


Votação Almoço API

Informações de acesso

| Usuário | Senha | Permissão de acesso |
|------------|-------|---------------------|
| admin | 123 | Todas |
| apurador | | /api/eleicoes/* |
| faminto_1 | | /api/votacoes/* |
| faminto_2 | | |
| faminto_3 | | |
| faminto_4 | | |
| faminto_5 | | |
| faminto_6 | | |
| faminto_7 | | |
| faminto_8 | | |
| faminto_9 | | |
| faminto_10 | | |

 **swagger**

Select a spec default

Votação Almoço API ^{1.0}

[Base URL: localhost:8080/votacao-almoco]
<http://localhost:8080/votacao-almoco/v2/api-docs>

[Contact Everton Gonçalves](#)

eleicao-controller Eleicao Controller

GET /api/eleicoes Retorna as eleições

POST /api/eleicoes/apurar Apura o restaurante com o maior número de votos do dia

restaurante-controller Restaurante Controller

GET /api/restaurantes Retorna todos os restaurantes

usuario-controller Usuario Controller

GET /api/usuarios Retorna todos os usuários

votacao-controller Votacao Controller

GET /api/votacoes Retorna todas as votações

POST /api/votacoes/restaurante/{id} Vota em um restaurante

Acesso ao Swagger: <http://localhost:8080/votacao-almoco/swagger-ui.html>

Utilização da ferramenta Postman

Usuários

GET

http://localhost:8080/votacao-almoco/api/usuarios

Send

ParamsAuthorizationHeaders (1)BodyPre-request ScriptTestsCookiesCode

TYPE

Basic Auth

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Username

admin

Password

123

☒ Show Password

Restaurantes

GET

http://localhost:8080/votacao-almoco/api/restaurantes

Send

ParamsAuthorizationHeaders (1)BodyPre-request ScriptTestsCookiesCode

TYPE

Basic Auth

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Username

admin

Password

123

☒ Show Password

Votações

GET

http://localhost:8080/votacao-almoco/api/votacoes

Send

ParamsAuthorizationHeaders (1)BodyPre-request ScriptTestsCookiesCode

TYPE

Basic Auth

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Username

admin

Password

123

☒ Show Password

Realizar votação em um restaurante

POST

http://localhost:8080/votacao-almoco/api/votacoes/restaurante/1

Send

ParamsAuthorizationHeaders (1)BodyPre-request ScriptTestsCookiesCode

TYPE

Basic Auth

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Username

faminto_1

Password

123

☒ Show Password

Apurar restaurante vencedor

POST

http://localhost:8080/votacao-almoco/api/eleicoes/apurar

Send

ParamsAuthorizationHeaders (1)BodyPre-request ScriptTestsCookiesCode

TYPE

Basic Auth

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Username

apurador

Password

123

☒ Show Password

Eleições

GET

http://localhost:8080/votacao-almoco/api/eleicoes

Send

ParamsAuthorizationHeaders (1)BodyPre-request ScriptTestsCookiesCode

TYPE

Basic Auth

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Username

apurador

Password

123

☒ Show Password

Informações

- O horário limite para a votação é configurado na propriedade **votacao.horaLimite** localizada no arquivo application.properties da aplicação e leva o formato HH:mm.
- Ao iniciar a aplicação, a base de dados é criada e o script localizado import.sql é executado automaticamente.

Execução da aplicação

Requisitos:

- Apache Maven 3.3.X
- Java 1.8

Iniciar a aplicação via Spring:

- mvn spring-boot:run

Iniciar a aplicação via comando java:

- mvn clean package
- java -jar target/votacao-almoco-api-1.0.0.jar

Tecnologias utilizadas

- Spring Boot: Framework de código aberto que facilita o desenvolvimento e configuração de APIs.
- H2: Sistema de gerenciamento de banco de dados relacional escrito em Java.
- Maven: Ferramenta para automação de builds e gerenciamento das dependências do projeto.
- JUnit: Framework de código aberto para implementação de testes de unidade.
- Lombok: Framework que provê implementações de código com o objetivo de auxiliar na produtividade e redução do mesmo.
- Swagger: Framework para documentar APIs Rest.

Pontos de destaque

- Utilização e configuração em memória do banco H2.
- Utilização do Spring Data para a persistência de dados.
- Delegação do controle transacional ao Spring na camada Service.
- Utilização de autenticação básica do Spring Security com usuários em memória e permissões para controle de acesso para as rotas.
- Tratamento de erros através da classe `ExceptionHandler` e apresentação customizada das mensagens através da classe `ErrorDetails`.
- Padronização de mensagens com suporte a internacionalização utilizando a classe utilitária `MessageSource` do Spring.
- Utilização do design pattern Bridge para separação de cada regra de negócio com o objetivo de reduzir a quantidade de código da camada Service e facilitar os testes unitários.
- Utilização do design pattern Adapter para converter um objeto para que outro o entenda.
- Utilização do design pattern Builder para a criação de objetos complexos através do Lombok.
- Utilização de generics na estrutura de validators.

Sugestão de melhoria

- Testes de integração.
- Implementação de programação orientada a aspectos (Spring AOP) para logs do sistema.
- Criação de arquitetura de microserviços desmembrando componentes mínimos e independentes (usuários e restaurantes).