Relatório do Projeto: Sistema de Gestão de Biblioteca

Introdução

Este projeto consiste no desenvolvimento de um **sistema de gestão de biblioteca em Java**, implementado de forma incremental ao longo de várias semanas, aplicando conceitos de programação orientada a objetos e padrões de design.

O objetivo principal é simular funcionalidades de uma biblioteca digital, permitindo a gestão de livros e utilizadores, bem como operações de empréstimo e devolução. Além disso, foram aplicados padrões de design conhecidos (Singleton, Factory, Observer) e boas práticas de arquitetura de software.

Primeiro Sprint

Funcionalidades Implementadas

Gestão de Livros

- Adicionar novos livros à biblioteca.
- Listar todos os livros disponíveis.
- Pesquisar livros por **título** ou **autor**.
- Guardar livros em ficheiro (livros.txt) e carregar automaticamente no arranque.

Gestão de Utilizadores

- Adicionar utilizadores à biblioteca.
- Diferenciar tipos de utilizador: **Aluno**, **Professor** e **Administrador**.
- Listar todos os utilizadores registados.
- Guardar utilizadores em ficheiro (utilizadores.txt) e carregar automaticamente no arranque.

Empréstimos e Devoluções

- Emprestar livros (movendo-os da lista de disponíveis para emprestados).
- Devolver livros (voltando para a lista de disponíveis).
- Notificação de devolução via **Observer**: todos os utilizadores registados são notificados quando um livro é devolvido.

Padrões de Design Aplicados

1. Singleton

O **BibliotecaService** foi implementado como um **Singleton**, garantindo que só existe uma única instância do serviço durante a execução do programa.

- Vantagem: permite que todas as operações sobre livros e utilizadores acedam ao mesmo estado global da biblioteca.

2. Factory

Foi criada a classe **UtilizadorFactory**, responsável pela criação de objetos `Utilizador` dos diferentes tipos (Aluno, Professor, Administrador).

- Vantagem: isola a lógica de criação de objetos, evitando erros ao usar construtores diretamente e tornando o código mais flexível e extensível.

3. Observer

Foi implementado um sistema de **notificações**:

- Quando um livro é devolvido, todos os utilizadores registados como observadores são notificados via `System.out.println`.

- Vantagem: simula um sistema de alertas em tempo real, aplicando o padrão **publish-subscribe**.

Estrutura do Projeto (Atual)

Atualmente, o projeto encontra-se organizado em pacotes:

- **model**
  - Livro
  - Utilizador
- **service**
  - BibliotecaService (lógica principal do sistema – Singleton, Observer)
  - UtilizadorFactory (padrão Factory)
- **main**
  - Main (responsável pela interação com o utilizador via consola, menus e input/output)

Conclusão e Próximos Passos

Até ao momento, o projeto já suporta as principais operações de uma biblioteca: registo de utilizadores, gestão de livros, empréstimos e devoluções. Além disso, foram aplicados padrões de design clássicos que tornam o código mais robusto e extensível.

🔹 **Próximo sprint:** aplicar **Arquitetura em Camadas (MVC)**, reorganizando o projeto em pacotes separados por model, repository, service e controller, tornando o código mais próximo de uma arquitetura profissional.