API CASA DE JAIRO - DOCUMENTAÇÃO COMPLETA

VISÃO GERAL

A **API Casa de Jairo** é um sistema backend desenvolvido em **Spring Boot** para suportar o website da ONG Casa de Jairo. A API oferece funcionalidades para gerenciamento de eventos, transparência organizacional, cadastro de voluntários e parcerias empresariais.

TECNOLOGIAS UTILIZADAS

Framework: Spring Boot 3.x Linguagem: Java Banco de Dados: MySQL Autenticação: JWT (JSON Web Tokens) Documentação: Swagger/OpenAPI 3 Upload de Arquivos: MultipartFile E-mail: Spring Mail (SMTP Gmail) Deploy: Fly.io

URLS DE ACESSO

Produção: https://back-sitecasadejairo.onrender.com Local: http://localhost:8080 Documentação Swagger: /swagger-ui.html API Docs: /v3/api-docs

ARQUITETURA E SEGURANÇA

SEGURANÇA (SecurityConfig)

A API implementa segurança robusta com as seguintes características:

CONFIGURAÇÃO CORS Origins Permitidos: Configuráveis via variável de ambiente

- Desenvolvimento: http://localhost:4200
- Produção: https://casadejairo.online, https://www.casadejairo.online Métodos Permitidos: GET, POST, PUT, DELETE, OPTIONS, PATCH Headers: Configuração completa com cache de preflight (1 hora)

AUTENTICAÇÃO JWT Filtro JWT: Intercepta todas as requisições autenticadas Encoder: BCrypt para senhas Sessões: Stateless (sem sessões no servidor)

PERMISSÕES POR ENDPOINT

Públicos (sem autenticação):

- GET /api/eventos/** - Visualização de eventos
- GET /api/transparencia/** - Consulta de transparência
- GET /api/voluntarios/** - Lista de voluntários
- GET /api/empresa-parceira/** - Lista de empresas parceiras
- POST /api/voluntarios/** - Cadastro de voluntários
- POST /api/empresa-parceira/** - Cadastro de empresas
- /api/auth/** - Endpoints de autenticação

HEADERS DE SEGURANÇA HSTS: HTTP Strict Transport Security habilitado: Secure, HttpOnly, SameSite=Strict

ENDPOINTS DA API

1. EVENTOS (/api/eventos)

Gerenciamento completo de eventos da ONG com suporte a imagens.

POST /api/eventos Descrição: Cria um novo evento com imagem opcional Autenticação: Requerida (ADMIN) Content-Type: multipart/form-data

Parâmetros:

- titulo (string, obrigatório): Título do evento
- descricao (string, obrigatório): Descrição detalhada
- data (string, obrigatório): Data no formato YYYY-MM-DD
- local (string, obrigatório): Local do evento
- imagem (file, opcional): Arquivo de imagem

Funcionalidades Especiais:

- Upload e armazenamento de imagem
- Notificação automática por e-mail para voluntários cadastrados
- Validação de formato de data

GET /api/eventos Descrição: Lista todos os eventos Autenticação: Não requerida

GET /api/eventos/{id} Descrição: Busca evento específico por ID Autenticação: Não requerida

PUT /api/eventos/{id} Descrição: Atualiza evento existente Autenticação: Requerida (ADMIN) Content-Type: multipart/form-data

DELETE /api/eventos/{id} Descrição: Remove evento Autenticação: Requerida (ADMIN)

GET /api/eventos/imagem/{id} Descrição: Recupera imagem do evento Response: Imagem JPEG com cache de 1 ano

2. TRANSPARÊNCIA (/api/transparencia)

Sistema de transparência organizacional com documentos e imagens.

GET /api/transparencia Descrição: Lista todos os registros de transparência Autenticação: Não requerida

POST /api/transparencia/com-imagem Descrição: Cria registro de transparência com imagem Autenticação: Requerida (ADMIN) Content-Type: multipart/form-data

Parâmetros:

- titulo (string, obrigatório): Título do registro
- descricao (string, obrigatório): Descrição detalhada
- data (string, obrigatório): Data no formato YYYY-MM-DD
- imagem (file, opcional): Arquivo de imagem (máx. 10MB)

Validações:

- Tamanho máximo da imagem: 20MB
- Tipos aceitos: image/*
- Validação de formato de data

PUT /api/transparencia/{id}/com-imagem Descrição: Atualiza registro de transparência Autenticação: Requerida (ADMIN)

DELETE /api/transparencia/{id} Descrição: Remove registro de transparência Autenticação: Requerida (ADMIN)

GET /api/transparencia/imagem/{postImagemId} Descrição: Recupera imagem do registro Response: Imagem JPEG com cache público

3. VOLUNTÁRIOS (/api/voluntarios)

Sistema de cadastro e gerenciamento de voluntários.

POST /api/voluntarios Descrição: Cadastra novo voluntário Autenticação: Não requerida Content-Type: application/json

Funcionalidades:

- Validação de e-mail único
- Cadastro público (qualquer pessoa pode se voluntariar)
- Tratamento de erros detalhado

GET /api/voluntarios Descrição: Lista todos os voluntários Autenticação: Não requerida

GET /api/voluntarios/email/{email} Descrição: Busca voluntário por e-mail Autenticação: Não requerida

DELETE /api/voluntarios/{id} Descrição: Remove voluntário Autenticação: Requerida

4. EMPRESAS PARCEIRAS (/api/empresa-parceira)

Gerenciamento de parcerias empresariais.

POST /api/empresa-parceira Descrição: Cadastra nova empresa parceira Autenticação: Não requerida Content-Type: application/json

GET /api/empresa-parceira Descrição: Lista todas as empresas parceiras Autenticação: Não requerida

GET /api/empresa-parceira/{id} Descrição: Busca empresa por ID Autenticação: Não requerida

PUT /api/empresa-parceira/{id} Descrição: Atualiza dados da empresa Autenticação: Requerida

DELETE /api/empresa-parceira/{id} Descrição: Remove empresa parceira Autenticação: Requerida

5. UPLOAD DE IMAGENS (/api/postImagem)

Sistema auxiliar para upload de imagens.

POST /api/postImagem Descrição: Upload de imagem com título e conteúdo Content-Type: multipart/form-data

GET /api/postImagem/{id} Descrição: Recupera imagem por ID Response: Imagem JPEG

6. E-MAIL (/email)

Sistema de envio de e-mails com templates.

POST /email/enviar Descrição: Envia e-mail usando template Content-Type: application/json

Parâmetros:

- para: E-mail destinatário
- assunto: Assunto do e-mail
- titulo: Título para o template
- mensagem: Conteúdo da mensagem

CONFIGURAÇÕES DO SISTEMA

Banco de Dados

- **Provider**: MySQL com Fly.io
- **Pool de Conexões**: HikariCP
    - Máximo: 2 conexões
    - Mínimo idle: 1
    - Timeout: 30s

E-mail

- **Provider**: Gmail SMTP
- **Configuração**: TLS habilitado
- **Templates**: Sistema de templates para notificações

Upload de Arquivos

- **Tamanho máximo**: 20MB por request/arquivo
- **Tipos suportados**: Imagens (image/*)

Tratamento de Erros

A API implementa tratamento robusto de erros com:

Respostas Padronizadas

{

"mensagem": "Descrição do erro",

"timestamp": "2024-01-01T10:00:00",

"status": 400

}

Códigos HTTP

- **200**: Sucesso
- **201**: Criado com sucesso
- **204**: Removido com sucesso
- **400**: Erro de validação/Bad Request
- **401**: Não autenticado
- **403**: Não autorizado
- **404**: Recurso não encontrado
- **500**: Erro interno do servidor

Validações Específicas

- **Imagens**: Validação de tipo e tamanho
- **Datas**: Formato YYYY-MM-DD obrigatório
- **E-mails**: Unicidade e formato válido
- **Campos obrigatórios**: Validação completa

Notificações por E-mail

O sistema possui notificação automática para eventos:

Trigger

- Criação de novo evento dispara e-mail para todos os voluntários

Template

- E-mail HTML personalizado com informações do evento
- Integração com sistema de templates Thymeleaf

Configuração SMTP

- Gmail como provedor
- Configuração via variáveis de ambiente
- Timeout configurado para estabilidade

Deployment e Ambiente

Variáveis de Ambiente Requeridas

Banco de Dados

DB_HOST=seu-host-mysql

DB_PORT=3306

DB_NAME=casajairo

DB_USER=usuario

DB_PASSWORD=senha

JWT

JWT_SECRET=sua-chave-secreta-jwt

E-mail

EMAIL_USERNAME=email@gmail.com

EMAIL_PASSWORD=senha-app-gmail

SMTP_HOST=smtp.gmail.com

SMTP_PORT=587

CORS

CORS_ORIGINS=http://localhost:4200,https://casadejairo.online

Ambiente

SPRING_PROFILES_ACTIVE=prod

Swagger/OpenAPI

Acesso

- **URL**: /swagger-ui.html
- **API Docs**: /v3/api-docs

Funcionalidades

- Documentação interativa completa
- Teste de endpoints diretamente na interface
- Autenticação JWT integrada
- Schemas de dados detalhados

Configuração

- Ordenação por método HTTP
- Persistência de autorização

- Informações de contato e licença

Considerações de Performance

Otimizações

- **Connection Pool**: Configurado para ambiente de produção
- **Cache de Imagens**: 1 ano para imagens estáticas
- **CORS Cache**: 1 hora para preflight requests
- **JPA**: open-in-view=false para performance

Limitações

- Pool de conexões limitado (adequado para pequena/média escala)
- Upload máximo de 20MB por request
- Rate limiting não implementado (pode ser adicionado se necessário)

Roadmap de Melhorias

1. **Rate Limiting**: Implementar controle de taxa de requisições
2. **Logs Estruturados**: Adicionar logging estruturado com ELK Stack
3. **Cache Redis**: Implementar cache distribuído para melhor performance
4. **Métricas**: Integrar Prometheus para métricas detalhadas
5. **Backup Automatizado**: Sistema de backup automático do banco

---

Contato: casadejairo.adm@hotmail.com
Website: https://casadejairo.online