SISTEMA DE GESTIÓN DE BIBLIOTECA

Proyecto Ingeniería del Software II

Descripción breve

El Sistema de Gestión de Biblioteca permite a los usuarios buscar, reservar y devolver libros de manera eficiente. Implementa patrones de diseño para garantizar escalabilidad y mantenibilidad, además de integrar notificaciones y distintos roles de usuario. Su desarrollo en Java con MySQL lo hace modular y flexible.

Pablo Carrasco Melero, Pablo Monda Cana, Pablo Aranda Cortés

pcarmel@alu.upo.es, pmoncan@alu.upo.es, paracor@alu.upo.es

Descripción General

El sistema de gestión de biblioteca tiene como objetivo facilitar a los usuarios la búsqueda, reserva y devolución de libros, así como la gestión eficiente de los recursos bibliotecarios. Para lograrlo, se implementará una arquitectura basada en patrones de diseño que proporcionará modularidad, escalabilidad y mantenibilidad al sistema.

Dado que el sistema manejará múltiples funcionalidades, como la gestión de usuarios, libros, préstamos y notificaciones, es fundamental estructurar el código de manera que sea flexible y reutilizable. Por ello, se emplearán diferentes **patrones de diseño** que permitirán desacoplar responsabilidades, optimizar la creación de objetos y facilitar la interacción entre los componentes del sistema.

Patrones de Diseño Implementados

1. Singleton

- **Descripción:** Asegura que una clase tenga una única instancia y proporciona un punto de acceso global a esa instancia.
- **Aplicación en el sistema:**
    - Se aplicará al gestor de base de datos para garantizar que todas las operaciones del sistema sean gestionadas a través de una única conexión centralizada, evitando la creación innecesaria de múltiples instancias y reduciendo el consumo de recursos.
- **¿Por qué se usa este patrón?**
    - Evita problemas de concurrencia y optimiza el rendimiento al asegurarse de que la base de datos solo tiene una única instancia en uso.

2. Factory Method

- **Descripción:** Define una interfaz para la creación de objetos, pero permite a las subclases decidir qué clase concreta instanciar.
- **Aplicación en el sistema:**
    - Se empleará para la creación de diferentes tipos de libros, como libros físicos y libros electrónicos, asegurando que el código de creación de objetos sea flexible y extensible.
- **¿Por qué se usa este patrón?**
    - Facilita la adición de nuevos tipos de libros sin modificar el código existente, promoviendo el principio de **abierto/cerrado**.

3. Abstract Factory

- **Descripción:** Proporciona una interfaz para crear familias de objetos relacionados sin especificar sus clases concretas.
- **Aplicación en el sistema:**
    - Se implementará para gestionar la creación de diferentes tipos de usuarios (estudiantes, profesores, investigadores), asegurando que cada tipo de usuario tenga sus propias reglas y permisos dentro del sistema.
- **¿Por qué se usa este patrón?**
    - Permite la creación flexible de diferentes perfiles de usuario sin alterar la lógica principal del sistema.

4. Builder

- **Descripción:** Permite construir un objeto complejo paso a paso y separa el proceso de construcción de su representación.
- **Aplicación en el sistema:**
    - Se utilizará para construir objetos de libros con atributos opcionales como autor, año de publicación, género, número de páginas, entre otros.
- **¿Por qué se usa este patrón?**
    - Evita la creación de múltiples constructores con parámetros diferentes y mejora la legibilidad del código.

5. Prototype

- **Descripción:** Permite la creación de nuevos objetos clonando una instancia existente.
- **Aplicación en el sistema:**
    - Se utilizará para la clonación de objetos de libros cuando sea necesario generar copias rápidas sin necesidad de crear nuevas instancias desde cero.
- **¿Por qué se usa este patrón?**
    - Reduce el costo computacional de crear objetos nuevos desde cero y permite copiar objetos manteniendo sus atributos originales.

6. Adapter

- **Descripción:** Convierte la interfaz de una clase en otra interfaz que los clientes esperan, permitiendo que dos sistemas incompatibles trabajen juntos.
- **Aplicación en el sistema:**
    - Se utilizará para integrar el sistema de gestión de biblioteca con una base de datos externa o servicios de terceros que tengan una interfaz diferente a la esperada.
- **¿Por qué se usa este patrón?**
    - Permite que el sistema funcione con bases de datos o servicios externos sin necesidad de modificar la lógica interna.

7. Decorator

- **Descripción:** Permite añadir funcionalidades adicionales a un objeto de forma dinámica sin alterar su estructura.
- **Aplicación en el sistema:**
    - Se aplicará para extender la funcionalidad de los libros, permitiendo agregar características como "favoritos", "reseñas de usuarios" o "marcar como leído" sin modificar la clase principal de Libro.
- **¿Por qué se usa este patrón?**
    - Proporciona una solución flexible para añadir funcionalidades sin necesidad de modificar la estructura de clases existentes.

8. Observer

- **Descripción:** Define una relación uno-a-muchos entre objetos de manera que cuando un objeto cambie su estado, todos sus observadores sean notificados automáticamente.
- **Aplicación en el sistema:**
    - Se empleará para notificar a los usuarios cuando un libro reservado esté disponible para su recogida o cuando haya nuevos libros añadidos en su categoría de interés.
- **¿Por qué se usa este patrón?**
    - Mantiene la sincronización automática entre la disponibilidad de libros y las notificaciones a los usuarios sin necesidad de verificaciones constantes en el código.

9. Strategy

- **Descripción:** Permite definir una familia de algoritmos, encapsular cada uno y hacerlos intercambiables sin modificar el código del cliente.
- **Aplicación en el sistema:**
    - Se usará para implementar distintos algoritmos de búsqueda de libros, como búsqueda por título, por autor o por género, permitiendo cambiar la estrategia sin afectar el resto del código.
- **¿Por qué se usa este patrón?**
    - Mejora la flexibilidad del sistema al permitir que los algoritmos de búsqueda sean seleccionados dinámicamente según la necesidad del usuario.

10. Command

- **Descripción:** Encapsula una solicitud como un objeto, permitiendo parametrizar clientes con diferentes solicitudes, encolar o registrar solicitudes y soportar operaciones que se pueden deshacer.
- **Aplicación en el sistema:**
    - Se aplicará a las operaciones de reserva y devolución de libros, encapsulando cada acción en un objeto de comando que puede ser ejecutado, deshecho o reejecutado.
- **¿Por qué se usa este patrón?**
    - Permite implementar un sistema de **deshacer/rehacer**, lo que resulta útil para revertir reservas accidentales o recuperar préstamos cancelados.

Conclusión

El uso de estos patrones de diseño en el sistema de gestión de biblioteca no solo mejora la **estructuración del código**, sino que también facilita la **mantenibilidad, reutilización y escalabilidad** del sistema. Cada patrón ha sido seleccionado cuidadosamente para resolver problemas específicos de diseño, asegurando que el sistema sea robusto y adaptable a futuras mejoras.