■ Documentação do Backend – Gerenciador de Eventos (NEKI)

1. Visão Geral

O backend do sistema **Gerenciador de Eventos** foi desenvolvido em **Spring Boot 3**, utilizando **PostgreSQL** como banco de dados. O sistema fornece autenticação baseada em **JWT**, documentação via **Swagger/OpenAPI** e segue boas práticas de API RESTful.

2. Estrutura do Projeto

Pacote	Descrição
config	Configurações globais (segurança, CORS, Swagger)
controller	Controladores REST responsáveis pelos endpoints
dto	Data Transfer Objects – objetos de entrada e saída da API
exception	Exceções personalizadas e tratador global
model	Entidades JPA mapeadas para o banco
repository	Interfaces JPA para acesso ao banco
security	Autenticação e validação JWT
service	Regras de negócio do sistema

3. Configurações Principais

- **OpenApiConfig** → Ativa a documentação Swagger em `/swagger-ui.html`. - **SecurityConfig** → Define autenticação JWT, libera rotas públicas (`/auth/**`, Swagger). - **WebConfig** → Configuração de CORS e acesso a recursos estáticos (/uploads). - **application.properties** → Configuração do PostgreSQL, JWT e Swagger.

4. Entidades (Models)

- **Administrador**: id, nome, email, senha (criptografada). Relacionamento 1:N com eventos. - **Evento**: id, nome, data, localização, imagem (binário ou URL). Relacionamento N:1 com administrador.

5. DTOs

- AdministradorRequestDTO → usado no cadastro de administrador. - AdministradorDTO → resposta com id, nome, email. - LoginRequestDTO → email e senha para login. - LoginResponseDTO → token JWT + dados do administrador. - EventoRequestDTO → criação de evento (upload ou URL). - EventoUpdateRequestDTO → atualização parcial (data/localização). - EventoDTO → resposta de listagens com dados do evento.

6. Serviços

- **AdministradorService**: cadastro com criptografia de senha, busca por email, validação de senha. - **EventoService**: criação de eventos (upload ou URL), listagem por administrador ou geral, atualização, exclusão e recuperação de imagens.

7. Segurança e Autenticação JWT

- **CustomUserDetailsService** → integra administradores com o Spring Security. - **JwtService** → geração e validação de tokens JWT. - **JwtAuthenticationFilter** → intercepta requisições e valida tokens. - **SecurityConfig** → define permissões de rotas e adiciona o filtro JWT.

8. Exceções

- **EmailJaCadastradoException** → lançada quando o e-mail já existe no sistema. - **GlobalExceptionHandler** → retorna respostas JSON padronizadas em caso de erro.

9. Endpoints Disponíveis

Método	Endpoint	Descrição
POST	/auth/register	Cadastro de administrador
POST	/auth/login	Login e retorno de token JWT
POST	/eventos/upload	Criação de evento com upload de imagem
POST	/eventos/url	Criação de evento com URL
GET	/eventos/admin/{id}	Listagem de eventos de um administrador
GET	/eventos	Listagem de todos os eventos
PUT	/eventos/{id}	Atualização de evento
DELETE	/eventos/{id}	Exclusão de evento
GET	/eventos/{id}/imagem	Recupera a imagem do evento

10. Segurança JWT

Todas as requisições (exceto /auth/** e Swagger) exigem token JWT no header: Authorization: Bearer O token expira em 24h (86.400.000 ms).

11. Exemplos de Requisições

Login

curl -X POST http://localhost:8080/auth/login -H "Content-Type: application/json" -d '{"email": "admin@neki.com", "senha": "123456"}'

Criar Evento (URL)

curl -X POST http://localhost:8080/eventos/url -H "Authorization: Bearer " -H "Content-Type: application/json" -d '{"nome":"Tech Talk","data":"2025-08-25T20:00:00","localizacao":"Petrópolis","administradorId":1,"imagemUrl":"https://exemplo.com/img.png"}'