigo);

            stmt.executeUpdate();

            stmt.close();

            conexion.close();

            // Eliminar la fila de la tabla

            modelo.removeRow(rowIndex);

        } catch (SQLException e) {

            e.printStackTrace();

            JOptionPane.showMessageDialog(null, "Error al eliminar el estudiante.", "Error", JOptionPane.ERROR\_MESSAGE);

        }

    }

    // Método para mostrar el diálogo para agregar un nuevo estudiante

    private void mostrarDialogoNuevoEstudiante() {

        codigoField = new JTextField(10); // Campo para ingresar el código del nuevo estudiante

        JTextField nombreField = new JTextField(30);

        JTextField apellidosField = new JTextField(30);

        JTextField facultadField = new JTextField(20);

        JTextField carreraField = new JTextField(20);

        JButton btnUploadPhoto = new JButton("Subir Foto");

        btnUploadPhoto.addActionListener(e -> {

            JFileChooser fileChooser = new JFileChooser();

            int returnValue = fileChooser.showOpenDialog(null);

            if (returnValue == JFileChooser.APPROVE\_OPTION) {

                File selectedFile = fileChooser.getSelectedFile();

                try {

                    // Leer la imagen seleccionada

                    BufferedImage originalImage = ImageIO.read(selectedFile);

                    // Redimensionar la imagen a 20x20 píxeles

                    BufferedImage resizedImage = new BufferedImage(20, 20, BufferedImage.TYPE\_INT\_ARGB);

                    Graphics2D g = resizedImage.createGraphics();

                    g.drawImage(originalImage, 0, 0, 20, 20, null);

                    g.dispose();

                    // Mostrar la imagen redimensionada en el JLabel

                    lblPhotoPreview.setIcon(new ImageIcon(resizedImage));

                } catch (IOException ex) {

                    ex.printStackTrace();

                }

            }

        });

        JPanel panel = new JPanel(new GridLayout(6, 2));

        panel.add(new JLabel("Código:"));

        panel.add(codigoField);

        panel.add(new JLabel("Nombres:"));

        panel.add(nombreField);

        panel.add(new JLabel("Apellidos:"));

        panel.add(apellidosField);

        panel.add(new JLabel("Facultad:"));

        panel.add(facultadField);

        panel.add(new JLabel("Carrera:"));

        panel.add(carreraField);

        panel.add(btnUploadPhoto);

        panel.add(lblPhotoPreview);

        int result = JOptionPane.showConfirmDialog(null, panel, "Nuevo estudiante",

                JOptionPane.OK\_CANCEL\_OPTION, JOptionPane.PLAIN\_MESSAGE);

        if (result == JOptionPane.OK\_OPTION) {

            String codigo = codigoField.getText();

            String nombre = nombreField.getText();

            String apellidos = apellidosField.getText();

            String facultad = facultadField.getText();

            String carrera = carreraField.getText();

            if (!codigo.isEmpty() && !nombre.isEmpty() && !apellidos.isEmpty() && !facultad.isEmpty()

                    && !carrera.isEmpty()) {

                // Guardar el nuevo estudiante en la base de datos y actualizar la tabla

                guardarNuevoEstudiante(codigo, nombre, apellidos, facultad, carrera);

                // Guardar la imagen en la base de datos y actualizar la tabla con la nueva

                // imagen

                guardarImagenEnBD(codigo);

            } else {

                JOptionPane.showMessageDialog(null, "Por favor, complete todos los campos.", "Error",

                        JOptionPane.ERROR\_MESSAGE);

            }

        }

    }

    // Método para guardar un nuevo estudiante en la base de datos y en la tabla

    private void guardarNuevoEstudiante(String codigo, String nombre, String apellidos, String facultad,

            String carrera) {

        String url = "jdbc:mysql://localhost/card\_bd1";

        String usuario = "root";

        String contrasena = ""; // Aquí debes agregar tu contraseña si la tienes configurada.

        try {

            Connection conexion = DriverManager.getConnection(url, usuario, contrasena);

            PreparedStatement stmt = conexion.prepareStatement(

                    "INSERT INTO estu\_table1 (codigo\_estu, nombre\_estu, apellidos\_estu, facultad\_estu, carrera\_estu, foto\_estu) VALUES (?, ?, ?, ?, ?, ?)");

            stmt.setString(1, codigo);

            stmt.setString(2, nombre);

            stmt.setString(3, apellidos);

            stmt.setString(4, facultad);

            stmt.setString(5, carrera);

            // Obtener la imagen del JLabel lblPhotoPreview

            Icon icon = lblPhotoPreview.getIcon();

            if (icon instanceof ImageIcon) {

                ImageIcon imageIcon = (ImageIcon) icon;

                Image image = imageIcon.getImage();

                BufferedImage bufferedImage = new BufferedImage(image.getWidth(null), image.getHeight(null),

                        BufferedImage.TYPE\_INT\_RGB);

                Graphics2D g2d = bufferedImage.createGraphics();

                g2d.drawImage(image, 0, 0, null);

                g2d.dispose();

                // Guardar la imagen en la base de datos

                ByteArrayOutputStream baos = new ByteArrayOutputStream();

                ImageIO.write(bufferedImage, "png", baos);

                byte[] imageBytes = baos.toByteArray();

                ByteArrayInputStream bais = new ByteArrayInputStream(imageBytes);

                stmt.setBinaryStream(6, bais);

            } else {

                // Si no hay imagen en lblPhotoPreview, almacenar un valor NULL en la base de

                // datos

                stmt.setNull(6, Types.BLOB);

            }

            stmt.executeUpdate();

            stmt.close();

            conexion.close();

            // Agregar el nuevo estudiante a la tabla

            modelo.addRow(new Object[] { codigo, nombre, apellidos, facultad, carrera, lblPhotoPreview.getIcon(),

                    "Editar", "Eliminar" });

            // Limpiar el campo de texto para el código del estudiante

            codigoField.setText("");

        } catch (SQLException | IOException e) {

            e.printStackTrace();

            JOptionPane.showMessageDialog(null, "Error al guardar el nuevo estudiante en la base de datos.", "Error",

                    JOptionPane.ERROR\_MESSAGE);

        }

    }

    private void generarInformePDF() {

        try {

            // Crear un documento PDF

            Document document = new Document(PageSize.A4);

            PdfWriter.getInstance(document, new FileOutputStream("informe\_estudiantes.pdf"));

            document.open();

            // Agregar el título al documento

            Paragraph titulo = new Paragraph("Informe de Estudiantes");

            titulo.setAlignment(Element.ALIGN\_CENTER);

            document.add(titulo);

            // Crear una tabla para el informe con las mismas columnas que la tabla mostrada

            // en la interfaz

            PdfPTable pdfTable = new PdfPTable(tabla.getColumnCount());

            pdfTable.setWidthPercentage(100);

            // Agregar las cabeceras de las columnas al informe

            for (int i = 0; i < tabla.getColumnCount(); i++) {

                pdfTable.addCell(new PdfPCell(new Paragraph(tabla.getColumnName(i))));

            }

            // Agregar los datos de las filas al informe

            for (int i = 0; i < tabla.getRowCount(); i++) {

                for (int j = 0; j < tabla.getColumnCount(); j++) {

                    Object value = tabla.getValueAt(i, j);

                    String cellData = (value == null) ? "" : value.toString();

                    pdfTable.addCell(new PdfPCell(new Paragraph(cellData)));

                }

            }

            // Agregar la tabla al documento

            document.add(pdfTable);

            document.close();

            JOptionPane.showMessageDialog(null, "Informe generado y guardado como 'informe\_estudiantes.pdf'");

        } catch (Exception ex) {

            ex.printStackTrace();

            JOptionPane.showMessageDialog(null, "Error al generar el informe PDF.", "Error", JOptionPane.ERROR\_MESSAGE);

        }

    }

    public static void main(String[] args) {

        SwingUtilities.invokeLater(() -> {

            InterfazTabla interfaz = new InterfazTabla();

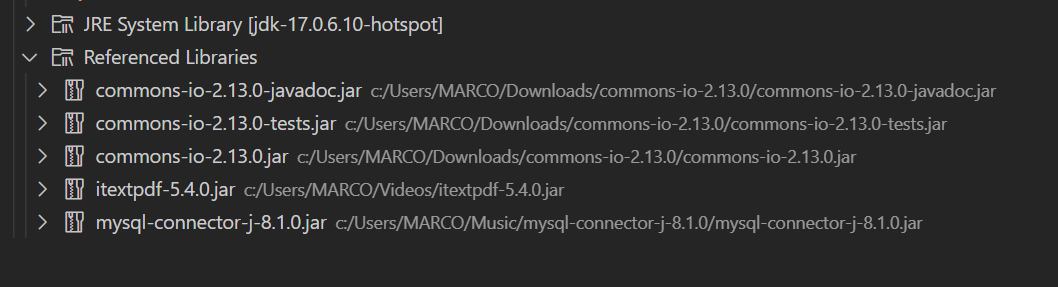
            interfaz.setVisible(true);

        });

    }

}

# BIBLIOTECAS:



**GUÍA DE USUARIO**

Es un proyecto de Java que implementa una interfaz gráfica para gestionar estudiantes y asignaturas. Permite visualizar, agregar, editar y eliminar estudiantes, además de generar informes en formato PDF. A continuación, se presenta una descripción de las principales funcionalidades y componentes del proyecto:

# Funcionalidades:

Mostrar lista de estudiantes y asignaturas en una tabla.

Agregar un nuevo estudiante con su información básica (código, nombre, apellidos, facultad, carrera) y foto.

Editar la información de un estudiante existente.

Eliminar un estudiante de la base de datos y de la tabla.

Subir una foto para un estudiante (también permite editar la foto existente).

Generar un informe en formato PDF con la lista de estudiantes y sus datos.

# Componentes y Detalles:

La interfaz gráfica está construida utilizando la librería Swing de Java.

Se utiliza una tabla para mostrar la lista de estudiantes y asignaturas, con columnas para código, nombre, apellidos, facultad, carrera, foto, botón "Editar" y botón "Eliminar".

Las celdas de la columna de foto contienen imágenes si hay una foto asociada al estudiante, o muestran "No existe Foto" si no hay foto.

Los botones "Editar" y "Eliminar" permiten editar y eliminar estudiantes respectivamente.

Se utiliza un diálogo para mostrar un formulario que permite agregar o editar los datos de un estudiante.

También hay un botón para subir una foto y mostrarla en un JLabel antes de guardarla en la base de datos.

Para la gestión de la base de datos se utiliza MySQL.

Se emplean diferentes clases internas para manejar la renderización y edición de celdas en la tabla, así como para generar el informe PDF.

# Nota:

Es importante mencionar que, para que este proyecto funcione adecuadamente, se debe tener instalado y configurado un servidor MySQL y tener la base de datos "card\_bd1" creada con la tabla "estu\_table1" conteniendo las columnas mencionadas (codigo\_estu, nombre\_estu, apellidos\_estu, facultad\_estu, carrera\_estu y foto\_estu).

REPOSITORIO GIT HUB: https://github.com/Maqui2404/STUDENT\_ADMIN\_JAVA.git