Documentação do Projeto Web

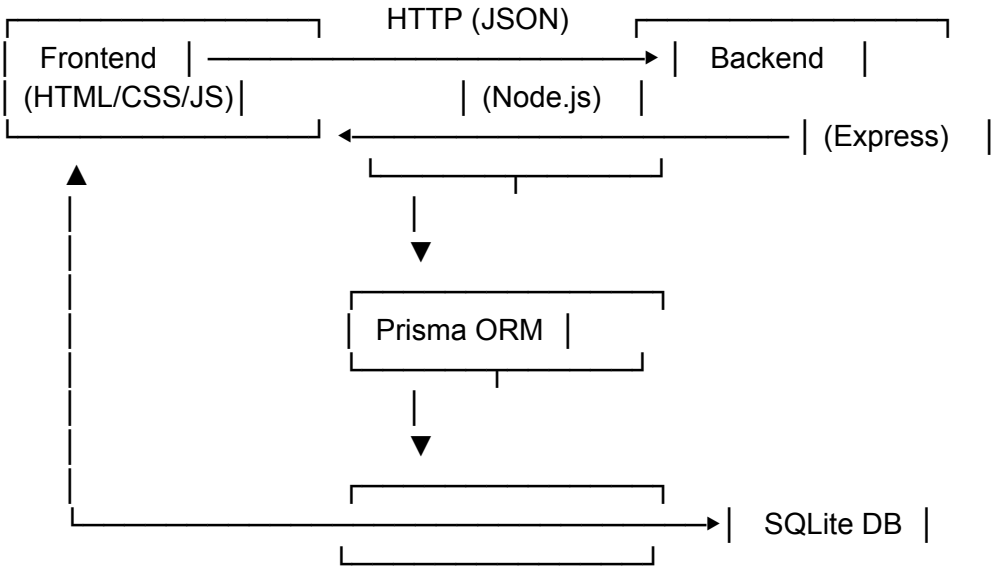
1. Visão Geral

Este projeto é uma aplicação web **full stack**, composta por **frontend estático em JavaScript (ES Modules)** e **backend em Node.js com Express**, utilizando **Prisma ORM** e **SQLite** como banco de dados.
O sistema oferece:

- Autenticação de usuários
- Gerenciamento de perfil
- Upload e listagem de arquivos
- Registro de logs
- Interface web com tema claro/escuro

---

2. Arquitetura Geral



---

3. Tecnologias Utilizadas

Frontend

- HTML5
- CSS3
- JavaScript ES Modules
- Fetch API
- Servidor estático (`npx serve`)

Backend

- Node.js
- Express
- Prisma ORM
- SQLite
- JWT para autenticação
- Multer para upload de arquivos

---

4. Estrutura de Pastas

Backend

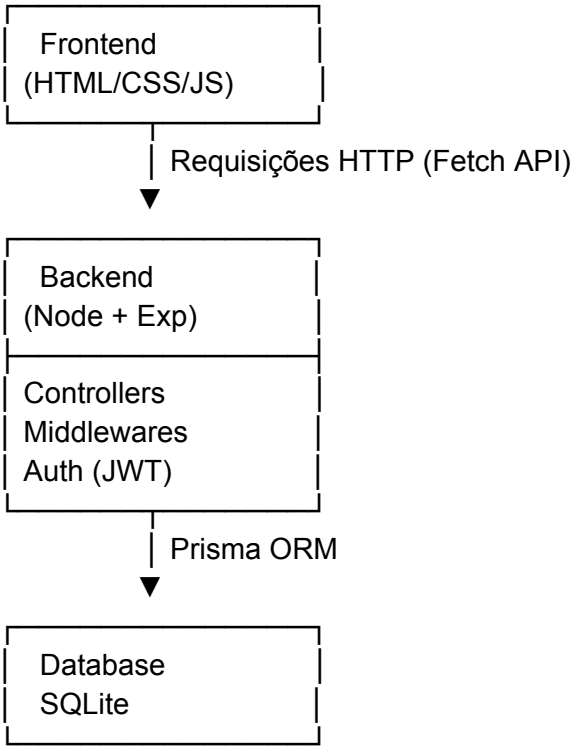
```
backend/
├── controllers/
│   ├── authController.js
│   └── fileController.js
├── middleware/
│   └── auth.js
├── database/
│   └── database.db
├── lib/
│   └── prisma.js
├── config/
│   └── multer.js
├── migrations/
├── server.js
└── .env
```

Frontend

```
frontend/
├── index.html
├── js/
│   ├── main.js
│   └── modules/
│       ├── api.js
│       ├── auth.js
│       ├── profile.js
│       ├── logs.js
│       └── notifications.js
└── css/
```

---

5. Diagrama de Componentes (UML)



Descrição

Este diagrama representa a **visão de componentes** do sistema SecureVault.
O frontend é responsável pela interação com o usuário e comunicação com o backend por meio de requisições HTTP.
O backend centraliza as regras de negócio, autenticação via JWT e controle de acesso, utilizando o Prisma ORM para comunicação com o banco de dados SQLite.

---

6. Fluxo de Autenticação

Usuário → Login → Frontend
↓
POST /login
↓
Backend valida
↓
JWT gerado
↓
Token salvo (localStorage)
↓
Requisições protegidas com Authorization Bearer

---

7. Comunicação Frontend ↔ Backend

Exemplo de requisição

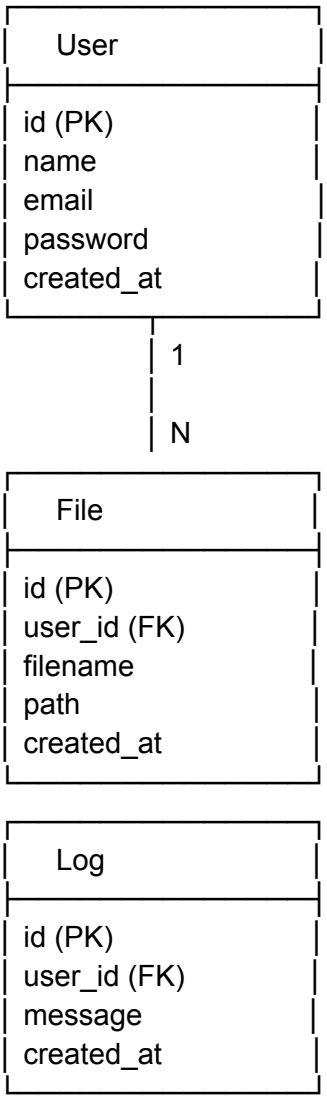
GET /profile
Headers:
Authorization: Bearer <token>

Exemplo de resposta

```
{
"id": 1,
"name": "Usuário",
"email": "user@email.com"
}
```

---

8. Diagrama Entidade-Relacionamento (ER)



Descrição

O diagrama entidade–relacionamento apresenta a estrutura lógica do banco de dados.

A entidade **User** possui relacionamento de um-para-muitos com as entidades **File** e **Log**, permitindo o controle de arquivos armazenados e o registro de ações realizadas no sistema

---

9. Diagrama de Módulos do Frontend

main.js
- auth.js
- api.js
- theme-switcher.js
- profile.js
- logs.js
- notifications.js

Responsabilidade:

main.js: inicialização global
api.js: comunicação HTTP
auth.js: login/logout/token
profile.js: dados do usuário
logs.js: histórico do sistema

---

10. Banco de Dados (Modelo Conceitual)

User
- id
- name
- email
- password
- createdAt

File
- id
- filename
- path
- userId
- createdAt

Log
- id
- message
- userId
- createdAt

---

11. Execução do Projeto

Backend

```
cd backend
npm install
npx prisma migrate dev --name init
npx prisma generate
node server.js
```

Frontend

```
cd frontend
npx serve
```

---

12. Conclusão

O projeto segue boas práticas de separação de responsabilidades, uso de ORM, autenticação segura e modularização do frontend. Ele é escalável, fácil de manter e ideal para demonstração acadêmica ou base para sistemas maiores.