



UNIFACS - UNIVERSIDADE SALVADOR

BRENO PIMENTEL - RA: 12722119792
NICHOLAS TORRES - RA : 1272220672
PABLO SANTANA - RA: 12722126096
PRISCILA SIMAS - RA: 12722123651
ROQUE CARLOS REIS LETO - RA 1272122005

Relatório do Projeto A3
UC: Sistemas distribuídos e mobile

SALVADOR
2023

BRENO PIMENTEL - RA: 12722119792
NICHOLAS TORRES - RA : 1272220672
PABLO SANTANA - RA: 12722126096
PRISCILA SIMAS - RA: 12722123651
ROQUE CARLOS REIS LETO - RA 1272122005

Relatório do Projeto A3
UC: Sistemas distribuídos e mobile

Trabalho apresentado à Universidade Salvador
como o requisito para obtenção de nota da A3

Orientadores: Adailton Cerqueira e Wellington Lacerda

SALVADOR
2023

Resumo

Neste relatório encontra-se uma descrição do trabalho de sistemas distribuídos no qual foi elaborada uma aplicação que simula a captação de dados de venda de uma rede de lojas. O exemplo que escolhemos para o nosso sistema foi a de uma padaria, sendo esse o motivo na qual os produtos do sistemas seguem esse padrão de produtos que se encontram numa padaria.

Esperamos que através deste relatório que seja possível:

- Apresentar a api, seus objetivos e suas funcionalidades;
- As classes e os atributos escolhidos para o funcionamento da mesma
- Dar instruções de como fazer corretamente as instruções das requisições
- Esclarecer o uso do CRUD no Banco de dados relacional PostgreSQL.

LISTA DE FIGURAS

Figura 1 - Java	7
Figura 2 - AS CLASSES	8
Figura 3 - A ARQUITETURA	8
Figura 4 - CADASTRO CLIENTE	9
Figura 5 - LISTAR CLIENTE	10
Figura 6 - ATUALIZAR CLIENTE	11
Figura 7 - EXCLUIR CLIENTE	11
Figura 8 - CADASTRAR PRODUTO	12
Figura 9 - LISTAR PRODUTO	13
Figura 10 - ATUALIZAR PRODUTO	14
Figura 11 - EXCLUIR PRODUTO	14
Figura 12 - CADASTRAR ESTOQUE	15
Figura 13 - LISTAR ESTOQUE	16
Figura 14- ATUALIZAR ESTOQUE	17
Figura 15 - EXCLUIR ESTOQUE	17
Figura 16 CADASTRAR PEDIDO	18
Figura 17 - LISTAR PEDIDO	19
Figura 18 - EXCLUIR PEDIDO	20

Sumário

1. INTRODUÇÃO.....	6
2. INSTRUÇÕES DO PROJETO.....	7
2.1 COMO INSTALAR O PROJETO.....	7
3.0 AS CLASSES E A ARQUITETURA.....	24
4.0 COMO FAZER AS REQUISIÇÕES.....	30
4.1 CLIENTE.....	
4.2 PRODUTO.....	
4.3 ESTOQUE.....	
4.4 PEDIDO.....	
5.0.....	

1.0 INTRODUÇÃO

Nesse projeto optamos por uma aplicação em Java 17, utilizando a framework Spring boot e utilizando as seguintes bibliotecas (Spring data JPA , Validation , Lombok , Spring web , Dev Tools, PostgreSQL driver).

A escolha da linguagem Java se deu por ser uma linguagem que já foi abordada em UCs passadas, e que por isso, foi a linguagem que nos sentimos mais confortáveis para fazer o trabalho, sobre utilizar a framework Spring Boot isso se deu pois o Spring facilita e abstrai muitos processos que são necessários para a criação de uma API, deixando o código muito mais limpo e aumentando a assim bastante a produtividade através de suas Annotations

Usamos as seguintes bibliotecas por :

Spring data JPA - abstrai as complexidades dos detalhes de implementação do JPA e oferece uma interface de programação mais simples e consistente. Isso permite que você mude facilmente entre diferentes provedores JPA sem alterar significativamente seu código .

Lombok - Através de uma única Annotation é possível criar todos os Getters e Setters.

Dev Tools - Reinicia o projeto automaticamente a cada nova alteração.

Spring web - Spring Web fornece recursos de integração, como funcionalidade de upload de arquivo multipart e inicialização do contêiner IoC usando ouvintes Servlet e um contexto de aplicativo orientado à web. Ele também contém um cliente HTTP e as partes relacionadas à web do suporte remoto Spring


PostgreSQL driver - Por estarmos utilizando o PostgreSql como banco de dados é preciso utilizar esse driver para o funcionamento da aplicação

nome controlar essas entidades, é nele que formamos o CRUD de todas as nossas classes

2.0 INSTRUÇÕES DO PROJETO

Figura 1- Java


2.1 COMO INSTALAR O PROJETO

2.2- Primeiramente Certifique-se de ter o Java instalado no seu sistema. O Spring Boot é baseado em Java, então você precisa ter uma versão compatível instalada. Você pode baixar o Java JDK no site oficial da Oracle ou acessando esse link 

link : <https://www.oracle.com/br/java/technologies/downloads>



2.3- Tenha uma IDE com suporte, nos usamos e recomendamos o IntelliJ IDEA

link para baixar 

<https://www.jetbrains.com/idea/download>

2.4 - Baixe o projeto e abra na sua IDE, aceite e baixe todas as dependências do projeto, é importante baixar corretamente as dependências para o funcionamento correto da aplicação

3.0 : AS CLASSES E A ARQUITETURA

As classes escolhidas foram clientes, pedido, produto e estoque como é possível observar

a imagem do lado e vamos descrevê-las mais detalhadamente sobre a escolha de atributos de cada classe logo abaixo, a escolha da arquitetura das pastas foi algo parecido com o que se vê em MVC(model , view, controller) porém sem o view uma vez que essa aplicação não possui front-end

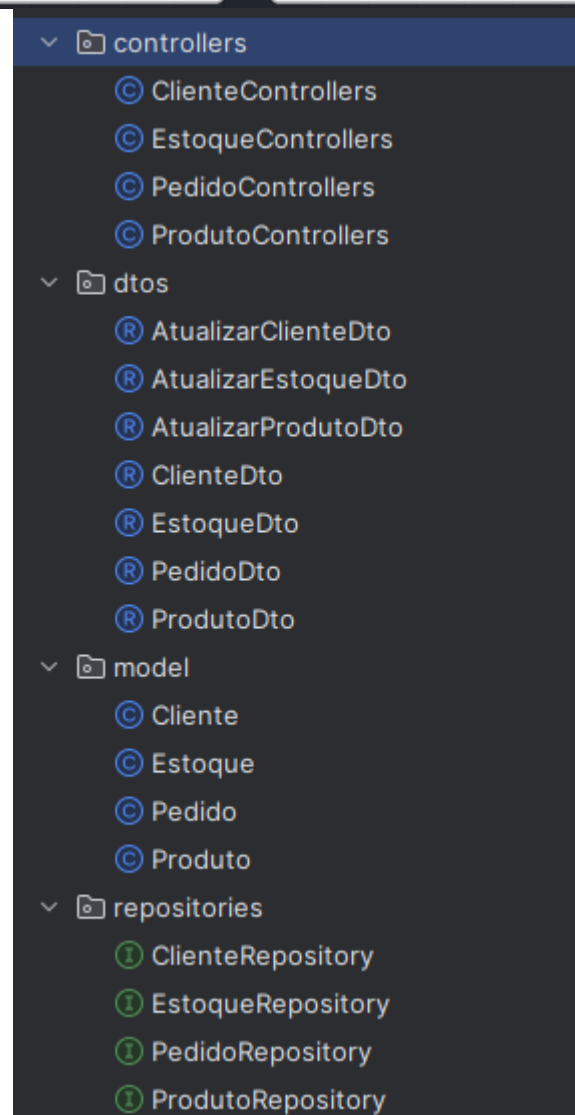
FIGURA 2 - A ARQUITETURA

Assim como no modelo MVC, o modelo é a camada responsável por representar os dados da aplicação, definindo que vai ser uma tabela seus atributos.

Além disso, o projeto possui DTOs que é um padrão de design de software usado para transportar dados entre diferentes camadas de uma aplicação. Ele encapsula os dados de forma organizada e eficiente, separando a lógica de negócio da camada de apresentação.

Além disso tem as repositories, que é uma interface que representa um repositório de dados. Ele é usado para acessar e manipular dados em um banco de dados ou em outra fonte de dados externa, dele vem métodos úteis como save(), findById(), findAll(), update() e delete() esses métodos serão úteis para fazer os controllers que como o próprio

FIGURA 1 - AS CLASSES

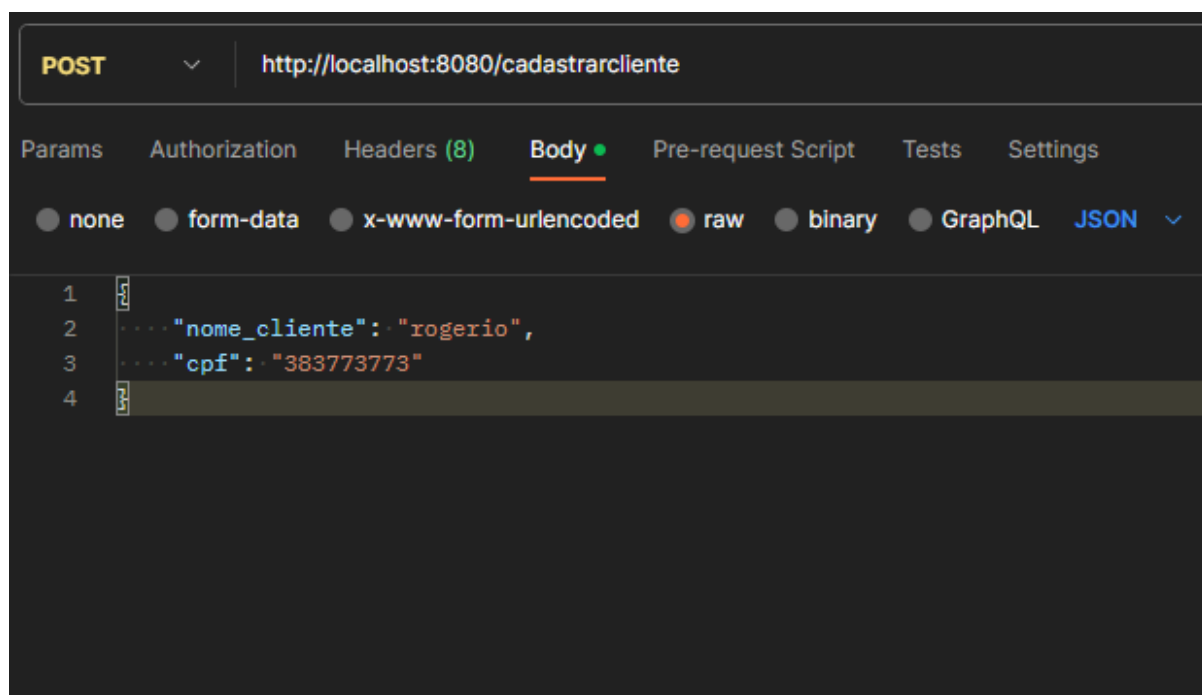


4.0 COMO FAZER AS REQUISIÇÕES

abaixo segue como fazer a requisições no Postman, porém é recomendado que você baixe o arquivo com todas as requisições já prontas-

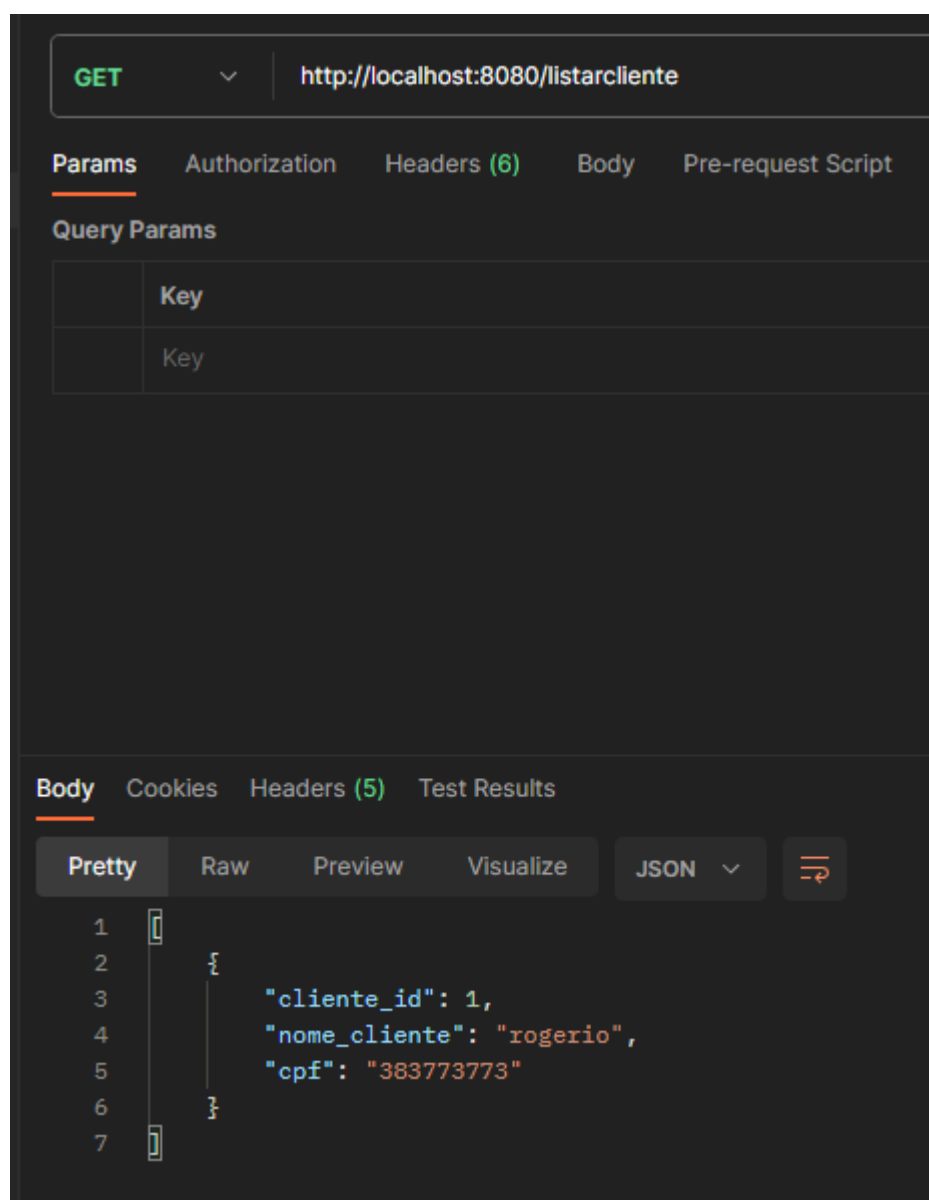
4.1 CLIENTE

FIGURA 3 - CADASTRO CLIENTE



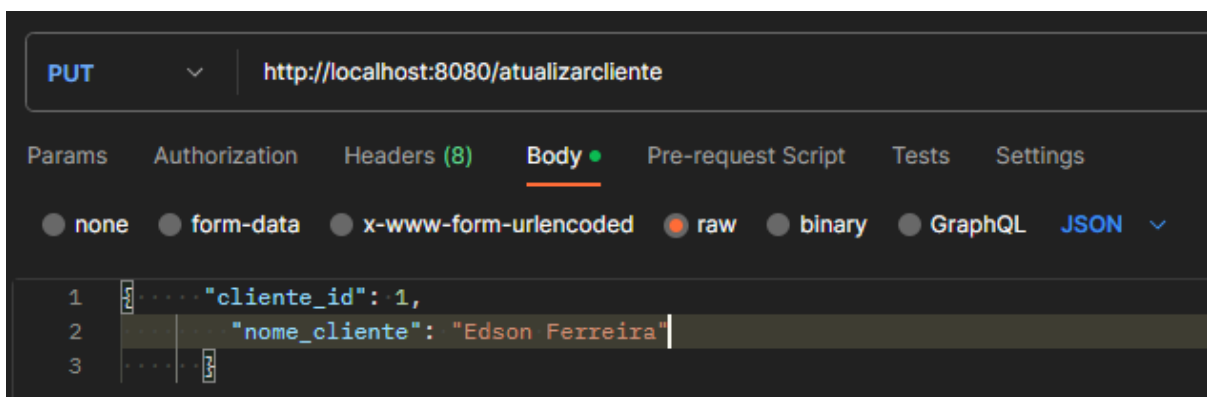
URL : `http://localhost:8080/cadastrarclien`

FIGURA 4 - LISTAR CLIENTE



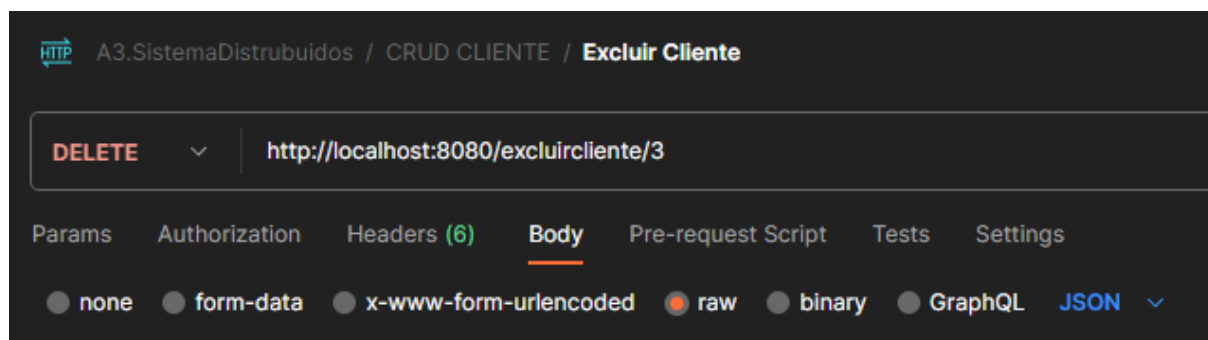
URL : `http://localhost:8080/listarcliente`

FIGURA 4 - ATUALIZAR CLIENTE



URL : `http://localhost:8080/atualizarcliente`

FIGURA 4 - EXCLUIR CLIENTE

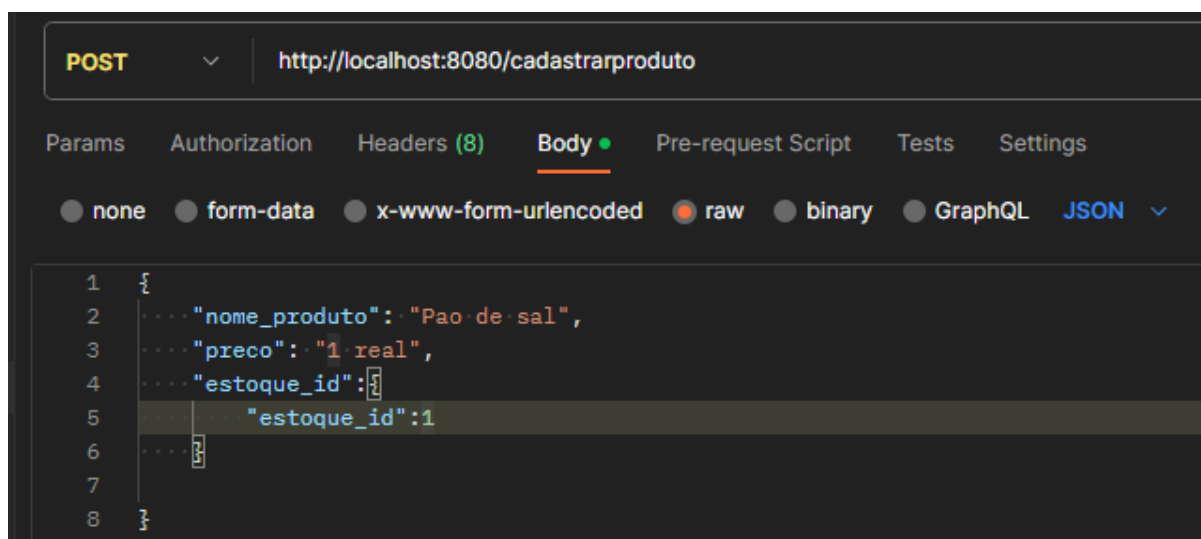


URL : `http://localhost:8080/excluircliente/id'`

obs: é preciso passar o id do pedido que deseja ser deletado na URL

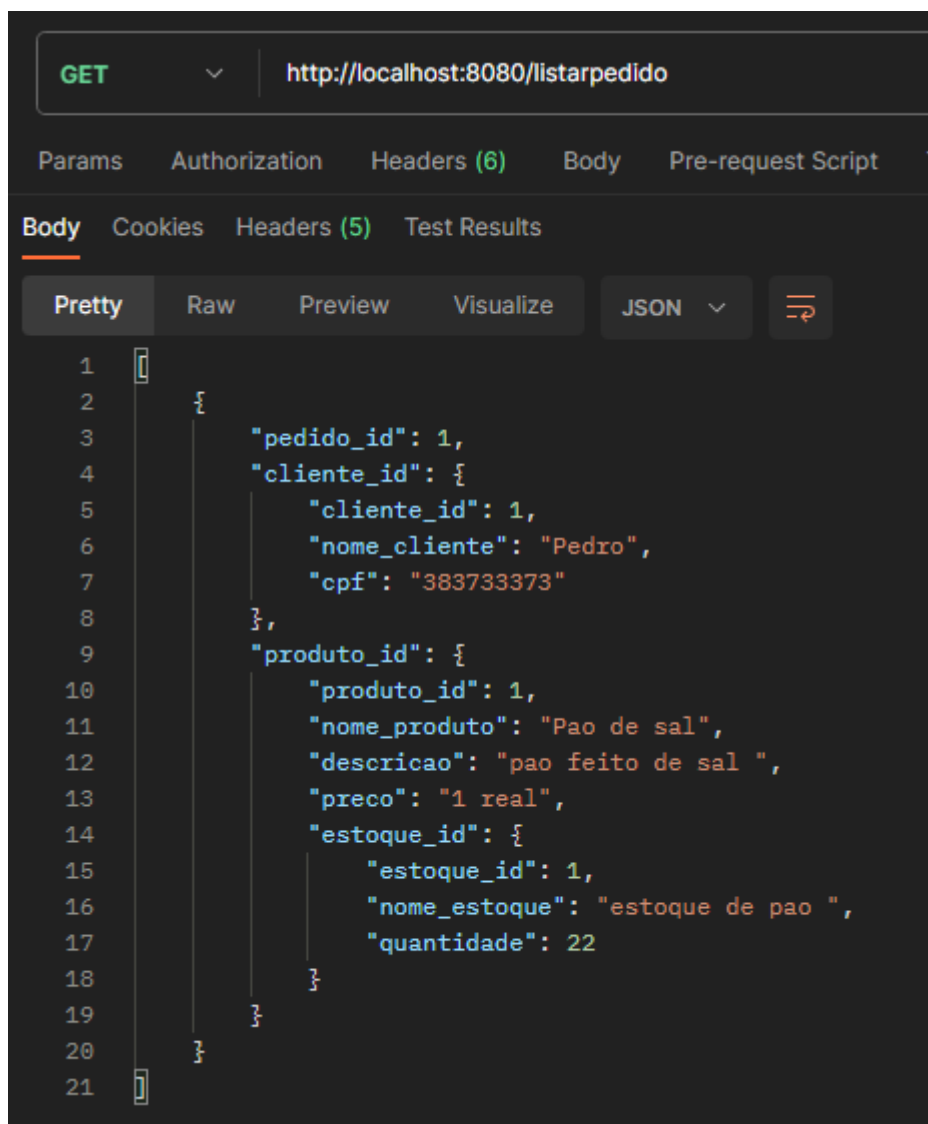
4.2 PRODUTO

FIGURA 4 - CADASTRAR PRODUTO



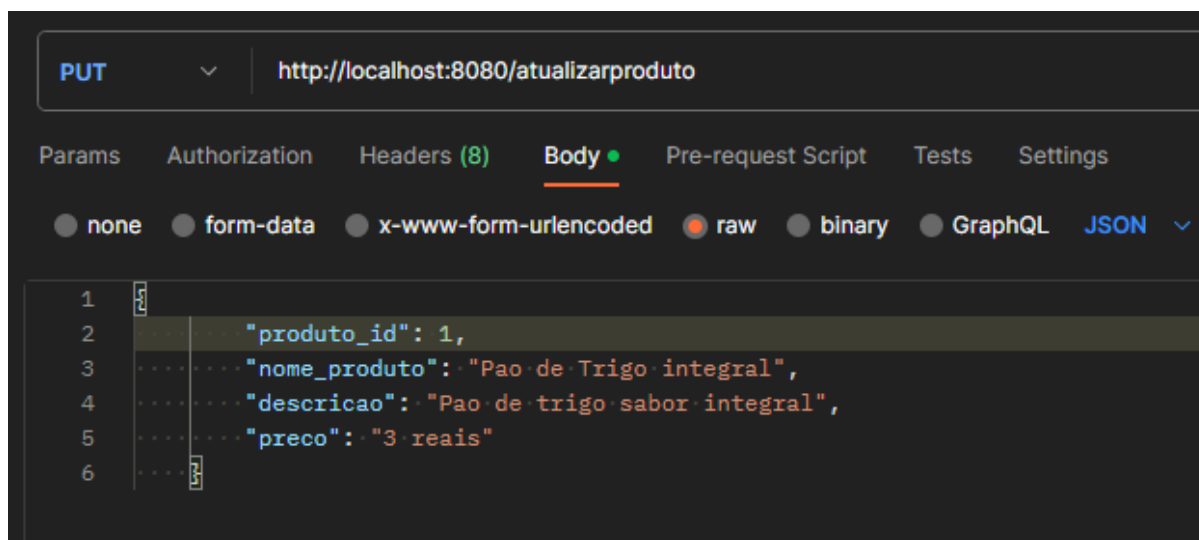
URL : `http://localhost:8080/cadastrarproduto`

FIGURA 4 - LISTAR PRODUTO



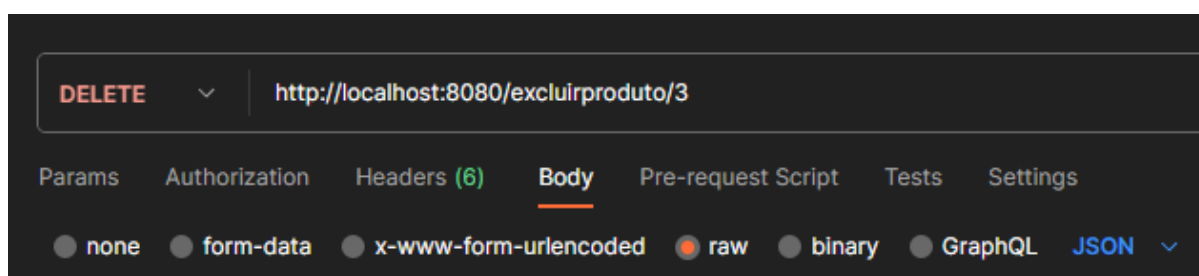
URL : `http://localhost:8080/listarproduto`

FIGURA 4 - ATUALIZAR PRODUTO



URL : `http://localhost:8080/atualizarproduto`

FIGURA 4 - EXCLUIR PRODUTO

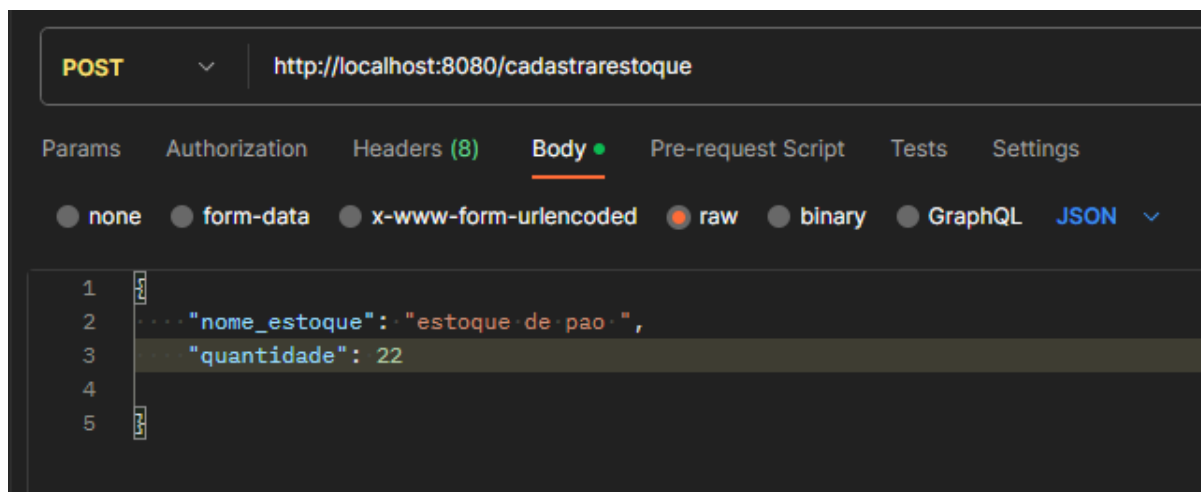


URL : `http://localhost:8080/excluirproduto/id`

obs: é preciso passar o id do pedido que deseja ser deletado na URL

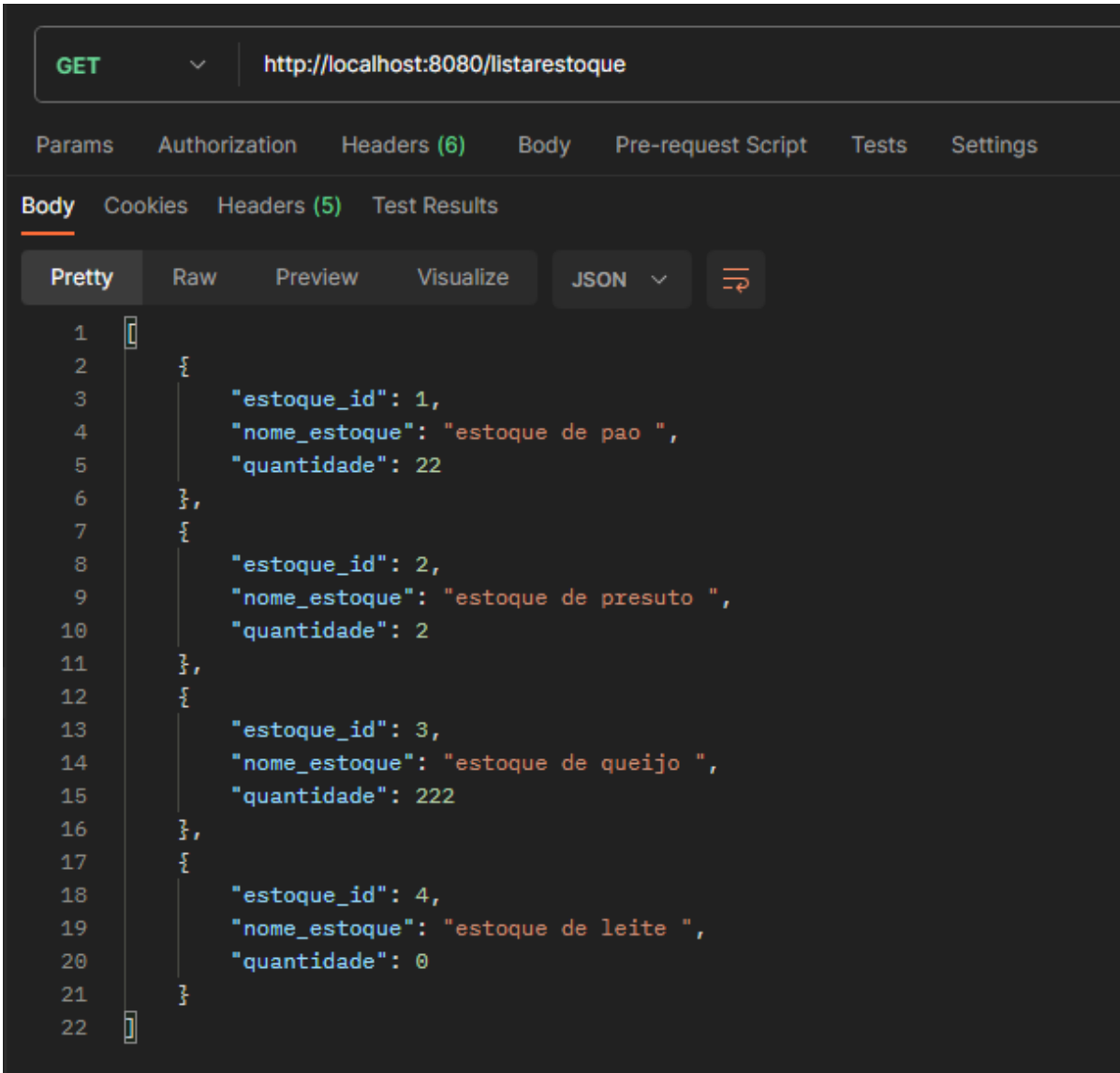
4.3 ESTOQUE

FIGURA 4 - CADASTRAR ESTOQUE



URL : `http://localhost:8080/cadastrarestoque`

FIGURA 4 - LISTAR ESTOQUE

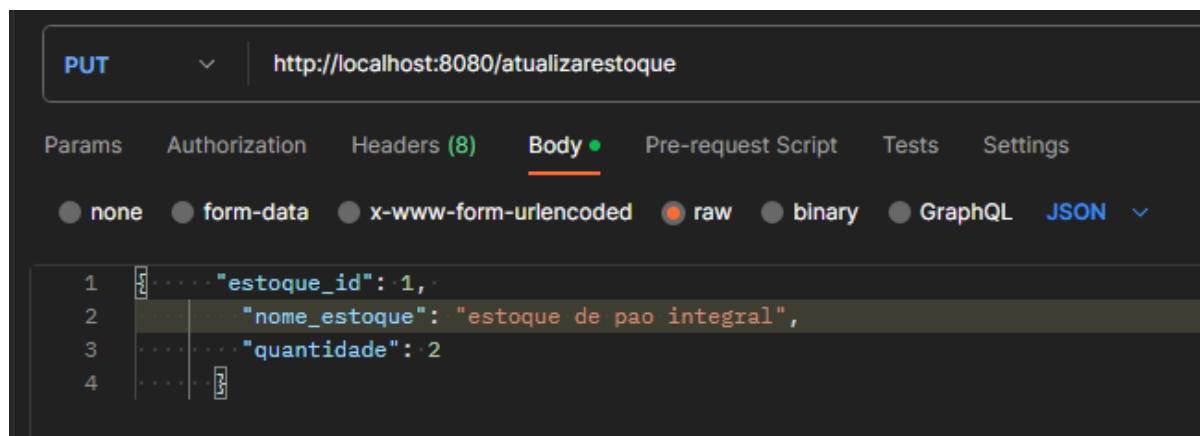


The screenshot shows a web browser's developer tools interface. At the top, a GET request is shown to the URL `http://localhost:8080/listarestoque`. Below this, the 'Body' tab is selected, displaying the response in JSON format. The JSON data is a list of four objects, each representing an inventory item with its ID, name, and quantity.

```
1  [
2    {
3      "estoque_id": 1,
4      "nome_estoque": "estoque de pao ",
5      "quantidade": 22
6    },
7    {
8      "estoque_id": 2,
9      "nome_estoque": "estoque de presunto ",
10     "quantidade": 2
11   },
12   {
13     "estoque_id": 3,
14     "nome_estoque": "estoque de queijo ",
15     "quantidade": 222
16   },
17   {
18     "estoque_id": 4,
19     "nome_estoque": "estoque de leite ",
20     "quantidade": 0
21   }
22 ]
```

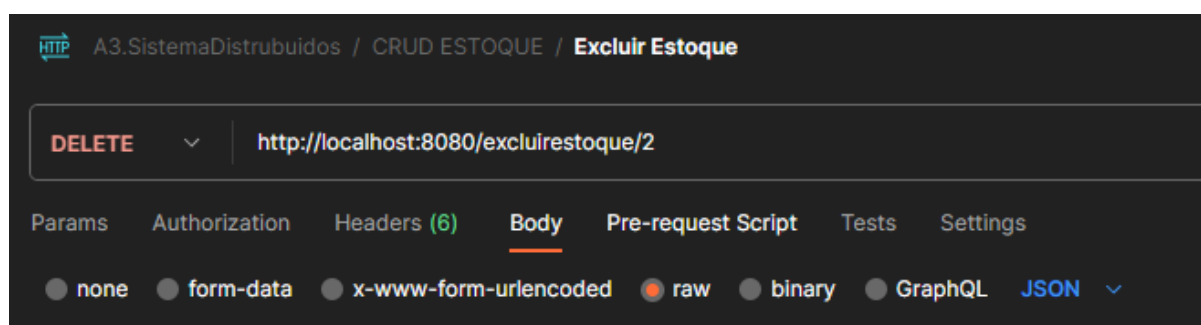
URL : `http://localhost:8080/listarestoque`

FIGURA 4 - ATUALIZAR ESTOQUE



URL : `http://localhost:8080/atualizarestoque`

FIGURA 4 - EXCLUIR ESTOQUE

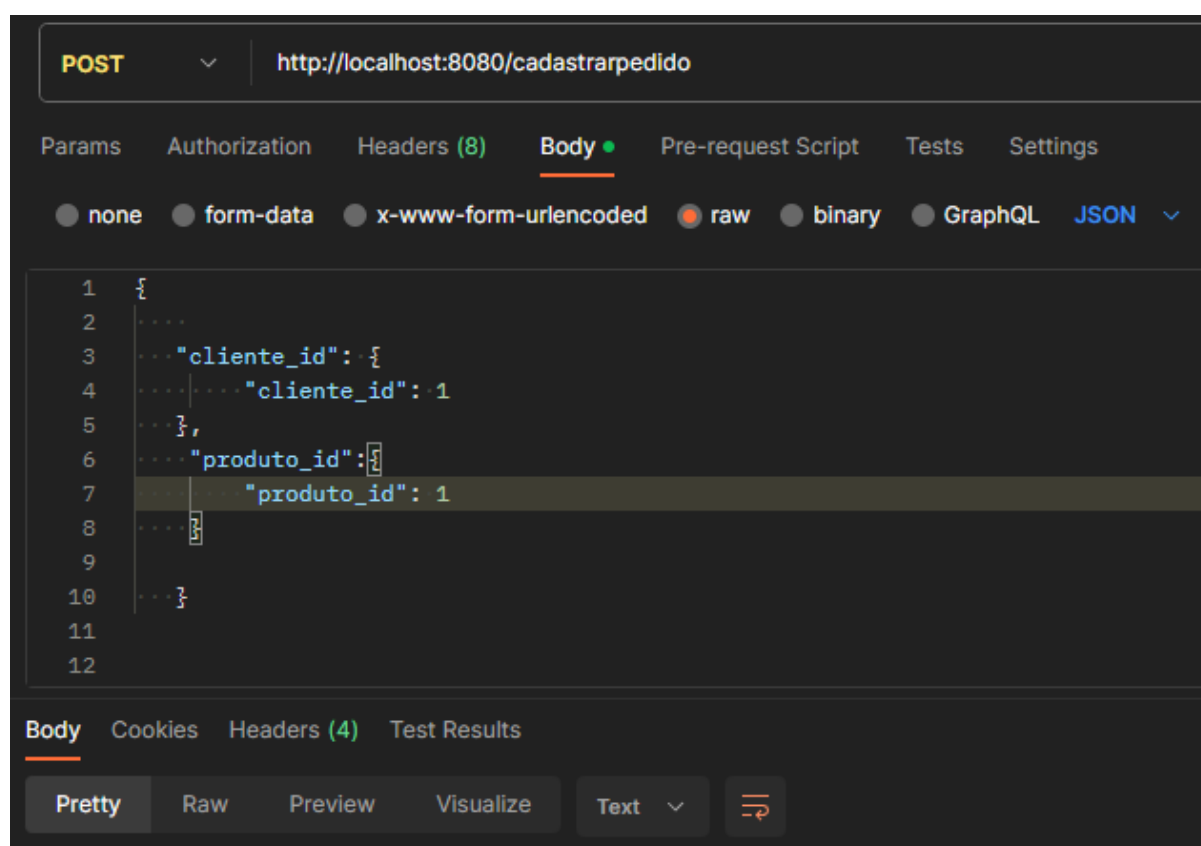


URL : `http://localhost:8080/excluirestoque/id`

obs: é preciso passar o id do pedido que deseja ser deletado na URL

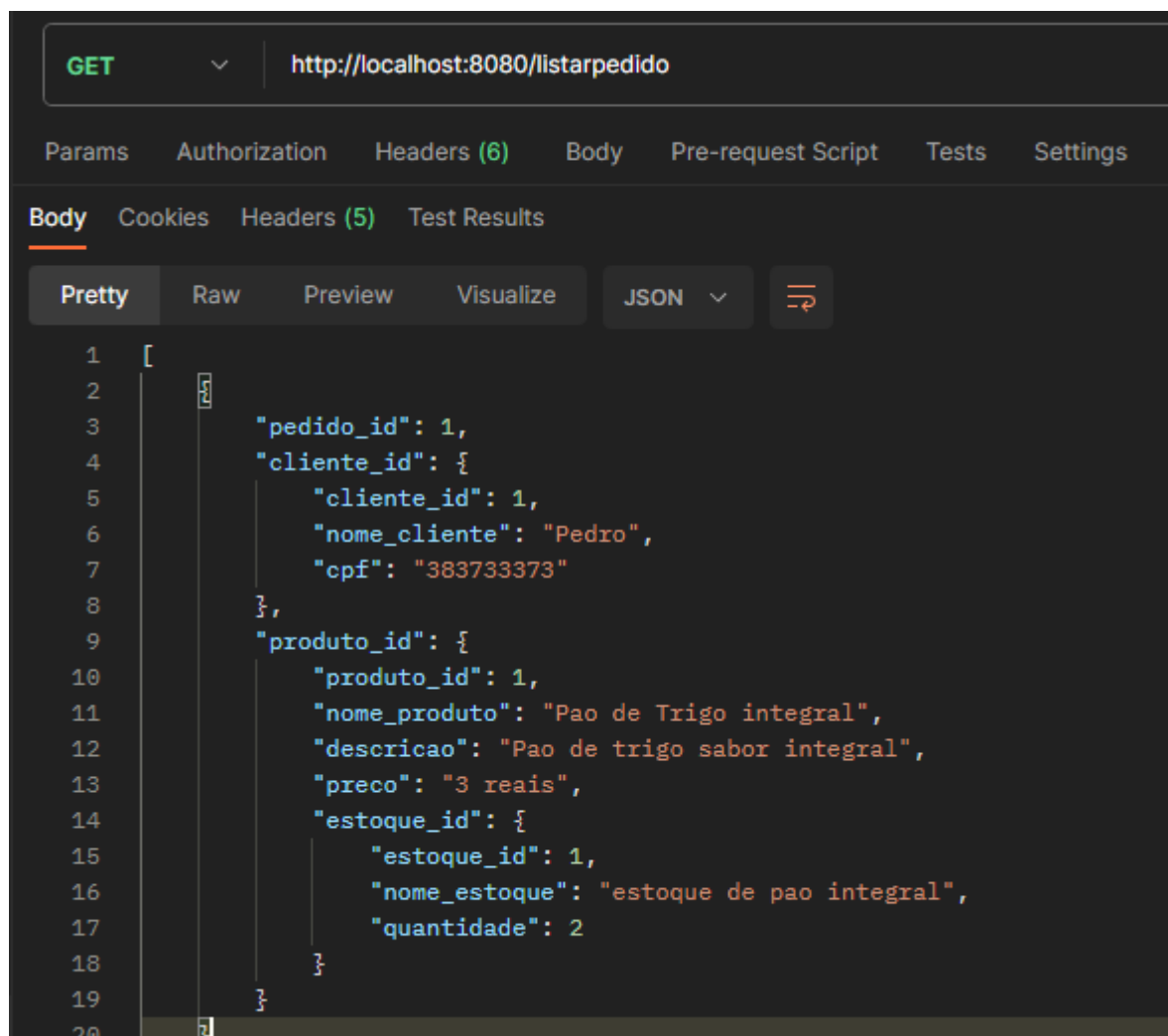
4.4 PEDIDO

FIGURA 4 - CADASTRAR PEDIDO



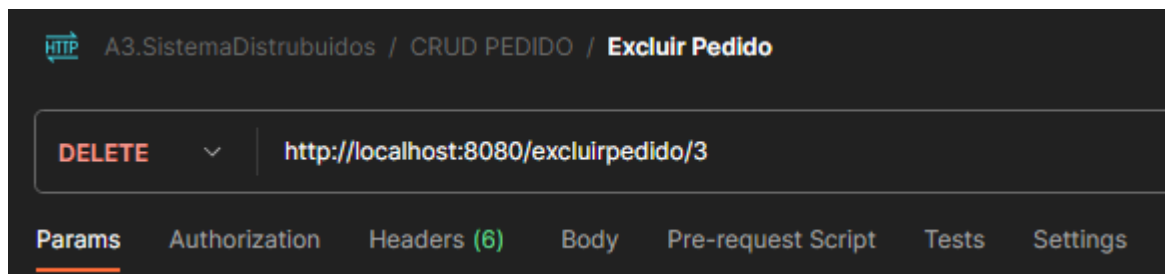
URL : `http://localhost:8080/cadastrarpedido`

FIGURA 4 - LISTAR PEDIDO



URL : `http://localhost:8080/listarpedido`

FIGURA 4 - EXCLUIR PEDIDO



URL : `http://localhost:8080/excluirpedido/id`

obs: é preciso passar o id do pedido que deseja ser deletado na URL