**Reto 1. Construcción de un Microservicio RESTful con Spring Boot 3 y Java 21**

Construir un microservicio RESTful en Java 21 utilizando Spring Boot 3 que gestione una lista de productos. El microservicio debe permitir realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre los productos. (Revisar el Instructivo del reto practico 1 en el video de la clase 13-07-2024 para realizar el reto completo)

**Instructivo reto práctico 1**

**Reto Práctico:** Construcción de un Microservicio RESTful con Spring Boot 3 y Java 21

Pasos Detallados:

**1. Crear un nuevo proyecto Spring Boot:**

- Ir a [Spring Initializr](https://start.spring.io/).

- Seleccionar las opciones:

- Project: Maven Project

- Language: Java

- Spring Boot: 3.x.x (Última versión disponible)

- Group: com.example

- Artifact: demo

- Dependencies: Spring Web, Spring Data JPA, H2 Database, lombok

- Java: 21

- Descargar el proyecto generado y descomprimirlo.

puntaje: 10

**2. Configurar 'application.properties':**

- Editar el archivo 'src/main/resources/application.properties' y agregar las

siguientes configuraciones:

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driver-class-name=org.h2.Driver

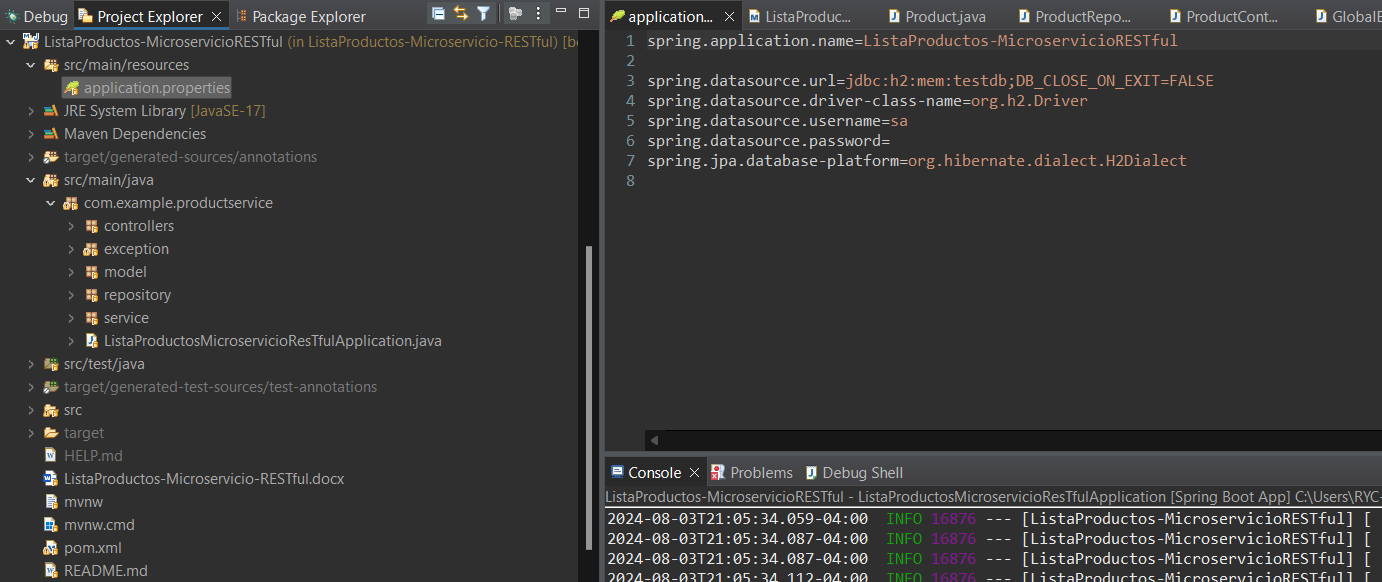
spring.datasource.username=sa

spring.datasource.password=password

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.h2.console.enabled=true

puntaje: 10



**3. Crear la entidad 'Product':**

package com.example.demo.model;

import jakarta.persistence.Id;

import lombok.Data;

@Entity

@Data

public class Product {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

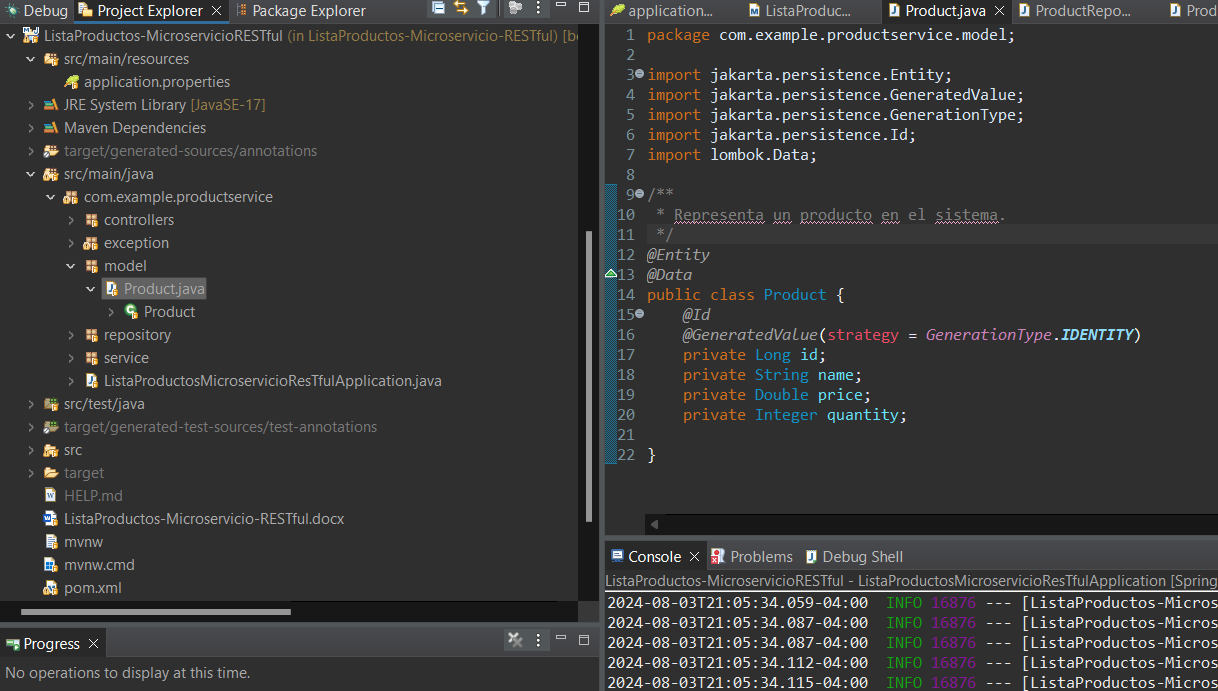
private Double price;

private Integer quantity;

// Getters and Setters

}

puntaje: 10



**4. Crear el repositorio 'ProductRepository':**

package com.example.demo.repository;

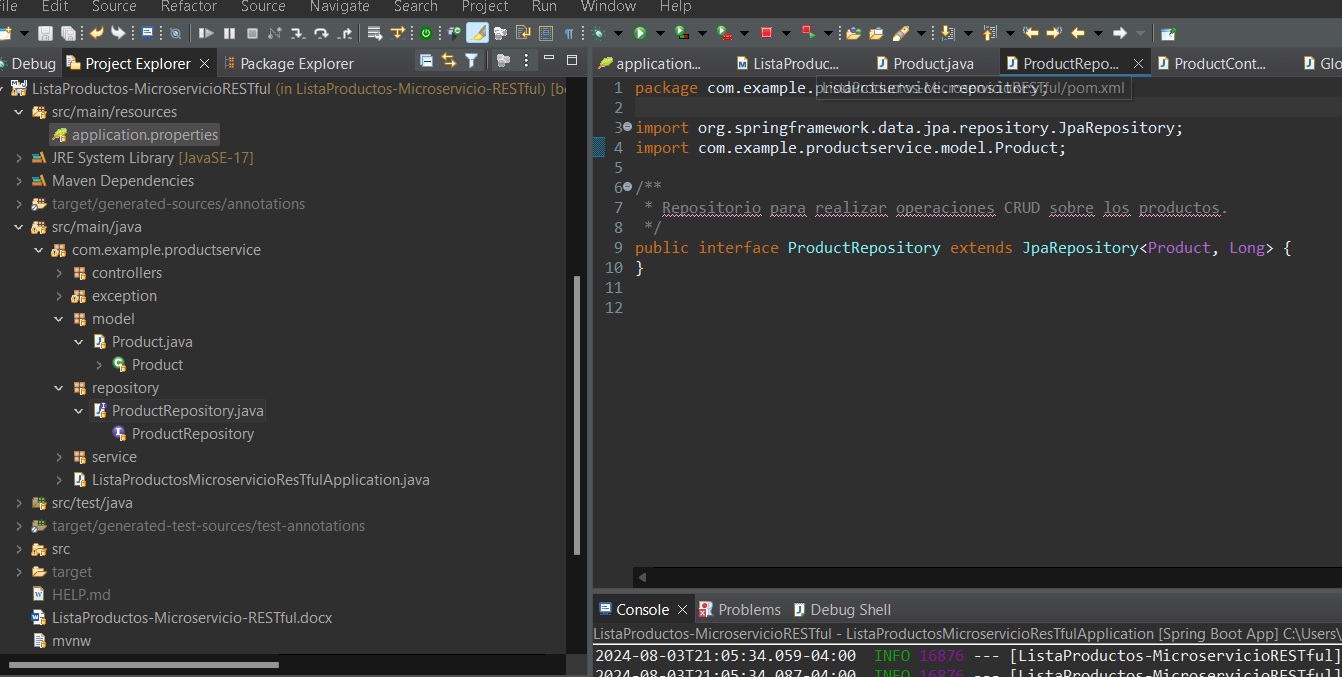
import com.example.demo.model.Product;

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

public interface ProductRepository extends JpaRepository<Product, Long> {

}

puntaje: 10



**5. Crear el controlador REST 'ProductController'**:

package com.example.demo.controller;

import com.example.demo.model.Product;

import com.example.demo.repository.ProductRepository;

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 ProductRepository productRepository;

@GetMapping

public List<Product> getAllProducts() {

return productRepository.findAll();

}

@GetMapping("/{id}")

public Product getProductById(@PathVariable Long id) {

return productRepository.findById(id).orElseThrow(() -> new

ProductNotFoundException(id));

}

@PostMapping

public Product createProduct(@RequestBody Product product) {

return productRepository.save(product);

}

@PutMapping("/{id}")

public Product updateProduct(@PathVariable Long id, @RequestBody Product

productDetails) {

Product product = productRepository.findById(id)

.orElseThrow(() -> new ProductNotFoundException(id));

product.setName(productDetails.getName());

product.setPrice(productDetails.getPrice());

product.setQuantity(productDetails.getQuantity());

return productRepository.save(product);

}

@DeleteMapping("/{id}")

public String deleteProduct(@PathVariable Long id) {

Product product = productRepository.findById(id).orElseThrow(() -> new

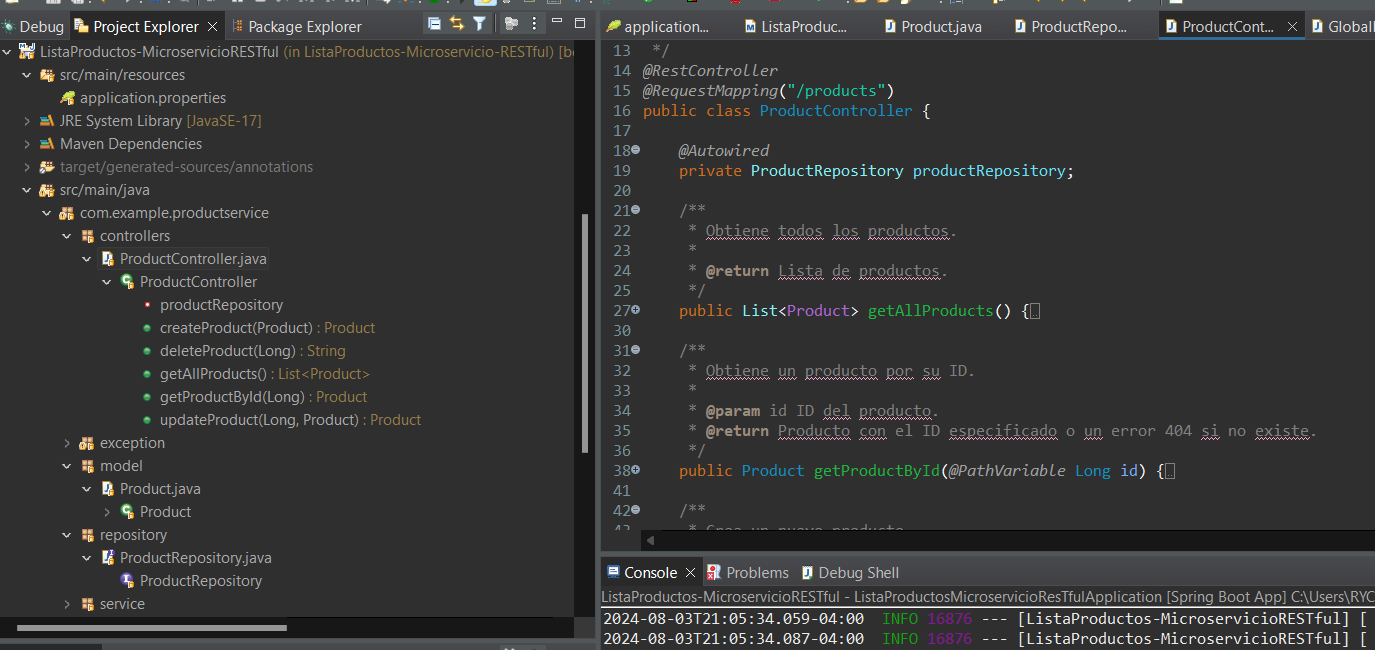
ProductNotFoundException(id));

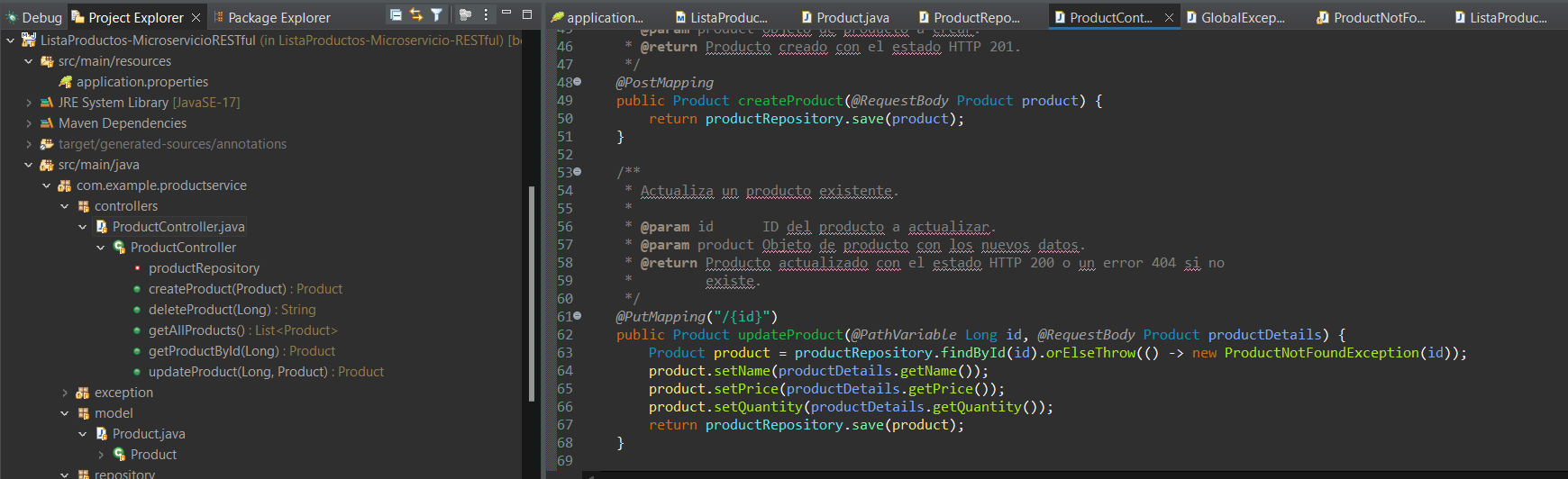
productRepository.delete(product);

return "Product " + id + " deleted successfully!";

}

} puntaje: 10





**6. Manejar excepciones:**

**- Crear la clase GlobalExceptionHandler**

package com.example.demo.exception;

import org.springframework.http.HttpStatus;

import org.springframework.web.bind.annotation.ExceptionHandler;

import org.springframework.web.bind.annotation.ResponseStatus;

import org.springframework.web.bind.annotation.RestControllerAdvice;

@RestControllerAdvice

public class GlobalExceptionHandler {

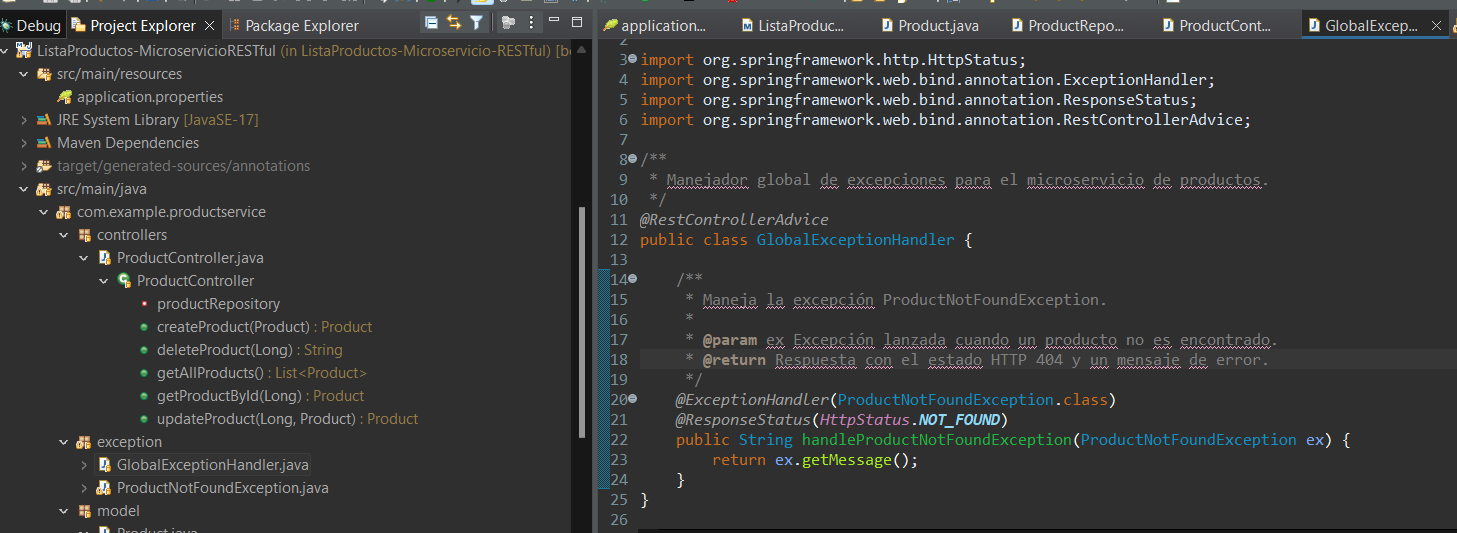
@ExceptionHandler(ProductNotFoundException.class)

@ResponseStatus(HttpStatus.NOT\_FOUND)

public String handleProductNotFoundException(ProductNotFoundException ex) {

return ex.getMessage();

} }



**- Crear la clase ProductNotFoundException**

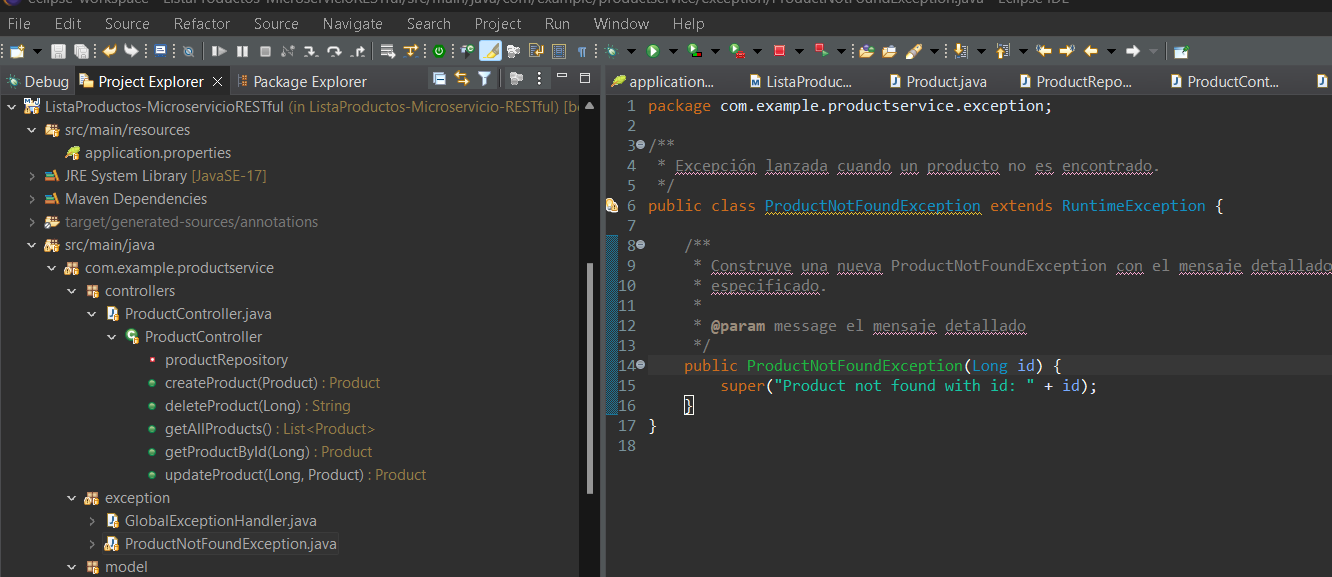
package com.example.demo.exception;

public class ProductNotFoundException extends RuntimeException {

public ProductNotFoundException(Long id) {

super("Product not found with id: " + id);

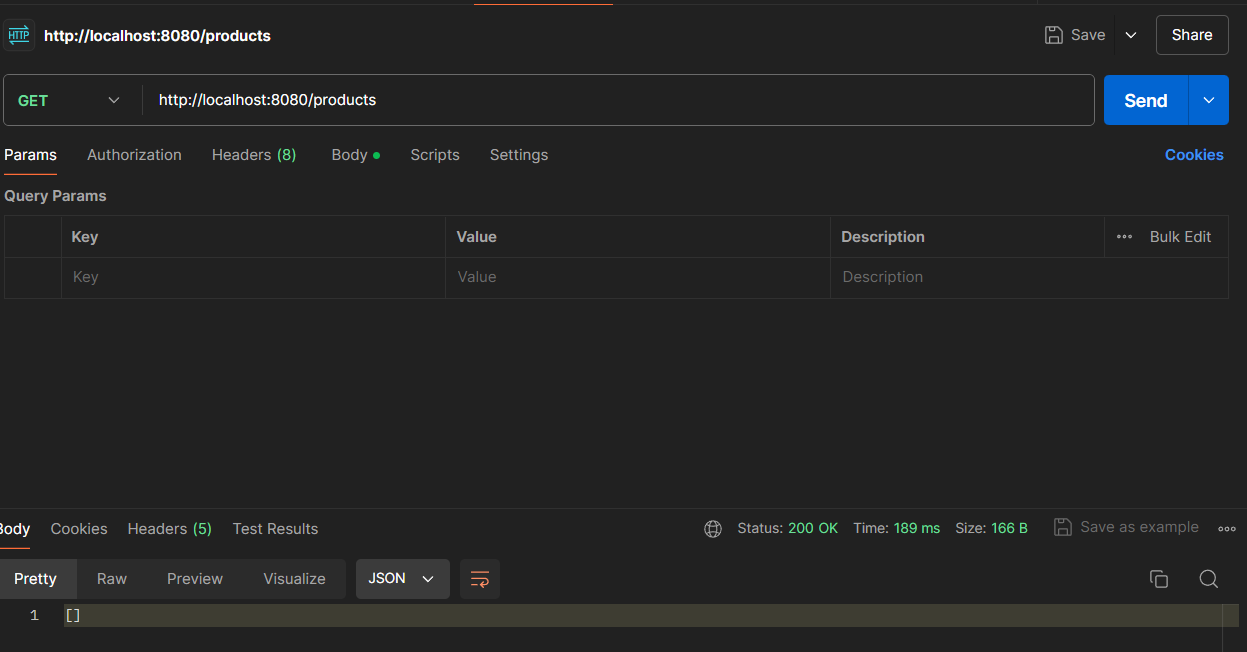
} } puntaje: 10

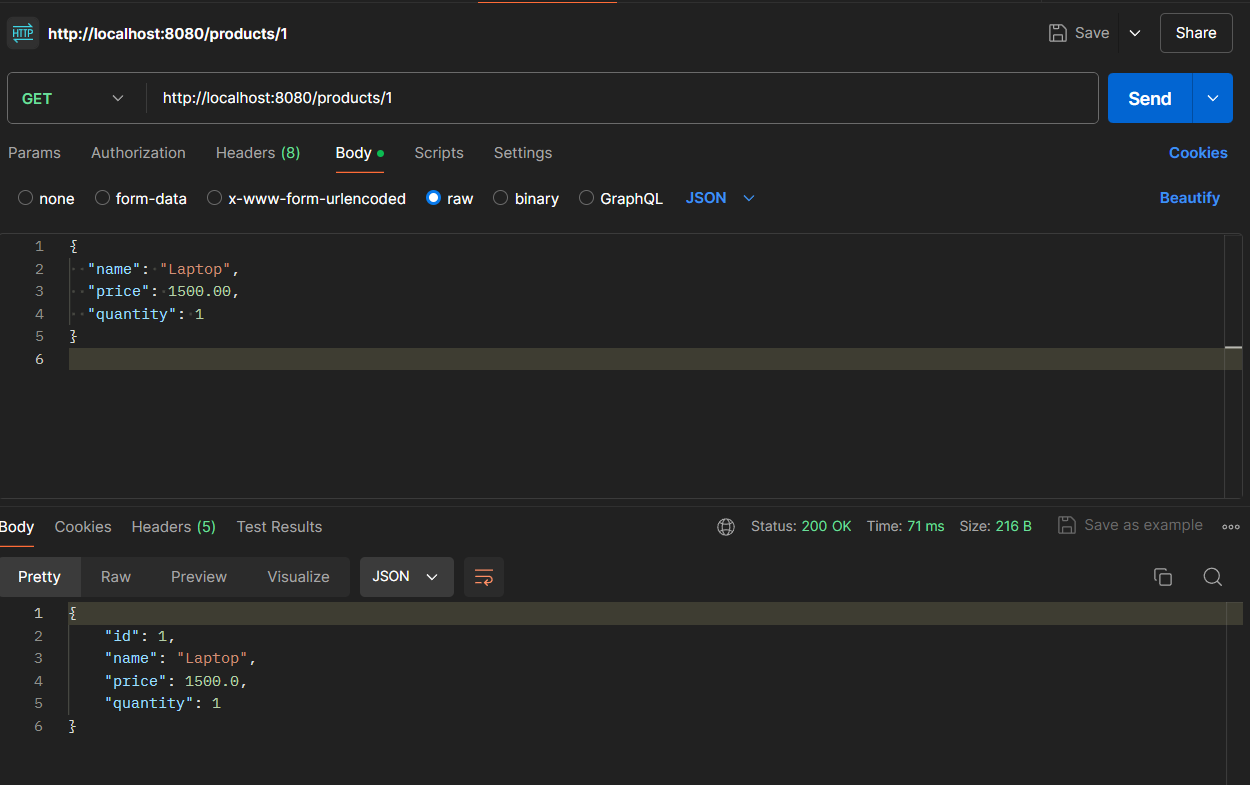


**7. Probar el microservicio:**

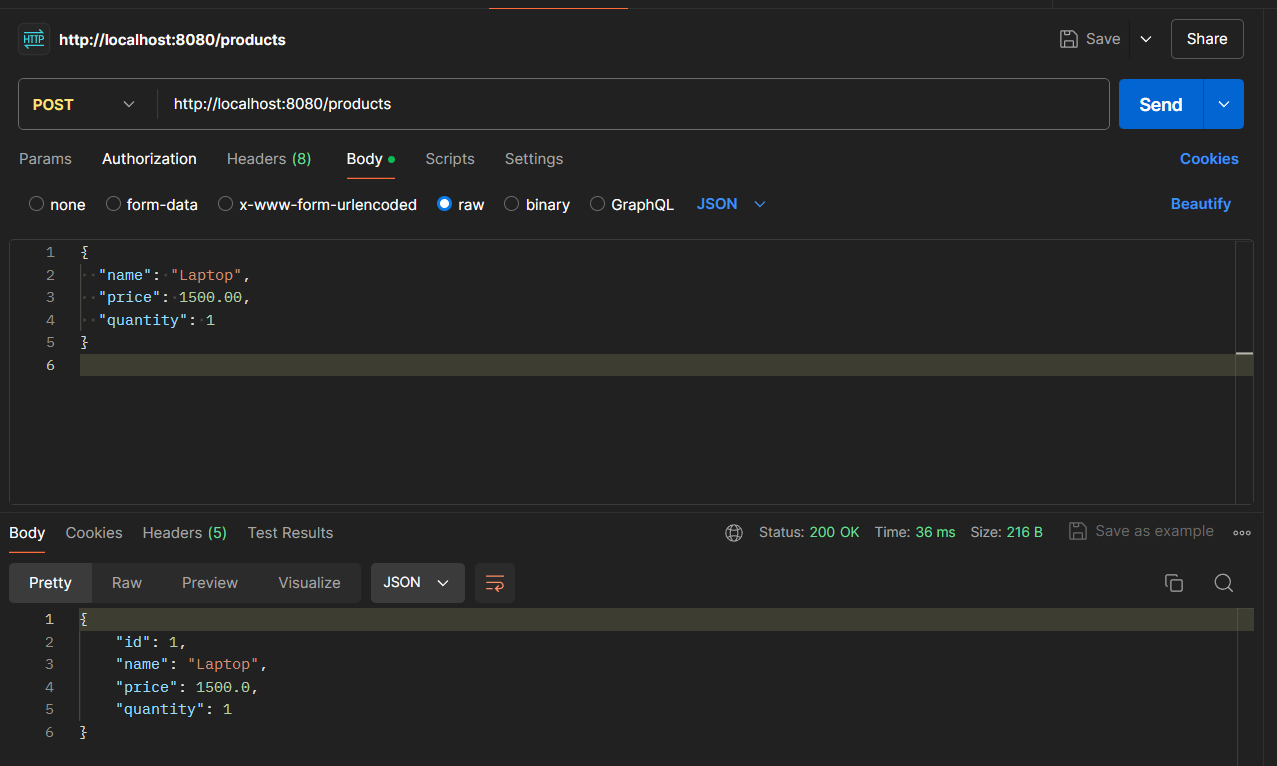
- Ejecutar la aplicación Spring Boot.

- Usar Postman para enviar solicitudes HTTP a las siguientes URLs:  
 'GET /products'

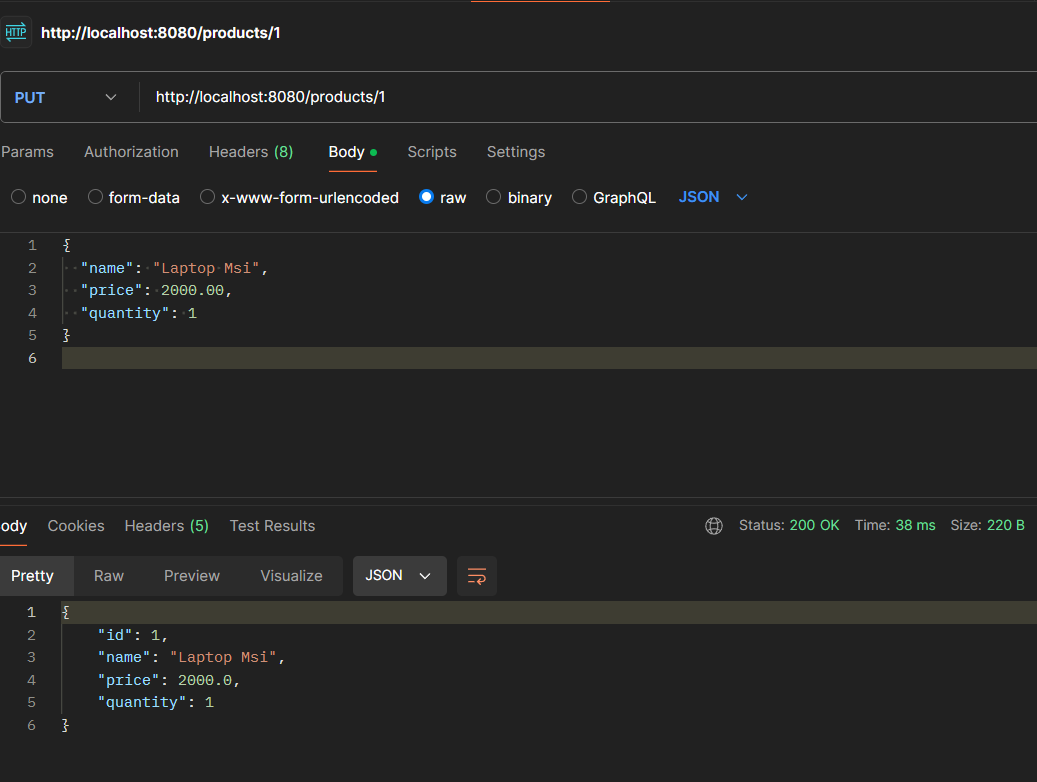


- 'GET /products/{id}'  


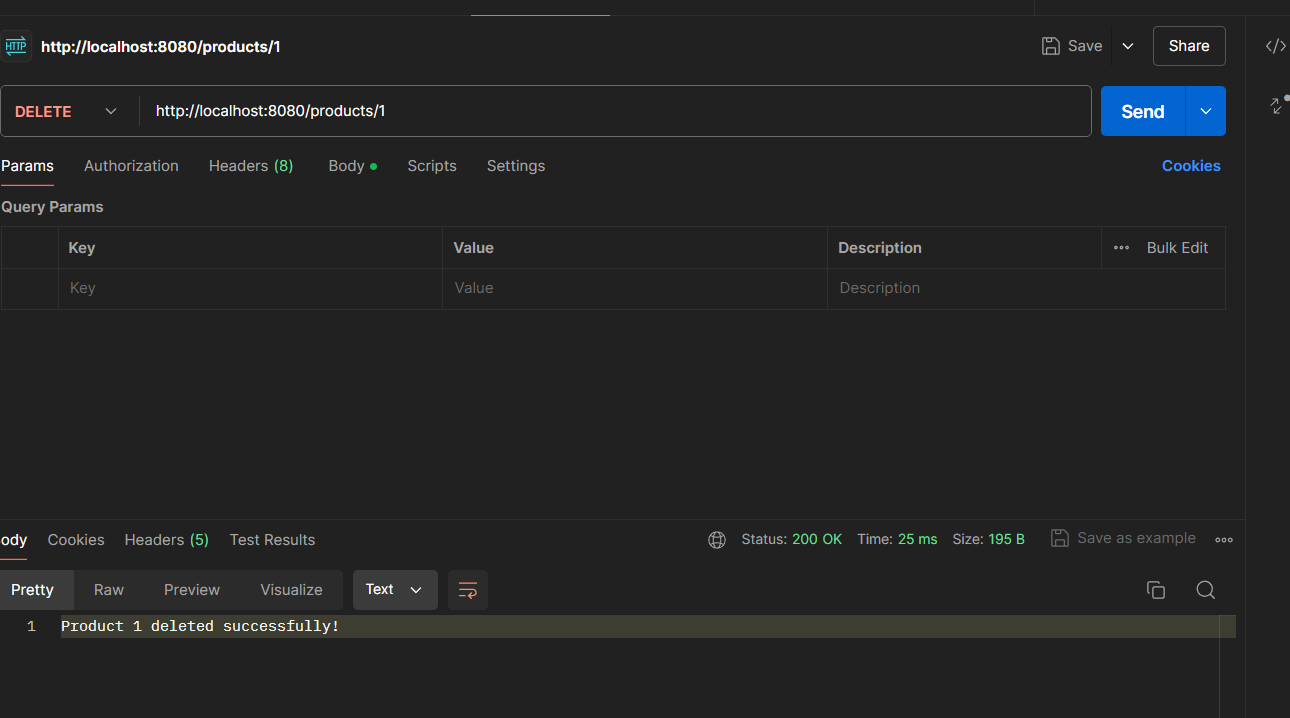
- 'POST /products'

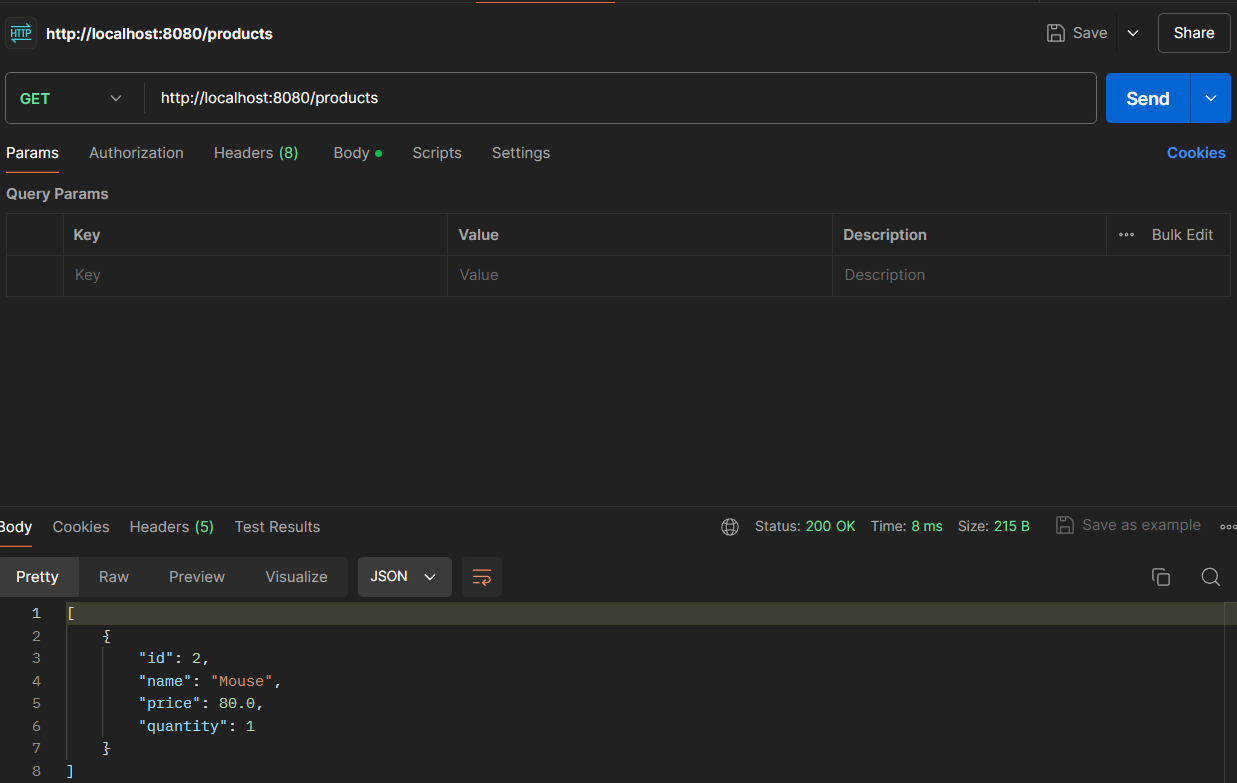
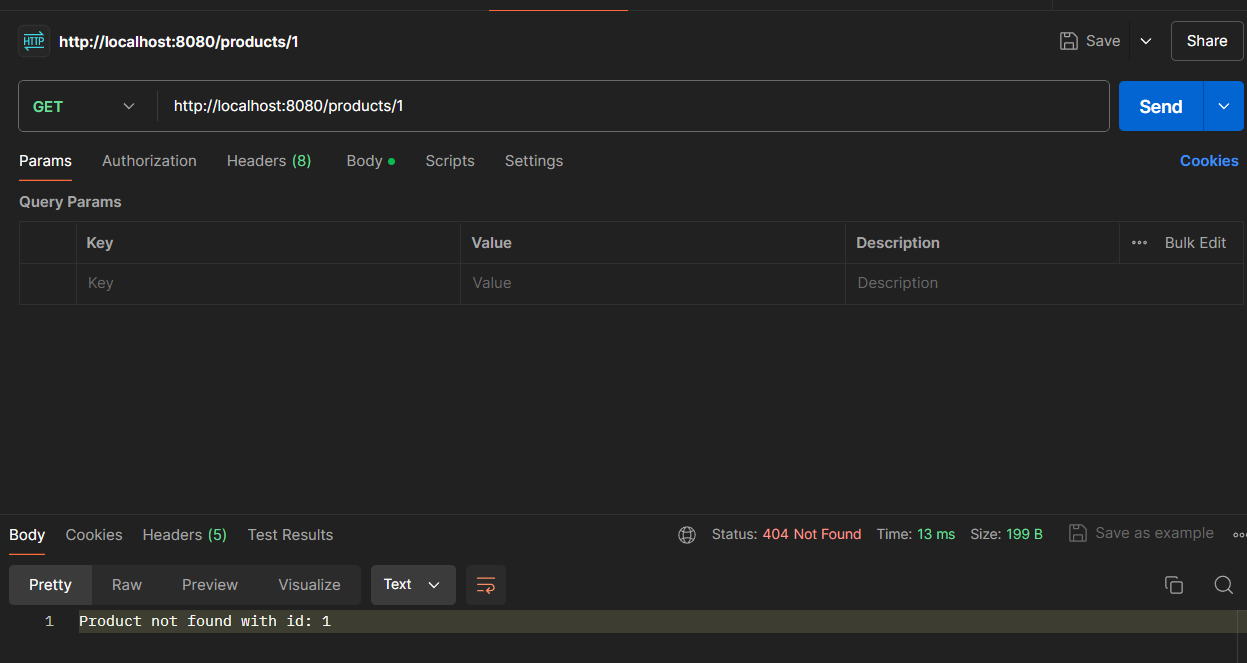


- 'PUT /products/{id}'



- 'DELETE /products/{id}'

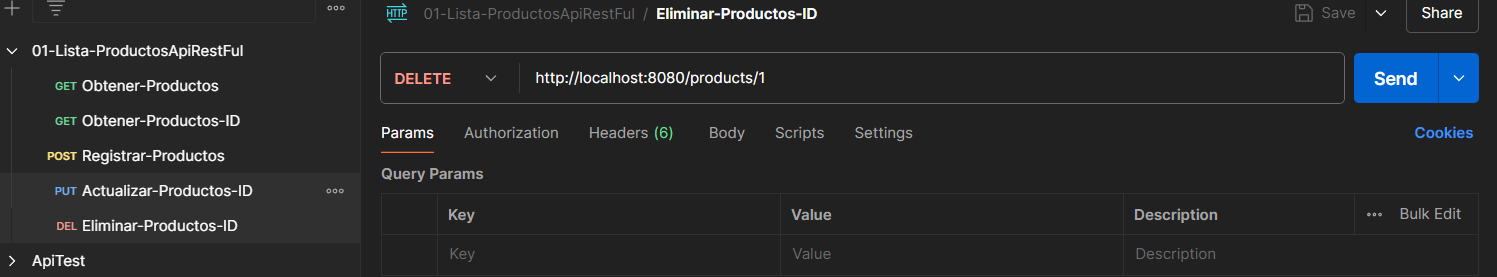




- Verificar que las operaciones funcionan correctamente.

**NOTA**: importar la collection de postman adjunta: reto.postman\_collection.json

puntaje: 10

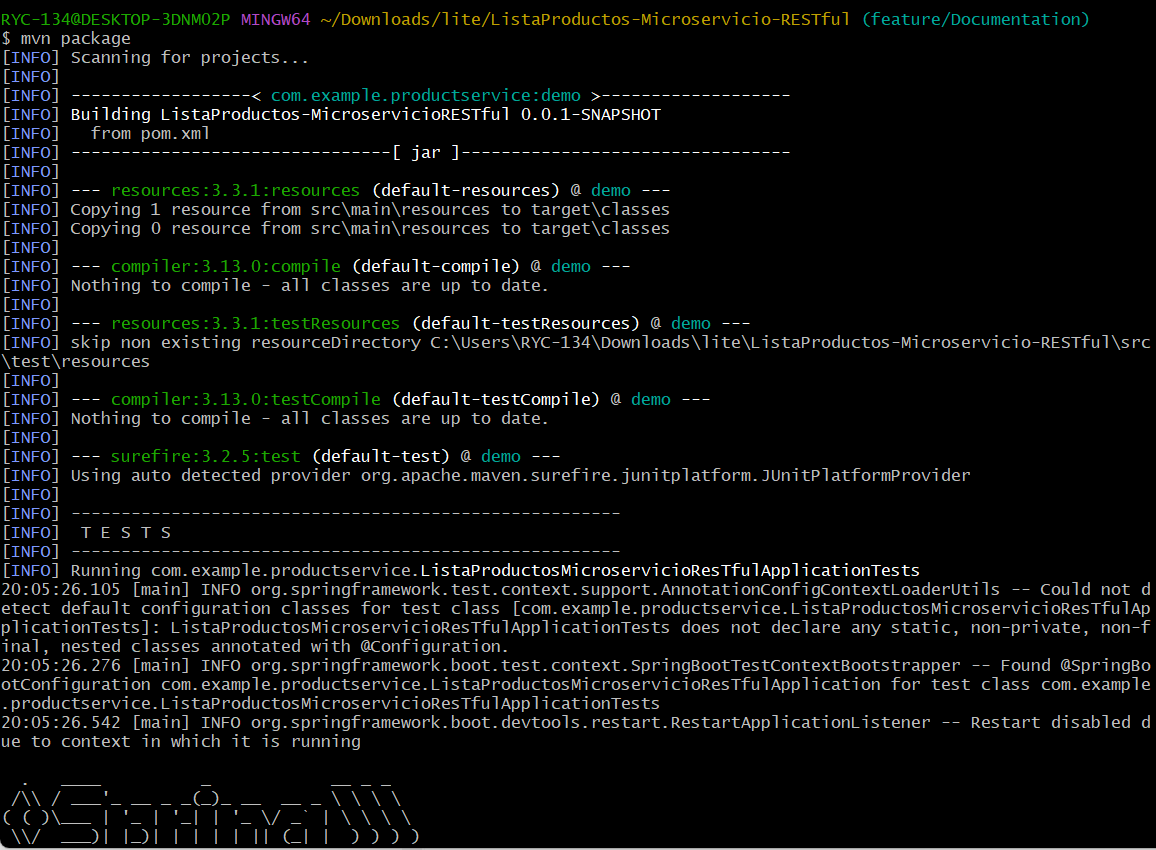


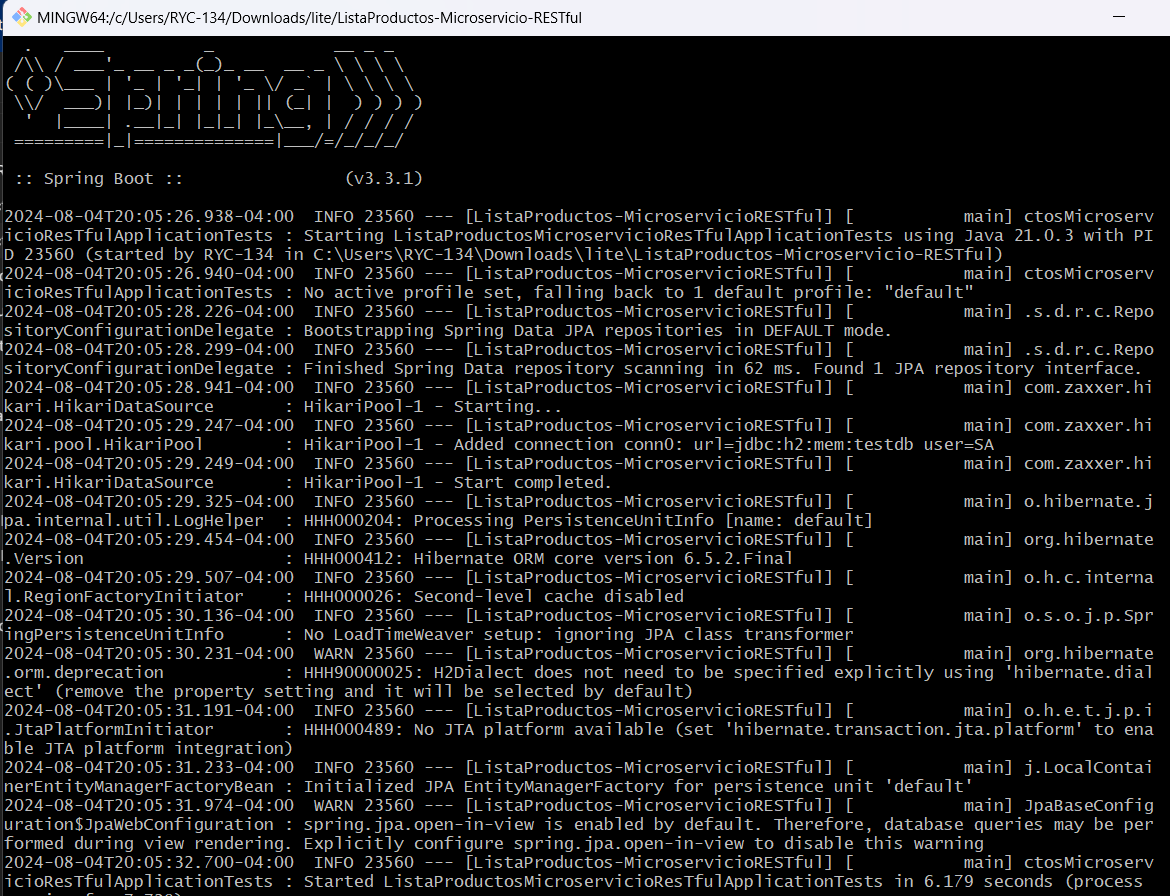
**8. Empaquetar el microservicio**

- empaquetar usando el comando maven

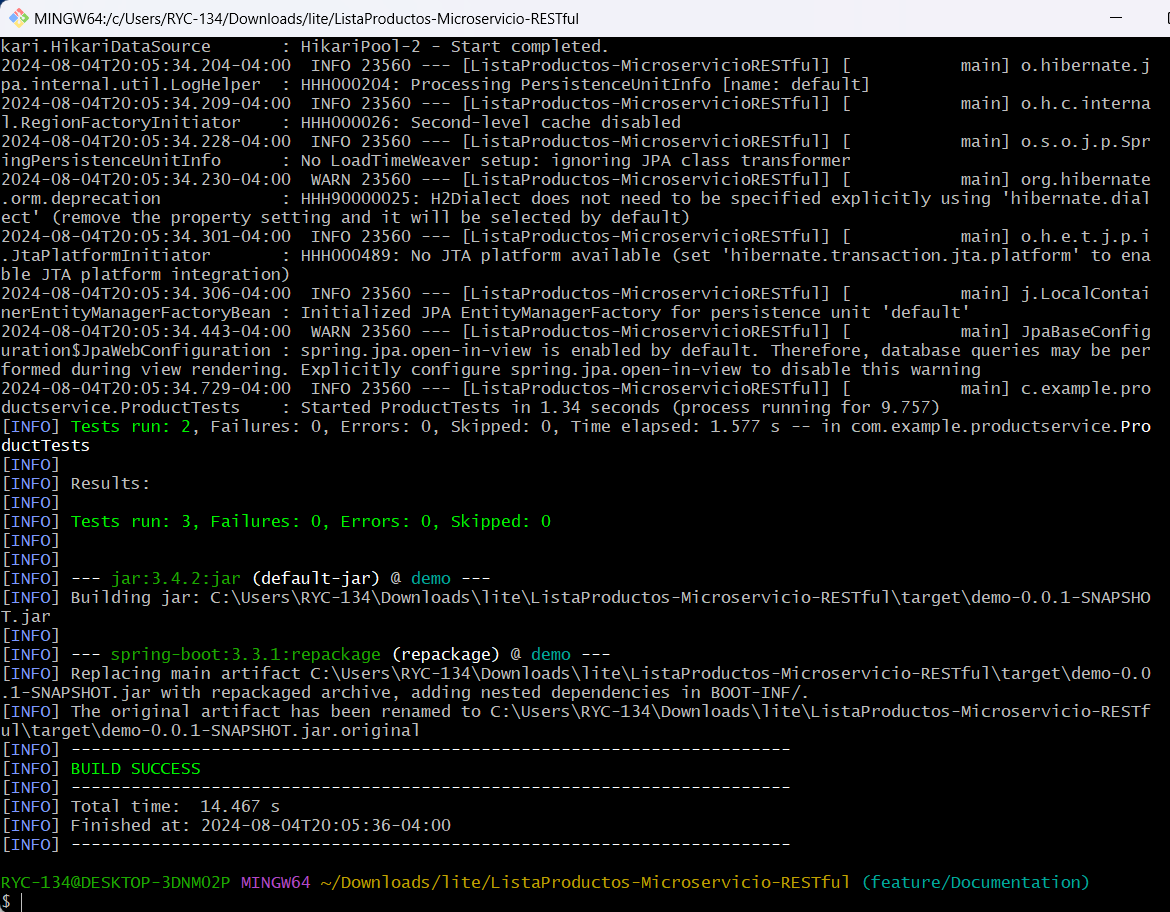
mvn package

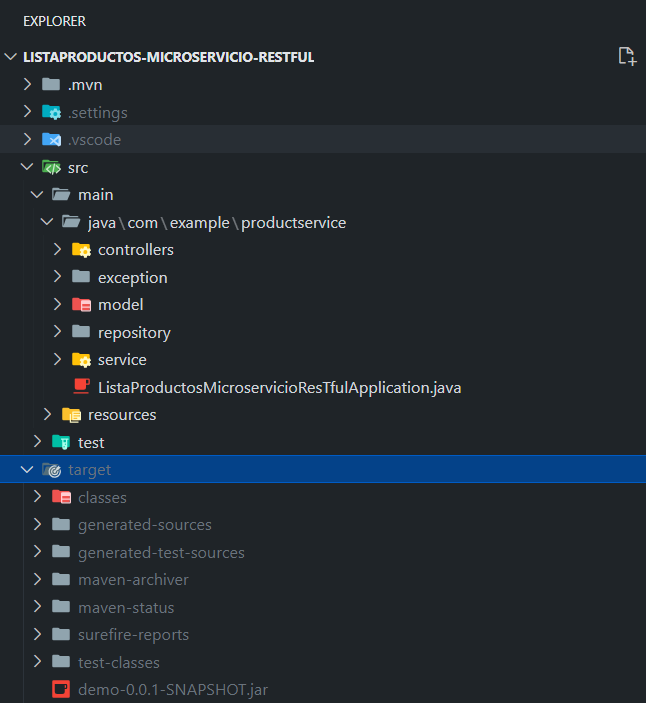
puntaje: 10









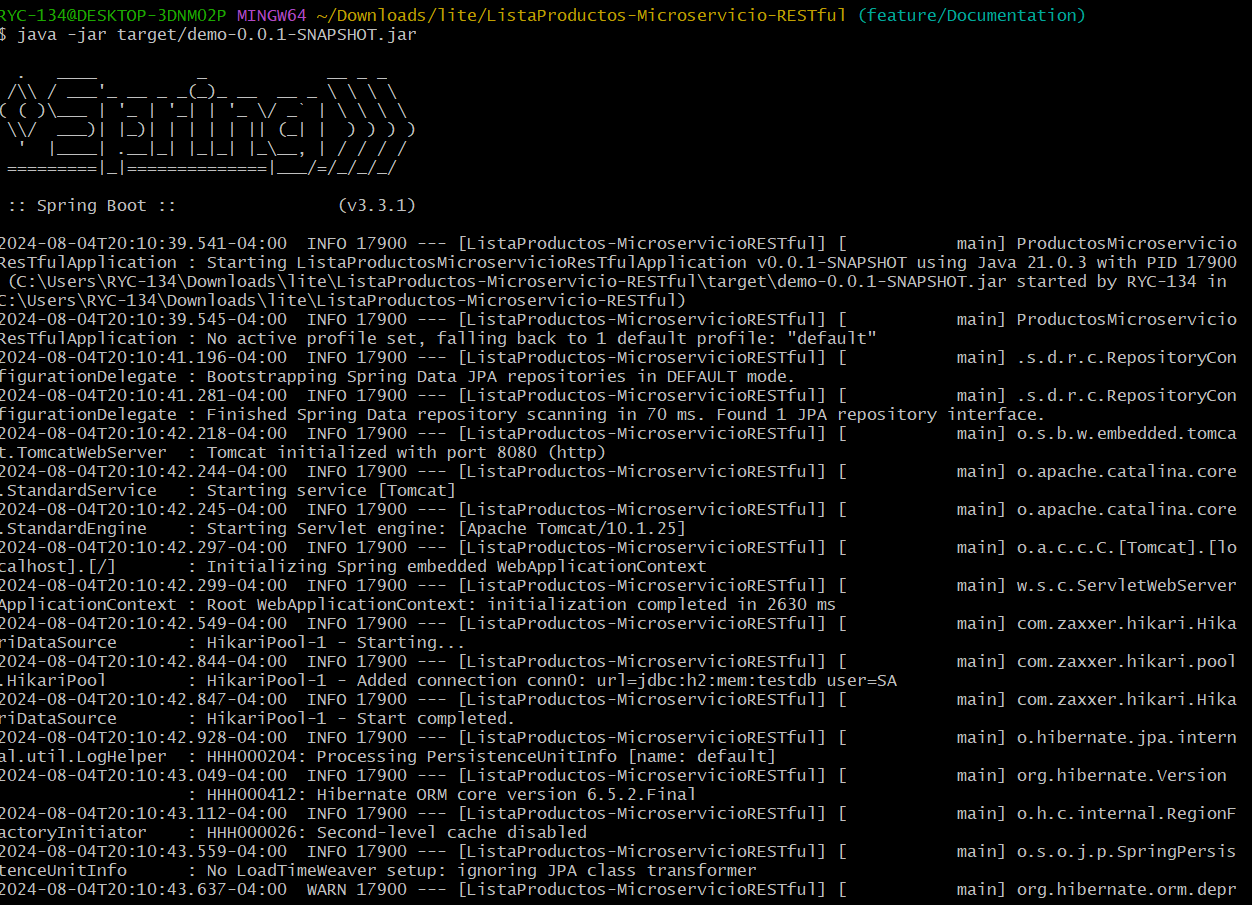


**9. Lanzar el microservicio desde linea de comando**

- Por fuera del IDE de desarrollo, lanzar la aplicación:

java -jar target/NOMBRE\_APP.jar

puntaje: 10



**10. Probar la aplicación y subir un pantallazo de los resultados**

puntaje: 10

