Projeto Integrador - Bootcamp Mercado Livre

MERCADO LIVRE - FRESCOS

Tecnologias utilizadas

:java: **Java**

:spring: **SpringBoot**

:maven: **Maven**

:docker: **Docker**

:mysql: **MySQL**

:junit: **Junit**

:jenkins: **Jenkins**

:pipeline: **Pipeline**

:sonar: **Sonarqube**

:jacoco: **Jacoco**

:postman: **Postman**

:redis: Redis

Tópicos

- **Descrição:** API de produtos frescos realizando toda logística de disponibilidade de compra e venda
- **Funcionalidade:** Comprar produtos adicionando a um carrinho e também vender
- **Deploy da Aplicação:** Push do branch master que se faz no repositório Git ou deploy automatizado é o Jenkins.
- **Pré-requisitos:** Ter devidamente instalado o JDK e JRE
- **Como rodar a aplicação:** docker-compose --file docker-compose.dev.yml up

---

Descrição do projeto

Construção de API para realizar a logística de produtos alimentícios em estado de congelados, refrigerados, fresco para serem armazenados em seus determinados setores com organização de volumes e sua respectiva venda em carrinho com suas ordens logística

---

Funcionalidades

Representante do armazém de distribuição, inseri um lote de produtos no armazém de distribuição para registrar a existência de estoque
Comprador ativo adiciona produtos ao carrinho de compras do marketplace para compra-los, se desejar
Representante pode consultar um produto em stock no armazém para saber a sua localização num setor e os diferentes lotes onde se encontra
Representante pode consultar um produto em todos os armazéns para saber o estoque em cada armazém do referido produto
Representante pode consultar os produtos em estoque que estão prestes a expirar no almoxarifado, afim de aplicar alguma ação comercial com eles

---

Layout ou Deploy da Aplicação

EndPoint REQ1	EndPoint REQ2	EndPoint REQ3	EndPoint REQ4	EndPoint REQ5	EndPoint REQ6
http://localhost:8080/api/v1/fresh-products/inboundorder/	http://localhost:8080/api/v1/fresh-products/	http://localhost:8080/api/v1/fresh-products/list?querytype=[idProducto]	http://localhost:8080/api/v1/fresh-products/warehouse/querytype=id product]	http://localhost:8080/api/v1/fresh-products/due-date/queryparam=[number of days] queryparam=[section]	http://localhost:8080/api/v1/fresh-products/buyer/[id]
http://localhost:8080/api/v1/fresh-products/inboundorder/	http://localhost:8080/api/v1/fresh-products/list?querytype=[categoría producto]	http://localhost:8080/api/v1/fresh-products/list?querytype=[idProducto] querytype=[L]		http://localhost:8080/api/v1/fresh-products/due-date/list?queryparam=[number of days] queryparam=[category] queryparam=[asc]	http://localhost:8080/api/v1/fresh-products/buyer?queryparam[id] &queryparam=[status]
	http://localhost:8080/api/v1/fresh-products/orders/				http://localhost:8080/api/v1/fresh-products/buyer?queryparam[id] &queryparam[David]&queryparam[email]
	http://localhost:8080/api/v1/fresh-products/orders/querytype=[idOrder]				http://localhost:8080/api/v1/fresh-products/buyer/[id]
	http://localhost:8080/api/v1/fresh-products/orders/query param=[idOrder]				

Pré-requisitos

- Instalação do Java 8 ou versão mais atualizada na maquina
- Utilização de uma IDE para start do serviço
- Instalação do Docker desktop
- Instalação do Postman para manuseio dos endpoint

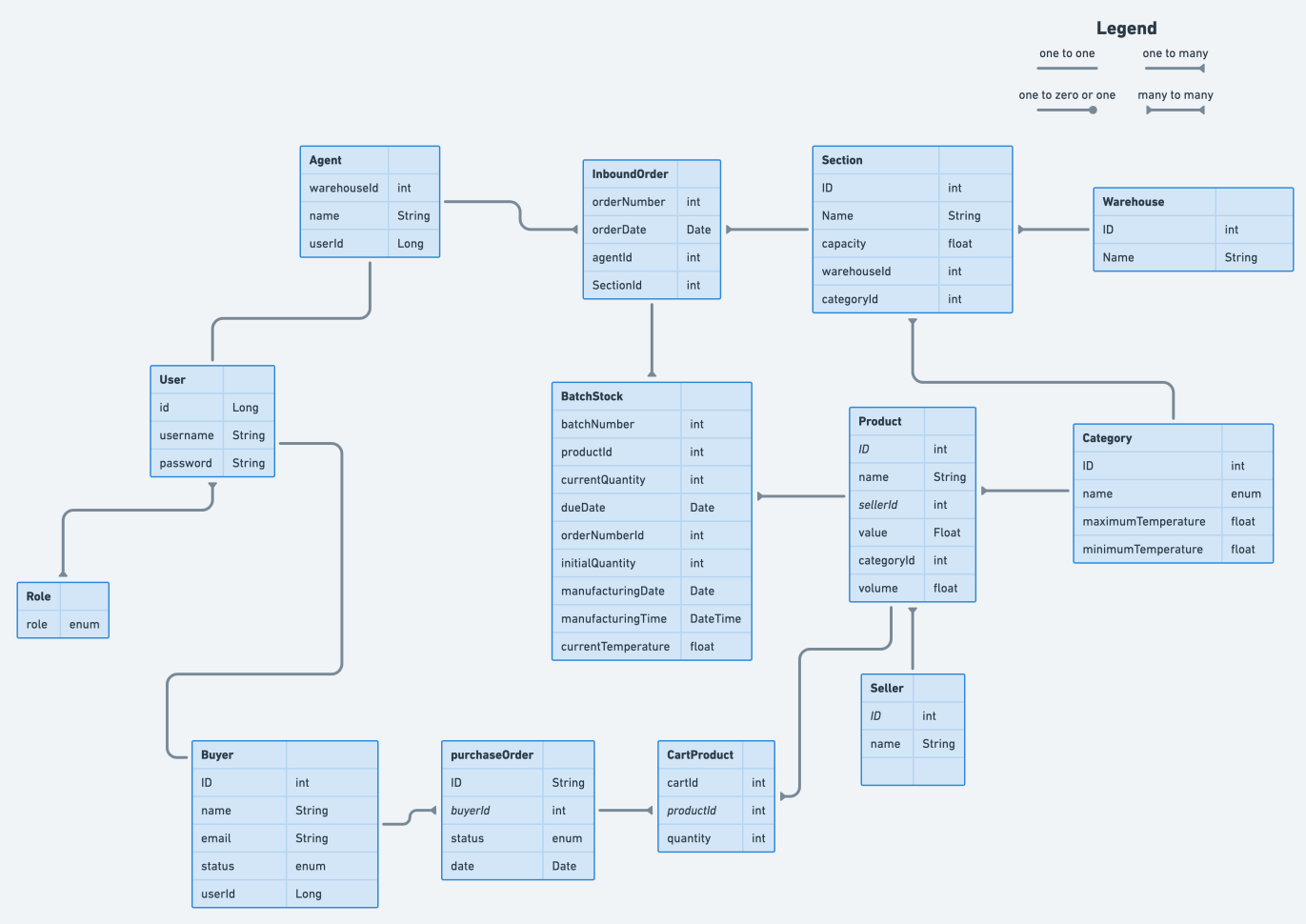
Como rodar a aplicação

```
No terminal, clone o projeto; git clone https ou ssh do projeto:
git clone https://github.com/maik-henrique/DH-Projeto-Integrador.git

Acesse a pasta do projeto via terminal, para iniciar o projeto:
mvn clean install

Em seguida execute o servidor Tomcat
mvn spring-boot:run

Inicie os conteiners referente a aplicacao com docker-compose
docker-compose --file docker-compose.dev.yml up

Inicialize a aplicacao springboot na IDE e acessse os endpoints
utilizando Postman
```

Como rodar os testes

```
Para iniciar os testes em linha de comando utilizar:
mvn tests

Para iniciar os testes no SonarQube
mvn sonar:sonar -Dsonar.projectKey={KEY DO PROJETO CRIADO NO SONAR} -
Dsonar.host.url=http://127.0.0.1:9000 -Dsonar.login={CHAVE DE
AUTENTICACAO DO SONAR}
```

Casos de Classes



Classes de teste referente ao Requisito 6

- Controller/BuyerController.java
- Service/BuyerService.java
- integration/BuyerControllerTest.java

unit/BuyerServiceTest.java

dto/response/BuyerDTO.java

Iniciando/Configurando banco de dados

```
Para iniciar o servico de banco de dados no docker:
docker-compose --file docker-compose.dev.yml up
```

Linguagens, dependencias e libs utilizadas

:java: **Java**

:spring: **SpringBoot**

:maven: **Maven**

:docker: **Docker**

:mysql: **MySQL**

:junit: **Junit**

:jenkins: **Jenkins**

:pipeline: **Pipeline**

:sonar: **Sonarqube**

:jacoco: **Jacoco**

:postman: **Postman**

:redis: Redis

Resolvendo Problemas

Os gadgets só podem ser visualizados nos navegadores

Os gadgets só podem ser visualizados nos navegadores

Os gadgets só podem ser visualizados nos navegadores

Tarefas em aberto

- Implementar microservices
- Expandir logica e regra de negocio
- Subir na cloud para implementar observability

Desenvolvedores

BM @ Bruno Mendes

@ David Alexandre Fernandes

MP @ Maik Henrique dos Santos Pereira

MV @ Mariana Galdino Vieira

MG @ Matheus dos Anjos Guerra

MA @ Micaela da Cruz Alves

PD @ Pedro Henrique Dalpa

Os gadgets só podem ser visualizados nos navegadores