“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 2

Alumno:

- Pariona Gozme Maicol

Docente: Fernandez Bejarano Raul Enrique

Asignatura: ARQUITECTURA DE SOFTWARE

Sección: B1

Ciclo: VII

Huancayo - Perú

2024

Sistema de Registro de Alumnos

El objetivo de este proyecto es desarrollar un sistema de registro de alumnos que permita gestionar la información de los estudiantes de una universidad. El sistema debe permitir agregar, modificar, buscar y eliminar registros de alumnos, proporcionando una interfaz de usuario sencilla y clara. La información que se gestionará incluye nombre, fecha de nacimiento, código de estudiante, teléfono, carrera, y semestre.

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: 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:	***R3: 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:	***R4: 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:	***R5: Visualización de Alumnos***
Resumen:	***El sistema debe mostrar una lista de todos los alumnos registrados en una tabla.***
Entrada:	***Ninguna entrada requerida.***
Resultados:	***El sistema muestra la lista de todos los alumnos registrados con sus detalles en una tabla.***

Nombre:	***R6: Cálculo de Edad***
Resumen:	***El sistema debe calcular automáticamente la edad de cada alumno basado en su fecha de nacimiento.***
Entrada:	***Fecha de nacimiento del alumno.***
Resultados:	***El sistema muestra la edad del alumno calculada automáticamente.***

Nombre:	***R4: Persistencia de Datos***
Resumen:	***El sistema puede permitir la persistencia de los datos de los alumnos en un archivo o base de datos.***
Entrada:	***Datos de los alumnos que deben guardarse.***
Resultados:	***Los datos se guardan en un archivo o base de datos para futuras consultas o recuperación.***

Diagrama de Clases para el Sistema de Registro de Alumnos

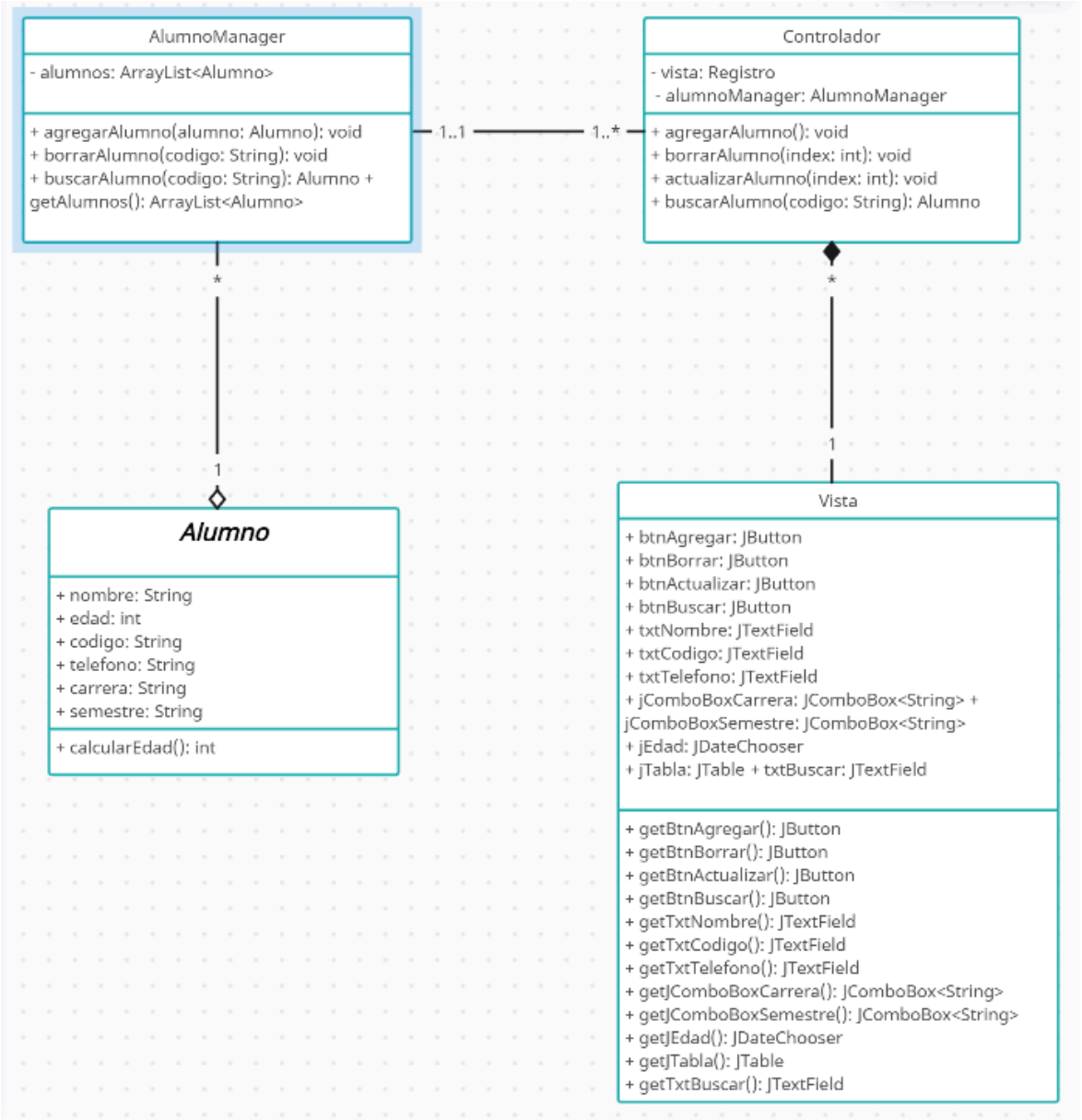
Clases y Relaciones

1. ALUMNO
    - Atributos:
        - -nombre: String
        - -edad: int
        - -codigo: String
        - -telefono: String
        - -carrera: String
        - -sede: String
    - Métodos:
        - +ALUMNO(nombre: String, fechaNacimiento: Date, codigo: String, telefono: String, carrera: String, sede: String)
        - +calcularEdad(fechaNacimiento: Date): int
        - +getNombre(): String
        - +setNombre(nombre: String): void
        - +getEdad(): int
        - +setEdad(edad: int): void
        - +getCodigo(): String
        - +setCodigo(codigo: String): void
        - +getTelefono(): String
        - +setTelefono(telefono: String): void
        - +getCarrera(): String
        - +setCarrera(carrera: String): void
        - +getSede(): String
        - +setSede(sede: String): void
2. ARRAYLIST
    - Atributos:
        - -listaAlumnos: List<ALUMNO>
    - Métodos:
        - +ARRAYLIST()
        - +agregarAlumno(alumno: ALUMNO): void
        - +borrarAlumno(index: int): void
        - +actualizarAlumno(index: int, alumno: ALUMNO): void
        - +getListaAlumnos(): List<ALUMNO>
        - +buscarAlumnoPorCodigo(codigo: String): ALUMNO
        - +buscarAlumnosPorSede(sede: String): List<ALUMNO>
3. CONTROLADOR
    - Atributos:
        - -vista: JFRAMEREGISTRO
        - -modelo: ARRAYLIST
    - Métodos:
        - +CONTROLADOR(vista: JFRAMEREGISTRO, modelo: ARRAYLIST)
        - +agregarAlumno(): void

        - +borrarAlumno(index: int): void
        - +actualizarAlumno(index: int): void
        - +buscarAlumno(codigo: String): void
4. JFRAMEREGISTRO
    - Atributos:
        - -controlador: CONTROLADOR
        - -btnAgregar: JButton
        - -btnBorrar: JButton
        - -btnActualizar: JButton
        - -btnBuscar: JButton
        - -txtNombre: JTextField
        - -txtCodigo: JTextField
        - -txtTelefono: JTextField
        - -jTabla: JTable
        - -jComboBoxCarrera: JComboBox<String>
        - -jComboBoxSede: JComboBox<String>
        - -jEdad: JDateChooser
        - -txtBuscar: JTextField
    - Métodos:
        - +JFRAMEREGISTRO()
        - +getBtnAgregar(): JButton
        - +getBtnBorrar(): JButton
        - +getBtnActualizar(): JButton
        - +getBtnBuscar(): JButton
        - +getTxtNombre(): JTextField
        - +getTxtCodigo(): JTextField
        - +getTxtTelefono(): JTextField
        - +getjTabla(): JTable
        - +getjComboBoxCarrera(): JComboBox<String>
        - +getjComboBoxSede(): JComboBox<String>
        - +getjEdad(): JDateChooser
        - +getTxtBuscar(): JTextField

Relaciones

- JFRAMEREGISTRO tiene una relación de asociación con CONTROLADOR (1 a 1).
- CONTROLADOR tiene una relación de asociación con ARRAYLIST (1 a 1).
- ARRAYLIST tiene una relación de agregación con ALUMNO (1 a muchos).



Código

Clase ALUMNO

```
package Modelo;

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

public class ALUMNO {
    private String nombre;
    private int edad;
    private String codigo;
    private String telefono;
    private String carrera;
```

```
    private String sede;

    // Constructor
    public ALUMNO(String nombre, Date fechaNacimiento, String codigo, String telefono,
String carrera, String sede) {
        this.nombre = nombre;
        this.edad = calcularEdad(fechaNacimiento);
        this.codigo = codigo;
        this.telefono = telefono;
        this.carrera = carrera;
        this.sede = sede;
    }

    public ALUMNO() {
    }

    // Método para calcular la edad a partir de la fecha de nacimiento
    private int calcularEdad(Date fechaNacimiento) {
        if (fechaNacimiento == null) return 0;

        Calendar hoy = Calendar.getInstance();
        Calendar nacimiento = Calendar.getInstance();
        nacimiento.setTime(fechaNacimiento);

        int edad = hoy.get(Calendar.YEAR) - nacimiento.get(Calendar.YEAR);
        if (hoy.get(Calendar.MONTH) < nacimiento.get(Calendar.MONTH) ||
            (hoy.get(Calendar.MONTH) == nacimiento.get(Calendar.MONTH) &&
             hoy.get(Calendar.DAY_OF_MONTH) <
nacimiento.get(Calendar.DAY_OF_MONTH))) {
            edad--;
        }
        return edad;
    }

    // Getters y Setters
    public String getNombre() { return nombre; }
    public void setNombre(String nombre) { this.nombre = nombre; }

    public int getEdad() { return edad; }
    public void setEdad(int edad) { this.edad = edad; }

    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 getSede() { return sede; }
    public void setSede(String sede) { this.sede = sede; }
}
```

Clase ARRAYLIST

```
package Modelo;

import java.util.ArrayList;
import java.util.List;

public class ARRAYLIST {
    private List<ALUMNO> listaAlumnos;

    public ARRAYLIST() {
        this.listaAlumnos = new ArrayList<>();
    }

    public void agregarAlumno(ALUMNO alumno) {
        listaAlumnos.add(alumno);
    }

    public void borrarAlumno(int index) {
        if (index >= 0 && index < listaAlumnos.size()) {
            listaAlumnos.remove(index);
        }
    }

    public void actualizarAlumno(int index, ALUMNO alumno) {
        if (index >= 0 && index < listaAlumnos.size()) {
            listaAlumnos.set(index, alumno);
        }
    }

    public List<ALUMNO> getListaAlumnos() {
        return listaAlumnos;
    }

    public ALUMNO buscarAlumnoPorCodigo(String codigo) {
        for (ALUMNO alumno : listaAlumnos) {
```

```
            if (alumno.getCodigo().equals(codigo)) {
                return alumno;
            }
        }
        return null; // Si no encuentra un alumno con ese código
    }

    public List<ALUMNO> buscarAlumnosPorSede(String sede) {
        List<ALUMNO> alumnosEnSede = new ArrayList<>();
        for (ALUMNO alumno : listaAlumnos) {
            if (alumno.getSede().equalsIgnoreCase(sede)) {
                alumnosEnSede.add(alumno);
            }
        }
        return alumnosEnSede;
    }
}
```

Clase JFRAMEREGISTRO

```
package Vista;

import Controlador.CONTROLADOR;
import com.toedter.calendar.JDateChooser;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JTable;
import javax.swing.JTextField;

public class JFRAMEREGISTRO extends javax.swing.JFrame {
    private CONTROLADOR controlador;

    public JFRAMEREGISTRO() {
        initComponents();
        controlador = new CONTROLADOR(this); // Pasar la vista al
controlador
    }

    public JButton getBtnAgregar() { return btnAgregar; }
    public JButton getBtnBorrar() { return btnBorrar; }
    public JButton getBtnActualizar() { return btnActualizar; }
```

```
    public JButton getBtnBuscar() { return btnBuscar; }
    public JTextField getTxtNombre() { return txtNombre; }
    public JTextField getTxtCodigo() { return txtCodigo; }
    public JTextField getTxtTelefono() { return txtTelefono; }
    public JTable getjTabla() { return jTabla; }
    public JComboBox<String> getjComboBoxCarrera() { return
jComboBoxCarrera; }
    public JComboBox<String> getjComboBoxSede() { return
jComboBoxSede; }
    public JDateChooser getjEdad() { return jEdad; }
    public JTextField getTxtBuscar() { return txtBuscar; }

    private void
btnAgregarActionPerformed(java.awt.event.ActionEvent evt) {
        controlador.agregarAlumno();
    }

    private void btnBorrarActionPerformed(java.awt.event.ActionEvent
evt) {
        controlador.borrarAlumno(jTabla.getSelectedRow());
    }

    private void
btnActualizarActionPerformed(java.awt.event.ActionEvent evt) {
        controlador.actualizarAlumno(jTabla.getSelectedRow());
    }

    private void btnBuscarActionPerformed(java.awt.event.ActionEvent
evt) {
        controlador.buscarAlumno(txtBuscar.getText());
    }

    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new JFRAMEREGISTRO().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
```

```
    private javax.swing.JButton btnActualizar;
    private javax.swing.JButton btnAgregar;
    private javax.swing.JButton btnBorrar;
    private javax.swing.JButton btnBuscar;
    private javax.swing.JComboBox<String> jComboBoxCarrera;
    private javax.swing.JComboBox<String> jComboBoxSede;
    private com.toedter.calendar.JDateChooser jEdad;
    private javax.swing.JTable jTabla;
    private javax.swing.JTextField txtBuscar;
    private javax.swing.JTextField txtCodigo;
    private javax.swing.JTextField txtNombre;
    private javax.swing.JTextField txtTelefono;
    // End of variables declaration
}
```

Clase CONTROLADOR

```
package Controlador;

import Modelo.ARRAYLIST;
import Modelo.ALUMNO;
import Vista.JFRAMEREGISTRO;

public class CONTROLADOR {
    private JFRAMEREGISTRO vista;
    private ARRAYLIST modelo;

    public CONTROLADOR(JFRAMEREGISTRO vista, ARRAYLIST modelo) {
        this.vista = vista;
        this.modelo = modelo;

        // Agregar listeners a los botones
        this.vista.getBtnAgregar().addActionListener(e ->
agregarAlumno());
        this.vista.getBtnBorrar().addActionListener(e ->
borrarAlumno(vista.getjTabla().getSelectedRow()));
        this.vista.getBtnActualizar().addActionListener(e ->
actualizarAlumno(vista.getjTabla().getSelectedRow()));
        this.vista.getBtnBuscar().addActionListener(e ->
buscarAlumno(vista.getTxtBuscar().getText()));
    }
```

```
    public void agregarAlumno() {
        String nombre = vista.getTxtNombre().getText();
        String codigo = vista.getTxtCodigo().getText();
        String telefono = vista.getTxtTelefono().getText();
        String carrera = (String)
vista.getjComboBoxCarrera().getSelectedItem();
        String sede = (String)
vista.getjComboBoxSede().getSelectedItem();
        java.util.Date fechaNacimiento = vista.getjEdad().getDate();

        ALUMNO nuevoAlumno = new ALUMNO(nombre, fechaNacimiento,
codigo, telefono, carrera, sede);
        modelo.agregarAlumno(nuevoAlumno);
        // Actualizar la tabla u otra acción después de agregar
    }

    public void borrarAlumno(int index) {
        modelo.borrarAlumno(index);
        // Actualizar la tabla u otra acción después de borrar
    }

    public void actualizarAlumno(int index) {
        String nombre = vista.getTxtNombre().getText();
        String codigo = vista.getTxtCodigo().getText();
        String telefono = vista.getTxtTelefono().getText();
        String carrera = (String)
vista.getjComboBoxCarrera().getSelectedItem();
        String sede = (String)
vista.getjComboBoxSede().getSelectedItem();
        java.util.Date fechaNacimiento = vista.getjEdad().getDate();

        ALUMNO alumnoActualizado = new ALUMNO(nombre,
fechaNacimiento, codigo, telefono, carrera, sede);
        modelo.actualizarAlumno(index, alumnoActualizado);
        // Actualizar la tabla u otra acción después de actualizar
    }

    public void buscarAlumno(String codigo) {
        ALUMNO alumno = modelo.buscarAlumnoPorCodigo(codigo);
        if (alumno != null) {
```

```
            vista.getTxtNombre().setText(alumno.getNombre());
            vista.getTxtCodigo().setText(alumno.getCodigo());
            vista.getTxtTelefono().setText(alumno.getTelefono());

vista.getjComboBoxCarrera().setSelectedItem(alumno.getCarrera());

vista.getjComboBoxSede().setSelectedItem(alumno.getSede());
            vista.getjEdad().setDate(alumno.getFechaNacimiento());
            // Mostrar otros detalles si es necesario
        } else {
            // Mensaje de error o acción si no se encuentra el
alumno
        }
    }
}
```

Clase SEMANA02

```
package com.mycompany.semana02;

import Vista.JFRAMEREGISTRO;
import Modelo.ARRAYLIST;

public class SEMANA02 {

    public static void main(String[] args) {
        // Configurar el Look and Feel (opcional)
        try {
            // Establecer Look and Feel de Texture desde JTattoo

javax.swing.UIManager.setLookAndFeel("com.jtattoo.plaf.texture.Textu
reLookAndFeel");
        } catch (ClassNotFoundException | InstantiationException |
IllegalAccessException | javax.swing.UnsupportedLookAndFeelException
ex) {

java.util.logging.Logger.getLogger(SEMANA02.class.getName()).log(jav
a.util.logging.Level.SEVERE, null, ex);
        }

        // Crear el modelo y la vista
```

```
        ARRAYLIST modelo = new ARRAYLIST();
        JFRAMEREGISTRO vista = new JFRAMEREGISTRO();

        // Crear el controlador y vincular la vista y el modelo
        new Controlador.CONTROLADOR(vista, modelo);

        // Mostrar la vista
        java.awt.EventQueue.invokeLater(() ->
vista.setVisible(true));
    }
}
```