

Teste Prático

Desenvolvedor Back End Pleno

Enredo do Teste: Sistema de Gestão de Aluguéis de Livros

Cenário:

A PGE deseja desenvolver uma API para gerenciar o empréstimo de livros em sua biblioteca interna. A API deve permitir o cadastro e consulta de livros, usuários (funcionários), e a realização de empréstimos, garantindo que o livro não seja emprestado a mais de um usuário ao mesmo tempo.

Desenvolver a API de empréstimos da biblioteca da PGE, baseada nos princípios de DDD, separando claramente responsabilidades de **domínio**, **aplicação**, **infraestrutura** e **interface (API)**.

Requisitos Funcionais:

1. Cadastro de Livros

- Título (obrigatório)
- Autor (obrigatório)
- ISBN (único)
- Quantidade disponível (≥ 0)

2. Cadastro de Usuários

- Nome
- Matrícula (única)
- Email (formato válido)

3. Empréstimo de Livro

- Um usuário pode pegar emprestado um ou mais livros (se disponíveis).
- Ao fazer um empréstimo, a quantidade disponível do livro deve ser reduzida.
- Cada empréstimo possui:
 - Data de início

- Data prevista de devolução (máximo 14 dias após o início)

4. Devolução de Livro

- A devolução de um livro aumenta sua quantidade disponível.
- Registrar a data de devolução.

5. Consulta

- Listar livros disponíveis.
- Listar empréstimos por usuário.
- Verificar se um usuário possui livros em atraso (baseado na data prevista de devolução).

Regras de Negócio:

- Não é possível emprestar um livro sem unidades disponíveis.
- A data prevista de devolução não pode ser superior a 14 dias.
- Um usuário não pode ter mais de 5 livros emprestados simultaneamente.

Design Patterns exigidos

Domain Service Pattern

Use um serviço de domínio para centralizar regras complexas que envolvem mais de uma entidade, como:

- Verificar disponibilidade de livros;
- Validar o limite de empréstimos por usuário;
- Calcular a data máxima de devolução;
- Esse serviço não conhece controladores nem banco de dados.

Factory Pattern

- Crie uma **LoanFactory** para encapsular a lógica de criação de um novo empréstimo com as regras corretas (ex: data prevista de devolução).

Avaliação Técnica

Critério	Observações
Estrutura com DDD	Separação clara entre camadas e responsabilidades
Design Patterns	Uso apropriado de Service e Factory Pattern
Clean Code	Código limpo, legível, com nomes expressivos, métodos coesos e responsabilidades bem definidas
Testes	JUnit + Mockito (domain services e controllers)
Tratamento de exceções	@ ControllerAdvice com retorno padronizado
DTOs	Separação entre entidade e modelo de transporte
Spring Boot idiomático	Uso de dependências corretas, validação via annotations, etc.
Persistência	JPA/Hibernate com PostgreSQL (ou H2)
Documentação	Swagger (OpenAPI)
Diferenciais (se incluir)	JWT, Docker, testes de integração

Observações Gerais

O candidato deverá disponibilizar o código-fonte em um repositório público utilizando uma plataforma de versionamento como **GitHub**, **GitLab** ou **Bitbucket**. Além disso, deverá gravar um **vídeo auto explicativo**, apresentando o funcionamento da aplicação e justificando as principais decisões técnicas adotadas no desenvolvimento.