



Universidade Federal do Ceará-Campus Quixadá

Disciplina de Sistemas Distribuídos

Docente: Prof. Dr. Rafael Braga

Davi Medeiros

Lucas Nóbrega

Relatório

API RESTful para Gerenciamento de Estoque

Quixadá

2025

1. Introdução

Este documento apresenta a implementação de uma API RESTful desenvolvida em Spring Boot para gerenciamento de estoque de produtos. A solução oferece operações básicas de CRUD (Create, Read, Update, Delete) através de serviços remotos acessíveis via protocolo HTTP, com troca de dados no formato JSON.

2. Objetivos

- Disponibilizar serviços remotos para cadastro e consulta de produtos
- Implementar padrão RESTful para garantia de interoperabilidade
- Manter arquitetura modular e escalável
- Facilitar a integração com diferentes clientes (web, mobile, outros sistemas)

3. Serviços Implementados

3.1. Cadastro de Produto

Método: POST /produtos

Descrição: Adicionar novo item ao estoque

Entrada (JSON):

```
{  
  "id": 1,  
  "nome": "Teclado",  
  "quantidade": 50,  
  "preco": 199.90  
}
```

Respostas: 200 OK (sucesso), 400 Bad Request (dados inválidos)

3.2. Listagem de Produtos

Método: GET /produtos

Descrição: Retorna todos os itens cadastrados

Resposta (JSON):

```
{  
  "id": 1,  
  "nome": "Teclado",  
  "quantidade": 50,  
  "preco": 199.90  
}
```

3.3. Consulta por ID

Método: GET /produtos/{id}

Descrição: Retorna produto específico

Respostas: 200 OK + JSON do produto, 404 Not Found (ID inexistente).

3.4. Atualização de Produto

Método: PUT /produtos/{id}

Descrição: Altera dados de produto existente

Entrada (JSON):

```
{  
  "nome": "Teclado Sem Fio",  
  "quantidade": 30,  
  "preco": 249.90  
}
```

Respostas: 204 No Content (sucesso), 404 Not Found (ID inexistente).

3.5. Remoção de Produto

Método: DELETE /produtos/{id}

Descrição: Remover item do estoque

Respostas: 200 OK + mensagem de confirmação, 404 Not Found (ID inexistente).

4. Arquitetura do Sistema

4.1. Camada de Modelo

Classe Produto.java:

Atributos: 'id', 'nome', 'quantidade', 'preco'

Getters/Setters

Construtores

Método toString()

4.2. Camada de Serviço

Interface EstoqueService.java:

Contrato com métodos CRUD

Classe EstoqueServiceImpl.java:

Implementação com ArrayList

Lógica de negócio

Anotação @Service

4.3. Camada de Controlador

Classe ProdutoController.java:

Anotação @RestController

Mapeamento @RequestMapping("/produtos")

Métodos com @GetMapping, @PostMapping, etc.

Uso de ResponseEntity

4.4. Configuração

application.properties:

server.port=8080

Configurações básicas

5. Fluxo de Processamento

1. Cliente envia requisição HTTP
2. Controller recebe e valida requisição
3. Service executa lógica de negócio
4. Resposta é retornada ao cliente

8. Conclusão

A API desenvolvida atende aos requisitos básicos de um sistema de gestão de estoque, seguindo as melhores práticas de desenvolvimento RESTful. A arquitetura em camadas permite fácil manutenção e expansão, sendo preparada para evoluções futuras.

Responsável: [Davi Medeiros, Lucas Nóbrega]

Data: [20/07/25]