“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:

- 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

Descripción:

El objetivo del proyecto es desarrollar un sistema de registro de alumnos para una universidad, que permita gestionar la información de los estudiantes, incluyendo nombre, edad, código, teléfono, carrera y semestre. El sistema proporcionará una interfaz gráfica amigable que permitirá a los usuarios agregar, actualizar, borrar y buscar registros de alumnos de manera eficiente.

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: 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: 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:	***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: 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:	***R7: 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:	***R8: Persistencia de Datos***
Resumen:	***Los datos de los alumnos deben almacenarse de manera persistente, utilizando archivos o bases de datos.***
Entrada:	***Información de los alumnos.***
Resultados:	***Los datos se almacenan y persisten entre sesiones.***

Nombre:	***R9: Accesibilidad***
Resumen:	***La interfaz debe ser accesible y fácil de usar, incluso para usuarios sin experiencia técnica.***
Entrada:	***Interacción del usuario con la interfaz gráfica.***
Resultados:	***La interfaz se muestra clara y accesible para todos los usuarios.***

Nombre:	***R10: Documentación***
Resumen:	***El sistema incluirá documentación para explicar su uso, junto con un manual de usuario y notas de instalación y configuración.***
Entrada:	***Documentación del sistema y manual de usuario.***
Resultados:	***Se proporciona una guía clara para los usuarios y administradores del sistema.***

Diagrama de Clases

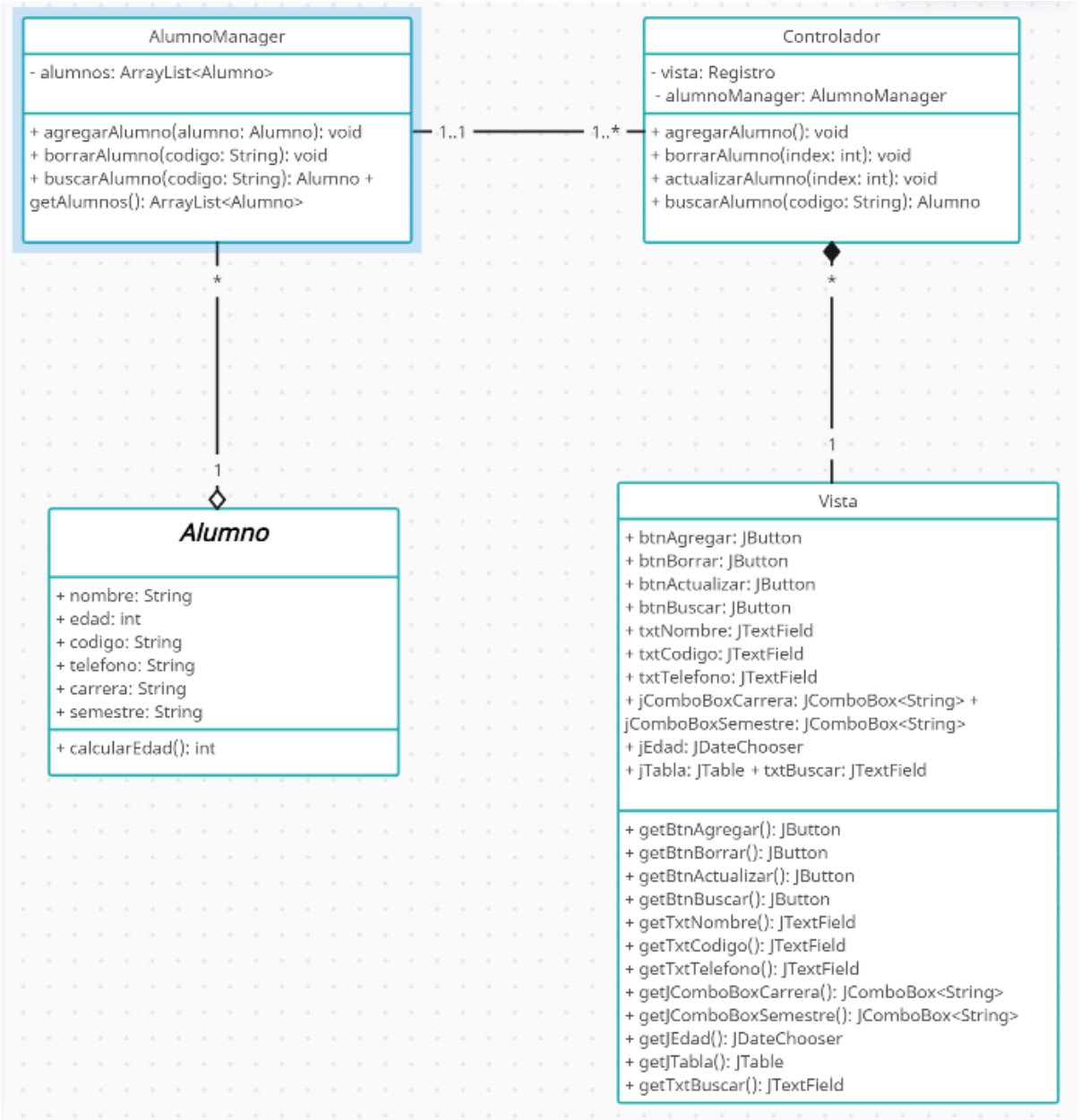
1. **Clase Alumno**
    - Atributos:
        - nombre: String
        - edad: int
        - codigo: String
        - telefono: String
        - carrera: String
        - semestre: String

- Métodos:
  - calcularEdad(): Calcula y devuelve la edad del alumno a partir de la fecha de nacimiento.

2. **Clase AlumnoManager**
   - Atributos:
     - alumnos: ArrayList<Alumno>
   - Métodos:
     - agregarAlumno(alumno: Alumno): Agrega un nuevo alumno a la lista.
     - borrarAlumno(codigo: String): Elimina un alumno basado en su código.
     - buscarAlumno(codigo: String): Alumno: Busca y devuelve un alumno por su código.
     - getAlumnos(): ArrayList<Alumno>: Devuelve la lista de alumnos.
3. **Clase Controlador**
   - Atributos:
     - vista: Registro
     - alumnoManager: AlumnoManager
   - Métodos:
     - agregarAlumno(): Maneja la lógica para agregar un nuevo alumno.
     - borrarAlumno(index: int): Maneja la lógica para borrar un alumno por su índice en la tabla.
     - actualizarAlumno(index: int): Maneja la lógica para actualizar un alumno existente.
     - buscarAlumno(codigo: String): Maneja la búsqueda de un alumno.
4. **Clase Registro (Vista)**
   - Atributos:
     - btnAgregar: JButton
     - btnBorrar: JButton
     - btnActualizar: JButton
     - btnBuscar: JButton
     - txtNombre: JTextField
     - txtCodigo: JTextField
     - txtTelefono: JTextField
     - jComboBoxCarrera: JComboBox<String>
     - jComboBoxSemestre: JComboBox<String>
     - jEdad: JDateChooser
     - jTabla: JTable
     - txtBuscar: JTextField
   - Métodos:
     - Métodos para obtener los componentes de la interfaz (getters para los botones y campos).

Relaciones

- Composición: Registro tiene una relación de composición con Controlador, ya que el controlador opera sobre la vista.
- Agregación: AlumnoManager contiene una lista de Alumno, lo que representa una relación de agregación.
- Asociación: Controlador utiliza AlumnoManager para manejar la lógica de negocio relacionada con los alumnos.



Código de la aplicación

Todo esto está hecho con el modelo vista controlador

Clase Controlador

```
package Controlador;
import Modelo.Alumno;
import Modelo.AlumnoManager;
import Vista.Registro;

public class Controlador {
    private Registro vista;
    private AlumnoManager alumnoManager;

    public Controlador(Registro vista) {
        this.vista = vista;
        this.alumnoManager = new AlumnoManager();
    }

    public void agregarAlumno() {
        String nombre = vista.getTxtNombre().getText();
        int edad = calcularEdad(vista.getjEdad().getDate());
        String codigo = vista.getTxtCodigo().getText();
        String telefono = vista.getTxtTelefono().getText();
        String carrera = (String) vista.getjComboBoxCarrera().getSelectedItem();
        String semestre = (String) vista.getjComboBoxSemestre().getSelectedItem();

        Alumno nuevoAlumno = new Alumno(nombre, edad, codigo, telefono, carrera,
semestre);
        alumnoManager.agregarAlumno(nuevoAlumno);
        actualizarTabla();
    }

    public void borrarAlumno(int index) {
        if (index >= 0) {
            String codigo = alumnoManager.getAlumnos().get(index).getCodigo();
            alumnoManager.borrarAlumno(codigo);
            actualizarTabla();
        }
    }

    public void actualizarAlumno(int index) {
```

```
        if (index >= 0) {
            Alumno alumno = alumnoManager.getAlumnos().get(index);
            // Lógica para actualizar los datos del alumno en la tabla
            // ...
            actualizarTabla();
        }
    }

    public void buscarAlumno(String codigo) {
        Alumno alumno = alumnoManager.buscarAlumno(codigo);
        if (alumno != null) {
            // Mostrar datos del alumno en la vista
            // ...
        }
    }

    private void actualizarTabla() {
        // Lógica para actualizar la tabla en la vista
        // ...
    }

    private int calcularEdad(Date fechaNacimiento) {
        // Lógica para calcular la edad
        return 0; // Placeholder
    }
}
```

Clase modelo

Clase Alumno

```
package Modelo;
public class Alumno {
    private String nombre;
    private int edad;
    private String codigo;
    private String telefono;
    private String carrera;
    private String semestre;

    public Alumno(String nombre, int edad, String codigo, String telefono, String carrera,
String semestre) {
        this.nombre = nombre;
```

```
        this.edad = edad;
        this.codigo = codigo;
        this.telefono = telefono;
        this.carrera = carrera;
        this.semestre = semestre;
    }

    // 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 getSemestre() { return semestre; }
    public void setSemestre(String semestre) { this.semestre = semestre; }

    public void calcularEdad() {
        // Lógica para calcular la edad basada en la fecha de nacimiento
    }
}
```

Clase AlumnoManager

```
package Modelo;
import java.util.ArrayList;

public class AlumnoManager {
    private ArrayList<Alumno> alumnos;

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

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

    public void borrarAlumno(String codigo) {
```

```
        alumnos.removeIf(alumno -> alumno.getCodigo().equals(codigo));
    }

    public Alumno buscarAlumno(String codigo) {
        for (Alumno alumno : alumnos) {
            if (alumno.getCodigo().equals(codigo)) {
                return alumno;
            }
        }
        return null; // Si no se encuentra
    }

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

Clase Vista

Clase Registro

```
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 Registro extends javax.swing.JFrame {
    private Controlador controlador;

    public Registro() {
        initComponents();
        controlador = new Controlador(this);
    }

    // Getters para los componentes
    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> getjComboBoxSemestre() { return jComboBoxSemestre; }
    public JDateChooser getjEdad() { return jEdad; }
    public JTextField getTxtBuscar() { return txtBuscar; }

    // initComponents y otros métodos...

    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());
    }
}
```

Main

```
package com.mycompany.semana02;

import Vista.Registro;

public class SEMANA02 {
    public static void main(String[] args) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Registro().setVisible(true);
            }
        });
    }
}
```