HUB Logística

Sistema de Gerenciamento Logístico para Controle e Rastreamento de Entregas

Aluno: Arthur Henrique Tscha Vieira

Curso: Engenharia de Software

Data de Entrega: Dezembro de 2025

Documentação Completa do Projeto

Centro Universitário Católica de Santa Catarina - Joinville
2025

Resumo

Este documento apresenta a documentação técnica completa do projeto HUB Logística, um sistema integrado de gerenciamento logístico composto por uma aplicação web frontend e uma API backend. O sistema foi projetado para facilitar o controle e rastreamento de entregas em tempo real, oferecendo uma interface intuitiva para gestores e operadores logísticos, além de uma API robusta para integração com diferentes transportadoras. A documentação aborda o contexto do projeto, sua justificativa, objetivos, especificações técnicas, considerações de design, stack tecnológica adotada e aspectos de segurança, tanto para o frontend quanto para o backend. O documento também apresenta os próximos passos para o desenvolvimento contínuo do projeto e as referências utilizadas em sua concepção.

Sumário

1 Introdução

1.1 Contexto

O setor de logística enfrenta desafios significativos relacionados à gestão eficiente de entregas, rastreamento em tempo real e otimização de rotas. As empresas de logística necessitam de ferramentas tecnológicas que permitam monitorar suas operações de forma centralizada, oferecendo visibilidade completa do processo aos gestores e clientes. Neste contexto, o HUB Logística surge como uma solução integrada composta por uma interface web frontend e uma API backend, unificando diferentes ferramentas e sistemas utilizados no processo logístico, proporcionando uma visão integrada das operações.

1.2 Justificativa

A fragmentação de sistemas logísticos é um problema recorrente que impacta negativamente a produtividade das equipes e a qualidade do serviço oferecido. Muitas empresas utilizam diferentes ferramentas para cada etapa do processo logístico, resultando em:

- Dificuldade na centralização de informações
- Perda de tempo com troca constante entre sistemas
- Inconsistência de dados entre plataformas
- Complexidade no treinamento de novos colaboradores
- Desafios na geração de relatórios consolidados

O desenvolvimento do HUB Logística justifica-se pela necessidade de unificar essas diferentes interfaces em uma única plataforma, melhorando a produtividade operacional e proporcionando maior visibilidade sobre todo o processo logístico.

1.3 Objetivos

1.3.1 Objetivo Principal

Desenvolver uma plataforma integrada que centralize informações sobre rastreamento e fretes de mercadorias que estão em múltiplos arquivos Excel e valide seus valores, proporcionando uma experiência unificada para gestão e monitoramento das operações logísticas relacionadas.

1.3.2 Objetivos Secundários

- Integrar ferramentas de diferentes etapas do processo logístico em uma única interface
- Implementar funcionalidades de rastreamento em tempo real de entregas
- Criar visualizações intuitivas para análise de indicadores logísticos
- Disponibilizar interface responsiva para acesso via diferentes dispositivos
- Desenvolver uma integração com diferentes transportadoras
- Implementar mecanismos de cache e resiliência para garantir alta disponibilidade

2 Descrição do Projeto

2.1 Tema do Projeto

O HUB Logística é um sistema integrado composto por uma aplicação web frontend e uma API backend, desenvolvido para centralizar e integrar diferentes ferramentas utilizadas no gerenciamento logístico. A plataforma atua como um hub central, permitindo o acesso a diversos sistemas a partir de uma única interface, simplificando o trabalho dos operadores logísticos. O frontend oferece uma experiência de usuário fluida, com design intuitivo e responsivo, enquanto o backend fornece uma API robusta para integração com diferentes transportadoras e serviços logísticos.

2.2 Problemas a Resolver

O HUB Logística busca resolver os seguintes problemas:

1. **Fragmentação de sistemas:** Unificação de diferentes ferramentas em uma única plataforma
2. **Ineficiência operacional:** Redução do tempo gasto na alternância entre sistemas diferentes
3. **Dificuldade de monitoramento:** Implementação de funcionalidades de rastreamento em tempo real
4. **Complexidade de gestão de usuários:** Criação de um sistema de gerenciamento de acessos centralizado
5. **Inconsistência de dados:** Integração entre sistemas para garantir a consistência das informações
6. **Falta de visibilidade completa:** Desenvolvimento de dashboards para visualização consolidada de dados
7. **Integração com transportadoras:** Desenvolvimento de adaptadores para diferentes APIs de transportadoras
8. **Disponibilidade de serviços:** Implementação de mecanismos de cache e retry para lidar com instabilidades

2.3 Limitações

O projeto apresenta as seguintes limitações em seu escopo:

- **Integração com sistemas legados:** O sistema atuará principalmente como um hub de acesso, não substituindo completamente sistemas específicos já existentes
- **Processamento offline:** O sistema depende de conexão com internet para funcionamento pleno
- **Operações financeiras:** O sistema não implementará funcionalidades de pagamento ou gestão financeira complexa
- **Integração com hardware:** O sistema não abordará integração direta com dispositivos IoT ou hardware específico para rastreamento
- **Otimização automatizada de rotas:** O sistema não implementará algoritmos avançados de otimização de rotas, apenas sua visualização
- **Integração com transportadoras:** Inicialmente, apenas as transportadoras Ouro Negro e Alfa serão integradas

3 Especificação Técnica

3.1 Arquitetura do Sistema

3.1.1 Visão Geral

O HUB Logística é composto por dois componentes principais:

- **Frontend:** Aplicação web responsiva desenvolvida com HTML5, CSS3 e JavaScript puro
- **Backend:** API RESTful desenvolvida em Node.js com Express

3.2 Requisitos de Software

3.2.1 Lista de Requisitos

Requisitos Funcionais (RF)

RF1: O sistema deve permitir autenticação de usuários via email e senha

RF2: O sistema deve disponibilizar um dashboard principal com acesso às diferentes ferramentas

RF3: O sistema deve permitir o rastreamento de entregas em tempo real

RF4: O sistema deve fornecer visualização de rotas de entrega

RF5: O sistema deve permitir a geração de relatórios sobre operações logísticas

RF6: O sistema deve possibilitar o gerenciamento de usuários por administradores

RF7: O sistema deve implementar controle de sessões ativas

RF8: O sistema deve fornecer acesso às diferentes ferramentas logísticas via uma única interface

RF9: O sistema deve permitir a personalização do perfil de usuário

RF10: O sistema deve permitir cadastro de novos usuários

RF11: O sistema deve permitir alteração de senha de usuários

RF12: O sistema deve permitir upload e visualização de fotos de perfil

RF13: O sistema deve integrar-se com APIs de transportadoras (Ouro Negro e Alfa)

RF14: O sistema deve consultar e exibir informações de notas fiscais

RF15: O sistema deve permitir filtrar entregas por período, transportadora e status

Requisitos Não-Funcionais (RNF)

RNF1: O sistema deve ser responsivo, adaptando-se a diferentes tamanhos de tela

RNF2: O sistema deve garantir a segurança dos dados por meio de autenticação robusta

RNF3: O sistema deve ter tempo de resposta inferior a 2 segundos para operações comuns

RNF4: O sistema deve ser intuitivo, facilitando a navegação entre as diferentes ferramentas

RNF5: O sistema deve seguir padrões de acessibilidade web

RNF6: O sistema deve ser compatível com os principais navegadores (Chrome, Firefox, Safari, Edge)

RNF7: O sistema deve implementar boas práticas de UX/UI para melhor experiência do usuário

RNF8: O sistema deve utilizar design consistente em todas as interfaces

RNF9: O sistema deve implementar feedback visual para ações do usuário

RNF10: O sistema deve suportar internacionalização (inicialmente português)

RNF11: O sistema deve garantir disponibilidade mínima de 99% do tempo

RNF12: O sistema deve implementar criptografia de senhas usando bcrypt

RNF13: O sistema deve seguir princípios REST na implementação da API

RNF14: O sistema deve implementar CORS para controle de acesso

RNF15: O sistema deve validar todos os dados de entrada para prevenir injeções e ataques

3.2.2 Representação dos Requisitos

Diagrama de Casos de Uso

O diagrama de casos de uso a seguir representa os principais requisitos funcionais do sistema HUB Logística:

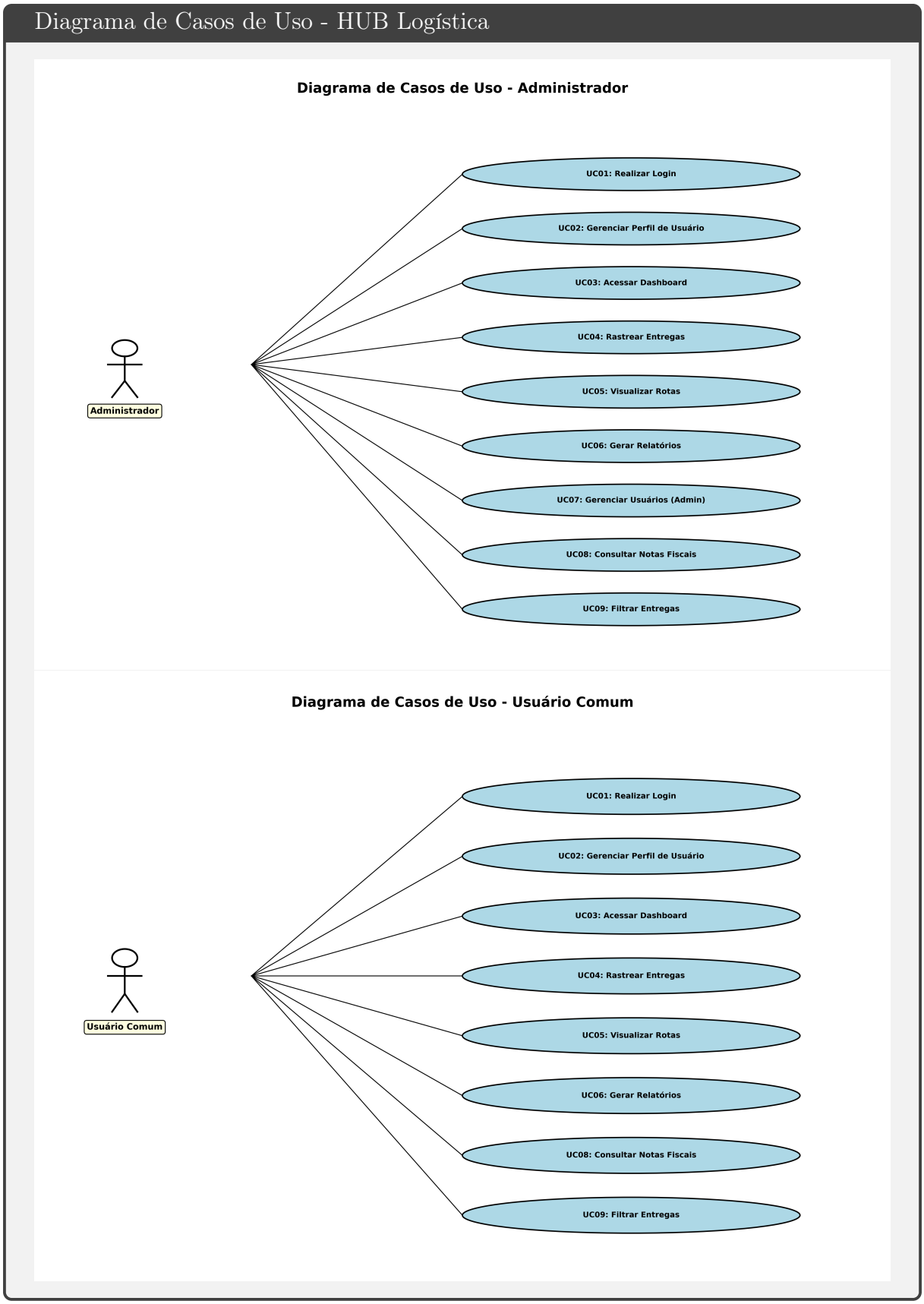

Figura 3.1: Diagrama de Casos de Uso do Sistema HUB Logística

3.3 Stack Tecnológica

3.3.1 Linguagens de Programação

- **JavaScript:** Linguagem principal utilizada tanto no frontend quanto no backend, escolhida pela sua versatilidade e amplo ecossistema de bibliotecas
- **HTML5:** Linguagem de marcação para estruturação do conteúdo das páginas web
- **CSS3:** Linguagem de estilo para apresentação visual e design responsivo

3.3.2 Frameworks e Bibliotecas

Frontend

- **Express.js:** Framework web para servir arquivos estáticos e roteamento básico
- **Vanilla JavaScript:** JavaScript puro para interatividade, evitando dependências desnecessárias
- **CSS Grid e Flexbox:** Para layout responsivo e moderno
- **Fetch API:** Para comunicação com o backend via requisições HTTP

Backend

- **Node.js:** Ambiente de execução JavaScript server-side
- **Express.js:** Framework web para criação da API RESTful
- **Sequelize:** ORM para abstração e manipulação do banco de dados
- **JWT (jsonwebtoken):** Biblioteca para autenticação baseada em tokens
- **Bcrypt:** Biblioteca para criptografia de senhas
- **Axios:** Cliente HTTP para requisições às APIs externas
- **CORS:** Middleware para controle de acesso cross-origin
- **Dotenv:** Gerenciamento de variáveis de ambiente
- **Nodemon:** Ferramenta para reinicialização automática durante desenvolvimento

3.3.3 Ferramentas de Desenvolvimento e Gestão de Projeto

- **Git:** Sistema de controle de versão distribuído
- **GitHub:** Plataforma para hospedagem de repositórios e colaboração
- **npm:** Gerenciador de pacotes para Node.js
- **PostgreSQL:** Sistema de gerenciamento de banco de dados relacional
- **Postman:** Ferramenta para teste de APIs
- **Visual Studio Code:** Editor de código com extensões para JavaScript/Node.js

3.3.4 Banco de Dados

- **PostgreSQL:** Escolhido pela robustez, confiabilidade e suporte a recursos avançados como JSON, índices complexos e transações ACID

3.3.5 APIs e Serviços Externos

- **API Ouro Negro:** Integração para rastreamento de entregas da transportadora Ouro Negro
- **API Alfa:** Integração para informações logísticas da transportadora Alfa

3.4 Considerações de Design

3.4.1 Visão Inicial da Arquitetura

O HUB Logística foi projetado seguindo uma arquitetura de separação clara entre frontend e backend, onde:

- **Frontend:** Aplicação web estática servida por Express.js, responsável pela interface do usuário e experiência de navegação
- **Backend:** API RESTful independente desenvolvida em Node.js, responsável pela lógica de negócios, autenticação e integração com transportadoras
- **Banco de Dados:** PostgreSQL para persistência de dados de usuários e cache de informações
- **Integrações Externas:** APIs das transportadoras Ouro Negro e Alfa para dados de rastreamento

Componentes Principais:

1. **Servidor Web Frontend:** Express.js servindo arquivos estáticos (HTML, CSS, JS)
2. **API Backend:** Servidor Node.js/Express com endpoints RESTful
3. **Sistema de Autenticação:** JWT para controle de sessões
4. **Adaptadores de Integração:** Módulos para comunicação com APIs externas
5. **Sistema de Cache:** Otimização de consultas frequentes
6. **Gerenciador de Logs:** Monitoramento e auditoria do sistema

3.4.2 Padrões de Arquitetura

O projeto adota os seguintes padrões de arquitetura:

1. **Client-Server:** Separação clara entre frontend (cliente) e backend (servidor)
2. **REST (Backend):** Arquitetura de API RESTful para comunicação
3. **MVC (Backend):** Separação entre Modelo, Visão e Controlador
4. **Repository:** Padrão para acesso a dados e abstração do banco
5. **Service Layer:** Padrão para encapsular lógica de negócios
6. **Adapter:** Padrão para integração com APIs externas de transportadoras
7. **Middleware:** Padrão para processamento de requisições (autenticação, CORS, etc.)

3.4.3 Modelos C4

Nível 1: Contexto

O sistema HUB Logística é utilizado por diferentes tipos de usuários através de uma interface web, que se comunica com uma API backend para acessar serviços de transportadoras e gerenciar dados logísticos.

Diagrama de Contexto - Nível 1

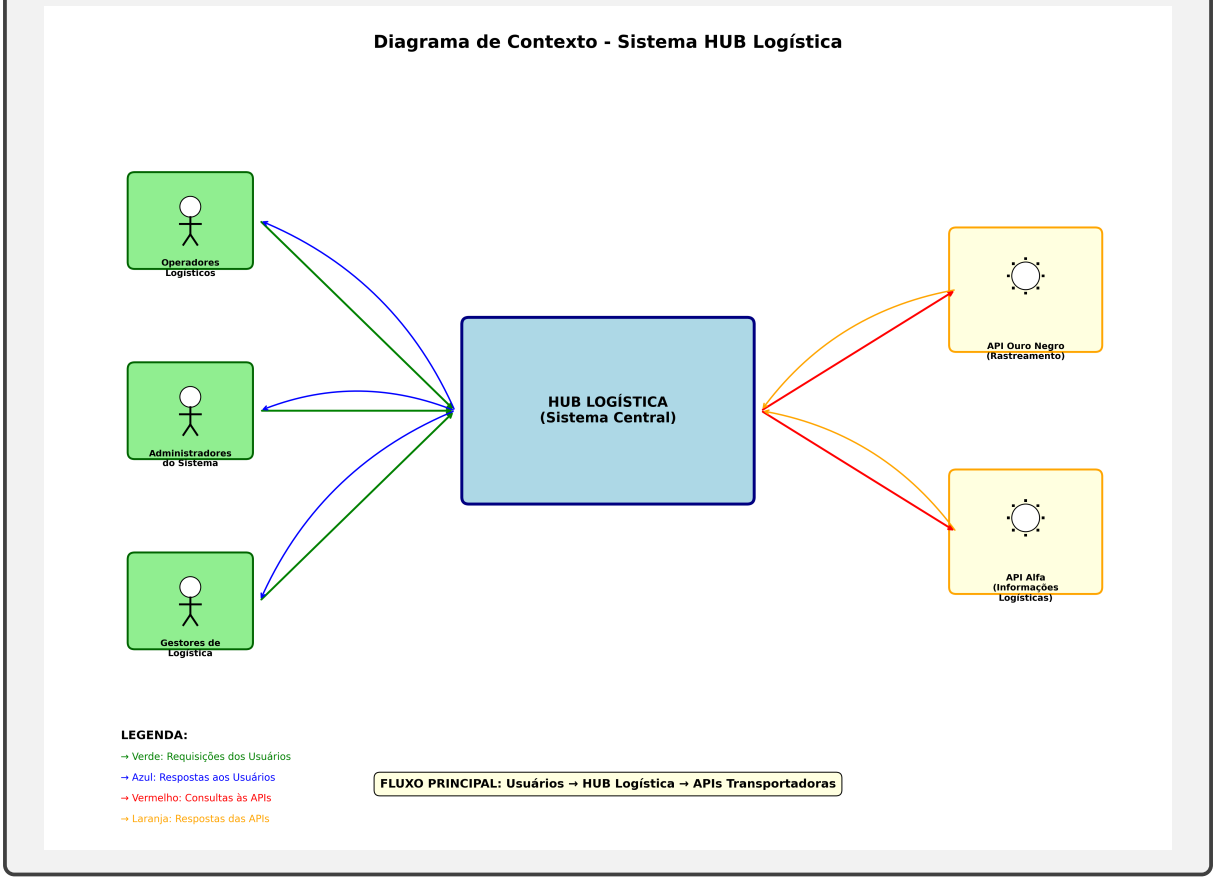

Figura 3.2: Diagrama de Contexto do Sistema

Nível 2: Contêineres

O sistema é composto pelos seguintes contêineres:

1. **Aplicação Web Frontend:** Servidor Express.js (porta 3060) servindo arquivos estáticos
2. **API Backend:** Servidor Node.js/Express (porta 4010) com endpoints RESTful
3. **Banco de Dados PostgreSQL:** Armazenamento de usuários e cache de dados
4. **APIs Externas:** Serviços de transportadoras Ouro Negro e Alfa

Diagrama de Contêineres - Nível 2

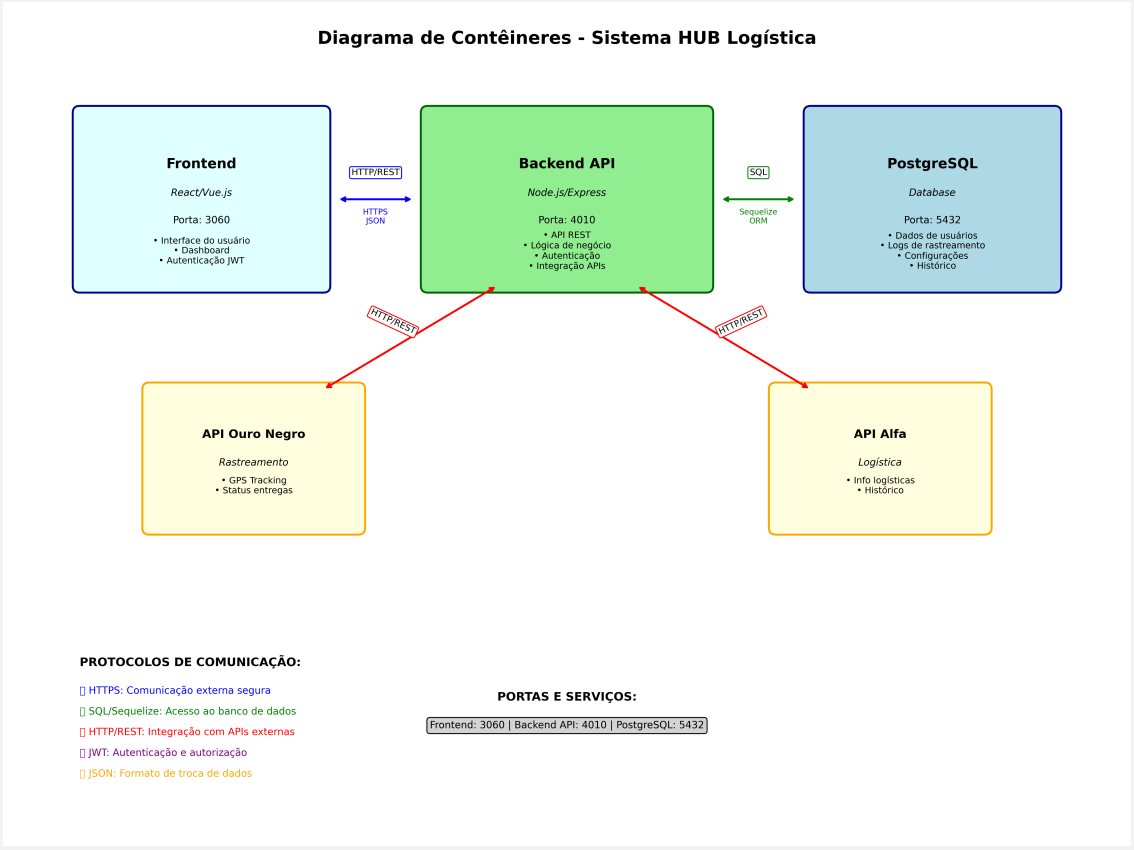

Figura 3.3: Diagrama de Contêineres do Sistema

Nível 3: Componentes

Os principais componentes do sistema incluem:

Frontend Components:

1. **Sistema de Autenticação:** Login e controle de sessão
2. **Dashboard Principal:** Interface central de navegação
3. **Módulo de Rastreamento:** Visualização de entregas
4. **Painel Administrativo:** Gestão de usuários

Backend Components:

1. **Controllers:** Gerenciamento de rotas e requisições
2. **Services:** Lógica de negócios e integração
3. **Models:** Representação de dados (Sequelize)
4. **Middleware:** Autenticação, CORS, validação
5. **Adaptadores:** Integração com APIs externas

3.5 Considerações de Segurança

O sistema implementa as seguintes medidas de segurança:

1. **Autenticação por JWT:** Tokens seguros e com expiração
2. **Hashing de senhas:** Utilização do bcrypt com salt
3. **Proteção contra injeção SQL:** Uso de queries parametrizadas
4. **Validação de entradas:** Validação rigorosa de dados
5. **Configuração de CORS:** Controle de acesso cross-origin
6. **Uso de HTTPS:** Conexões seguras
7. **Proteção de segredos:** Variáveis de ambiente
8. **Controle de acesso:** Diferentes níveis de permissão
9. **Tratamento de erros:** Feedback seguro
10. **Rate limiting:** Prevenção de ataques

4 Próximos Passos

O desenvolvimento do HUB Logística seguirá as seguintes etapas:

4.1 Portfólio I (Curto Prazo)

1. **Frontend:**
   (a) Implementação da estrutura básica
   (b) Desenvolvimento do sistema de autenticação
   (c) Criação do dashboard principal
   (d) Implementação do painel administrativo
2. **Backend:**
   (a) Estruturação inicial do projeto
   (b) Implementação dos modelos de dados
   (c) Desenvolvimento do sistema de autenticação
   (d) Criação da estrutura básica de rotas

4.2 Portfólio II (Médio Prazo)

1. **Frontend:**
   (a) Desenvolvimento do módulo de rastreamento
   (b) Implementação da visualização de rotas
   (c) Desenvolvimento de relatórios
   (d) Refinamentos na interface
2. **Backend:**
   (a) Implementação dos serviços de integração
   (b) Desenvolvimento do sistema de cache
   (c) Implementação de mecanismos de retry
   (d) Testes e documentação

4.3 Cronograma Estimado

Etapa	**Atividades**	**Prazo**
Análise	Levantamento de requisitos	2 semanas
Projeto	Design da arquitetura	3 semanas
Desenvolvimento I	Estrutura básica	4 semanas
Desenvolvimento II	Integrações	5 semanas
Testes	Testes e ajustes	3 semanas
Documentação	Documentação final	2 semanas
Implantação	Deploy e configuração	1 semana

Tabela 4.1: Cronograma estimado de desenvolvimento

5 Referências

Referências Bibliográficas

[1] Node.js. Disponível em: `https://nodejs.org/`. Acesso em: Maio 2025.

[2] Express.js: Framework web rápido, minimalista e flexível para Node.js. Disponível em: `https://expressjs.com/`. Acesso em: Maio 2025.

[3] Bootstrap: Framework front-end para desenvolvimento web mais rápido e responsivo. Disponível em: `https://getbootstrap.com/`. Acesso em: Maio 2025.

[4] Font Awesome: O conjunto de ícones mais popular da web. Disponível em: `https://fontawesome.com/`. Acesso em: Maio 2025.

[5] JWT: JSON Web Tokens. Disponível em: `https://jwt.io/`. Acesso em: Maio 2025.

[6] C4 Model: Um modelo para visualizar arquitetura de software. Disponível em: `https://c4model.com/`. Acesso em: Maio 2025.

[7] Sequelize ORM Documentation. Disponível em: `https://sequelize.org/master/`. Acesso em: Maio 2025.

[8] PostgreSQL Documentation. Disponível em: `https://www.postgresql.org/docs/`. Acesso em: Maio 2025.

[9] REST API Best Practices. Disponível em: `https://restfulapi.net/`. Acesso em: Maio 2025.

6 Apêndices

6.1 Estrutura do Projeto

6.1.1 Frontend (hub-logistica)

```
hub-logistica/
|-- public/                      # Arquivos estaticos
|   |-- assets/                  # Recursos multimidia
|   |   |-- icons/               # Icones do sistema
|   |   `-- images/              # Imagens e graficos
|   |-- html/                    # Paginas HTML
|   |   |-- administration.html  # Painel administrativo
|   |   |-- index.html           # Dashboard principal
|   |   `-- login.html           # Pagina de login
|   |-- javascripts/             # Scripts JavaScript
|   |   |-- admin-check.js       # Verificacao de admin
|   |   |-- admin.js             # Funcoes administrativas
|   |   |-- administration.js    # Logica do painel admin
|   |   |-- auth-check.js        # Verificacao de autenticacao
|   |   |-- dashboard.js         # Logica do dashboard
|   |   |-- index.js             # Script principal
|   |   |-- login-check.js       # Verificacao de login
|   |   |-- login.js             # Logica de autenticacao
|   |   `-- rastreamento.js      # Funcoes de rastreamento
|   `-- styles/                  # Arquivos CSS
|       |-- admin.css            # Estilos administrativos
|       |-- dashboard.css        # Estilos do dashboard
|       |-- index.css            # Estilos principais
|       |-- login.css            # Estilos de login
|       `-- rastreamento.css     # Estilos de rastreamento
|-- views/                       # Templates EJS
|   `-- rastreamento.ejs         # Template de rastreamento
|-- app.js                       # Servidor Express principal
|-- package.json                 # Dependencias e scripts
|-- package-lock.json            # Lock de dependencias
|-- diretrizes-doc.md            # Diretrizes de documentacao
|-- documentacao-hub-logistica.tex # Documentacao LaTeX
|-- .gitignore                   # Arquivos ignorados pelo Git
`-- README.md                    # Documentacao do projeto
```

Listing 6.1: Estrutura de arquivos do frontend

6.1.2 Backend (hub-logistica-backend)

```
hub-logistica-backend/
|-- config/                  # Configuracoes do sistema
|   |-- database.js          # Configuracao do banco de dados
|   `-- cors.js              # Configuracao de CORS
|-- controllers/             # Controladores da aplicacao
|   |-- userController.js    # Controlador de usuarios
|   |-- sessionController.js # Controlador de sessoes
|   |-- ouroNegroController.js # Controlador Ouro Negro
|   `-- alfaController.js    # Controlador Alfa
|-- models/                  # Modelos de dados Sequelize
|   |-- user.model.js        # Modelo de usuario
|   `-- index.js             # Configuracao dos modelos
|-- routes/                  # Rotas da API
|   |-- userRoutes.js        # Rotas de usuarios
|   |-- sessionRoutes.js     # Rotas de autenticacao
|   |-- ouroNegroRoutes.js   # Rotas Ouro Negro
|   `-- alfaRoutes.js        # Rotas Alfa
|-- services/                # Camada de servicos
|   |-- userService.js       # Servicos de usuario
|   |-- authService.js       # Servicos de autenticacao
|   |-- ouroNegroService.js  # Servicos Ouro Negro
|   `-- alfaService.js       # Servicos Alfa
|-- app.js                   # Arquivo principal da aplicacao
|-- .env                     # Variaveis de ambiente (nao versionado)
|-- .env.example             # Exemplo de variaveis de ambiente
|-- .gitignore               # Arquivos ignorados pelo Git
|-- package.json             # Dependencias e scripts
|-- package-lock.json        # Lock de dependencias
`-- README.md                # Documentacao do projeto
```

Listing 6.2: Estrutura de arquivos do backend

6.2 Exemplos de Código

6.2.1 Configuração do Servidor Frontend (app.js)

```
const express = require('express');
const path = require('path');
const app = express();
const PORT = 3060;

// Configuracao de arquivos estaticos
app.use(express.static(path.join(__dirname, 'public')));
app.set('view␣engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));

// Rotas principais
app.get('/', (req, res) => {
    res.sendFile(path.join(__dirname, 'public', 'html', 'login.html'));
});

app.get('/home', (req, res) => {
    res.sendFile(path.join(__dirname, 'public', 'html', 'index.html'));
});

```

```
app.get('/administration', (req, res) => {
    res.sendFile(path.join(__dirname, 'public', 'html', 'administration.
        html'));
});

app.get('/rastreamento/:codigo', (req, res) => {
    const codigo = req.params.codigo;
    res.render('rastreamento', { codigo });
});

// Inicializacao do servidor
app.listen(PORT, () => {
    console.log(`Servidor frontend rodando na porta ${PORT}`);
});
```

Listing 6.3: Servidor Express do Frontend

6.2.2 Autenticação Frontend (login.js)

```
class AuthManager {
    constructor() {
        this.apiUrl = 'http://localhost:4010/api';
    }

    async login(email, password) {
        try {
            const response = await fetch(`${this.apiUrl}/sessions`, {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify({ email, password })
            });

            if (!response.ok) {
                throw new Error('Credenciais invalidas');
            }

            const data = await response.json();
            localStorage.setItem('token', data.token);
            localStorage.setItem('user', JSON.stringify(data.user));

            return data;
        } catch (error) {
            throw error;
        }
    }

    logout() {
        localStorage.removeItem('token');
        localStorage.removeItem('user');
        window.location.href = '/';
    }

    isAuthenticated() {
        return localStorage.getItem('token') !== null;
    }
```

```
39
40      getToken() {
41          return localStorage.getItem('token');
42      }
43  }
```

Listing 6.4: Sistema de autenticação do frontend

6.2.3 Rota de Autenticação Backend

```
1  const express = require("express");
2  const router = express.Router();
3  const jwt = require("jsonwebtoken");
4  const bcrypt = require("bcrypt");
5  const User = require("../models/user.model");
6
7  router.post("/login", async (req, res) => {
8    try {
9      const { email, password } = req.body;
10     const user = await User.findOne({ where: { email } });
11
12     if (!user) {
13       return res.status(401).json({ message: "Usuario nao encontrado" })
             ;
14     }
15
16     const validPassword = await bcrypt.compare(password, user.password);
17     if (!validPassword) {
18       return res.status(401).json({ message: "Senha invalida" });
19     }
20
21     const token = jwt.sign(
22       { id: user.id, email: user.email },
23       process.env.JWT_SECRET,
24       { expiresIn: "1d" }
25     );
26
27     res.json({ token });
28   } catch (error) {
29     res.status(500).json({ message: "Erro interno do servidor" });
30   }
31 });
32
33 module.exports = router;
```

Listing 6.5: Exemplo de rota de autenticacao

6.2.4 Integração com APIs de Rastreamento

```
1  class TrackingService {
2      constructor() {
3          this.apiUrl = 'http://localhost:4010/api';
4      }
5
6      async trackOuroNegro(codigo) {
```

```
        try {
            const token = localStorage.getItem('token');
            const response = await fetch(`${this.apiUrl}/ouro-negro/
                track/${codigo}`, {
                headers: {
                    'Authorization': `Bearer ${token}`,
                    'Content-Type': 'application/json'
                }
            });

            if (!response.ok) {
                throw new Error('Erro ao rastrear pedido');
            }

            return await response.json();
        } catch (error) {
            throw error;
        }
    }

    async trackAlfa(codigo) {
        try {
            const token = localStorage.getItem('token');
            const response = await fetch(`${this.apiUrl}/alfa/track/${
                codigo}`, {
                headers: {
                    'Authorization': `Bearer ${token}`,
                    'Content-Type': 'application/json'
                }
            });

            if (!response.ok) {
                throw new Error('Erro ao rastrear pedido');
            }

            return await response.json();
        } catch (error) {
            throw error;
        }
    }

    async trackAll(codigo) {
        const results = await Promise.allSettled([
            this.trackOuroNegro(codigo),
            this.trackAlfa(codigo)
        ]);

        return results.filter(result => result.status === 'fulfilled')
                      .map(result => result.value);
    }
}
```

Listing 6.6: Serviço de rastreamento frontend

7 Conclusão

O Hub Logística representa uma solução inovadora para a centralização e otimização do rastreamento de entregas no setor logístico brasileiro. Através da integração com as APIs das transportadoras Ouro Negro e Alfa, o sistema oferece uma plataforma unificada que simplifica significativamente o processo de acompanhamento de pedidos.

7.1 Resultados Alcançados

O desenvolvimento do sistema resultou em:

- **Interface Unificada**: Dashboard centralizado para rastreamento de múltiplas transportadoras
- **Autenticação Segura**: Sistema robusto com JWT e criptografia bcrypt
- **Arquitetura Escalável**: Separação clara entre frontend e backend, facilitando manutenção e expansão
- **Integração Eficiente**: Comunicação otimizada com APIs externas das transportadoras
- **Experiência do Usuário**: Interface responsiva e intuitiva para diferentes tipos de usuários

7.2 Impacto e Benefícios

A implementação do Hub Logística proporciona:

- **Redução de Tempo**: Eliminação da necessidade de consultar múltiplos sistemas
- **Maior Visibilidade**: Acompanhamento centralizado de todas as entregas
- **Melhoria Operacional**: Otimização dos processos de monitoramento logístico
- **Escalabilidade**: Base sólida para integração com novas transportadoras

8 Avaliações de Professores

8.1 Considerações Professor/a 1

Considerações do(a) Professor(a)

Assinatura: _____

8.2 Considerações Professor/a 2

Considerações do(a) Professor(a)

Assinatura: _____

8.3 Considerações Professor/a 3

Considerações do(a) Professor(a)

Assinatura: _____