Documento de Arquitetura de Software

LibShow - Gerenciamento Inteligente de Biblioteca

Versão 0.1

Conteúdo

1. Introdução

1.1 Objetivo

Este documento descreve a arquitetura do sistema **LibShow**, projetado para gerenciar o acervo, empréstimos, devoluções e reservas de uma biblioteca universitária. A arquitetura fornece uma base técnica para orientar o desenvolvimento, padronizar decisões e facilitar a comunicação entre os membros da equipe.

1.2 Escopo

O **LibShow** permitirá que bibliotecários cadastrem e mantenham registros de livros e usuários, que alunos e professores realizem consultas e reservas no acervo, e que administradores acompanhem relatórios de uso e estatísticas. O sistema será acessível via navegador.

2. Visão Geral da Arquitetura

2.1 Estilo Arquitetural

O sistema seguirá uma Arquitetura em Camadas combinada ao padrão MVC (Model-View-Controller). A API será exposta como serviços REST.

2.2 Camadas

- **Apresentação (Front-end):** Interface do usuário (ReactJS).
- **Aplicação/Serviço:** Controladores REST e lógica de negócio.
- **Domínio:** Entidades como Usuário, Livro, Empréstimo, Reserva.
- **Persistência:** Repositórios com JPA/Hibernate.
- **Infraestrutura:** Banco de dados, autenticação e configurações gerais.

3. Componentes Principais

Componente	Descrição
UsuarioController	Endpoints REST para cadastro e autenticação de usuários.

LivroController	Endpoints para gerenciamento do acervo (cadastro, atualização, consulta).
EmprestimoController	Endpoints para empréstimos e devoluções.
ReservaController	Endpoints para gerenciamento de reservas.
UsuarioService	Lógica de autenticação e perfis de acesso.
LivroService	Regras de negócio para cadastro e atualização de livros.
EmprestimoService	Controle de empréstimos, devoluções e atrasos.
ReservaService	Controle de reservas e disponibilidade de exemplares.
RelatorioService	Geração de relatórios para administradores.
Repositórios (DAO)	Persistência de dados de usuários, livros, empréstimos e reservas.

4. Tecnologias Utilizadas

Camada	Tecnologias
Front-end	ReactJS
Back-end	Spring Boot
Banco de Dados	PostgreSQL ou MySQL
ORM	Hibernate / JPA
Autenticação	JWT (JSON Web Token)
Versionamento	Git + GitHub
Deploy	Docker (opcional)

Diagrama de Componentes

(FAZER)

5. Padrões e Convenções

- Uso de DTOs para transporte de dados na API.
- Separação entre Service e Repository.
- Organização em pacotes por camadas.
- Tratamento de erros centralizado com @ControllerAdvice.
- Padrão REST para endpoints.

6. Requisitos Não Funcionais

Requisito	Descrição
Usabilidade	Interface intuitiva para bibliotecários, alunos e administradores.
Desempenho	Suporte para até 10 usuários simultâneos.
Segurança	Autenticação com login/senha, perfis de acesso e criptografia de dados sensíveis.
Compatibilidade	Suporte a navegadores modernos.
Escalabilidade	Arquitetura preparada para expansão de funcionalidades.
Manutenibilidade	Código modular, reutilizável e de fácil manutenção.

7. Riscos Arquiteturais

- Curva de aprendizado da equipe com Spring Boot e JPA.
- Integração entre versões Web e Mobile.
- Desafios na implementação de testes automatizados.
- Dependência de conexão de rede para acesso ao sistema.

8. Decisões Arquiteturais

Decisão	Justificativa
Uso de Spring Boot	Framework consolidado, rápido para configurar e com ampla documentação.
Banco PostgreSQL ou MySQL	Robusto, gratuito e bem adaptado a aplicações acadêmicas.
Arquitetura em Camadas	Facilita a separação de responsabilidades e a evolução do sistema.
JWT para autenticação	Padrão amplamente utilizado e seguro para aplicações Web/Mobile.

9. Diagrama de Domínio