

Procedimentos da API

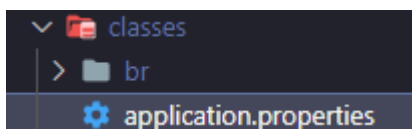
instalação:

1- Baixar os arquivos no repositório do github:

<https://github.com/brunobrasil/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:

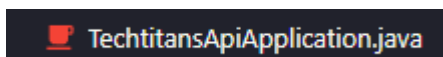


4- Colocar o seguinte código:

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

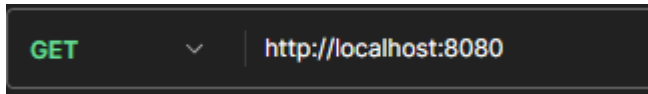
```
application.properties X
target > classes > application.properties
1  #PORTA ONDE VAI SER EXECUTADO O TOMCAT
2  server.port = 8080
3
4  #INFORMAÇÕES PARA CONEXÃO COM O BANCO DE DADOS MYSQL
5  spring.datasource.url=jdbc:mysql://localhost/api_spring?useTimezone=true&serverTimezone=UTC
6  spring.datasource.username=root
7  spring.datasource.password=root
8  spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
9
10 #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:



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

Todos os testes da API foram feitos no POSTMAN



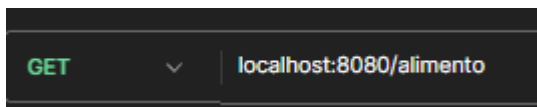
1- Colocar o link de acesso para conseguir a API e fazer os testes

2- Escolher o método a ser executado: GET, POST, PUT ou DELETE

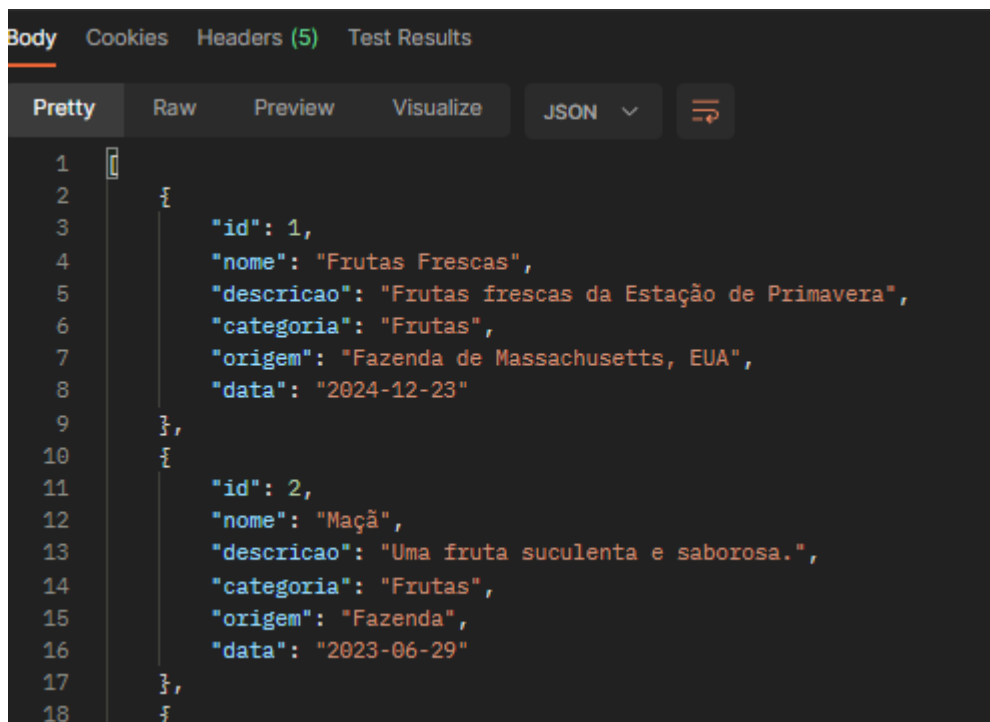
3- Colocar no final do link o endpoint a ser requisitado, exemplo: `http://localhost:8080/alimento`

4- No caso dos **GET**, ele só retornará os dados que já foram inseridos

GET:



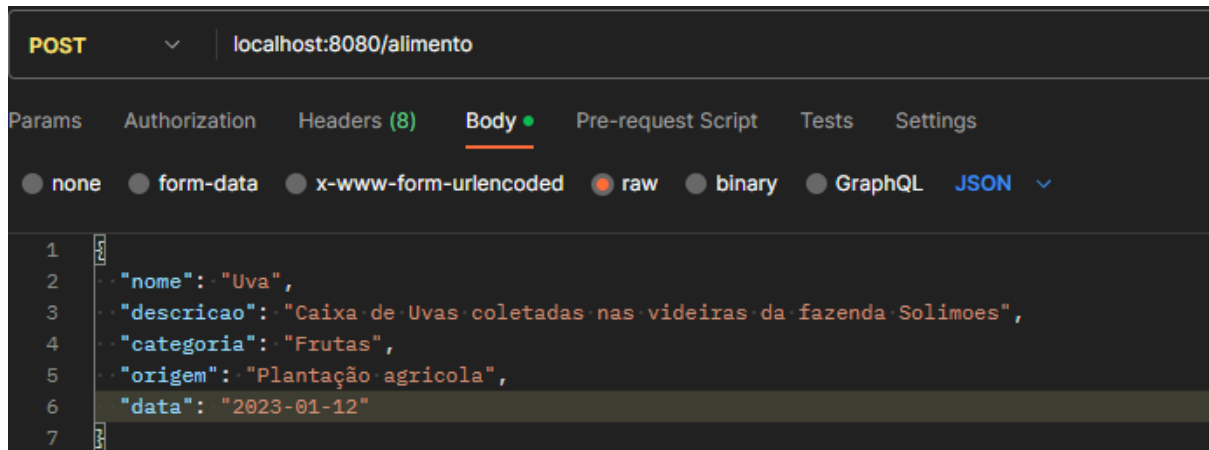
Vai retornar todos os dados inseridos na API, que foram inseridos naquele endpoint



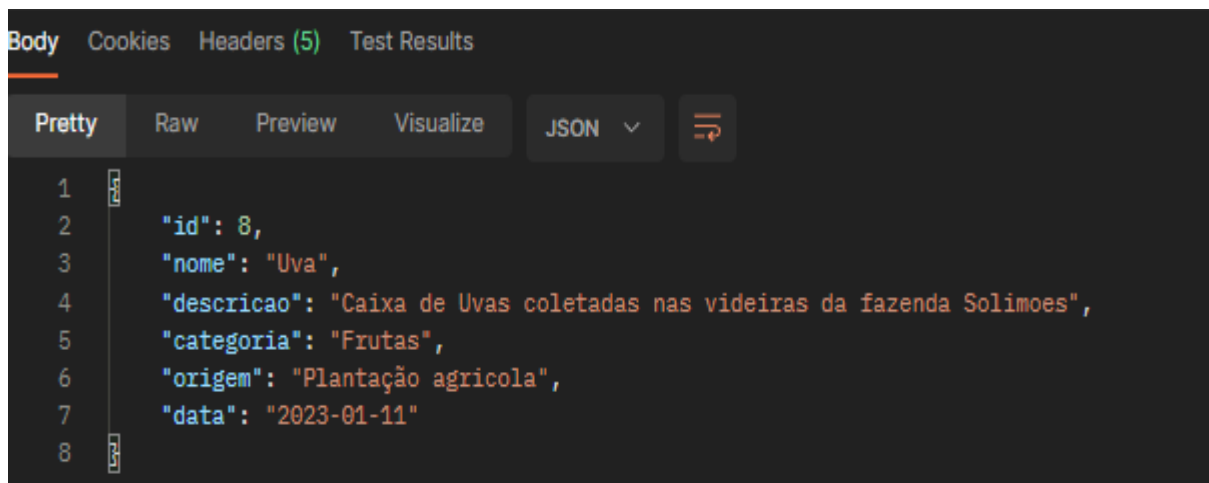
Inserindo ou Editando dados:

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

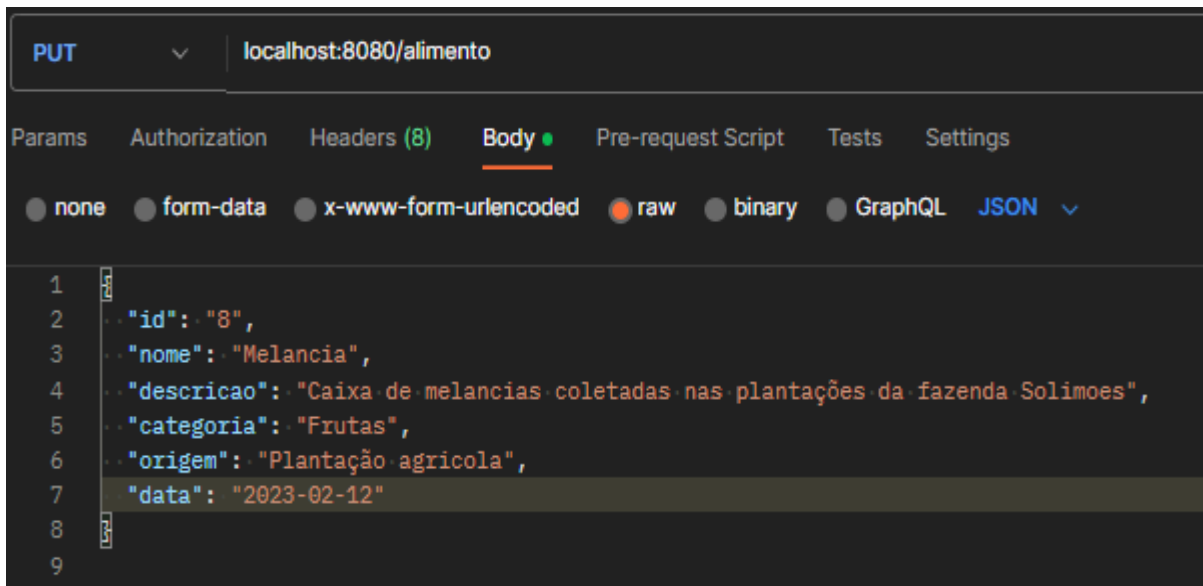
Inserindo um Alimento usando o Body request em formato JSON



Ele vai devolver o que foi inserido



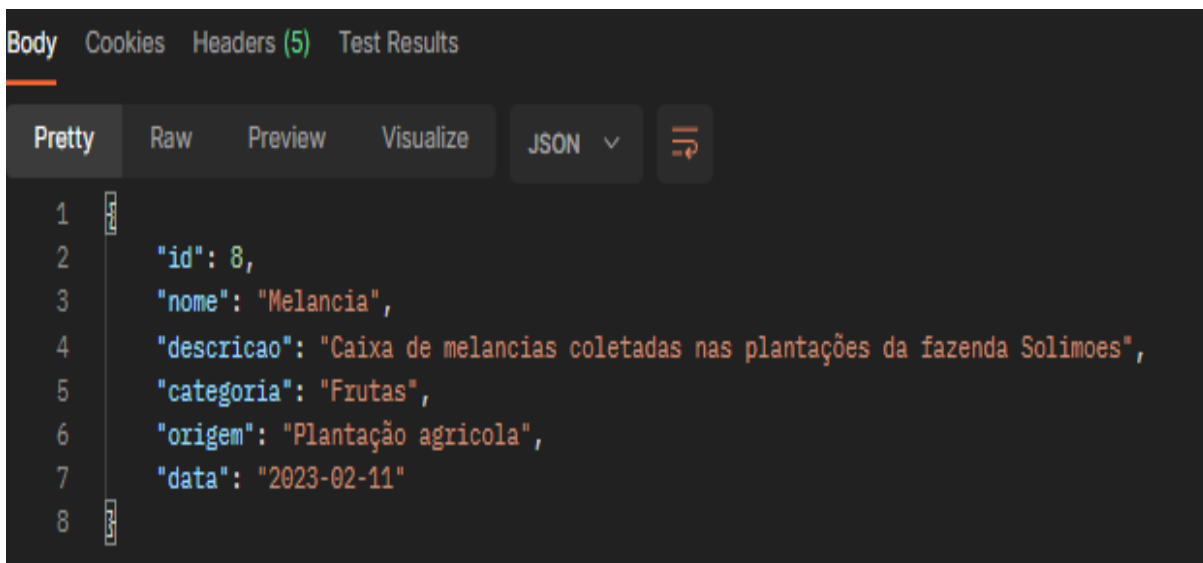
Agora vamos editar esse alimento utilizando **PUT**



The screenshot shows a REST client interface with the method 'PUT' selected and the URL 'localhost:8080/alimento'. The 'Body' tab is active, and the 'JSON' format is chosen. The JSON body contains the following data:

```
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"  
8 }  
9
```

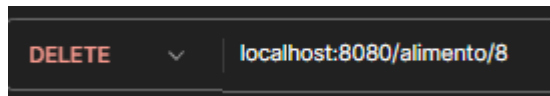
Ele vai retornar o objeto que foi alterado



The screenshot shows the response of the PUT request in the REST client. The 'Body' tab is active, and the 'JSON' format is chosen. The response is a JSON object with the following data:

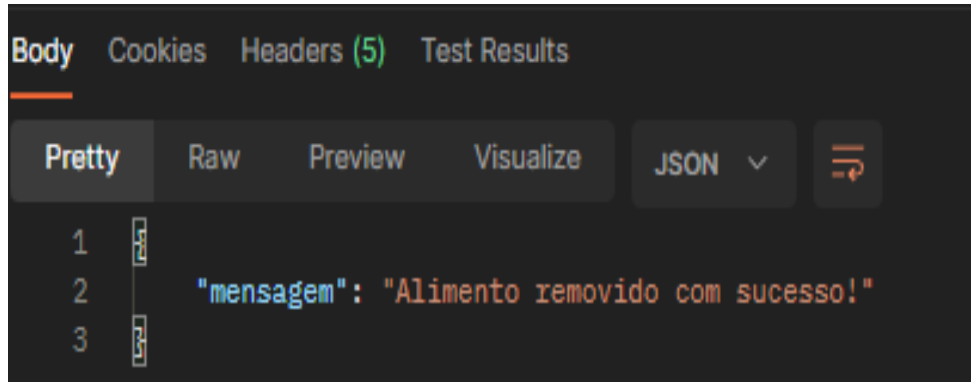
```
1 {  
2   "id": 8,  
3   "nome": "Melancia",  
4   "descricao": "Caixa de melancias coletadas nas plantações da fazenda Solimoes",  
5   "categoria": "Frutas",  
6   "origem": "Plantação agricola",  
7   "data": "2023-02-11"  
8 }
```

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



Lista de endpoints:

/alimento : Este endpoint retorna uma lista de todos os produtos alimentícios.

O corpo da resposta é uma matriz JSON de objetos, cada um representando um produto alimentício. Cada objeto tem as seguintes propriedades:

id_alimento: O ID do produto alimentício.

nm_alimento: O nome do produto alimentício.

desc_alimento: Uma descrição do produto alimentar.

cat_alimento: A categoria de comida.

origem_alimento: A origem do produto alimentício.

data_validade: A data de vencimento do produto alimentício.

/alimentos/{id_alimento}: Este endpoint retorna um único produto alimentício por ID. O corpo da resposta é um objeto JSON, que tem as mesmas propriedades que o corpo da resposta para o endpoint **/alimento**.

/fazenda: esse endpoint retorna uma lista de todos os farms.

O corpo da resposta é uma matriz JSON de objetos, cada um representando um farm. Cada objeto tem as seguintes propriedades:

id_fazenda: O ID da fazenda.

nm_fazenda: O nome da fazenda.

end_fazenda: O endereço da fazenda.

cont_fazenda: as informações de contato da fazenda.

/distribuidor: Este endpoint retorna uma lista de todos os distribuidores. O corpo da resposta é uma matriz JSON de objetos, cada um representando um distribuidor. Cada objeto tem as seguintes propriedades:

id_distribuidor: O ID do distribuidor.

nm_distribuidor: O nome do distribuidor.

end_distribuidor: O endereço do distribuidor.

cont_distribuidor: as informações de contato do distribuidor.

/nutricao: Este endpoint retorna uma lista de todas as informações nutricionais de produtos alimentícios. O corpo da resposta é uma matriz JSON de objetos, cada um representando uma informação nutricional para um produto alimentar. Cada objeto tem as seguintes propriedades:

id_alimento: O ID do produto alimentício.

calorias: O número de calorias no produto alimentar.

gord_totais: A quantidade total de gordura no produto alimentar.

gord_saturada: A quantidade de gordura saturada no produto alimentar.

gord_trans: A quantidade de gordura trans no produto alimentar.

colesterol: A quantidade de colesterol no produto alimentar.

sódio: A quantidade de sódio no produto alimentar.

carbo: A quantidade de hidratos de carbono no produto alimentar.

açúcar: A quantidade de açúcar no produto alimentar.

proteínas: A quantidade de proteínas no produto alimentar.

fibra_alimentar: A quantidade de fibra dietética no produto alimentar.