UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE

Departamento de las Ciencias de Computación

Carrera de Ingeniería de Software

Curso:

NRC 22412

Estudiantes:

Ronny Joel Ibarra Gaona

Carlos Sebastián Rivera Espín

Angelo Patricio Sánchez Sarabia

Instructor:

Ing. Jenny Ruiz

Fecha:

12-06-2025

Contenido

1. Introducción

En el desarrollo de software, aplicar buenas prácticas arquitectónicas permite construir sistemas más ordenados, mantenibles y escalables. Este proyecto consistió en implementar un sistema CRUD para gestionar estudiantes, utilizando Java con arquitectura en 3 capas. Además, se integraron patrones de diseño como Composite e Interpreter, reforzando conceptos estructurales y de comportamiento en el diseño de software. La aplicación incluye una interfaz gráfica construida con Java Swing, que permite al usuario interactuar de forma intuitiva con los datos registrados.

2. Objetivo General

Desarrollar una aplicación de escritorio en Java que gestione estudiantes mediante operaciones CRUD, aplicando la arquitectura en 3 capas y los patrones de diseño Composite e Interpreter.

3. Marco Teórico

El desarrollo de aplicaciones robustas y mantenibles requiere el uso de patrones arquitectónicos y de diseño que garanticen una correcta separación de responsabilidades y faciliten la escalabilidad del software. Uno de los enfoques más utilizados en aplicaciones empresariales es la arquitectura en 3 capas, la cual divide el sistema en: presentación, lógica de negocio y acceso a datos.

La capa de presentación se encarga de la interacción con el usuario final, generalmente mediante una interfaz gráfica. La capa de lógica de negocio implementa las reglas del sistema, validaciones y procesos que controlan el flujo de la aplicación. Finalmente, la capa de datos gestiona la persistencia, ya sea en memoria o mediante una base de datos. Esta separación permite que cada capa evolucione de forma independiente y que el código sea más fácil de mantener, probar y reutilizar.

Además, durante este proyecto se aplicaron patrones de diseño reconocidos. El patrón Composite, perteneciente a los patrones estructurales, permite componer objetos en estructuras jerárquicas de árbol, donde los objetos compuestos y los individuales se tratan de manera uniforme. En el contexto de interfaces gráficas, como con Java Swing, este patrón facilita la organización de componentes como paneles, etiquetas, botones y tablas, mejorando la modularidad y reutilización visual.

Por otro lado, el patrón Interpreter, que pertenece a los patrones de comportamiento, se utiliza cuando se necesita interpretar o evaluar expresiones. En este caso, se lo aplicó para procesar comandos simples desde el usuario (por ejemplo, búsquedas por ID o nombre), simulando una gramática básica. Este patrón resulta útil cuando se busca flexibilidad en la ejecución de acciones según entradas dinámicas.

Finalmente, el concepto CRUD (Create, Read, Update, Delete) representa las operaciones básicas de manipulación de datos. Son fundamentales en aplicaciones que gestionan registros, ya que permiten mantener actualizada y organizada la información. Estas operaciones fueron implementadas en este proyecto utilizando estructuras en memoria para simplificar el alcance inicial, pero con posibilidad de ser migradas a persistencia en bases de datos.

El uso de Java Swing como tecnología para la interfaz gráfica permitió implementar una aplicación visualmente funcional y modular, en la que se conectan eficientemente todas las capas del sistema, aplicando los principios mencionados.

4. Desarrollo

4.1.Patrón Composite

Se crearon las siguientes clases:

- **EstudianteComponente:** Clase abstracta o interfaz que define la estructura común para componentes simples y compuestos.
- **EstudianteHoja:** Representa un estudiante individual (hoja en la jerarquía).
- **EstudianteCompuesto:** Representa un grupo o colección de estudiantes, permite agregar o eliminar componentes.

Funcionalidad:

Permite tratar a un grupo de estudiantes y estudiantes individuales de manera uniforme, facilitando la manipulación de conjuntos de estudiantes.

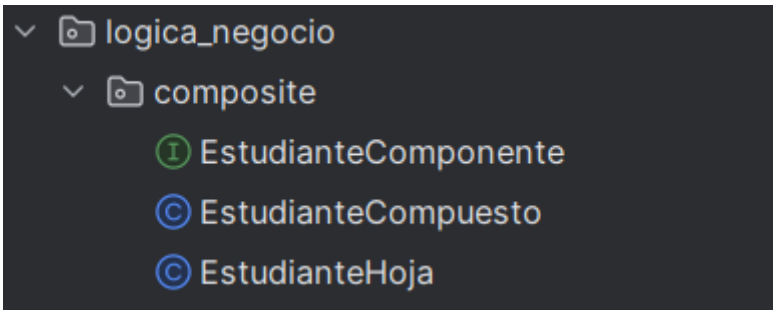

Ilustración 1 Estructura Composite

Resultado:

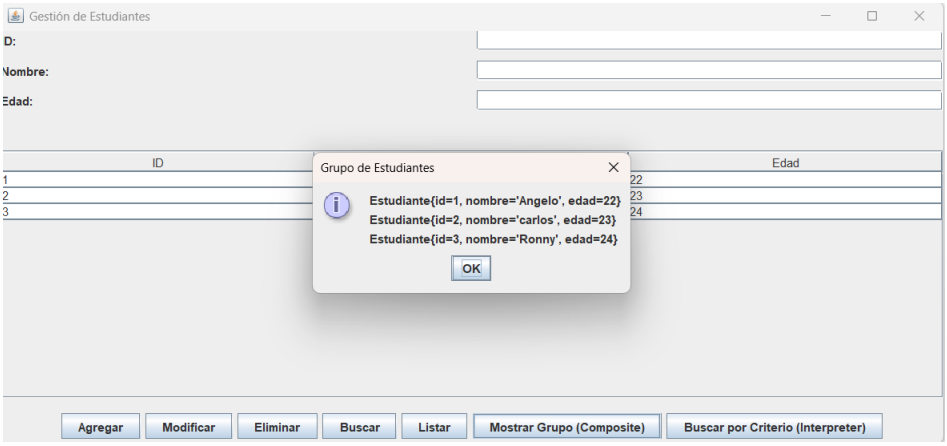

Ilustración 2 Ejecución composite

4.2.Patrón Interpreter

- Se definió la interfaz Expresion para interpretar condiciones o criterios sobre estudiantes.

- Se implementaron expresiones concretas, por ejemplo:
    - **ExpresionEdadMayor:** Evalúa si un estudiante tiene edad mayor a un valor dado.
    - **ExpresionNombreContiene:** Evalúa si el nombre del estudiante contiene una cadena dada.

Funcionalidad:

Facilita la interpretación dinámica de criterios de búsqueda, permitiendo filtrar estudiantes según condiciones definidas en tiempo de ejecución.

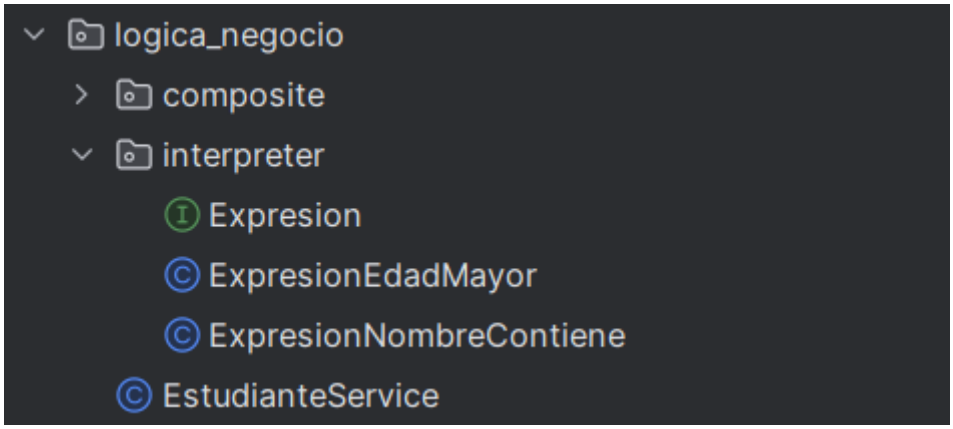

Ilustración 3 Estructura Interpreter

5. Resultados:

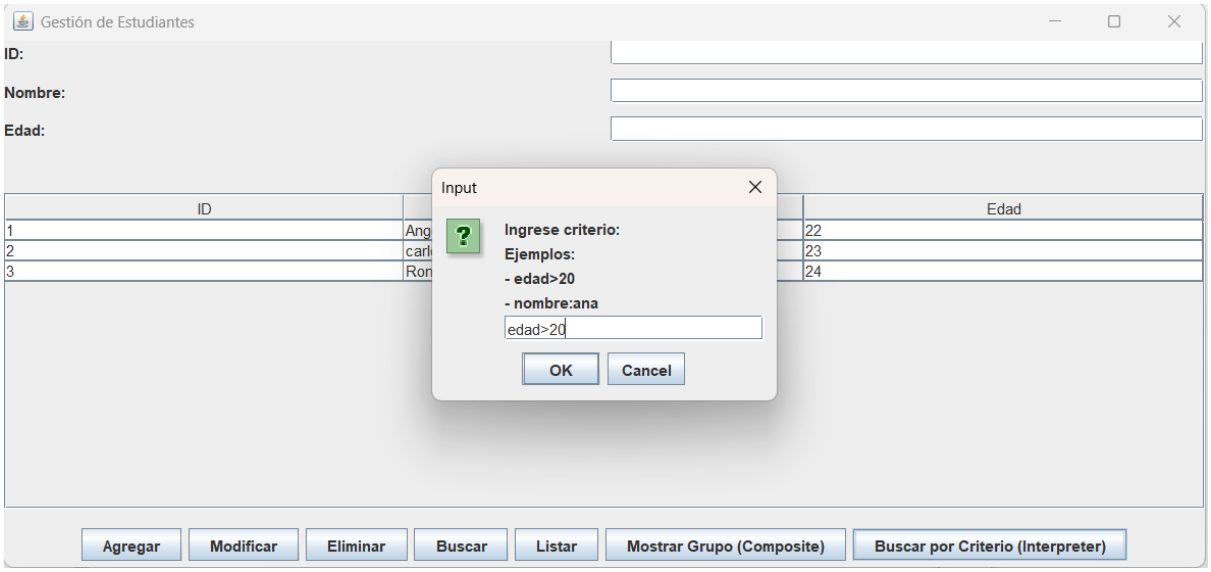

Ilustración 4 Ejecución Interpreter 1

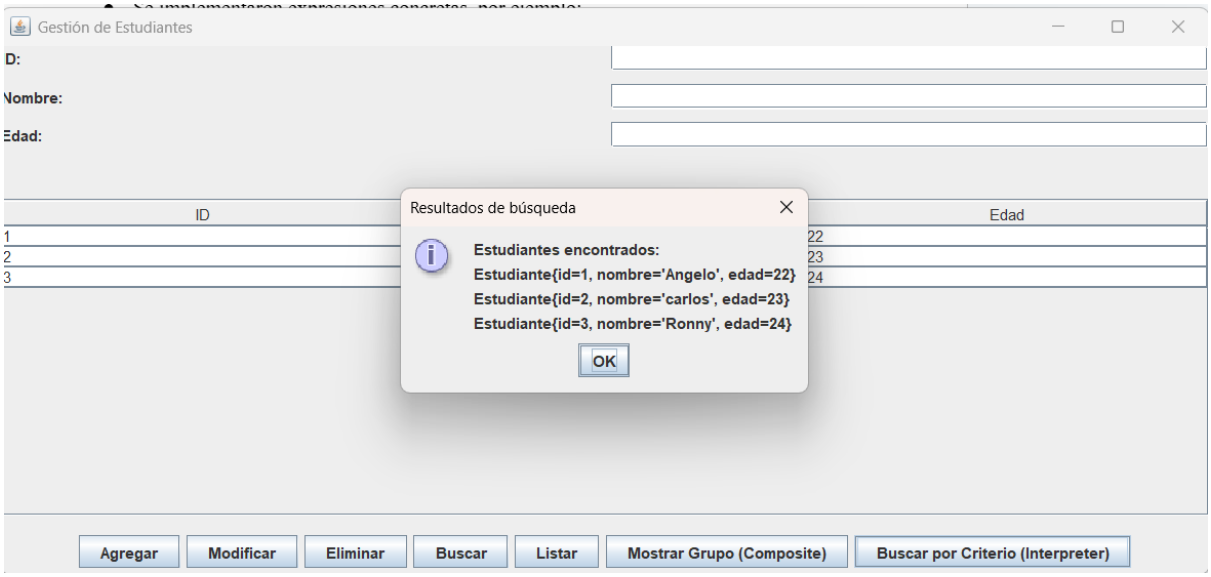

Ilustración 5 Ejecución interpreter 2

5.1.Cambios en la Interfaz Gráfica

Se creó la clase EstudianteUI basada en Java Swing, con los siguientes componentes:

- Campos de entrada para ID, Nombre y Edad.
- Tabla para mostrar la lista de estudiantes con sus atributos.

- Botones para operaciones CRUD: Agregar, Modificar, Eliminar, Buscar y Listar estudiantes.

Nuevos botones:

- Mostrar Grupo (Composite): Muestra en un cuadro de diálogo el listado de estudiantes agrupados usando el patrón Composite.
- Buscar por Criterio (Interpreter): Permite al usuario ingresar un criterio (como edad>20 o nombre:Angelo) y muestra los estudiantes que cumplen ese criterio usando el patrón Interpreter.

Integración:

Los eventos de los botones están conectados con la lógica del servicio EstudianteService, que a su vez opera con el modelo de datos.

5.2.Beneficios Obtenidos

- Modularidad y escalabilidad: Gracias a la organización en paquetes y al uso de patrones de diseño, el proyecto es más fácil de mantener y ampliar.
- Flexibilidad en búsquedas: El patrón Interpreter permite agregar nuevos criterios de búsqueda sin modificar código existente.
- Manejo uniforme de estudiantes individuales y grupos: El patrón Composite facilita el tratamiento conjunto o individual sin diferencias en la lógica.
- Interfaz gráfica amigable: La UI con Swing permite una interacción directa y sencilla para el usuario final, con retroalimentación visual clara.

6. Conclusiones

La implementación de los patrones Composite e Interpreter en el proyecto de gestión de estudiantes ha aportado una arquitectura más robusta y extensible. La integración con una interfaz gráfica mediante Swing ha permitido una experiencia de usuario práctica y funcional, cubriendo los requerimientos de manejo, visualización y búsqueda avanzada de estudiantes.

7. Recomendaciones

- Mantener la separación de capas

  Mantén claramente separada la lógica de negocio, la presentación (UI) y el acceso a datos para facilitar mantenimiento y escalabilidad.
- Validar correctamente las entradas de usuario

  Siempre valida las entradas en la interfaz para evitar errores y datos inconsistentes.
- Extender el patrón Interpreter para nuevas búsquedas

  Usa el patrón Interpreter para agregar fácilmente nuevos criterios sin modificar código existente.
- Implementar manejo adecuado de excepciones

  Controla las excepciones para evitar fallos inesperados y mejorar la experiencia del usuario.
- Documentar bien el uso de patrones y código

  La documentación facilita la comprensión y el mantenimiento del proyecto.