**Relatório do Projeto: Sistema de Gerenciamento de Eventos**

**1. Visão Geral**

O **Sistema de Gerenciamento de Eventos** é uma API RESTful desenvolvida em Node.js com o objetivo de gerenciar eventos em uma plataforma. Ela permite aos usuários autenticados criar, editar e deletar eventos, enquanto as listagens de eventos são acessíveis publicamente.

O projeto utiliza as seguintes tecnologias e bibliotecas principais:

* **Node.js** para o backend.
* **Express.js** como framework para criação de rotas e middlewares.
* **Passport.js** para autenticação de usuários.
* **PostgreSQL** como banco de dados relacional.
* **JWT (JSON Web Tokens)** para controle de autenticação.

**2. Configuração do Ambiente**

**2.1. Pré-requisitos**

* **Node.js** (versão 16 ou superior)
* **PostgreSQL** (versão 12 ou superior)
* **Postman** (para testes das rotas)

**2.2. Instalação**

1. Clone o repositório:

git clone <url-do-repositorio>

1. Entre na pasta do projeto:

cd sistema-gerenciamento-eventos

1. Instale as dependências:

npm install

1. Configure o arquivo .env: Crie um arquivo .env na raiz do projeto com o seguinte conteúdo:

env

PORT=3000

DB\_USER=seu\_usuario

DB\_PASSWORD=sua\_senha

DB\_HOST=localhost

DB\_PORT=5432

DB\_NAME=sistema\_eventos

JWT\_SECRET=sua\_chave\_secreta

1. Execute as migrações do banco:

npx sequelize-cli db:migrate

1. Inicie o servidor:

npm start

**3. Estrutura do Projeto**

sistema-gerenciamento-eventos/

│

├── controllers/ # Controladores das rotas

├── middlewares/ # Middlewares para validações e autenticação

├── docs # documentacao

├── routes/ # Definição das rotas da API

├── node\_modules # dependencias

├── config/ # Configuração do banco de dados

├── .env # Configurações do ambiente

├── package.json # Gerenciamento de dependências

├── app.js # Ponto de entrada da aplicação

└──

**4. Funcionalidades**

**4.1. Registro de Usuários**

* **Rota:** POST /api/register
* Permite que novos usuários sejam cadastrados na plataforma.

**4.2. Login de Usuários**

* **Rota:** POST /api/login
* Retorna um token JWT para autenticação nas rotas protegidas.

**4.3. Criação de Eventos**

* **Rota:** POST /api/events/create
* Apenas usuários autenticados podem criar eventos. Os dados do evento incluem título, descrição, data e local.

**4.4. Listagem de Eventos**

* **Rota:** GET /api/events
* Disponível para qualquer usuário (rota pública).

**4.5. Edição de Eventos**

* **Rota:** PUT /api/events/:id/edit
* Apenas o criador do evento pode editá-lo.

**4.6. Deleção de Eventos**

* **Rota:** DELETE /api/events/:id/delete
* Apenas o criador do evento pode deletá-lo.

**5. Exemplos de Requisições**

**5.1. Registro de Usuário**

**Método:** POST  
**URL:** /api/register  
**Body:**

{

"username": "usuario\_teste",

"password": "senha123"

}

**5.2. Login**

**Método:** POST  
**URL:** /api/login  
**Body:**

{

"username": "usuario\_teste",

"password": "senha123"

}

**Resposta:**

{

"token": "seu\_token\_jwt\_aqui"

}

**5.3. Criação de Evento**

**Método:** POST  
**URL:** /api/events/create  
**Headers:**

Authorization: Bearer seu\_token\_jwt\_aqui

**Body:**

{

"title": "Workshop de Node.js",

"description": "Aprenda a criar APIs com Node.js",

"event\_date": "2024-12-01",

"location": "São Paulo"

}

**6. Autenticação e Autorização**

* A autenticação utiliza JWT, e o token deve ser incluído no cabeçalho das requisições protegidas.
* **Exemplo de Header:**

Authorization: Bearer <seu\_token\_jwt>

**7. Considerações Finais**

**7.1. Melhorias Futuras**

* Implementação de paginação na listagem de eventos.
* Adicionar suporte para uploads de imagens para os eventos.
* Criar um sistema de notificações para participantes.

**7.2. Dificuldades Encontradas**

* Configuração inicial do Passport.js e integração com JWT.
* Gerenciamento de erros em rotas protegidas.