Projeto Digi System

Objetivo: levantar um ambiente

Escolha: Backend de uma loja usando spring, Swagger, mysql

Sumário

[Mysql Queries 1](#_Toc152269037)

[Projeto Java 2](#_Toc152269038)

[Arquivo Pom.xml 2](#_Toc152269039)

[Arquivo application.properties 3](#_Toc152269040)

[Classes 3](#_Toc152269041)

[Classe Customer 3](#_Toc152269042)

[Classe CustomerRepository 4](#_Toc152269043)

[Classe Product 4](#_Toc152269044)

[Classe Purchase 4](#_Toc152269045)

# Mysql Queries

Tabela Cliente:

CREATE TABLE customer (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL

);

Tabela Produto

CREATE TABLE product (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

price DOUBLE NOT NULL

);

Tabela Compra:

CREATE TABLE purchase (

id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_id INT,

FOREIGN KEY (customer\_id) REFERENCES customer(id)

);

Tabela Compra\_Produtos (Relação Many to Many):

CREATE TABLE purchase\_products (

purchase\_id INT,

product\_id INT,

PRIMARY KEY (purchase\_id, product\_id),

FOREIGN KEY (purchase\_id) REFERENCES purchase(id),

FOREIGN KEY (product\_id) REFERENCES product(id)

);

# Projeto Java

## Arquivo Pom.xml

O arquivo pom.xml define as dependências do projeto

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>org.example</groupId>  
 <artifactId>DigiSystem</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <properties>  
 <java.version>18</java.version>  
 <spring.boot.version>2.6.1</spring.boot.version>  
 </properties>  
  
 <repositories>  
 <repository>  
 <id>central</id>  
 <url>https://repo.maven.apache.org/maven2</url>  
 </repository>  
 </repositories>  
  
 <dependencies>  
 <!-- Spring Boot Starter Web -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 <!-- <version>${spring.boot.version}</version>-->  
 </dependency>  
  
 <!-- Spring Boot Starter Data JPA -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 <!-- <version>${spring.boot.version}</version>-->  
 </dependency>  
  
  
 <!-- MySQL Connector Java -->  
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 </dependency>  
  
 <!-- Lombok -->  
 <dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <version>1.18.22</version>  
 <scope>provided</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
</project>

## Arquivo application.properties

O arquivo application.properties existe para definir propriedades como portas, acesso a banco etc

server.port = 8081  
  
# Configurações do banco de dados MySQL  
spring.datasource.url=jdbc:mysql://localhost:3306/local  
spring.datasource.username=root  
spring.datasource.password=root  
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  
  
  
# Configurações JPA  
spring.jpa.hibernate.ddl-auto=update  
  
spring.jpa.show-sql=true

## Classes

### Classe Customer

import javax.persistence.\*;

@Entity

@Data

public class Customer {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String email;

}

### Classe CustomerRepository

import org.springframework.data.jpa.repository.JpaRepository;

public interface CustomerRepository extends JpaRepository<Customer, Long> {

}

### Classe Product

import lombok.Data;

import javax.persistence.\*;

@Entity

@Data

public class Product {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private double price;

}

### Classe Customer Service

package com.digisystem.luiz.service;  
  
import com.digisystem.luiz.model.Customer;  
import com.digisystem.luiz.repository.CustomerRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
  
@Service  
public class CustomerService {  
  
 @Autowired  
 private CustomerRepository customerRepository;  
  
 public List<Customer> getAllCustomers() {  
 return customerRepository.findAll();  
 }  
  
 public Customer getCustomerById(Long id) {  
 return customerRepository.findById(id).orElse(null);  
 }  
  
 public void saveCustomer(Customer customer) {  
 customerRepository.save(customer);  
 }  
  
 public void deleteCustomer(Long id) {  
 customerRepository.deleteById(id);  
 }  
}

@Transactional  
public void update(Long id, Product updatedProd) {  
 productRepository  
 .findById(id) // returns Optional<User>  
 .ifPresent(prod -> {  
 prod.setName(updatedProd.getName());  
 prod.setPrice(updatedProd.getPrice());  
  
  
 productRepository.save(prod);  
 });  
}

### Classe Product Service

package com.digisystem.luiz.service;  
  
import com.digisystem.luiz.model.Product;  
import com.digisystem.luiz.repository.ProductRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
  
@Service  
public class ProductService {  
  
 @Autowired  
 private ProductRepository productRepository;  
  
 public List<Product> getAllProducts() {  
 return productRepository.findAll();  
 }  
  
 public Product getProductById(Long id) {  
 return productRepository.findById(id).orElse(null);  
 }  
  
 public void saveProduct(Product product) {  
 productRepository.save(product);  
 }  
  
 public void deleteProduct(Long id) {  
 productRepository.deleteById(id);  
 }  
}

@Transactional  
public void update(Long id, Customer updatedCust) {  
 customerRepository  
 .findById(id) // returns Optional<User>  
 .ifPresent(cust-> {  
 cust.setName(updatedCust.getName());;  
 cust.setEmail(updatedCust.getEmail());;  
  
  
 customerRepository.save(cust);  
 });  
}

### Classe Customer Controller

package com.digisystem.luiz.controladores;  
  
import com.digisystem.luiz.model.Customer;  
import com.digisystem.luiz.service.CustomerService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/customers")  
public class CustomerController {  
  
 @Autowired  
 private CustomerService customerService;  
  
 @GetMapping  
 public List<Customer> getAllCustomers() {  
 return customerService.getAllCustomers();  
 }  
  
 @GetMapping("/{id}")  
 public Customer getCustomerById(@PathVariable Long id) {  
 return customerService.getCustomerById(id);  
 }  
  
 @PostMapping  
 public void saveCustomer(@RequestBody Customer customer) {  
 customerService.saveCustomer(customer);  
 }  
  
 @DeleteMapping("/delete/{id}")  
public void deleteCustomer(@PathVariable Long id) {  
 customerService.deleteCustomer(id);  
}

}

@PutMapping("/update/{id}")  
public void updateCustomer(@PathVariable Long id, @RequestBody Customer customer) {  
 customerService.update(id, customer);  
  
}

### Classe Product Controller

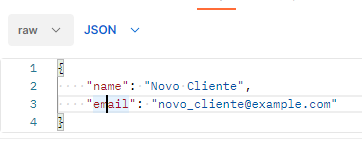
package com.digisystem.luiz.controladores;  
  
import com.digisystem.luiz.model.Product;  
import com.digisystem.luiz.service.ProductService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/products")  
public class ProductController {  
  
 @Autowired  
 private ProductService productService;  
  
 @GetMapping  
 public List<Product> getAllProducts() {  
 return productService.getAllProducts();  
 }  
  
 @GetMapping("/{id}")  
 public Product getProductById(@PathVariable Long id) {  
 return productService.getProductById(id);  
 }  
  
 @PostMapping  
 public void saveProduct(@RequestBody Product product) {  
 productService.saveProduct(product);  
 }  
  
 @DeleteMapping("/delete/{id}")  
public void deleteProduct(@PathVariable Long id) {  
 productService.deleteProduct(id);  
}

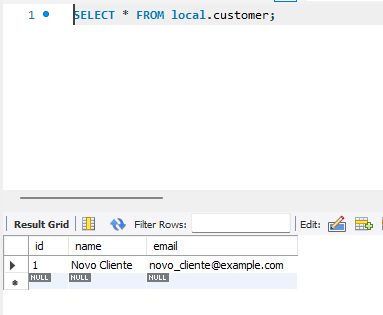
}

@PutMapping("/update/{id}")  
public void updateProduct(@PathVariable Long id, @RequestBody Product product) {  
 productService.update(id, product);  
  
}

# Testes Postman

## Criar Cliente

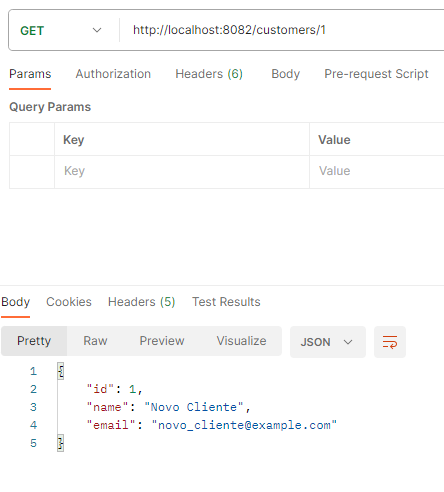




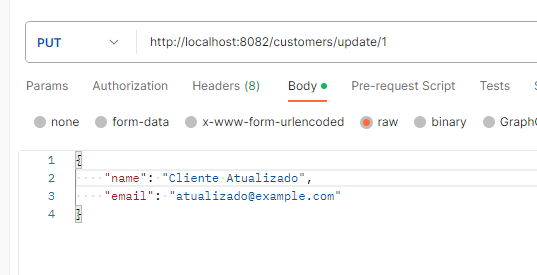
## Obter todos clients

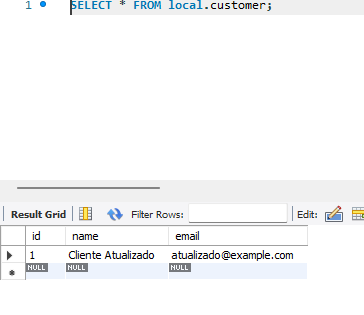
## 

## Obter cliente por id

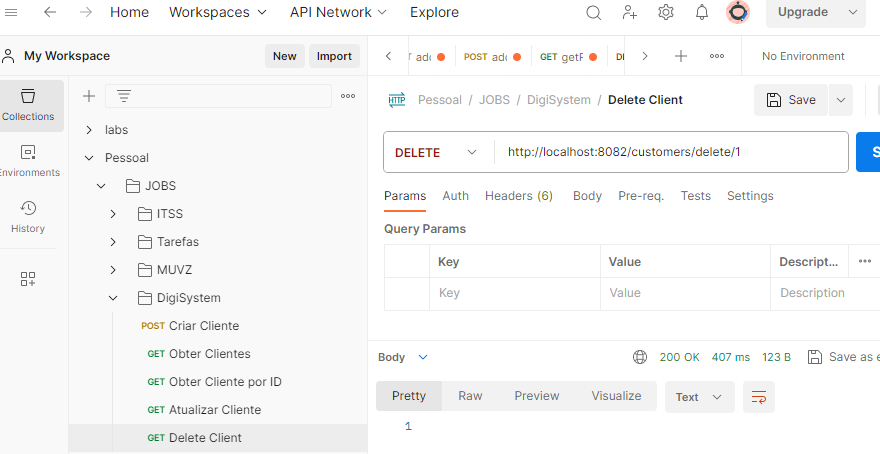


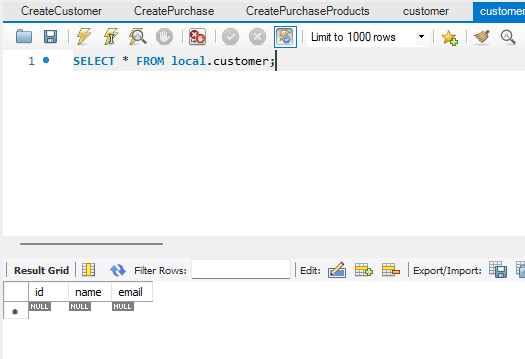
## Atualizar cliente



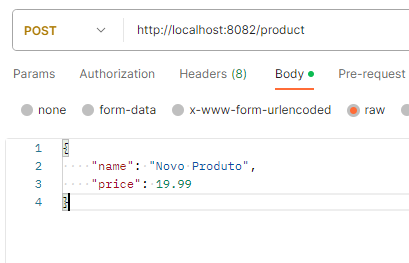


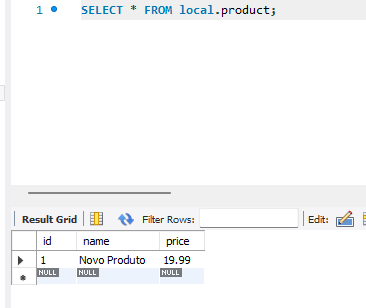
## Excluir cliente



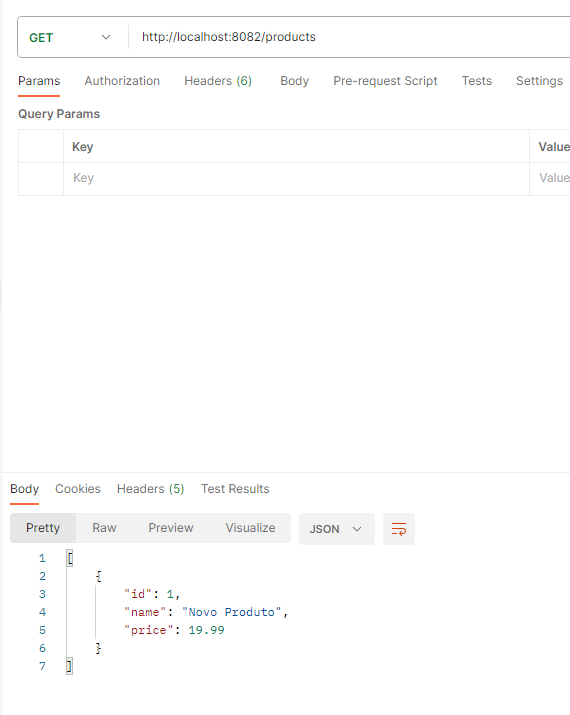


## Criar Produto

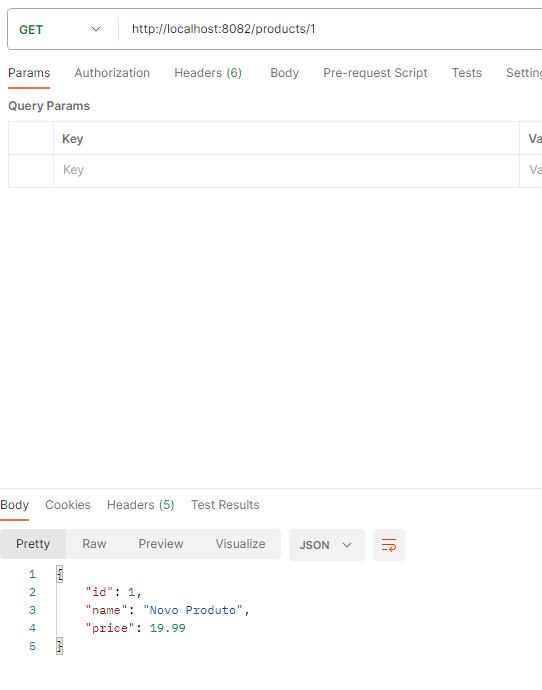


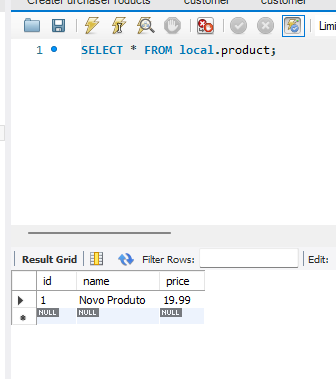


## Obter Todos Produtos

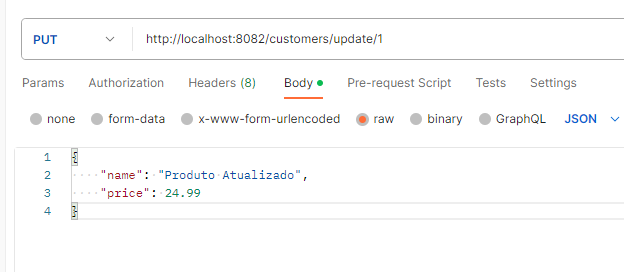


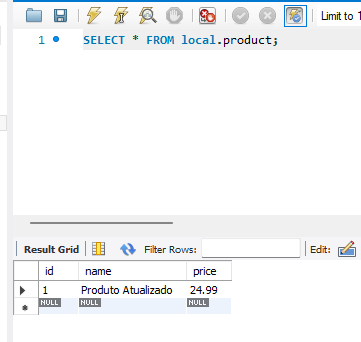
## Obter Produto por ID





## Atualizar Produto por ID





## Excluir Produto

