Procedimentos da API

instalação:

- 1- Baixar os arquivos no repositório do github: https://github.com/brunobrasillI/API-TechTitans
- 2- Abrir em algum editor de código(utilizaremos o VS Code no exemplo)
- 3- Mudar as definições antes de rodar (utilizamos o MySql no exemplo), Abrindo o seguinte arquivo:



4- Colocar o seguinte código:

no nosso exemplo utilizamos o usuário root com a senha root e um comando para atualizar as tabelas automaticamente.

```
target > classes >  application.properties

1  #PORTA ONDE VAI SER EXECUTADO O TOMCAT

2  server.port = 8080

3  #INFORMAÇÕES PARA CONEXÃO COM O BANCO DE DADOS MYSQL

5  spring.datasource.url=jdbc:mysql://localhost/api_spring?useTimezone=true&serverTimezone=UTC

6  spring.datasource.username=root

7  spring.datasource.password=root

8  spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

9  #MANIPULANDO ESTRUTURAS (TABELAS)

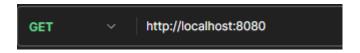
11  spring.jpa.hibernate.ddl-auto=update
```

5- Pronto agora podemos rodar a aplicação no servidor local, através dessa classe, na pasta test/java/br/com/fiap/techtitansapi:

```
TechtitansApiApplication.java
```

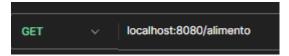
6- Com a aplicação rodando podemos fazer os testes agora.

Todos os testes da API foram feitos no POSTMAN



- 1- Colocar o link de acesso para conseguir a API e fazer os testes
- 2- Escolher o método a ser executado: GET, POST, PUT ou DELETE
- 3- Colocar no final do link o endpoint a ser requisitado, exemplo: http://localhost:8080/alimento
- 4-No caso dos **GET**, ele só retornará os dados que já foram inseridos

GET:



Vai retornar todos os dados inseridos na API, que foram inseridos naquele endpoint

```
Body
    Cookies Headers (5) Test Results
 Pretty
          Raw Preview Visualize
                                       JSON V
               "id": 1,
               "nome": "Frutas Frescas",
               "descricao": "Frutas frescas da Estação de Primavera",
               "categoria": "Frutas",
               "origem": "Fazenda de Massachusetts, EUA",
               "data": "2024-12-23"
               "id": 2,
               "nome": "Maçã",
               "descricao": "Uma fruta suculenta e saborosa.",
               "categoria": "Frutas",
               "origem": "Fazenda",
               "data": "2023-06-29"
```

Inserindo ou Editando dados:

Inserindo um alimento no banco de dados através do método **POST**, usando o Body

Inserindo um Alimento usando o Body request em formato JSON

```
Post v localhost:8080/alimento

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON v

1 2 "nome": "Uva",
3 "descricao": "Caixa de Uvas coletadas nas videiras da fazenda Solimoes",
4 "categoria": "Frutas",
5 "origem": "Plantação agricola",
6 "data": "2023-01-12"
```

Ele vai devolver o que foi inserido

Agora vamos editar esse alimento utilizando **PUT**

```
Put v localhost:8080/alimento

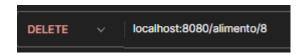
Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON v

1 2 "id": "8",
3 "nome": "Melancia",
4 "descricao": "Caixa de melancias coletadas nas plantações da fazenda Solimoes",
5 "categoria": "Frutas",
6 "origem": "Plantação agricola",
7 "data": "2023-02-12"
```

Ele vai retornar o objeto que foi alterado

E por fim para deletar esse alimento utilizamos o DELETE, chamando só o id do produto a ser deletado



Para deletar utilizamos o link do ID que queremos deletar, especificando o número depois do /alimento/{id}

E ele retorna a mensagem que foi deletado com sucesso

