**“Año del Bicentenario, de la consolidación de nuestra Independencia, y de la conmemoración de las heroicas batallas de Junín y Ayacucho”**

**“UNIVERSIDAD PERUANA LOS ANDES”**

**FACULTAD DE INGENIERÍA**

**ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN**

# Trabajo 1


**Alumno:**

**- Rojas Pariona Miguel Angel**


**Docente: Fernandez Bejarano Raul Enrique**

**Asignatura: ARQUITECTURA DE SOFTWARE**

**Sección: B1**

**Ciclo: VII**

**Huancayo - Perú**

**2024**

# ***Sistema de Registro de Alumnos***

**Enunciado del Proyecto**

Desarrollar un sistema de registro para alumnos que permita gestionar la información básica de los estudiantes, incluyendo nombre, código, fecha de nacimiento, teléfono, carrera y semestre. El sistema debe permitir agregar, actualizar, borrar y buscar registros de alumnos.

## *Requerimientos Funcionales*

| | |
|---|---|
| ***Nombre:*** | ***R1: Registro de Alumno*** |
| ***Resumen:*** | ***Permite al usuario ingresar los datos de un nuevo alumno.*** |
| ***Entrada:*** | ***Nombre, Fecha de nacimiento, Código, Teléfono, Carrera, Semestre.*** |
| ***Resultados:*** | ***El alumno es registrado en el sistema.*** |

| | |
|---|---|
| ***Nombre:*** | ***R2: Guardar Alumno*** |
| ***Resumen:*** | ***Almacena la información del alumno en una lista al hacer clic en "Agregar".*** |
| ***Entrada:*** | ***Datos del alumno ingresados en los campos correspondientes.*** |
| ***Resultados:*** | ***El alumno se añade a la lista de alumnos (ArrayList).*** |

| | |
|---|---|
| ***Nombre:*** | ***R3: Actualizar Información de Alumno*** |
| ***Resumen:*** | ***Permite modificar los datos de un alumno seleccionado de la lista.*** |
| ***Entrada:*** | ***Datos del alumno a modificar (Nombre, Fecha de nacimiento, Código, etc.) y selección del alumno en la lista.*** |
| ***Resultados:*** | ***Los datos actualizados se guardan al hacer clic en "Actualizar".*** |

.

| | |
|---|---|
| *Nombre:* | *R4: Eliminar Alumno* |
| *Resumen:* | *Elimina un alumno seleccionado de la lista.* |
| *Entrada:* | *Selección de un alumno en la lista. Confirmación de eliminación por parte del usuario.* |
| *Resultados:* | *El alumno es eliminado de la lista, previa confirmación.* |

| | |
|---|---|
| *Nombre:* | *R5: Buscar Alumno* |
| *Resumen:* | *Permite buscar alumnos por nombre.* |
| *Entrada:* | *Nombre del alumno o parte del nombre.* |
| *Resultados:* | *Se muestra una lista filtrada con los alumnos que coinciden con la búsqueda.* |

| | |
|---|---|
| *Nombre:* | *R6: Mostrar Lista de Alumnos* |
| *Resumen:* | *Muestra todos los alumnos registrados en una tabla o lista.* |
| *Entrada:* | *Ninguna entrada requerida.* |
| *Resultados:* | *La interfaz muestra todos los alumnos registrados, permitiendo la navegación entre ellos.* |

| | |
|---|---|
| *Nombre:* | *R7: Interfaz Gráfica de Usuario* |
| *Resumen:* | *Proporciona una interfaz gráfica amigable para interactuar con el sistema.* |
| *Entrada:* | *Botones para agregar, actualizar, buscar y eliminar alumnos, campos de texto para ingresar o modificar datos.* |
| *Resultados:* | *Interfaz clara y accesible que permite gestionar alumnos de manera intuitiva.* |

| | |
|---|---|
| ***Nombre:*** | ***R8: Validación de Datos*** |
| ***Resumen:*** | ***Valida los campos obligatorios antes de permitir el registro de un nuevo alumno.*** |
| ***Entrada:*** | ***Datos del alumno (Nombre, Fecha de nacimiento, Código, Teléfono, Carrera, Semestre).*** |
| ***Resultados:*** | ***Si faltan datos, muestra advertencias sin utilizar alertas emergentes; si todos los datos son válidos, permite el registro.*** |

| | |
|---|---|
| ***Nombre:*** | ***R9: Estilo y Diseño*** |
| ***Resumen:*** | ***La aplicación debe tener un diseño visual coherente y atractivo.*** |
| ***Entrada:*** | ***Aplicación de un Look and Feel personalizado.*** |
| ***Resultados:*** | ***La interfaz presenta un diseño visual atractivo y coherente, siguiendo un estilo unificado para todos los componentes gráficos.*** |

# *Diagrama de Clases*

## *Clase Alumno*

- Atributos:
  - nombre: String
  - fechaNacimiento: Date
  - codigo: String
  - telefono: String
  - carrera: String
  - semestre: String
- Métodos:
  - calcularEdad(): int
  - getNombre(): String
  - setNombre(nombre: String): void
  - getFechaNacimiento(): Date
  - setFechaNacimiento(fechaNacimiento: Date): void
  - getCodigo(): String
  - setCodigo(codigo: String): void
  - getTelefono(): String
  - setTelefono(telefono: String): void
  - getCarrera(): String

    - setCarrera(carrera: String): void
    - getSemestre(): String
    - setSemestre(semestre: String): void

### *Clase GestorArrayList*

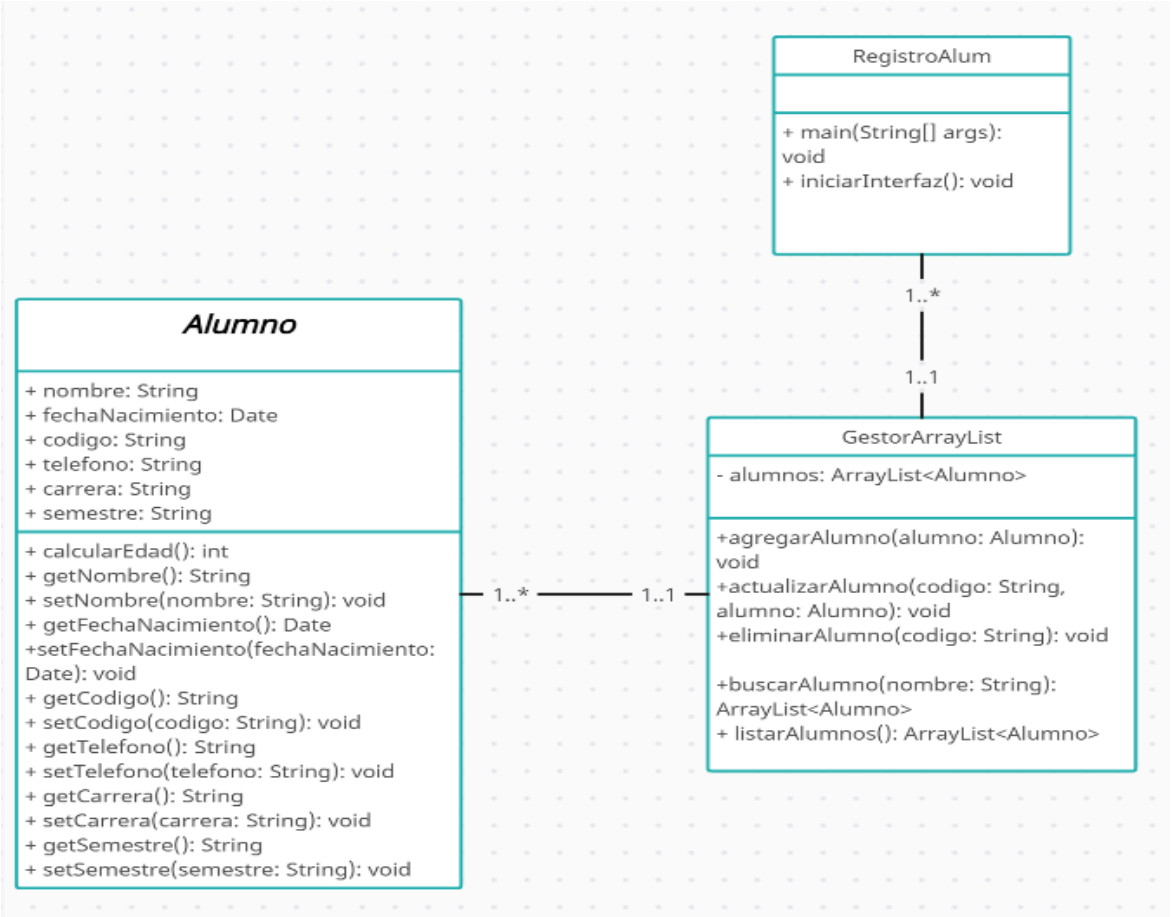
- Atributos:
    - alumnos: ArrayList<Alumno>
- Métodos:
    - agregarAlumno(alumno: Alumno): void
    - actualizarAlumno(codigo: String, alumno: Alumno): void
    - eliminarAlumno(codigo: String): void
    - buscarAlumno(nombre: String): ArrayList<Alumno>
    - listarAlumnos(): ArrayList<Alumno>

### *Clase RegistroAlum*

- Métodos:
    - main(String[] args): void
    - iniciarInterfaz(): void (inicializa y muestra la interfaz gráfica)

### *Relaciones*

- La clase GestorArrayList tiene una relación de composición con la clase Alumno, ya que gestiona una lista de objetos Alumno.
- La clase RegistroAlum utiliza GestorArrayList para interactuar con los datos de los alumnos y mostrar la interfaz gráfica.




# *Código de la aplicación*

## 1. Clase Alumno

```
package com.mycompany.registroalum;

import java.util.Calendar;
import java.util.Date;

public class Alumno {
    private String nombre;
    private Date fechaNacimiento;
    private String codigo;
    private String telefono;
    private String carrera;
    private String semestre;
```

```
    public Alumno(String nombre, Date fechaNacimiento, String codigo, String telefono, String
carrera, String semestre) {
        this.nombre = nombre;
        this.fechaNacimiento = fechaNacimiento;
        this.codigo = codigo;
        this.telefono = telefono;
        this.carrera = carrera;
        this.semestre = semestre;
    }

    public int calcularEdad() {
        Calendar today = Calendar.getInstance();
        Calendar birthDate = Calendar.getInstance();
        birthDate.setTime(fechaNacimiento);
        int age = today.get(Calendar.YEAR) - birthDate.get(Calendar.YEAR);
        if (today.get(Calendar.DAY_OF_YEAR) < birthDate.get(Calendar.DAY_OF_YEAR)) {
            age--;
        }
        return age;
    }

    // Getters y Setters
    public String getNombre() { return nombre; }
    public void setNombre(String nombre) { this.nombre = nombre; }
    public Date getFechaNacimiento() { return fechaNacimiento; }
    public void setFechaNacimiento(Date fechaNacimiento) { this.fechaNacimiento =
fechaNacimiento; }
    public String getCodigo() { return codigo; }
    public void setCodigo(String codigo) { this.codigo = codigo; }
    public String getTelefono() { return telefono; }
    public void setTelefono(String telefono) { this.telefono = telefono; }
    public String getCarrera() { return carrera; }
    public void setCarrera(String carrera) { this.carrera = carrera; }
    public String getSemestre() { return semestre; }
    public void setSemestre(String semestre) { this.semestre = semestre; }
}
```

## 2. Clase GestorArrayList

```
package com.mycompany.registroalum;

import java.util.ArrayList;

public class GestorArrayList {
    private ArrayList<Alumno> alumnos;
```

```
    public GestorArrayList() {
        alumnos = new ArrayList<>();
    }

    public void agregarAlumno(Alumno alumno) {
        alumnos.add(alumno);
    }

    public void actualizarAlumno(String codigo, Alumno alumnoActualizado) {
        for (int i = 0; i < alumnos.size(); i++) {
            if (alumnos.get(i).getCodigo().equals(codigo)) {
                alumnos.set(i, alumnoActualizado);
                return;
            }
        }
    }

    public void eliminarAlumno(String codigo) {
        alumnos.removeIf(alumno -> alumno.getCodigo().equals(codigo));
    }

    public ArrayList<Alumno> buscarAlumno(String nombre) {
        ArrayList<Alumno> encontrados = new ArrayList<>();
        for (Alumno alumno : alumnos) {
            if (alumno.getNombre().equalsIgnoreCase(nombre)) {
                encontrados.add(alumno);
            }
        }
        return encontrados;
    }

    public ArrayList<Alumno> listarAlumnos() {
        return new ArrayList<>(alumnos);
    }
}
```

## 3. Clase RegistroAlum

```
package com.mycompany.registroalum;

import javax.swing.*;

public class RegistroAlum {
    public static void main(String[] args) {
        try {
            // Set HiFi Look and Feel from JTattoo
```

```
            javax.swing.UIManager.setLookAndFeel("com.jtattoo.plaf.hifi.HiFiLookAndFeel");
        } catch (Exception e) {
            e.printStackTrace();
        }

        java.awt.EventQueue.invokeLater(() -> new InterfazRegistro().setVisible(true));
    }
}
```

## 4. Clase InterfazRegistro (para gestionar la interfaz gráfica)

```
public class alumno extends javax.swing.JFrame {

    // ArrayList para almacenar los registros de alumnos
    private ArrayList<Alumnos> listaAlumnos = new ArrayList<>();

    public alumno() {
        initComponents();
        this.setTitle("Registro de Alumnos");
        this.setLocationRelativeTo(null);
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */

   private void btnAgregarActionPerformed(java.awt.event.ActionEvent evt) {
         // Obtener los valores de los campos de texto y combo box
    String nombre = txtNombre.getText();

    // Obtener la fecha desde JDateChooser
    Date fechaSeleccionada = Calendario.getDate();
    String fecha = "";  // Inicia la variable de fecha vacía
    if (fechaSeleccionada != null) {
        // Convertir la fecha a String utilizando SimpleDateFormat
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");  // Formato de fecha
        fecha = sdf.format(fechaSeleccionada);
    }

    String codigoEstudiante = txtCodigo.getText();
    String telefono = txtTelefono.getText();
```

```
    String carrera = jComboBoxCarrera.getSelectedItem().toString();
    String semestre = jComboBoxSemestre.getSelectedItem().toString();

    // Verificar si el registro ya existe (puedes usar el código de estudiante como identificador)
    int selectedRow = tblRegistro.getSelectedRow();
    if (selectedRow != -1) {
        // Actualizar los datos en el ArrayList
        Alumnos alumnoActualizado = listaAlumnos.get(selectedRow);
        alumnoActualizado.nombre = nombre;
        alumnoActualizado.fecha = fecha;
        alumnoActualizado.codigo = codigoEstudiante;
        alumnoActualizado.telefono = telefono;
        alumnoActualizado.carrera = carrera;
        alumnoActualizado.semestre = semestre;

        // Actualizar la tabla
        DefaultTableModel model = (DefaultTableModel) tblRegistro.getModel();
        model.setValueAt(nombre, selectedRow, 0);
        model.setValueAt(codigoEstudiante, selectedRow, 1);
        model.setValueAt(fecha, selectedRow, 2);
        model.setValueAt(telefono, selectedRow, 3);
        model.setValueAt(carrera, selectedRow, 4);
        model.setValueAt(semestre, selectedRow, 5);

        // Mostrar mensaje de éxito
        JOptionPane.showMessageDialog(this, "Registro actualizado con éxito");
    } else {
        // Crear un nuevo objeto Registro con los datos ingresados
        Alumnos nuevoRegistro = new Alumnos(nombre, fecha, codigoEstudiante, carrera,
semestre, telefono);

        // Agregar el nuevo registro a la lista
        listaAlumnos.add(nuevoRegistro);

        // Agregar el nuevo registro a la tabla
        DefaultTableModel model = (DefaultTableModel) tblRegistro.getModel();
        model.addRow(new Object[]{nombre, codigoEstudiante, fecha, telefono, carrera,
semestre});

        // Mostrar mensaje de éxito
        JOptionPane.showMessageDialog(this, "Registro agregado con éxito");
    }
       // Limpiar los campos después de actualizar
        txtNombre.setText("");
```

```
        Calendario.setDate(null);
        txtCodigo.setText("");
        txtTelefono.setText("");
        jComboBoxCarrera.setSelectedIndex(0);
        jComboBoxSemestre.setSelectedIndex(0);
    }

    private void btnBorrarActionPerformed(java.awt.event.ActionEvent evt) {
         // Obtener la fila seleccionada
    int selectedRow = tblRegistro.getSelectedRow();

    if (selectedRow != -1) {
        // Eliminar el registro de la lista
        listaAlumnos.remove(selectedRow);

        // Eliminar la fila seleccionada del modelo de la tabla
        DefaultTableModel model = (DefaultTableModel) tblRegistro.getModel();
        model.removeRow(selectedRow);

        // Mostrar mensaje de confirmación
        JOptionPane.showMessageDialog(this, "Registro borrado con éxito");
    } else {
        // Si no hay fila seleccionada, mostrar mensaje de error
        JOptionPane.showMessageDialog(this, "Por favor selecciona un registro para borrar.");
    }

    }

    private void btnBuscarActionPerformed(java.awt.event.ActionEvent evt) {
       // Obtener la letra inicial del campo de texto
    String letraInicial = txtBuscar.getText().trim().toUpperCase(); // Asegúrate de tener un
campo de texto para buscar

    // Limpiar la tabla antes de mostrar los resultados
    DefaultTableModel model = (DefaultTableModel) tblRegistro.getModel();
    model.setRowCount(0); // Limpiar la tabla

    // Si el campo de búsqueda está vacío, mostrar todos los registros
    if (letraInicial.isEmpty()) {
        for (Alumnos alumno : listaAlumnos) {
            model.addRow(new Object[]{
                alumno.nombre,
                alumno.codigo,
                alumno.fecha,
```

```
                alumno.telefono,
                alumno.carrera,
                alumno.semestre
            });
        }
    } else {
        // Filtrar y buscar en el ArrayList
        for (Alumnos alumno : listaAlumnos) {
            if (alumno.nombre.toUpperCase().startsWith(letraInicial)) {
                // Agregar el registro a la tabla
                model.addRow(new Object[]{
                    alumno.nombre,
                    alumno.codigo,
                    alumno.fecha,
                    alumno.telefono,
                    alumno.carrera,
                    alumno.semestre
                });
            }
        }

        // Mostrar mensaje si no se encontraron resultados
        if (model.getRowCount() == 0) {
            JOptionPane.showMessageDialog(this, "No se encontraron registros que empiecen
con la letra: " + letraInicial);
        }
    }
    }

    private void txtBuscarActionPerformed(java.awt.event.ActionEvent evt) {
      // Obtener la letra inicial del campo de texto
    String letraInicial = txtBuscar.getText().trim().toUpperCase(); // Asegúrate de tener un
campo de texto para buscar

    // Limpiar la tabla antes de mostrar los resultados
    DefaultTableModel model = (DefaultTableModel) tblRegistro.getModel();
    model.setRowCount(0); // Limpiar la tabla

    // Si el campo de búsqueda está vacío, mostrar todos los registros
    if (letraInicial.isEmpty()) {
        for (Alumnos alumno : listaAlumnos) {
            model.addRow(new Object[]{
                alumno.nombre,
                alumno.codigo,
```

```
                alumno.fecha,
                alumno.telefono,
                alumno.carrera,
                alumno.semestre
            });
        }
    } else if (letraInicial.length() == 1) {
        // Filtrar y buscar en el ArrayList
        for (Alumnos alumno : listaAlumnos) {
            if (alumno.nombre.toUpperCase().startsWith(letraInicial)) {
                // Agregar el registro a la tabla
                model.addRow(new Object[]{
                    alumno.nombre,
                    alumno.codigo,
                    alumno.fecha,
                    alumno.telefono,
                    alumno.carrera,
                    alumno.semestre
                });
            }
        }

        // Mostrar mensaje si no se encontraron resultados
        if (model.getRowCount() == 0) {
            JOptionPane.showMessageDialog(this, "No se encontraron registros que empiecen
con la letra: " + letraInicial);
        }
    } else {
        JOptionPane.showMessageDialog(this, "Por favor ingresa una letra inicial válida.");
    }
    }

    private void btnActualizarActionPerformed(java.awt.event.ActionEvent evt) {
 // Obtener la fila seleccionada
    int filaSeleccionada = tblRegistro.getSelectedRow();

    // Verificar si hay una fila seleccionada
    if (filaSeleccionada != -1) {
        // Obtener los valores actuales de la fila seleccionada
        String nombre = (String) tblRegistro.getValueAt(filaSeleccionada, 0);
        String codigo = (String) tblRegistro.getValueAt(filaSeleccionada, 1);
        String fecha = (String) tblRegistro.getValueAt(filaSeleccionada, 2);
        String telefono = (String) tblRegistro.getValueAt(filaSeleccionada, 3);
        String carrera = (String) tblRegistro.getValueAt(filaSeleccionada, 4);
```

```
        String semestre = (String) tblRegistro.getValueAt(filaSeleccionada, 5);

        // Crear una instancia de Alumnos con los datos actuales
        Alumnos alumno = new Alumnos(nombre, fecha, codigo, telefono, carrera, semestre);

        // Llenar los campos del formulario con los datos del alumno
        llenarCampos(alumno);
    } else {
        JOptionPane.showMessageDialog(this, "Por favor selecciona un registro para
actualizar.");
    }

    }
private void llenarCampos(Alumnos alumno) {
    txtNombre.setText(alumno.getNombre());
    txtCodigo.setText(alumno.getCodigo()); // Permitir modificación
    txtTelefono.setText(alumno.getTelefono());
    jComboBoxCarrera.setSelectedItem(alumno.getCarrera());
    jComboBoxSemestre.setSelectedItem(alumno.getSemestre());

    // No actualizamos el campo de fecha directamente, ya que requiere conversión
    // Si se desea, descomentar la siguiente línea para actualizar el JDateChooser:
    // Calendario.setDate(java.sql.Date.valueOf(alumno.getFecha()));
}
```