UNIVERSIDAD PERUANA LOS ANDES

FACULTAD DE INGENIERÍA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

PRACTICA N°4

ASIGNATURA: ARQUITECTURA DE SOFTWARE

DOCENTE: FERNÁNDEZ BEJARANO RAUL ENRIQUE

INTEGRANTE: AQUINO CANO, MOISES ISRAEL

CICLO: VII

HUANCAYO-2024
PERÚ

INFORME DE TRABAJO N°4

Título: Sistema de Registro de Alumnos

Este programa implementa un sistema de registro de alumnos utilizando el patrón de diseño Modelo-Vista-Controlador (MVC) en Java. El patrón MVC es una arquitectura de software que separa la lógica de negocio (Modelo), la interfaz de usuario (Vista) y el control del flujo de la aplicación (Controlador) para mantener un diseño más organizado y modular.

Modelo (MODELO): En esta parte del código se gestionan las estructuras de datos y la lógica de negocio relacionada con los alumnos. El modelo Alumno almacena la información de cada estudiante, como código, nombre, dirección, teléfono, fecha de nacimiento y edad. La clase AlumnoArray maneja la colección de alumnos, permitiendo agregar, actualizar y guardar los datos en un archivo TXT para su persistencia.

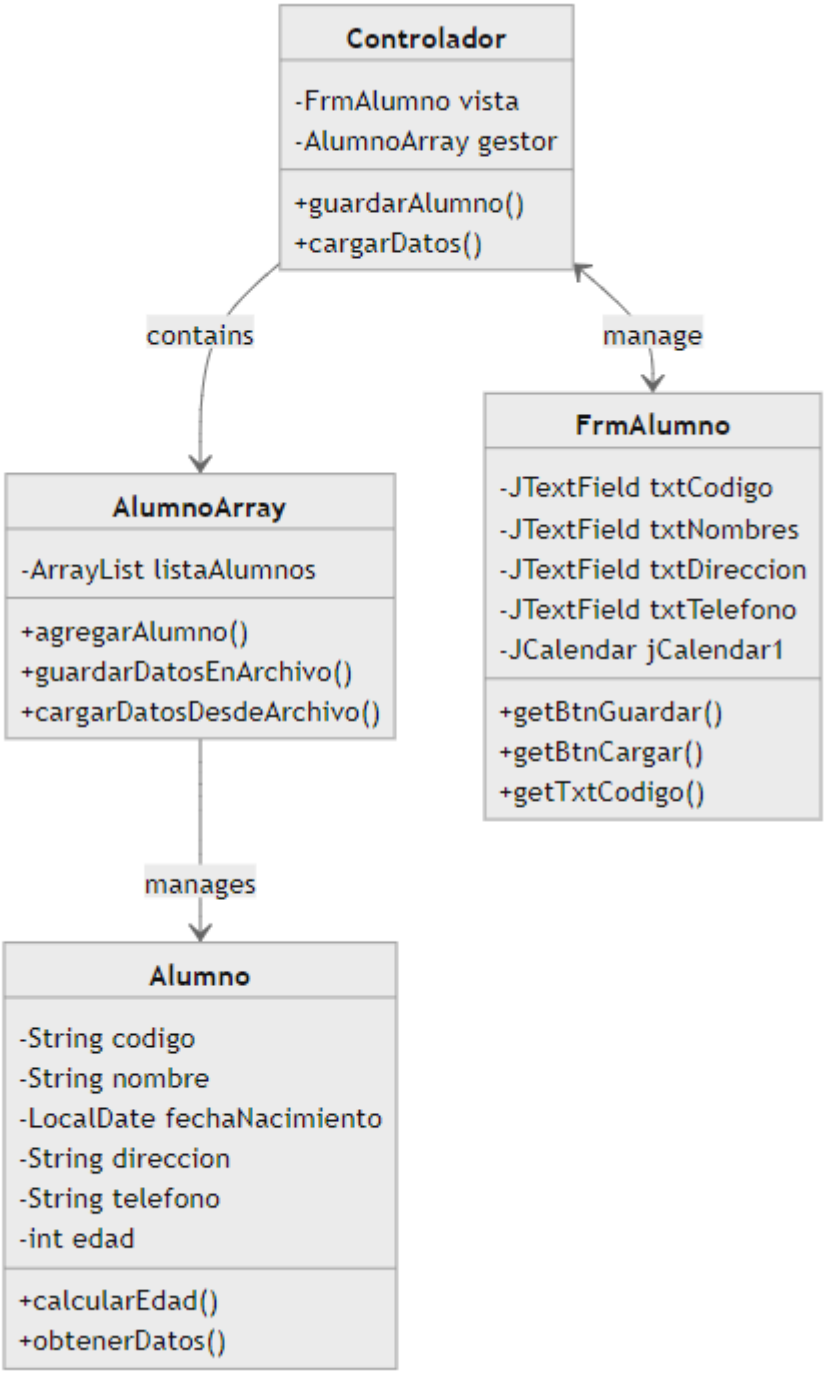
Vista (VISTA): La clase FrmAlumno es la interfaz gráfica del usuario (GUI) diseñada con componentes de Swing, que permite a los usuarios introducir datos de alumnos (como nombre, teléfono y fecha de nacimiento) y visualizar esta información en una tabla.

Controlador (CONTROLADOR): La clase Controlador actúa como el intermediario entre el modelo y la vista. Escucha las acciones del usuario, como hacer clic en el botón "Guardar" o "Cargar", y actualiza el modelo o la vista en consecuencia. Este controlador también gestiona la persistencia de datos en archivos TXT.

Requerimientos Funcionales

1. **Registrar Alumno**: El sistema debe permitir al usuario registrar un nuevo alumno introduciendo su código, nombre, dirección, teléfono y fecha de nacimiento. Una vez registrado, los datos se almacenan en un archivo TXT y se muestran en una tabla dentro de la interfaz gráfica.
2. **Guardar Datos en Archivo**: El sistema debe guardar los datos de los alumnos en un archivo de texto (`estudiantes.txt`) para garantizar la persistencia de los datos, de modo que los registros puedan recuperarse en futuras sesiones.
3. **Cargar Datos desde Archivo**: El sistema debe ser capaz de cargar datos de alumnos desde un archivo TXT, mostrando los registros existentes en la tabla de la interfaz gráfica.
4. **Visualizar Información**: La información de los alumnos debe mostrarse en una tabla, donde se visualizan las columnas de código, nombre, dirección, teléfono y edad.
5. **Validar Campos Obligatorios**: El sistema debe validar que todos los campos requeridos (código, nombre, dirección, teléfono y fecha de nacimiento) sean completados antes de permitir el registro del alumno.
6. **Calcular Edad**: El sistema debe calcular automáticamente la edad del alumno a partir de su fecha de nacimiento.

Diagrama de Clases:



MODELO

CLASE ALUMNO: La clase Alumno representa a un alumno. Tiene atributos que almacenan información como su código, nombre, dirección, teléfono, fecha de nacimiento y edad.

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package MODELO;

import java.time.LocalDate;
import java.time.Period;

/**
 *
 * @author ADMIN
 */
public class Alumno {
    // Atributos
    // Atributos
   private String codigo;
   private String nombre;
   private LocalDate fechaNacimiento;
   private String direccion;
   private String telefono;
   private int edad;

   // Constructor con todos los parámetros
    public Alumno(String codigo, String nombre, LocalDate fechaNacimiento, String direccion, String telefono) {
       this.codigo = codigo;
       this.nombre = nombre;
       this.fechaNacimiento = fechaNacimiento;
       this.direccion = direccion;
       this.telefono = telefono;
       this.edad = calcularEdad();  // Calcular la edad automáticamente
   }
```

```java
// Método para calcular la edad
public int calcularEdad() {
    return Period.between(this.fechaNacimiento, LocalDate.now()).getYears();
}

// Getters y Setters
public String getCodigo() {
    return codigo;
}

public void setCodigo(String codigo) {
    this.codigo = codigo;
}

public String getNombre() {
    return nombre;
}

public void setNombre(String nombre) {
    this.nombre = nombre;
}

public LocalDate getFechaNacimiento() {
    return fechaNacimiento;
}

public void setFechaNacimiento(LocalDate fechaNacimiento) {
    this.fechaNacimiento = fechaNacimiento;
    this.edad = calcularEdad();  // Recalcular la edad cuando cambie la fecha
}

public String getDireccion() {
    return direccion;
}

public void setDireccion(String direccion) {
    this.direccion = direccion;
}

public String getTelefono() {
    return telefono;
}
```

```
    public void setTelefono(String telefono) {
        this.telefono = telefono;
    }

    public int getEdad() {
        return edad;
    }

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

    @Override
    public String toString() {
          return "Alumno{" + "codigo=" + codigo + ", nombre=" + nombre + ", edad=" +
edad + " años, dirección=" + direccion + ", teléfono=" + telefono + '}';
    }
}
```

MODELO

CLASE ALUMNOARRAY: La clase **AlumnoArray** actúa como el gestor o la lógica de negocio para manejar una lista de alumnos. Esta clase permite agregar alumnos, guardar y cargar datos desde un archivo, y actualizar la vista de la tabla.

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package MODELO;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.time.LocalDate;
import java.time.Period;
import java.time.ZoneId;
import java.util.ArrayList;
import java.util.Date;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author ADMIN
 */
public class AlumnoArray {
    private ArrayList<Alumno> listaAlumnos;
    private DefaultTableModel modelo;

    public AlumnoArray(DefaultTableModel modelo) {
        this.listaAlumnos = new ArrayList<>();
        this.modelo = modelo;
    }

    public void agregarAlumno(Alumno alumno, Date fechaNacimiento) {
        // Convertimos la fecha a LocalDate
```

```
                                                    LocalDate nacimiento =
fechaNacimiento.toInstant().atZone(ZoneId.systemDefault()).toLocalDate();
        alumno.setFechaNacimiento(nacimiento);
        listaAlumnos.add(alumno);
        actualizarTabla();
    }

    public void actualizarTabla() {
        modelo.setRowCount(0);
        for (Alumno alumno : listaAlumnos) {
            modelo.addRow(new Object[]{
                alumno.getCodigo(),
                alumno.getNombre(),
                alumno.getDireccion(),
                alumno.getTelefono(),
                alumno.getEdad() + " años"
            });
        }
    }

    public void guardarTablaEnArchivo(String rutaArchivo) {
    try {
        File file = new File(rutaArchivo);
        if (!file.exists()) {
            file.getParentFile().mkdirs();  // Crear directorios si no existen
            file.createNewFile();  // Crear el archivo si no existe
        }

        // Escribir los datos de la tabla en el archivo
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(file))) {
            for (int i = 0; i < modelo.getRowCount(); i++) {
                String codigo = modelo.getValueAt(i, 0).toString();
                String nombre = modelo.getValueAt(i, 1).toString();
                String direccion = modelo.getValueAt(i, 2).toString();
                String telefono = modelo.getValueAt(i, 3).toString();
                String edad = modelo.getValueAt(i, 4).toString();

                    writer.write(codigo + "," + nombre + "," + direccion + "," + telefono + "," +
edad);
                writer.newLine();  // Nueva línea después de cada registro
            }
                    JOptionPane.showMessageDialog(null, "Datos de la tabla guardados
exitosamente en " + rutaArchivo);
        }
```

```
    } catch (IOException e) {
        JOptionPane.showMessageDialog(null, "Error al guardar los datos de la tabla: "
+ e.getMessage());
    }
}


    public void cargarDatosDesdeArchivo(String rutaArchivo) {
                        try (BufferedReader reader = new BufferedReader(new
FileReader(rutaArchivo))) {
            String linea;
            listaAlumnos.clear();
            modelo.setRowCount(0);

            while ((linea = reader.readLine()) != null) {
                String[] datos = linea.split(",");
                if (datos.length == 5) {
                        Alumno alumno = new Alumno(datos[0], datos[1], LocalDate.now(),
datos[2], datos[3]);
                    listaAlumnos.add(alumno);
                        modelo.addRow(new Object[]{datos[0], datos[1], datos[2], datos[3],
datos[4]});
                }
            }
        } catch (IOException e) {
                JOptionPane.showMessageDialog(null, "Error al cargar los datos: " +
e.getMessage());
        }
    }

    public void guardarDatosEnArchivo(String rutaArchivo) {
    try {
        File file = new File(rutaArchivo);
        if (!file.exists()) {
            file.getParentFile().mkdirs();  // Crear directorios si no existen
            file.createNewFile();  // Crear el archivo si no existe
        }

        // Escribir los datos en el archivo con el formato deseado
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(file, true))) {
            int contador = 1; // Para enumerar los alumnos
            for (Alumno alumno : listaAlumnos) {
                writer.write("ALUMNO " + contador); // Nombre del alumno con número
                writer.newLine(); // Salto de línea
```

```
                writer.write("Nombre: " + alumno.getNombre()); // Nombre del alumno
                writer.newLine(); // Salto de línea
                writer.write("Dirección: " + alumno.getDireccion()); // Dirección del alumno
                writer.newLine(); // Salto de línea
                writer.write("Edad: " + alumno.getEdad() + " años"); // Edad del alumno
                writer.newLine(); // Salto de línea
                writer.write("Teléfono: " + alumno.getTelefono()); // Teléfono del alumno
                writer.newLine(); // Salto de línea
                writer.write("----------------------------------------"); // Separador entre alumnos
                writer.newLine();  // Nueva línea para separar los datos de cada alumno
                contador++; // Incrementa el número de alumno
            }
             JOptionPane.showMessageDialog(null, "Datos guardados exitosamente en "
+ rutaArchivo);
        }
    } catch (IOException e) {
                JOptionPane.showMessageDialog(null, "Error al guardar los datos: " +
e.getMessage());
    }
}


    public void limpiar(javax.swing.JTextField txtCodigo,
        javax.swing.JTextField txtNombre,
        javax.swing.JTextField txtDireccion,
        javax.swing.JTextField txtTelefono,
        com.toedter.calendar.JCalendar calendario) {
        txtCodigo.setText("");
        txtNombre.setText("");
        txtDireccion.setText("");
        txtTelefono.setText("");
        calendario.setDate(null);  // Limpiar el calendario
    }

}
```

VISTA

JFRAME FRMALUMNO: La clase **FrmAlumno** es la interfaz gráfica del usuario (GUI), diseñada usando componentes de **Swing**. Contiene todos los elementos gráficos que el usuario ve e interactúa, como los botones, campos de texto y la tabla donde se muestran los alumnos.

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package CONTROLADOR;

import MODELO.Alumno;
import MODELO.AlumnoArray;
import VISTA.FrmAlumno;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.time.LocalDate;
import java.util.Date;
import javax.swing.JOptionPane;

/**
 *
 * @author ADMIN
 */
public class Controlador {
    private AlumnoArray gestor;
    private FrmAlumno vista;

    public Controlador(FrmAlumno vista, AlumnoArray gestor) {
        this.vista = vista;
        this.gestor = gestor;

        this.vista.getBtnGuardar().addActionListener(e -> guardarAlumno());
        this.vista.getBtnCargar().addActionListener(e -> cargarDatosDesdeArchivo());
    }

    public void guardarAlumno() {
```

```
                                    if (vista.getTxtCodigo().getText().isEmpty() ||
vista.getTxtNombres().getText().isEmpty()
                                        || vista.getTxtDireccion().getText().isEmpty() ||
vista.getTxtTelefono().getText().isEmpty()
            || vista.getJCalendar().getDate() == null) {
                JOptionPane.showMessageDialog(vista, "Por favor, complete todos los
campos.");
            return;
        }

        String codigo = vista.getTxtCodigo().getText();
        String nombre = vista.getTxtNombres().getText();
        String direccion = vista.getTxtDireccion().getText();
        String telefono = vista.getTxtTelefono().getText();
        Date fechaNacimiento = vista.getJCalendar().getDate();
                                        LocalDate fechaNacimientoLocal =
fechaNacimiento.toInstant().atZone(java.time.ZoneId.systemDefault()).toLocalDate();

            Alumno alumno = new Alumno(codigo, nombre, fechaNacimientoLocal,
direccion, telefono);
        gestor.agregarAlumno(alumno, fechaNacimiento);

                        gestor.limpiar(vista.getTxtCodigo(), vista.getTxtNombres(),
vista.getTxtDireccion(), vista.getTxtTelefono(), vista.getJCalendar());
        gestor.actualizarTabla();

                String rutaArchivo = "D:\\trabajo arquitectura software\\SEMANA
04\\Ejemplo\\estudiantes.txt";
        gestor.guardarDatosEnArchivo(rutaArchivo);
    }

    public void cargarDatosDesdeArchivo() {
            String rutaArchivo = "D:\\trabajo arquitectura software\\SEMANA
04\\Ejemplo\\estudiantes.txt";

    // Guardar los datos actuales de la tabla en un archivo
    gestor.guardarTablaEnArchivo(rutaArchivo);

     // Si también deseas cargar los datos desde el archivo, puedes mantener esta
línea
    // gestor.cargarDatosDesdeArchivo(rutaArchivo);
}

}
```

CONTROLADOR

CLASE CONTROLADOR: La clase **Controlador** tiene como función principal ser el intermediario entre la vista (**FrmAlumno**) y el modelo (**AlumnoArray**). Se encarga de gestionar las acciones del usuario en la interfaz gráfica y realizar las operaciones correspondientes en el modelo.

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package CONTROLADOR;

import MODELO.Alumno;
import MODELO.AlumnoArray;
import VISTA.FrmAlumno;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.time.LocalDate;
import java.util.Date;
import javax.swing.JOptionPane;
/**
 *
 * @author ADMIN
 */
public class Controlador {
    private AlumnoArray gestor;
    private FrmAlumno vista;

    public Controlador(FrmAlumno vista, AlumnoArray gestor) {
        this.vista = vista;
        this.gestor = gestor;

        this.vista.getBtnGuardar().addActionListener(e -> guardarAlumno());
        this.vista.getBtnCargar().addActionListener(e -> cargarDatosDesdeArchivo());
    }

    public void guardarAlumno() {
```

```
                                    if (vista.getTxtCodigo().getText().isEmpty() ||
vista.getTxtNombres().getText().isEmpty()
                                        || vista.getTxtDireccion().getText().isEmpty() ||
vista.getTxtTelefono().getText().isEmpty()
            || vista.getJCalendar().getDate() == null) {
                JOptionPane.showMessageDialog(vista, "Por favor, complete todos los
campos.");
            return;
        }

        String codigo = vista.getTxtCodigo().getText();
        String nombre = vista.getTxtNombres().getText();
        String direccion = vista.getTxtDireccion().getText();
        String telefono = vista.getTxtTelefono().getText();
        Date fechaNacimiento = vista.getJCalendar().getDate();
                                        LocalDate fechaNacimientoLocal =
fechaNacimiento.toInstant().atZone(java.time.ZoneId.systemDefault()).toLocalDate();

            Alumno alumno = new Alumno(codigo, nombre, fechaNacimientoLocal,
direccion, telefono);
        gestor.agregarAlumno(alumno, fechaNacimiento);

                        gestor.limpiar(vista.getTxtCodigo(), vista.getTxtNombres(),
vista.getTxtDireccion(), vista.getTxtTelefono(), vista.getJCalendar());
        gestor.actualizarTabla();

                String rutaArchivo = "D:\\trabajo arquitectura software\\SEMANA
04\\Ejemplo\\estudiantes.txt";
        gestor.guardarDatosEnArchivo(rutaArchivo);
    }

    public void cargarDatosDesdeArchivo() {
            String rutaArchivo = "D:\\trabajo arquitectura software\\SEMANA
04\\Ejemplo\\estudiantes.txt";

    // Guardar los datos actuales de la tabla en un archivo
    gestor.guardarTablaEnArchivo(rutaArchivo);

     // Si también deseas cargar los datos desde el archivo, puedes mantener esta
línea
    // gestor.cargarDatosDesdeArchivo(rutaArchivo);
}

}
```