ANA CAMILA FONSECA SOARES

**DESAFIO TÉCNICO**

**VERTICAL LOGÍSTICA**

EXTREMA – MG

2025

ANA CAMILA FONSECA SOARES

**DESAFIO TÉCNICO**

**VERTICAL LOGÍSTICA**

Desafio Técnico para a Empresa Vertical Logística, com foco na integração de dois sistemas, utilizando a linguagem Java, Spring Boot e PostgreSQL, usando API REST.

Orientação: Vinicius Mussak | LuizaLabs

EXTREMA – MG

2025

**SUMÁRIO**

[PREPARAÇÃO DO AMBIENTE 4](#_Toc198485781)

[CRIAÇÃO DO SPRING BOOT 4](#_Toc198485782)

[CONFIGURAÇÃO DO BANCO DE DADOS POSTGRESQL 7](#_Toc198485783)

[Configurações no java 8](#_Toc198485784)

[Modelagem das Entidades JPA 8](#_Toc198485785)

[Criação da camada de repositórios 9](#_Toc198485786)

[TESTES 9](#_Toc198485787)

[DEPOIS DA CRIAÇÃO DAS DTOS 9](#_Toc198485788)

[Teste final no navegador 17](#_Toc198485789)

# PREPARAÇÃO DO AMBIENTE

* Instalação IDE Eclipse
* Instalação do PostgreSQL
* Instalação do Maven – Gerenciador de Dependências:

<https://www.youtube.com/watch?v=-ucX5w8Zm8s>

[Clique aqui para instalar](https://maven.apache.org/download.cgi)

**Verificação do Ambiente:**

Texto

O conteúdo gerado por IA pode estar incorreto.

Figura 1- Verificação da instalação do Maven e versão do Java

# CRIAÇÃO DO SPRING BOOT

Existem várias opções para criar uma **API REST** em Java, mas a escolha foi o uso do framework Spring Boot que possui uma abordagem simplificada e rápida para desenvolvimento.

**Facilidade de Uso:** O Spring Boot simplifica a configuração e o desenvolvimento de aplicações Java, permitindo a criação da APIs REST rapidamente.

**Anotações:** Utiliza anotações como @RestController, @GetMapping, @PostMapping, etc., para definir endpoints de forma clara e concisa.

**Integração com Banco de Dados:** Suporta facilmente a integração com bancos de dados através do Spring Data JPA.

**Segurança:** Integração com Spring Security para proteger suas APIs.

**Acesse o site Spring Initializr:**

<https://start.spring.io/>

**Configurações do projeto:**

Project: Maven Project.

Language: Java.

Spring Boot: versão 3.4.5

Project Metadata:

Group: identificador do grupo: com.vert.logistica

Artifact: Nome do projeto: pedidoapi

Packaging: Jar

Java: 24 (pois a versão do meu java é 22)

Tela de celular com aplicativo aberto

O conteúdo gerado por IA pode estar incorreto.

Figura 2 - Configurações realizadas do spring boot

**Dependências adicionadas**

* Spring Web (para criar a API REST)
* Spring Data JPA (para interagir com o banco de dados)
* PostgreSQL Driver (para conectar ao PostgreSQL)
* Lombok (para reduzir código boilerplate)
* Spring Boot DevTools (para facilitar desenvolvimento)

Tela de computador com texto preto sobre fundo branco

O conteúdo gerado por IA pode estar incorreto.

Figura 3 - Dependências inseridas no spring boot.

Depois de gerar o *spring boot*, descompactei o arquivo, inseri no diretório do eclipse workspace e abri na IDE eclipse como Maven, que é o gerenciador de dependências.

Interface gráfica do usuário, Texto

O conteúdo gerado por IA pode estar incorreto.

Figura 4 - Inserção do projeto spring boot na IDE Eclipse via Maven

# CONFIGURAÇÃO DO BANCO DE DADOS POSTGRESQL

Criei o Banco de Dados com o nome de pedidos. Certifique na imagem abaixo:

Interface gráfica do usuário, Aplicativo, Word

O conteúdo gerado por IA pode estar incorreto.

**Configuração do arquivo Java application.properties**

**Informações do Banco de dados pelo sql shell**

Texto

O conteúdo gerado por IA pode estar incorreto.

Figura 5 - Informações do Banco de dados para configuração

# Configurações no java

Texto

O conteúdo gerado por IA pode estar incorreto.

# Modelagem das Entidades JPA

Criei uma classe chamada Pedidos para representar os dados do banco.

Declarei as variáveis idUsuario, idPedido e idProduto como Long pois podemos declarar esses números como literais ou eu poderia ter colocado como String, que não foi o caso.

Texto

O conteúdo gerado por IA pode estar incorreto.

Figura 6 - Print da Classe Pedidos

# Criação da camada de repositórios

No projeto Spring Boot com JPA, a **Camada de Repositórios** é responsável por gerenciar o acesso aos dados no banco — ela atua como uma interface entre sua aplicação e o banco de dados. Para criar essa camada, você cria interfaces que estendem as interfaces do Spring Data JPA, como a **JpaRepository**. O Spring Data JPA automaticamente implementa essas interfaces para você, fornecendo os métodos básicos para CRUD (Create, Read, Update, Delete) e ainda permite definir consultas personalizadas facilmente.

# TESTES

**Comandos do CMD:**

curl -X POST -F "arquivo=@C:\\Users\\anaca\\Desktop\\luizaLabsArquivos\data\_1.txt" http://localhost:8080/api/pedidos/importarArquivo

curl http://localhost:8080/api/pedidos

# DEPOIS DA CRIAÇÃO DAS DTOS

**Teste no prompt de comando:**

curl http://localhost:8080/api/pedidos/listarUsuariosComPedidos

**Teste no navegador:**

http://localhost:8080/api/pedidos/listarUsuariosComPedidos

**Desafios enfrentados:**

Os pedidos não estavam sendo salvos corretamente no banco de dados. Consegui importar o Arquivo, mas por algum motivo não consegui salvar no banco.

Após a execução do método **processarArquivo**, recebi uma lista vazia ( **[ ]**), isso pode indicar que os pedidos não estão sendo salvos corretamente no banco de dados.

**Prints dos erros e logo após as correções**

Tela de computador com texto preto sobre fundo branco

O conteúdo gerado por IA pode estar incorreto.

QUASE LÁ: Análise das posições do arquivo:

Uma imagem contendo Interface gráfica do usuário

O conteúdo gerado por IA pode estar incorreto.

0000000046 Vanetta Bogan00000004840000000004 332.3820211001

**Finalmente, posições entendidas e corrigidas**

Interface gráfica do usuário

O conteúdo gerado por IA pode estar incorreto.

**Teste para resolução do próximo erro:**

Tela de computador com texto preto sobre fundo branco

O conteúdo gerado por IA pode estar incorreto.

Inseri um id automático como chave primária. Antes, eu havia colocado como chave primária o idUsuario, mas tem duplicidade no sistema desnormalizado. Apaguei a tabela do banco para ser criada novamente com essa inserção.

**Banco recriado**

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

**No prompt de comando, o arquivo foi importado e salvo**

Texto

O conteúdo gerado por IA pode estar incorreto.

**No Spring Boot mostrou que salvou 2352 linhas o arquivo:**

Tela de computador com texto preto sobre fundo branco

O conteúdo gerado por IA pode estar incorreto.

**No navegador, antes de aplicar o estilo de formatação:**

Texto

O conteúdo gerado por IA pode estar incorreto.

**No navegador, depois de aplicar o estilo de formatação:**

Texto

O conteúdo gerado por IA pode estar incorreto.

Inserção no Banco de dados

Interface gráfica do usuário, Tabela

O conteúdo gerado por IA pode estar incorreto.

**Instalação opcional de uma extensão json no navegador**

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

**Visual mais apresentável**

Texto

O conteúdo gerado por IA pode estar incorreto.

Obs: Retirei essa extensão, pois estava conflitando com a estrutura de resposta de saída.

**Teste Filtro**

Texto

O conteúdo gerado por IA pode estar incorreto.

**Certificação no Banco**

Tabela

O conteúdo gerado por IA pode estar incorreto.

**Teste do filtro de data**

Texto

O conteúdo gerado por IA pode estar incorreto.

# Teste final no navegador

<http://localhost:8080/api/pedidos>

Texto

O conteúdo gerado por IA pode estar incorreto.

Abaixo, a inserção de DTOs no projeto:

O uso de DTOs (Data Transfer Objects) é uma prática comum e recomendada para separar a lógica de negócios da estrutura de dados para expor na API.

O link abaixo demonstra o arquivo json com a estrutura de saída payload de response pedida:

<http://localhost:8080/api/pedidos/listarUsuariosComPedidos>

Texto

O conteúdo gerado por IA pode estar incorreto.

Texto

O conteúdo gerado por IA pode estar incorreto.