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:

