Api - Processo de vaga de Estágio AIKO

Documentação da Aplicação

### Autor(a): Claudiane Costa Monteiro da Silva

Data publicação : 25/04/2022

Salvador - BA, Brasil

PT-BR



### Introdução

A documentação a seguir foi desenvolvida para o processo seletivo do estágio na empresa Aiko, onde foi proposto o desafio da construção de uma aplicação API Rest com CRUD, sendo fornecido no repositório do projeto um banco de dados construído com base na seguinte situação:

“Você é o desenvolvedor backend de uma empresa que coleta dados de equipamentos utilizados em uma operação florestal. Dentre esses dados estão o histórico de posições e estados desses equipamentos. O estado de um equipamento é utilizado para saber o que o equipamento estava fazendo em um determinado momento, seja *operando*, *parado* ou em *manutenção*. O estado é alterado de acordo com o uso do equipamento na operação, já a posição do equipamento é coletada através do GPS e é enviada e armazenada de tempo em tempo pela aplicação.

Seu objetivo é, de posse desses dados, desenvolver uma aplicação backend que exponha esses dados através de uma API.”

## Processo de construção:

Foi feito o download da ferramenta [Spring Boot Tool Suite 4](https://spring.io/tools) . Onde foi criado um projeto com o nome api\_aiko\_processo e nele foi gerado a seguinte estrutura:

* src/main/java => com.aikoprocesso.api (**esse pacote sendo artifactId no Maven**)
* com.aikoprocesso.api.controller
* com.aikoprocesso.api.controller.status
* com.aikoprocesso.api.model
* com.aikoprocesso.api.repository

### 1.1 - Backup do Banco de Dados

Foi criado um banco de dados de nome **processoaiko** vazio e feito o backup com o arquivo data.backup, tendo o primeiro contato fiz o estudo e realizei o mapeamento de todos os dados, colunas e tabelas fornecidas.

## Tomada de decisão

Optei pela escolha de utilizar como biblioteca de mapeamento objeto/relacional o JPA ou [Jakarta Persistence 3.0](https://jakarta.ee/specifications/persistence/3.0/) para juntar as informações do banco de dados utilizando Java. Foram criadas 6 classes no pacote **com.aikoprocesso.model***,*  todas com o mesmo nome que consta nas tabelas do banco de dados.

Fui ate o arquivo applicattion.properties e fiz a seguinte configuração para o banco de dados:

spring.jpa.properties.hibernate.jdbc.lob.non\_contextual\_creation=true

spring.datasource.url=jdbc:postgresql://localhost:2456/processoaiko

spring.datasource.username=postgres

spring.datasource.password=1\*\*

spring.jpa.hibernate.ddl-auto=validate

Foram feitos primeiramente somente teste com a tabela *equipment*, criado no diretório **com.aikoprocesso.api.repository***.* Na classe **equipment.java**, criei os atributos que referencia as colunas no banco de dados. No diretório **com.aikoprocesso.api.repository**, foi criado a interface que estende a configuração do repositório JPA, passando como parâmetros o nome da classee o tipo do Id. Em seguida criei a classe que recebe os métodos HTTP, utilizando o método **GET** para poder listar todos os dados da coluna e ao realizar o teste no (Postman), foi retornado o status de sucesso caracterizado como 200 OK.

Depois da realização e o retorno do teste, comecei a estruturar o projeto e durante o desenvolvimento deste processo, comecei a observar os relacionamentos das tabelas e foi observado que três tabelas não possuem chaves primárias, somente chaves estrangeiras. Depois de testes sem sucesso e estudando a documentação do JPA, percebi que durante o uso da annotation **@Entity**, era obrigatório o uso do **@Id** que é responsável por informar qual campo/atributo de uma entidade está relacionada a respectiva tabela no banco de dados, porém, ainda havia uma observação de que essa anotação era de uso obrigatório e que resultaria em erro durante a execução caso ela não esteja presente. Com isso, tomei a decisão de inserir uma coluna id em cada tabela e gerar valores do tipo UUID, para prosseguir com a construção da API.

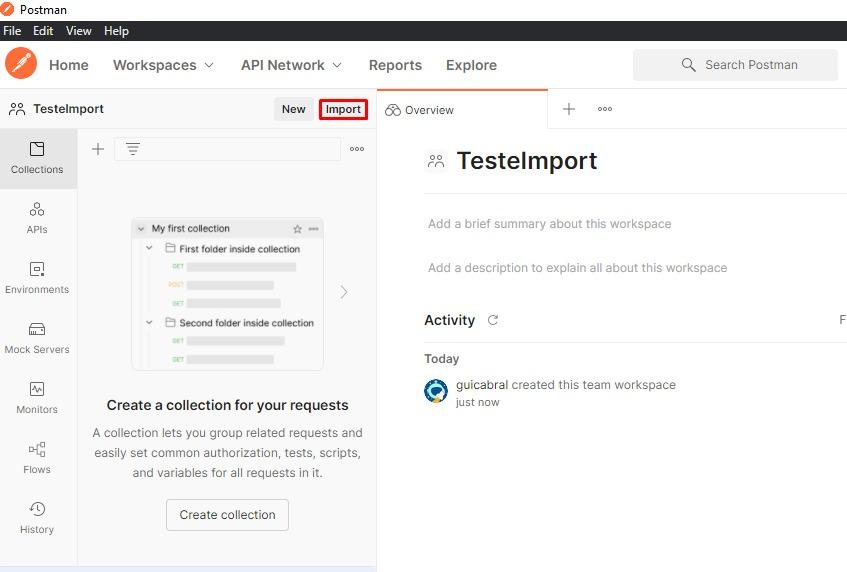
### 2.1 - Conclusão da tomada de decisão

Devido às regras impostas pelo framework JPA, e o meu conhecimento que possuía e adquiri durante a realização desse projeto,optei por essa decisão visando garantir assim a funcionalidade correta da aplicação e entrega do projeto dentro do prazo da data de entrega prevista.

## Instrução de uso:

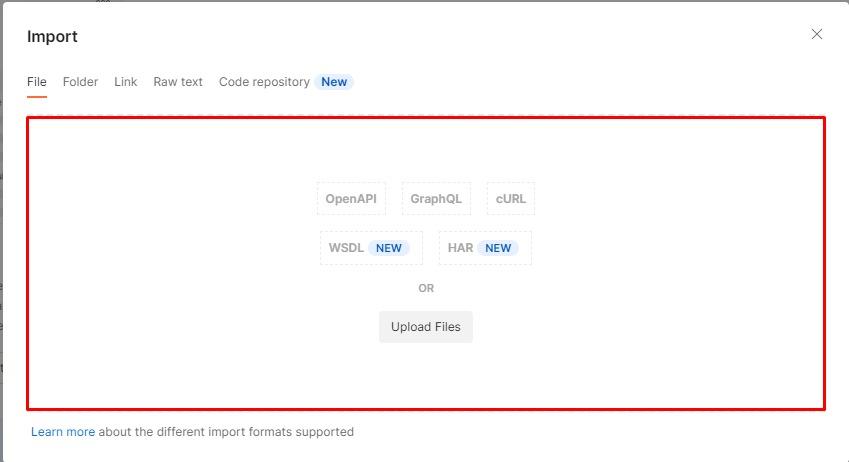
Foi anexado ao projeto postado no Github um arquivo que contém a estrutura de Collection onde está organizado todos os métodos HTTP com os EndPoint proposto no desafio, onde é possível realizar os testes juntamente ao backup do servidor localmente na máquina.

**Instruções:**

* Na aba Collection clique em **Import**:

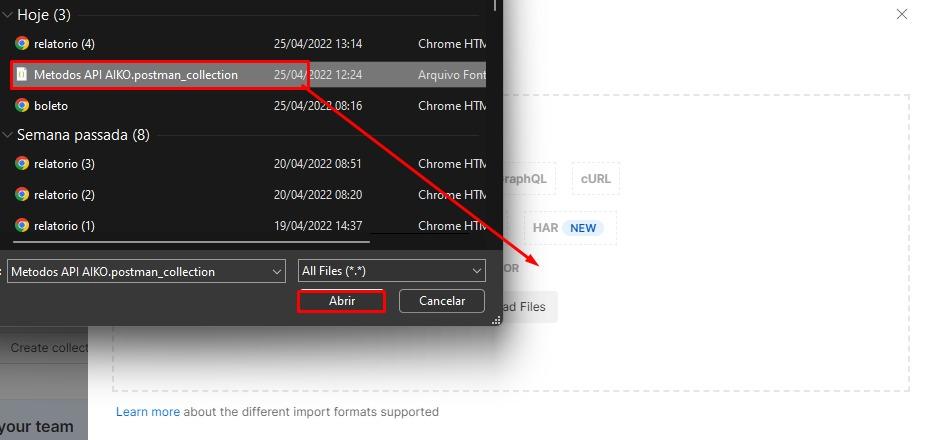
## 

* Ao abrir essa pagina iremos fazer o upload da collection:

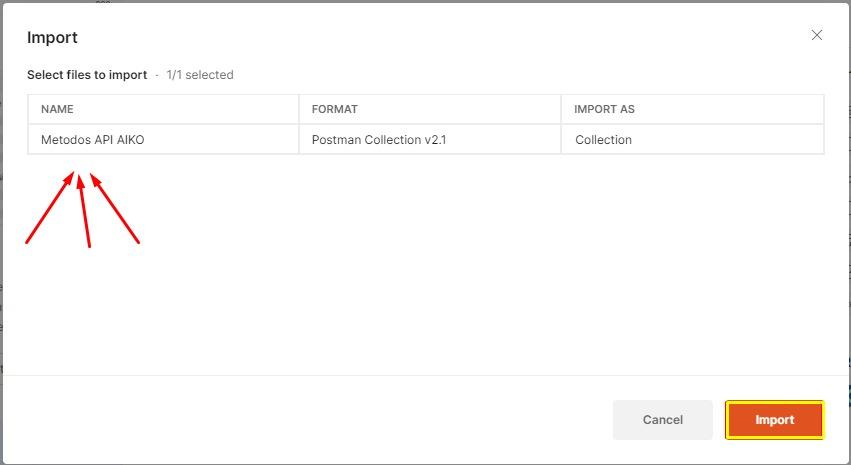


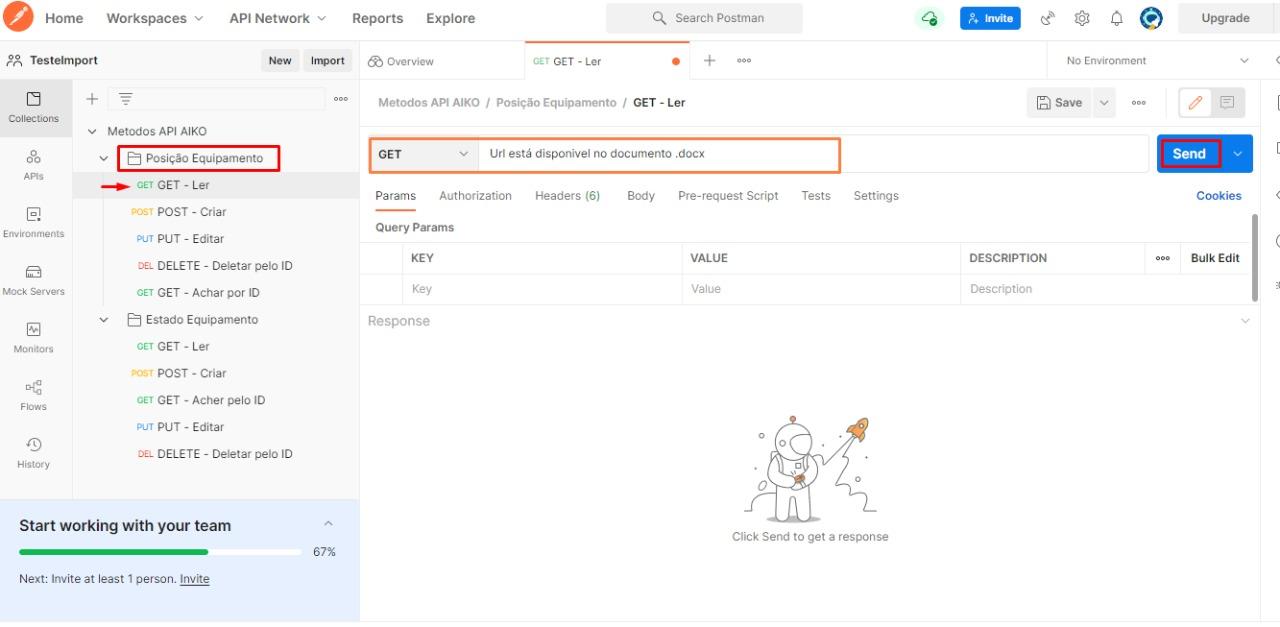
## 

* Selecione a collection que foi disponibilizada no projeto dentro do diretório (..api\_aiko\_processo\documentation\collection\Metodos API AIKO.postman\_collection.json).



* Verifique se o arquivo selecionado é o correto, em seguida, clique em **import.**



* Depois de ter importado a collection, abra as pastas para localizar a requisição a ser solicitada. Insira a url que vai está disponível no arquivo que vai está dentro do diretório (..api\_aiko\_processo\documentation\url\URLs.txt), faça a requisição e aguarde o status da resposta.

## 

## 

## 

## 

## 

## 4. Ferramentas utilizadas:

* Java
* Spring Boot Tool Suite 4
* PostgreSQL (PgAdmin 4)
* Postman

## 5. Considerações finais:

Obrigado a todos da Aiko pelo desafio, pela oportunidade de ampliar meu conhecimento, espero ter mostrado o que eu sei fazer e aprendi durante esse período. Atenciosamente, Claudiane Costa.