Análisis patrón de diseño controlador de bases de datos

d.barbosam@uniandes.edu.co

Información general

Este proyecto es un controlador sencillo de bases de datos, una especie de implementación de SQL en Java que permite operaciones como la carga y unión de tablas, eliminación y modificación de registros, actualización de datos, selección y creación de nuevas tablas. Además, incluye una función de encriptación que se aplica de forma independiente a cada campo dentro de una tabla.

El reto principal de este proyecto radica en la necesidad de realizar una acción distinta para cada instrucción. Esta diversidad genera un alto acoplamiento y una cohesión muy débil o casi nula. Cada instrucción requiere un manejo específico de datos, lo que implica un procesamiento variado y extenso de la información. Además, abordar las funciones de encriptación y desencriptación por separado podría resultar en problemas de desacoplamiento, lo que podría dejar la clase sin funcionalidad o, en el otro extremo, generar altos niveles de dependencia al mover datos entre partes del código.

Patrón, aplicación ventajas y desventajas

El patrón 'Chain of Responsibility' es un patrón de comportamiento que implica una cadena por la cual se transmite una instrucción. En esencia, se crea una interfaz o clase abstracta que recibe la instrucción y la pasa a un receptor. Este receptor decide si ejecutar la instrucción o pasarla al siguiente elemento de la cadena. El objetivo es que el emisor de la instrucción no conozca directamente su receptor, reduciendo así el acoplamiento. Sin embargo, este enfoque puede llevar a incluir una instrucción específica para cada acción, lo que incrementa el acoplamiento y dificulta modificar el sistema sin afectar su totalidad.

En este proyecto, la clase 'Main' sirve como emisor y las clases que extienden la clase abstracta funcionan como receptores. El usuario ingresa comandos en lugar de seleccionar opciones de una lista, lo que simplifica la interacción. Aunque este patrón minimiza el acoplamiento y la complejidad del código al separar la asignación de acciones, no todas las instrucciones pueden ser ejecutadas. Esto puede requerir una acción predeterminada y también puede aumentar el tiempo de respuesta al recorrer la cadena de clases.

Una alternativa específica para este proyecto sería crear una única clase que contenga múltiples métodos para acciones como actualización, eliminación o selección, aplicadas a una tabla determinada. El usuario elegiría la acción mediante un número, lo que reduciría el acoplamiento. Sin embargo, otras acciones como modificación o unión podrían requerir clases adicionales, lo que no sería tan eficiente como el patrón mencionado anteriormente.