Architecture Notebook

Sistema de Gestão de Feiras

Higor Roger de Freitas Santos - 221006440
Victor Eneias Oliveira - 221038364

Engenharia de Software CIC0105 Turma 01 2025.1

29 de junho de 2025

Conteúdo

1 Visão Geral da Arquitetura

O Sistema de Gestão de Feiras implementa uma arquitetura de três camadas (3-tier) baseada no padrão cliente-servidor, com separação clara entre interface de usuário, lógica de aplicação e persistência de dados. A comunicação entre as camadas ocorre através de uma API REST, permitindo independência tecnológica e facilidade de manutenção.

1.1 Elementos Principais

O sistema é composto por três elementos fundamentais:

- **Frontend (Cliente)**: Interface web desenvolvida em React.js que executa no navegador do usuário
- **Backend (Servidor)**: API REST desenvolvida em FastAPI que processa requisições e implementa regras de negócio
- **Banco de Dados**: SQLite para persistência de dados com estrutura relacional

1.2 Comunicação e Fluxo de Dados

A comunicação entre os elementos segue o protocolo HTTP/HTTPS com troca de dados em formato JSON. O frontend realiza requisições para endpoints específicos do backend, que processa as operações e retorna respostas estruturadas. A autenticação é mantida através de tokens JWT armazenados no navegador.

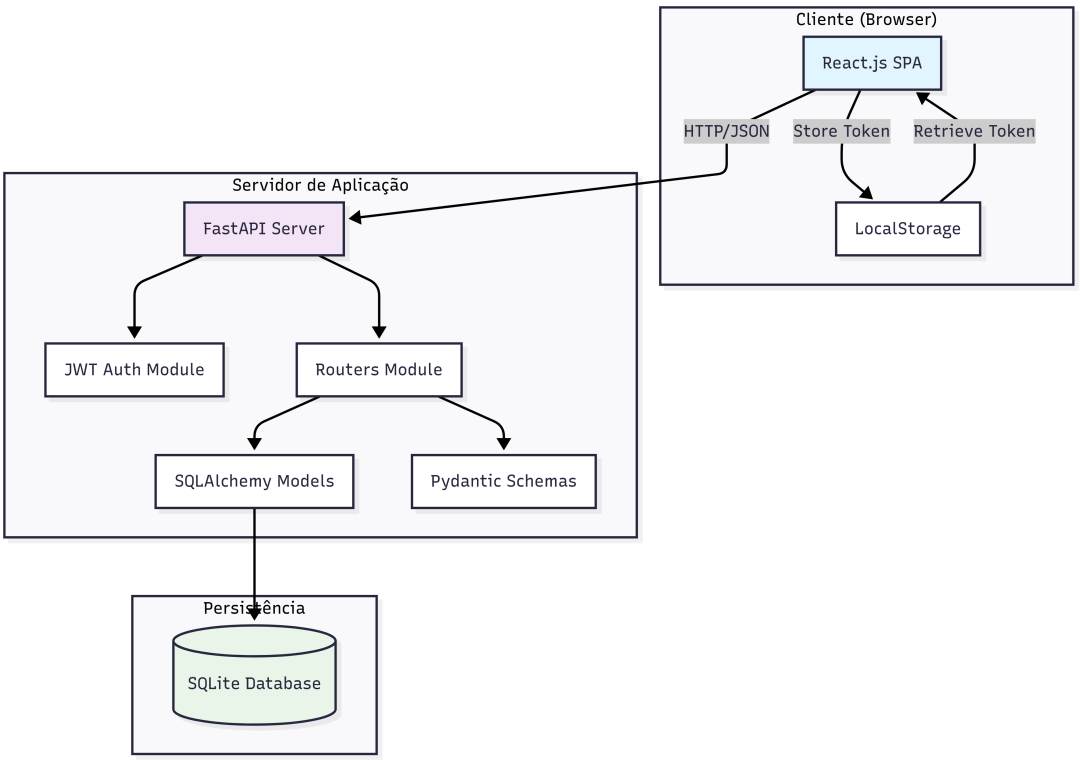

Figura 1: Arquitetura Geral do Sistema

2 Elementos e Relacionamentos

2.1 Camada de Apresentação (Frontend)

O frontend é uma Single Page Application (SPA) desenvolvida em React.js que executa no navegador do usuário. Seus principais componentes são:

- **Componente Principal (App)**: Gerencia o estado de autenticação e controla a navegação
- **Componente de Login**: Responsável pela autenticação e registro de usuários
- **Componentes de Gestão**: Interfaces para CRUD de feiras, expositores, produtos e ingressos
- **Módulo API**: Centraliza a comunicação com o backend e gerencia tokens JWT

2.2 Camada de Aplicação (Backend)

O backend implementa uma API REST usando FastAPI, organizando a lógica em módulos especializados:

- **Routers**: Definem endpoints HTTP e processam requisições (`usuarios.py`, `feiras.py`, etc.)
- **Modelos**: Representam entidades do banco de dados usando SQLAlchemy (`models.py`)
- **Schemas**: Validam dados de entrada e saída usando Pydantic (`schemas.py`)
- **Autenticação**: Gerencia tokens JWT e autorização (`auth.py`)
- **CRUD**: Implementa operações de banco de dados (`crud.py`)

2.3 Camada de Persistência

O banco de dados SQLite armazena informações em cinco tabelas principais:

- **Usuários**: Dados de autenticação e identificação
- **Feiras**: Eventos principais do sistema
- **Expositores**: Participantes vinculados a feiras específicas
- **Produtos**: Catálogo de itens por expositor
- **Ingressos**: Tickets de acesso às feiras

3 Tecnologias e Ferramentas

3.1 FastAPI (Backend)

Descrição: Framework web Python para desenvolvimento de APIs REST com foco em performance e facilidade de uso.

Por que foi escolhido:

- **Documentação Automática**: Gera automaticamente documentação OpenAPI/Swagger sem configuração adicional
- **Validação Automática**: Integração nativa com Pydantic para validação de dados
- **Performance**: Oferece alta performance comparável a frameworks Node.js
- **Tipagem**: Suporte nativo para type hints Python, melhorando qualidade do código

Limitações:

- Ecossistema menor comparado ao Django
- Requer conhecimento de Python assíncrono para recursos avançados

3.2 React.js (Frontend)

Descrição: Biblioteca JavaScript para construção de interfaces de usuário baseada em componentes.

Por que foi escolhido:

- **Componentização**: Facilita reutilização e manutenção de código
- **Ecossistema Maduro**: Ampla comunidade e disponibilidade de recursos
- **Virtual DOM**: Otimiza atualizações de interface
- **Curva de Aprendizado**: Relativamente simples para desenvolvedores JavaScript

Limitações:

- Requer conhecimento de JavaScript moderno (ES6+)
- Pode ser excessivo para aplicações muito simples

3.3 SQLite (Banco de Dados)

Descrição: Sistema de banco de dados relacional embarcado, sem necessidade de servidor dedicado.

Por que foi escolhido:

- **Simplicidade**: Zero configuração para desenvolvimento
- **Portabilidade**: Arquivo único facilita distribuição
- **Padrão SQL**: Compatível com SQL padrão

- **Adequação ao Escopo**: Suficiente para demonstração acadêmica

Limitações:

- Limitações de concorrência para múltiplos escritores
- Não adequado para aplicações de alta escala
- Funcionalidades avançadas limitadas comparado a PostgreSQL/MySQL

3.4 JWT (Autenticação)

Descrição: JSON Web Tokens para autenticação stateless entre cliente e servidor.

Por que foi escolhido:

- **Stateless**: Elimina necessidade de armazenamento de sessão no servidor
- **Portabilidade**: Funciona bem em arquiteturas distribuídas
- **Padrão da Indústria**: Amplamente adotado em APIs REST
- **Simplicidade**: Implementação direta com bibliotecas existentes

Limitações:

- Tokens não podem ser revogados facilmente
- Tamanho maior que cookies de sessão tradicionais
- Requer cuidado com tempo de expiração

4 Diagramas de Fluxo

4.1 Fluxo de Autenticação

O processo de autenticação segue o padrão JWT, onde o usuário fornece credenciais, o sistema valida e retorna um token que será usado nas requisições subsequentes.

Etapas do processo:

1. Usuário insere email e senha no frontend
2. Frontend envia requisição POST para `/usuarios/login`
3. Backend valida credenciais no banco de dados
4. Se válidas, backend gera token JWT com expiração de 60 minutos
5. Token é retornado ao frontend e armazenado no LocalStorage
6. Frontend inclui token em todas as requisições subsequentes

4.2 Fluxo de Criação de Feira

Este fluxo demonstra como funciona a criação de recursos com autenticação e autorização.

Etapas do processo:

1. Usuário preenche formulário de criação de feira
2. Frontend envia requisição POST para `/feiras/` incluindo token JWT
3. Backend verifica validade do token e extrai ID do usuário
4. Dados são validados usando schemas Pydantic
5. Nova feira é criada no banco com `id_criador` do usuário autenticado
6. Dados da feira criada são retornados ao frontend
7. Interface é atualizada com a nova feira

4.3 Controle de Autorização

O sistema implementa autorização baseada em propriedade, onde apenas o criador de um recurso pode modificá-lo.

Etapas do processo:

1. Usuário solicita edição de uma feira específica
2. Frontend envia requisição PUT incluindo token JWT
3. Backend extrai ID do usuário do token
4. Sistema busca a feira no banco de dados
5. Compara `id_criador` da feira com ID do usuário autenticado
6. Se diferentes, retorna erro 403 (Forbidden)
7. Se iguais, permite a operação de edição

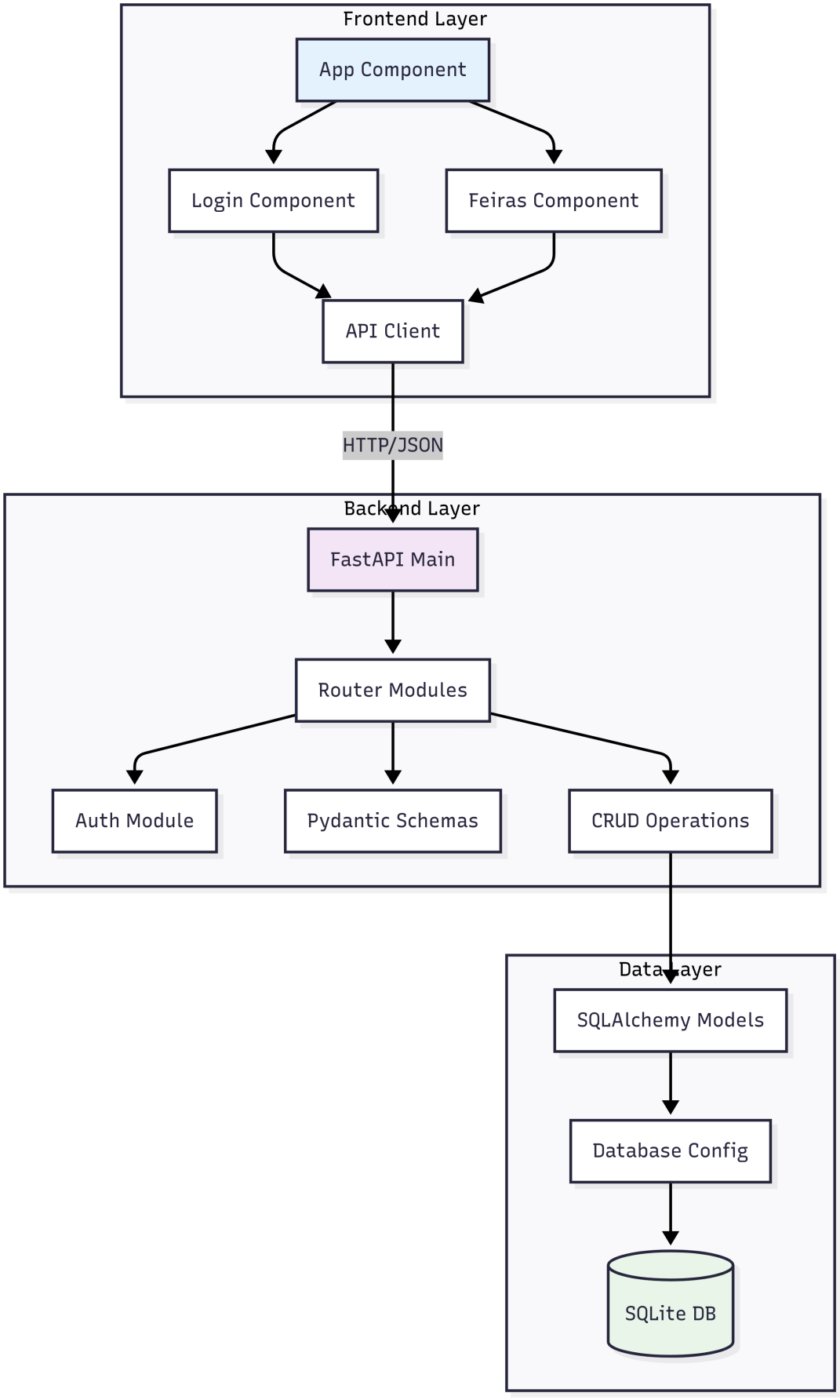

Figura 2: Relacionamentos entre Componentes

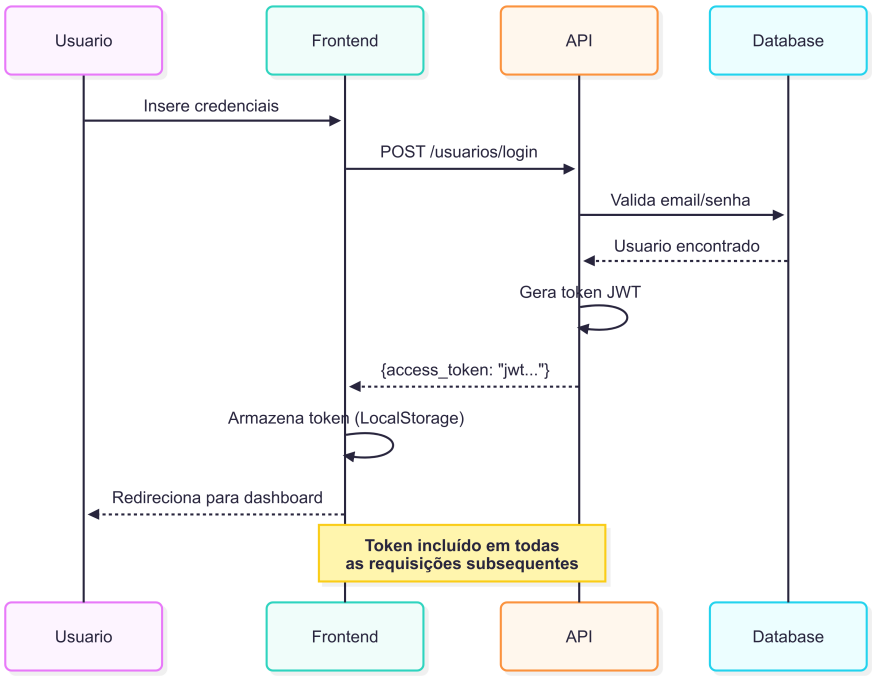

Figura 3: Sequência de Autenticação JWT

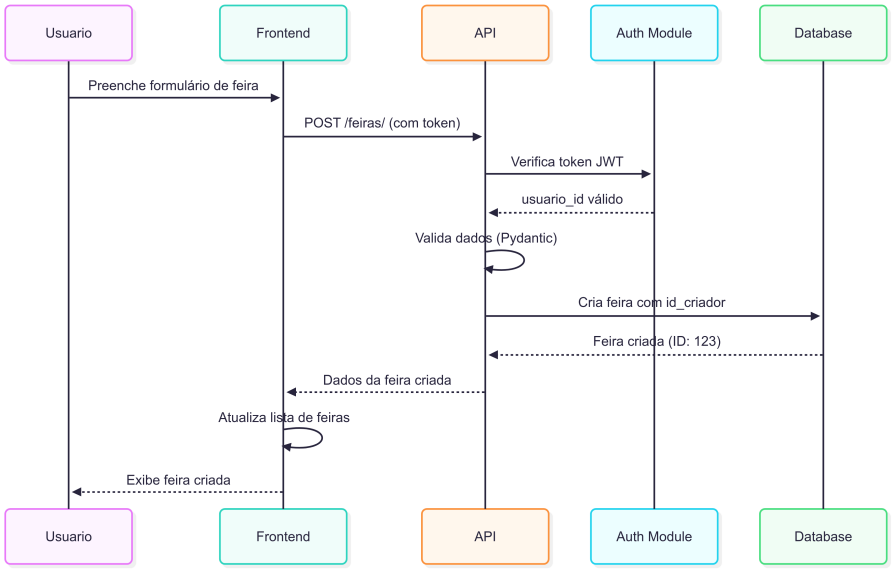

Figura 4: Sequência de Criação de Feira

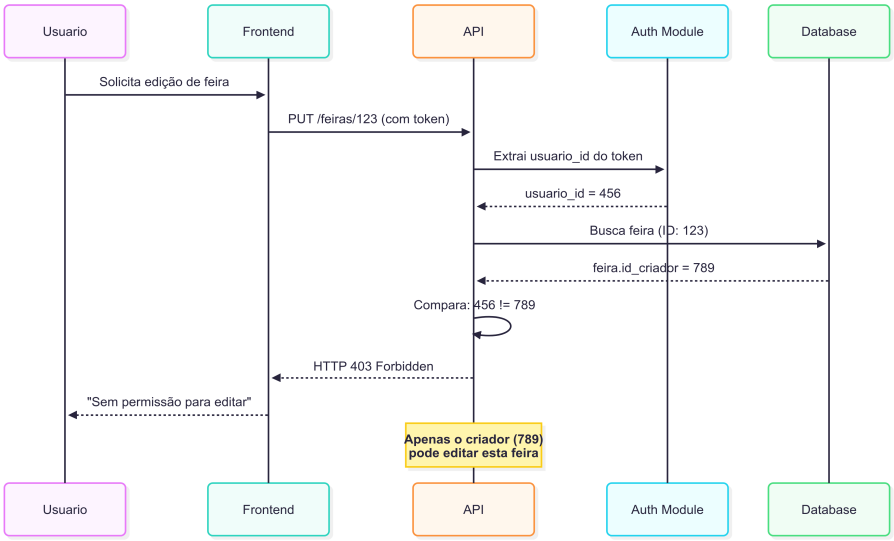

Figura 5: Sequência de Autorização por Propriedade