2025

LENGUAJE DE PROGRAMACIÓN AVANZADO 1 - 2501B04G1

UNIREMINGTON®
CORPORACIÓN UNIVERSITARIA REMINGTON
RES. 2661 MEN JUNIO 21 DE 1996

INGENIERIA DE SISTEMAS

Realizado por:

Carlos Andres Gaviria Ocampo

Presentado a:

Martha Nicolasa Amaya Becerra

📌 Evaluación de API RESTful

Objetivo: Evaluar el conocimiento básico de los estudiantes en **API RESTful con Spring Boot** a través de 5 ejercicios prácticos y sencillos.

📂 Instrucciones Generales

✓ Los estudiantes deben **desarrollar los ejercicios en Spring Boot 3+**.
✓ Pueden probar sus endpoints con **Postman o el navegador**.
✓ Se recomienda documentar el código con comentarios.
✓ **Subir el código a GitHub** y compartir el enlace en Moodle.

📝 Ejercicio 1: Crear un Endpoint Simple

📌 **Descripción:** Crea un endpoint en un controlador REST que devuelva el mensaje "¡Hola, API RESTful!" cuando el usuario acceda a /api/saludo.

📄 Código Esperado:

java

```
@RestController
@RequestMapping("/api")
public class SaludoController {

@GetMapping("/saludo")     public
```

String obtenerSaludo() {

return "¡Hola, API RESTful!";

} }

📌 **Prueba:**

- **GET:** http://localhost:8080/api/saludo
- **Salida esperada:** "¡Hola, API RESTful!"

📝 **Ejercicio 2: Crear un CRUD Básico de Productos**

📌 **Descripción:** Implementa una API RESTful que maneje un CRUD (Crear, Leer, Actualizar, Eliminar) para un modelo **Producto**.

📄 **Código Esperado:**

java

@Data

@AllArgsConstruc

tor @NoArgsConstructor

public class Producto {

private String id; private

String nombre; private

double precio; }

📄 **Endpoints esperados:**

- POST /api/productos → Agregar un producto.
- GET /api/productos → Listar todos los productos.
- GET /api/productos/{id} → Obtener un producto por ID.
- PUT /api/productos/{id} → Actualizar un producto.
- DELETE /api/productos/{id} → Eliminar un producto.

📌 **Prueba:**

- **POST:** Agregar un producto con Postman. **GET:** Consultar todos los productos.

📝 **Ejercicio 3: Implementar Internacionalización (i18n)**

📌 **Descripción:** Modifica el **Ejercicio 1** para que el mensaje de saludo pueda ser traducido según el idioma solicitado en la URL (?lang=es o ?lang=en).

📄 **Archivos messages.properties:**

📄 **messages_es.properties**

properties

saludo=¡Hola, API RESTful en Español!

📄 **messages_en.properties**

properties

saludo=Hello, RESTful API in English!

📄 **Código Esperado:**

java

```
@RestController
@RequestMapping("/api")
public class SaludoController {

    @Autowired
    private MessageSource messageSource;

    @GetMapping("/saludo")
    public String obtenerSaludo(@RequestHeader(name = "Accept-Language",
required = false) Locale locale) {        return messageSource.getMessage("saludo", null,
locale);


} }
```

📌 **Prueba:**

 GET: http://localhost:8080/api/saludo?lang=es → "¡Hola, API RESTful en Español!" **GET:** http://localhost:8080/api/saludo?lang=en → "Hello, RESTful API in English!"

📝 **Ejercicio 4: Crear un Endpoint Reactivo con WebFlux**

📌 **Descripción:** Modifica el **Ejercicio 2** para que la lista de productos sea manejada de manera **reactiva** usando Flux.

📄 **Código Esperado:**

java

```
@RestController
@RequestMapping("/api/productos")
public class ProductoController {

    @GetMapping     public
Flux<Producto> listarProductos() {
return Flux.just(
            new Producto("1", "Laptop",
1200.0),          new Producto("2", "Mouse", 25.0),
            new Producto("3", "Teclado", 45.0)
        );
    }
}
```

📌 **Prueba:**

- **GET:** http://localhost:8080/api/productos
- **Salida esperada:** JSON con los productos en formato reactivo.

📝 **Ejercicio 5: Implementar Pruebas con StepVerifier**

📌 **Descripción:** Crea una prueba unitaria con **StepVerifier** para validar que el endpoint de productos devuelve los valores esperados.

📄 **Código Esperado:**

java

```
@SpringBootTest
@AutoConfigureMockMvc
public class ProductoControllerTest {

    @Autowired
    private ProductoController productoController;

    @Test
    public void testListaProductos() {
        Flux<Producto> productos = productoController.listarProductos();

        StepVerifier.create(productos)
```

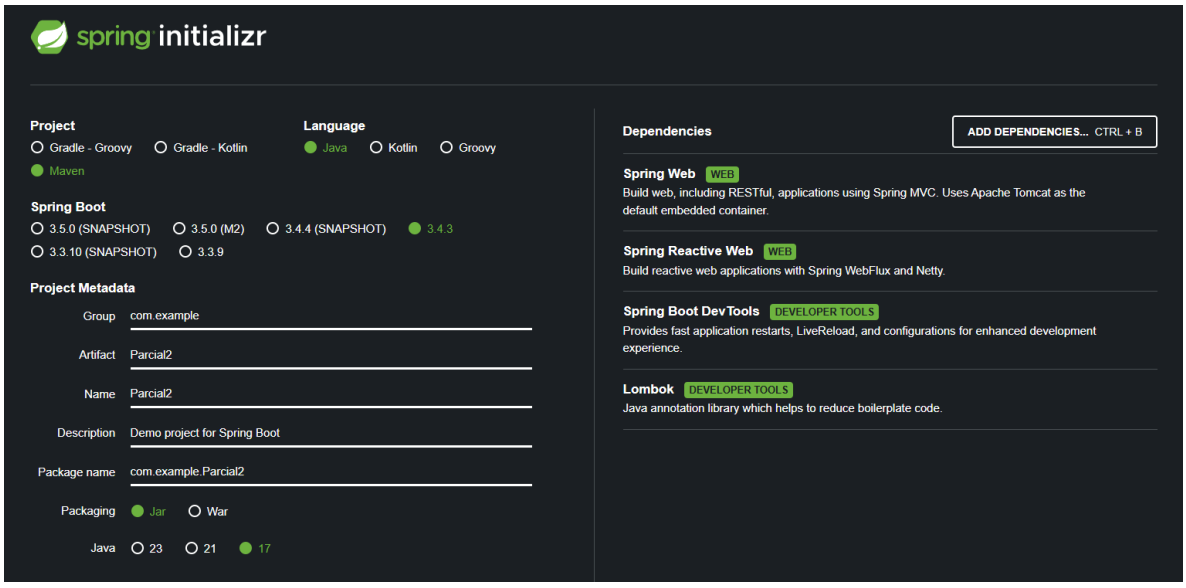
```
                .expectNextMatches(p -> p.getNombre().equals("Laptop"))

                .expectNextMatches(p -> p.getNombre().equals("Mouse"))

                .expectNextMatches(p -> p.getNombre().equals("Teclado"))
.verifyComplete();



} }
```

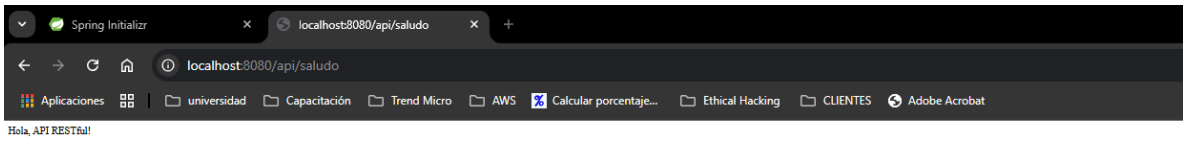
📌 **Prueba:**

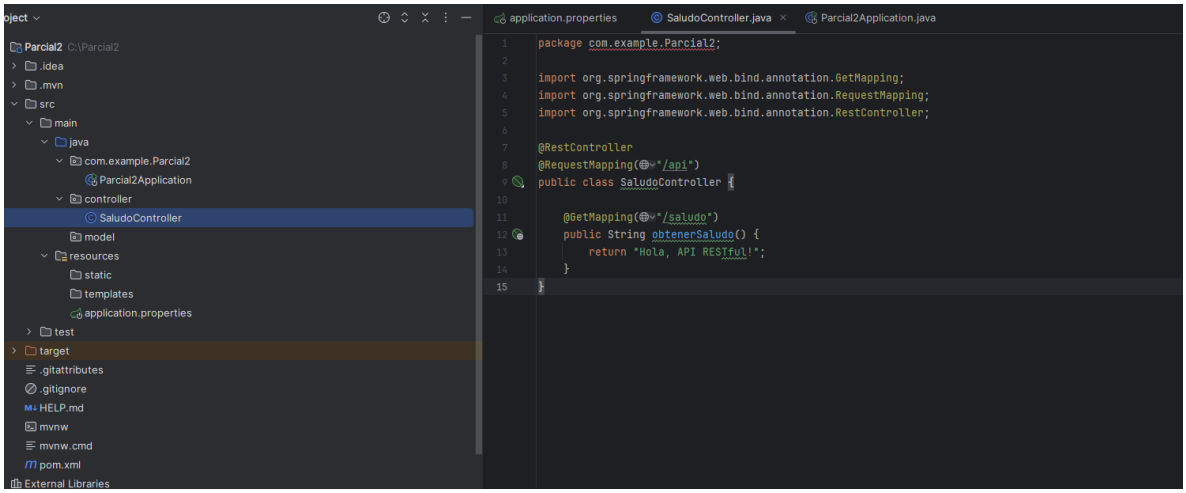
- **Ejecutar la prueba con JUnit y StepVerifier**.
- **Salida esperada:** La prueba debe pasar correctamente.

SOLUCION

Punto 1







Saludo

package com.example.parcial2.controller;

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

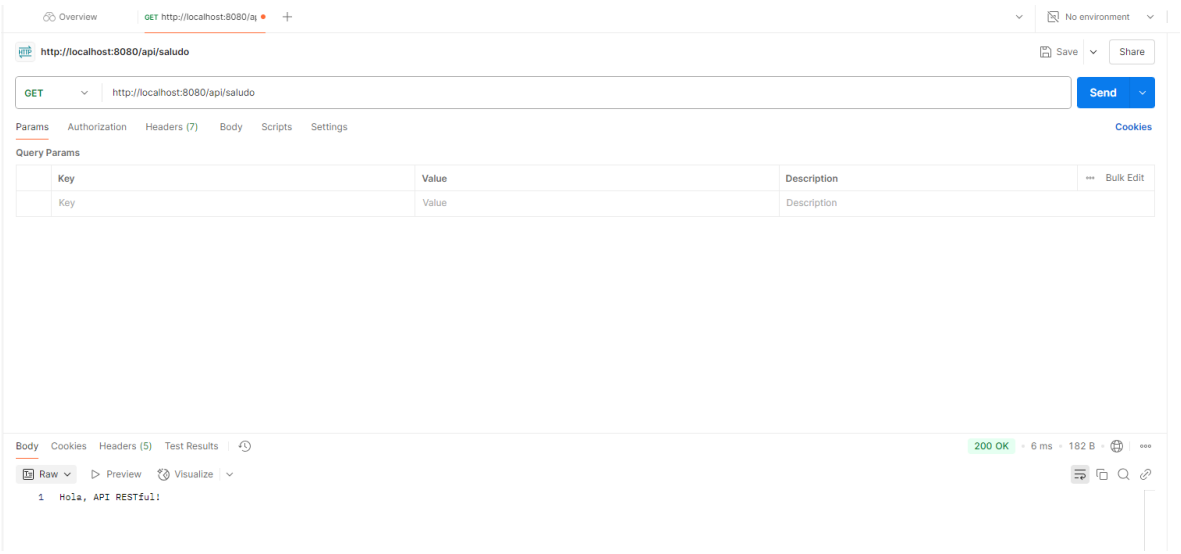
import org.springframework.web.bind.annotation.RequestMapping;

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

@RestController

```
@RequestMapping("/api")
public class SaludoController {

    @GetMapping("/saludo")
    public String obtenerSaludo() {
        return "Hola, API RESTful!";
    }
}
```

Postman



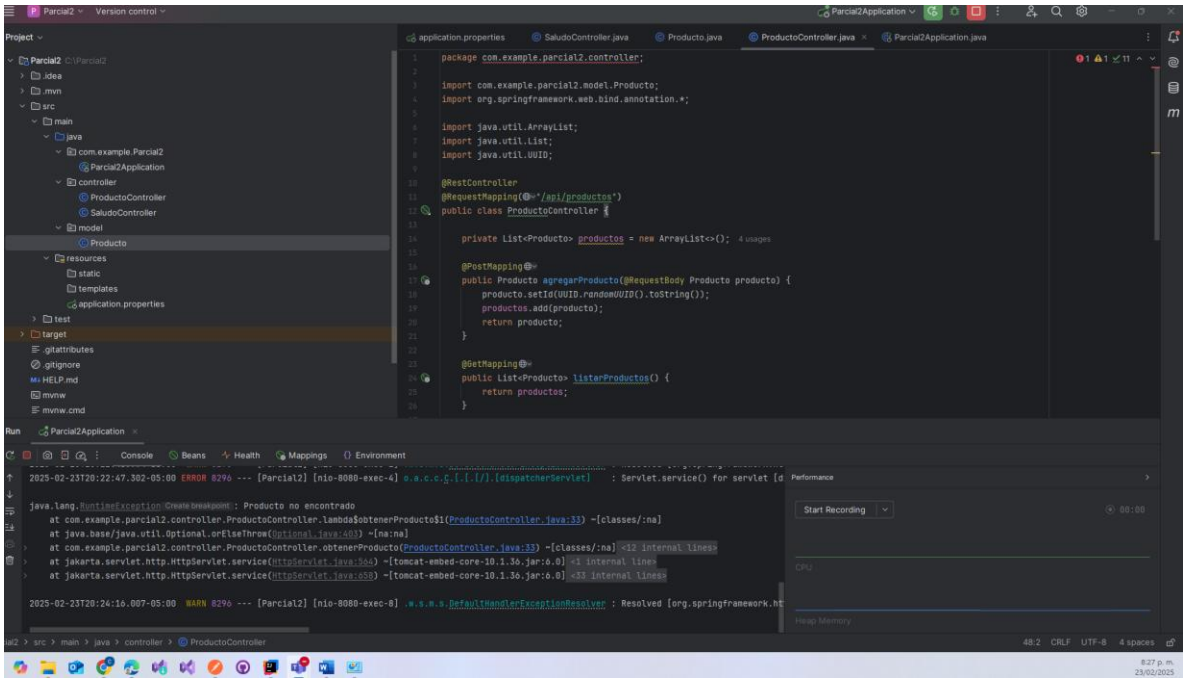
Punto 2

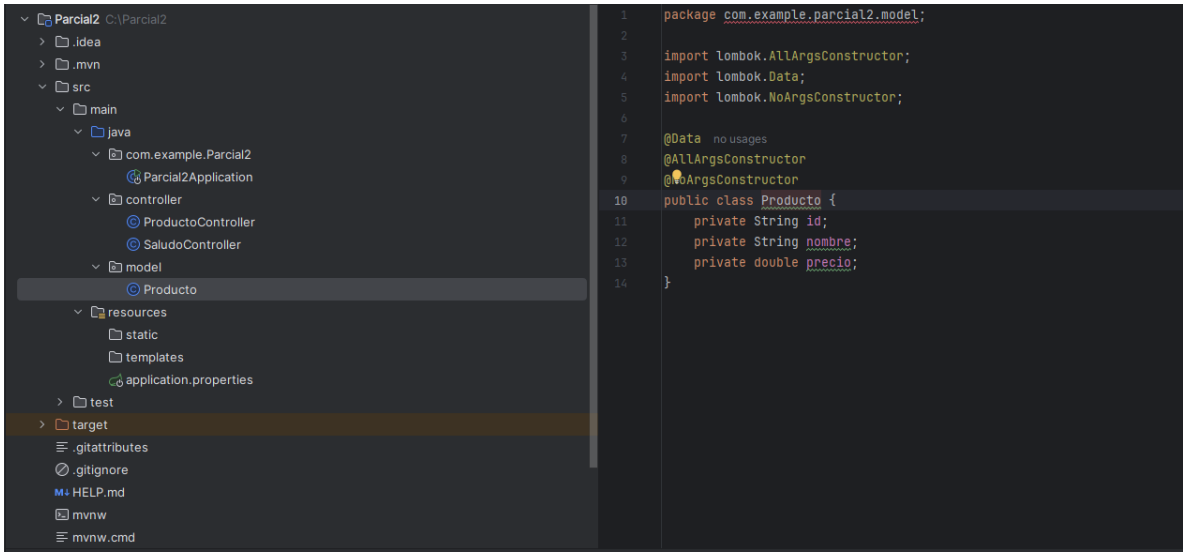
package com.example.parcial2.controller;

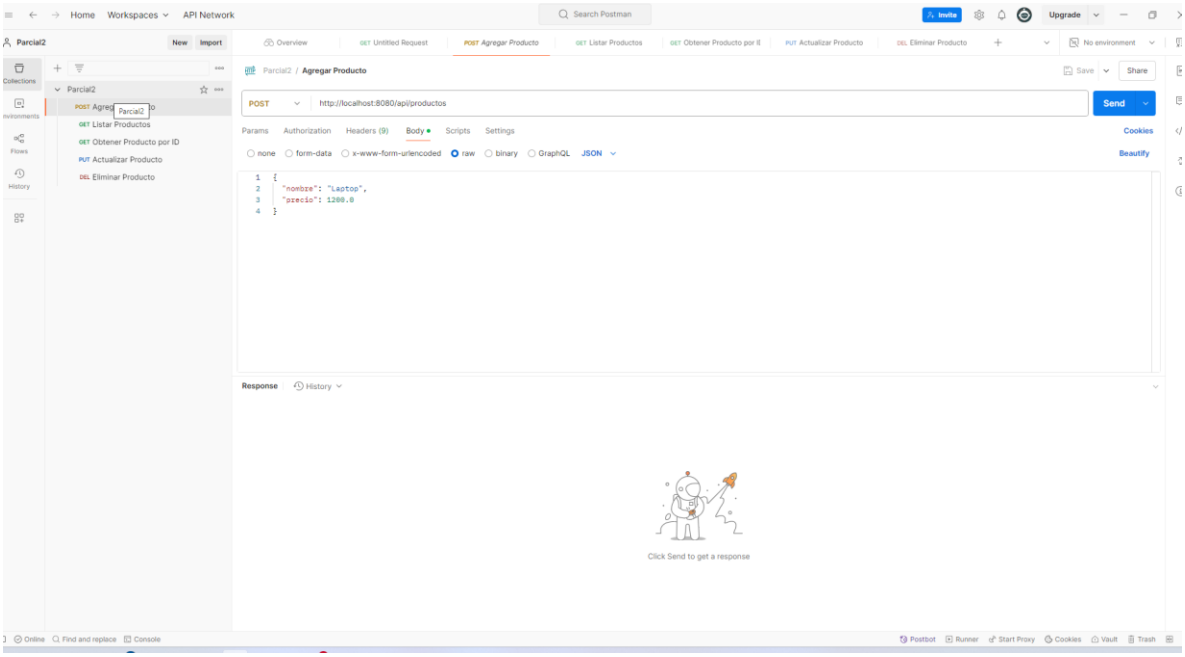
```
import com.example.parcial2.model.Producto;
import org.springframework.web.bind.annotation.*;

import java.util.ArrayList;
import java.util.List;
import java.util.UUID;

@RestController
@RequestMapping("/api/productos")
public class ProductoController {

    private List<Producto> productos = new ArrayList<>();

    @PostMapping
    public Producto agregarProducto(@RequestBody Producto producto) {
        producto.setId(UUID.randomUUID().toString());
        productos.add(producto);
        return producto;
    }

    @GetMapping
    public List<Producto> listarProductos() {
```

```
    return productos;
}

@GetMapping("/{id}")
public Producto obtenerProducto(@PathVariable String id) {
    return productos.stream()
            .filter(p -> p.getId().equals(id))
            .findFirst()
            .orElseThrow(() -> new RuntimeException("Producto no encontrado"));
}

@PutMapping("/{id}")
public Producto actualizarProducto(@PathVariable String id, @RequestBody Producto
producto) {
    Producto productoExistente = obtenerProducto(id);
    productoExistente.setNombre(producto.getNombre());
    productoExistente.setPrecio(producto.getPrecio());
    return productoExistente;
}

@DeleteMapping("/{id}")
public void eliminarProducto(@PathVariable String id) {
```

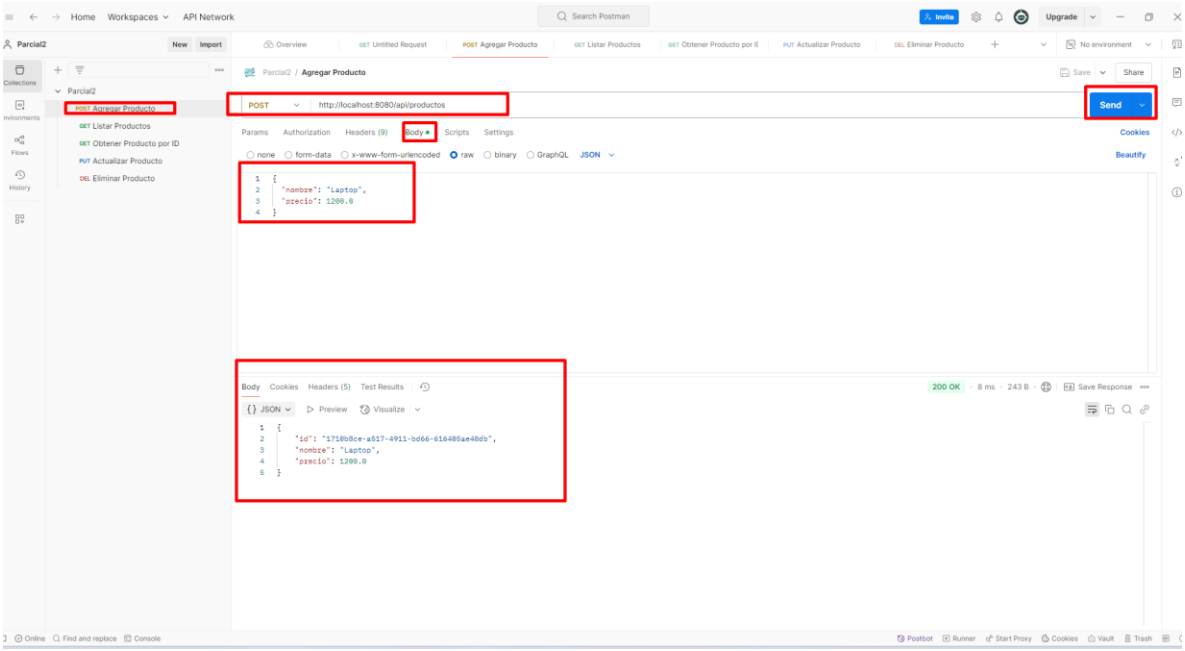
```
        productos.removeIf(p -> p.getId().equals(id));
    }
}
```



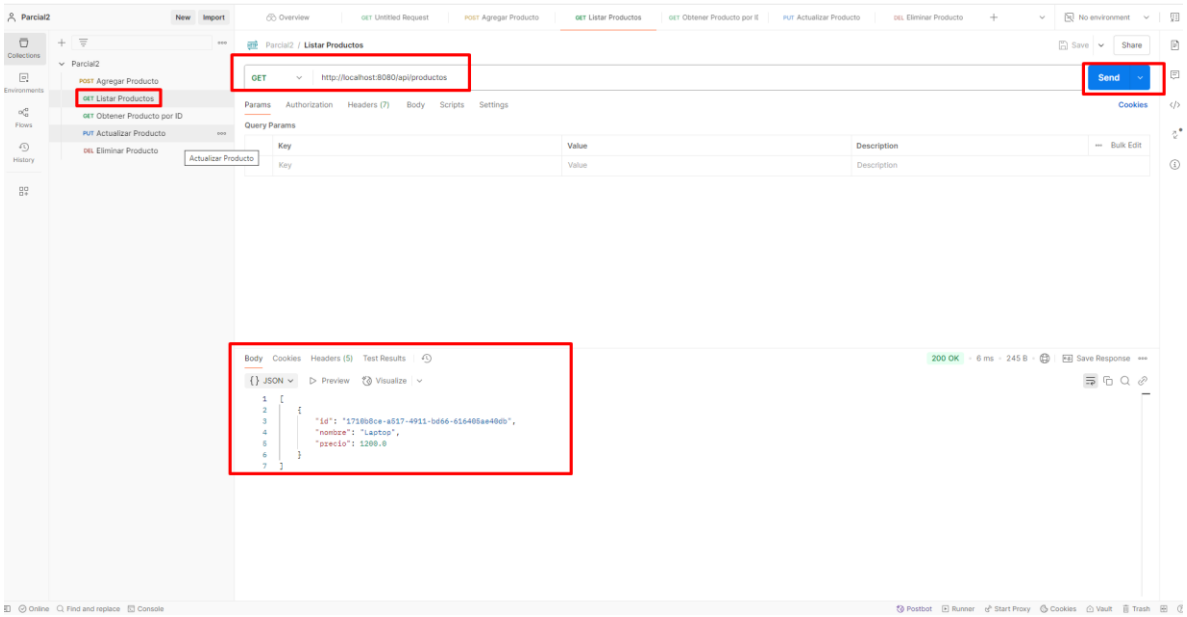




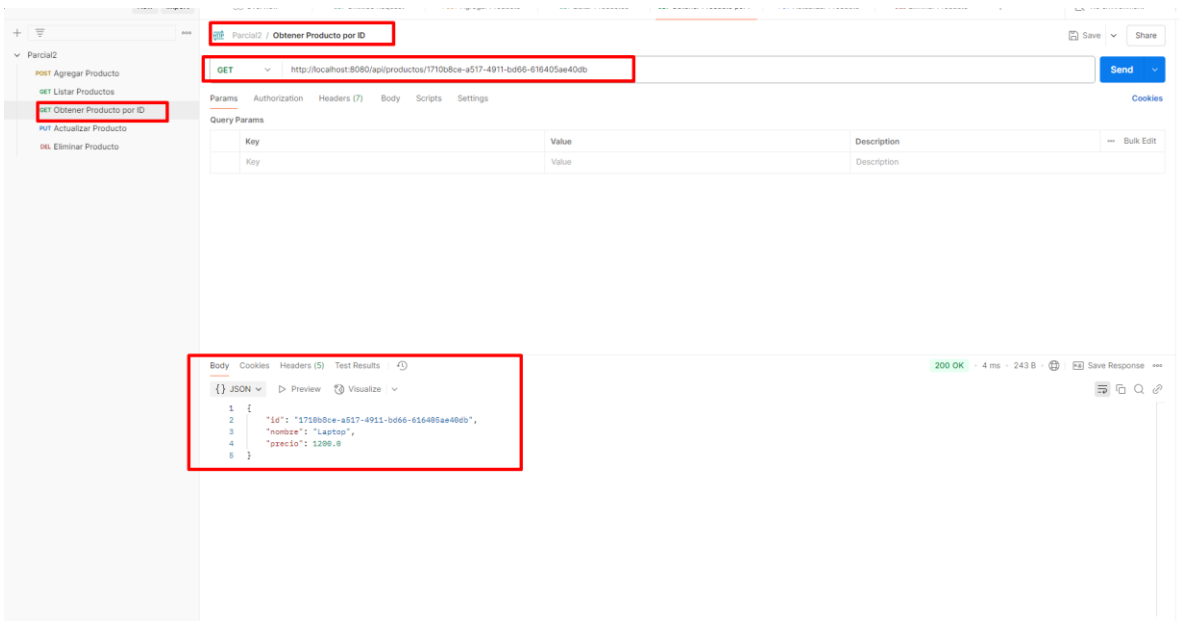
Agregar producto



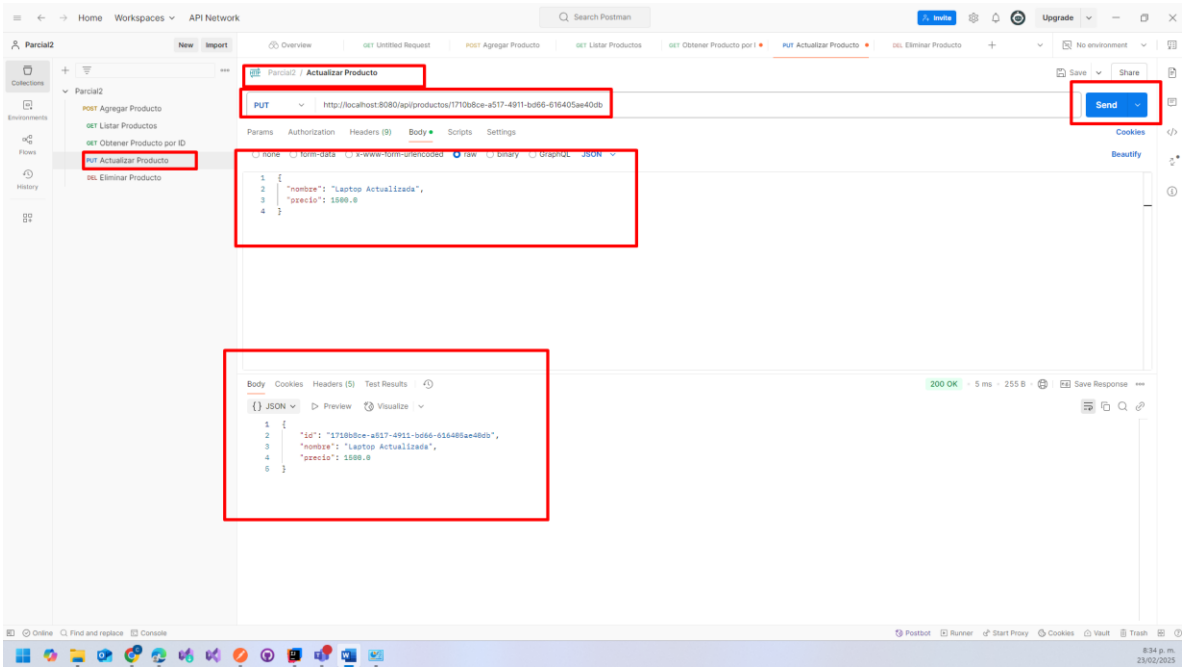
Listar Productos



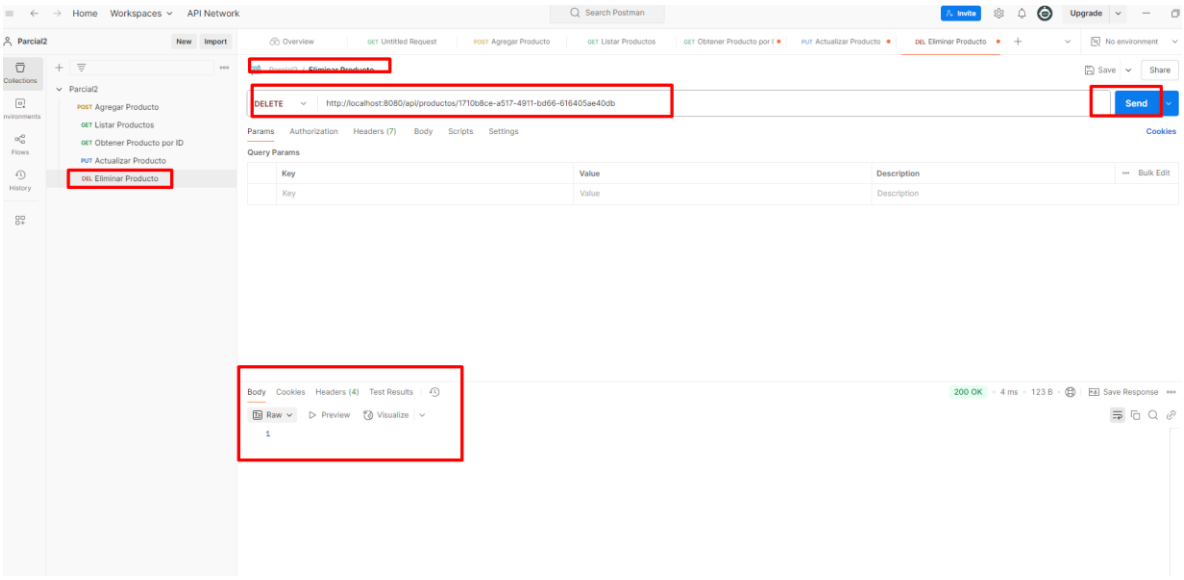
Obtener Producto por ID



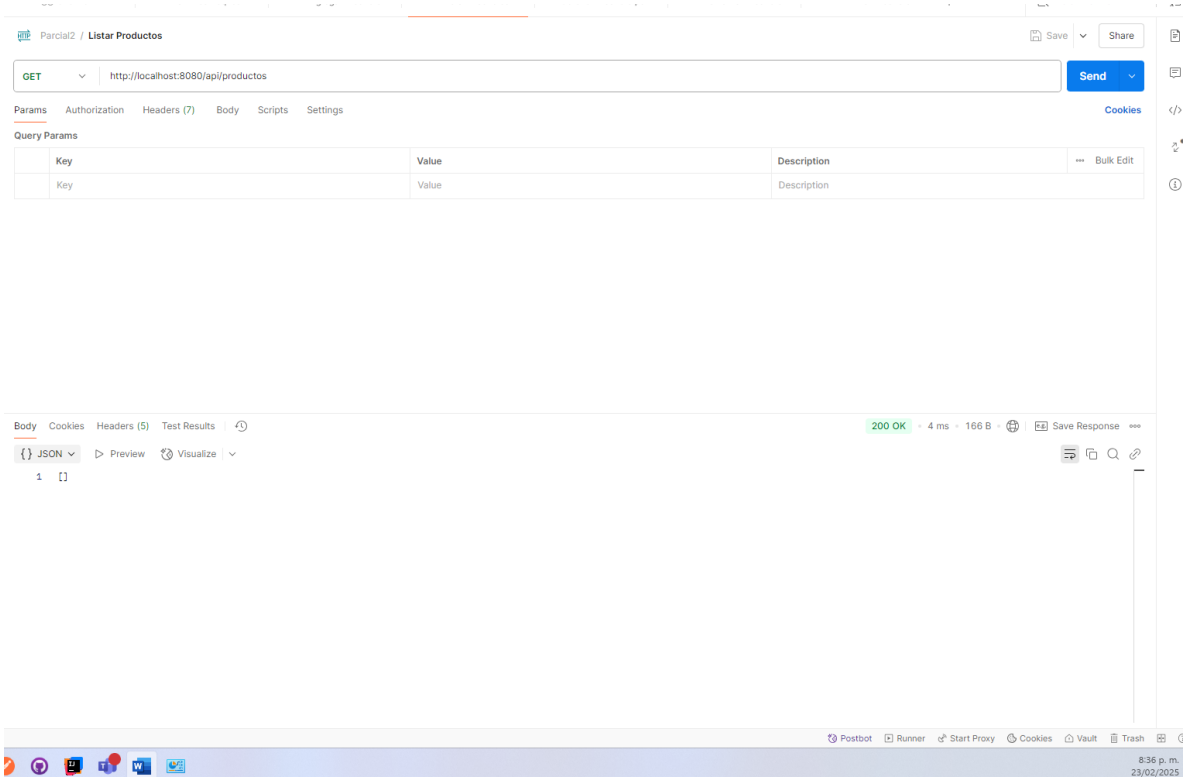
Actualizar Producto



Eliminar producto

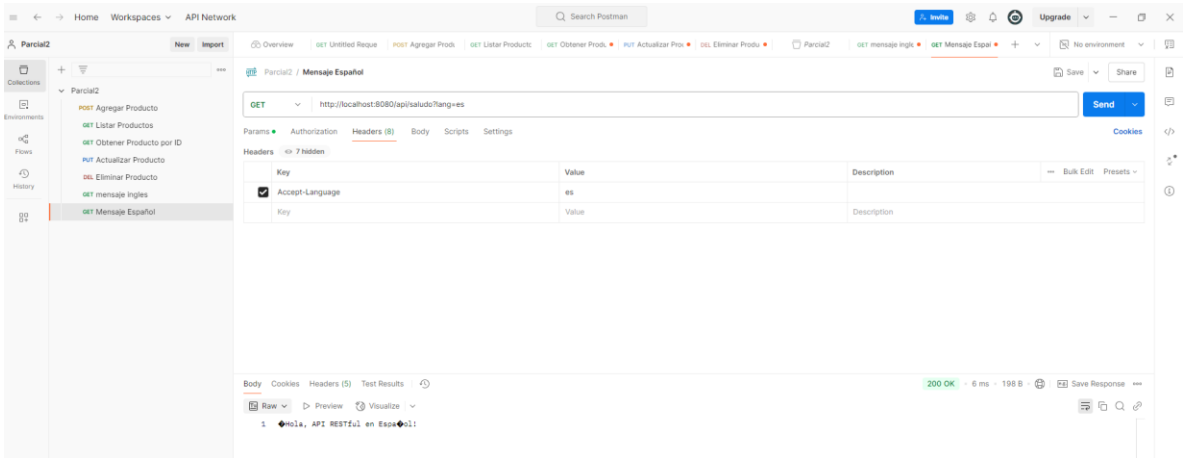


Consulta final

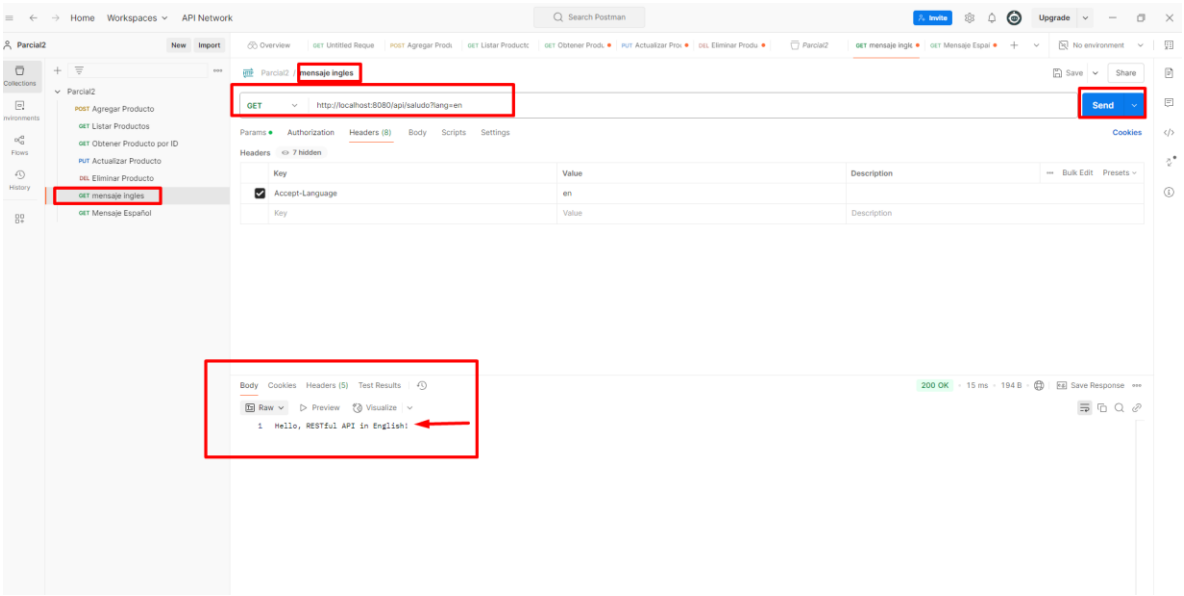


Punto 3 postman

Español



Ingles



Punto 4

Web flux

package com.example.parcial2.controller;

import com.example.parcial2.model.Producto;

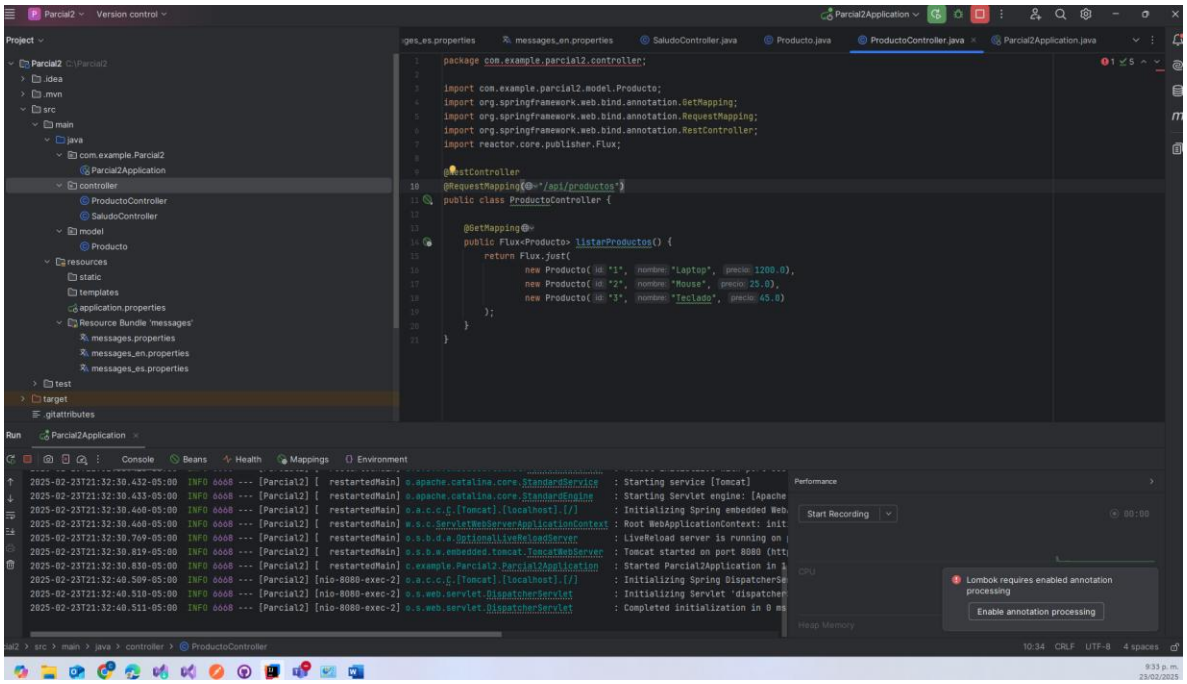
import org.springframework.web.bind.annotation.GetMapping;

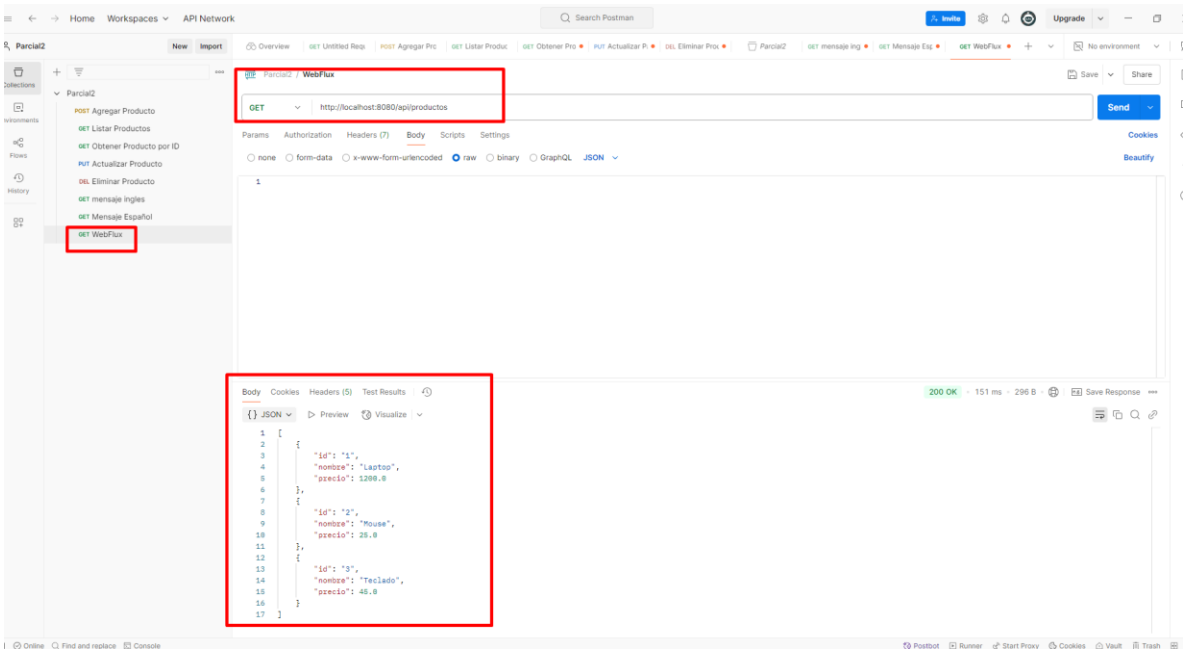
import org.springframework.web.bind.annotation.RequestMapping;

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

import reactor.core.publisher.Flux;

```
@RestController
@RequestMapping("/api/productos")
public class ProductoController {


    @GetMapping
    public Flux<Producto> listarProductos() {
        return Flux.just(
            new Producto("1", "Laptop", 1200.0),
            new Producto("2", "Mouse", 25.0),
            new Producto("3", "Teclado", 45.0)
        );
    }
}
```


Parcial2 Version control
Parcial2Application
Project
Parcial2 C:\Parcial2
.idea
.mvn
src
main
java
com.example.Parcial2
Parcial2Application
controller
ProductoController
SaludoController
model
Producto
resources
static
templates
application.properties
Resource Bundle 'messages'
messages.properties
messages_en.properties
messages_es.properties
test
target
.gitattributes
messages_es.properties
messages_en.properties
SaludoController.java
Producto.java
ProductoController.java
Parcial2Application.java
package com.example.parcial2.controller;
import com.example.parcial2.model.Producto;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import reactor.core.publisher.Flux;
@RestController
@RequestMapping("/api/productos")
public class ProductoController {
@GetMapping
public Flux<Producto> listarProductos() {
return Flux.just(
new Producto("1", "Laptop", 1200.0),
new Producto("2", "Mouse", 25.0),
new Producto("3", "Teclado", 45.0)
);
}
}
Run
Parcial2Application
Console
Beans
Health
Mappings
Environment
Performance
Start Recording
Lombok requires enabled annotation processing
Enable annotation processing
Heap Memory
10:34 CRLF UTF-8 4 spaces


Home
Workspaces
API Network
Search Postman
Upgrade
Parcial2
New
Import
POST Agregar Producto
GET Listar Productos
GET Obtener Producto por ID
PUT Actualizar Producto
DEL Eliminar Producto
GET mensaje ingles
GET Mensaje Español
GET WebFlux
Parcial2 / WebFlux
GET
http://localhost:8080/api/productos
Send
Params
Authorization
Headers (7)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
Body
Cookies
Headers (5)
Test Results
200 OK
151 ms
296 B
Save Response
{} JSON
Preview
Visualize
[
{
"id": "1",
"nombre": "Laptop",
"precio": 1200.0
},
{
"id": "2",
"nombre": "Mouse",
"precio": 25.0
},
{
"id": "3",
"nombre": "Teclado",
"precio": 45.0
}
]

Punto 5

Producto controller test

```
package com.example.Parcial2;

import com.example.parcial2.controller.ProductoController;
import com.example.parcial2.model.Producto;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.context.ApplicationContext;
import reactor.core.publisher.Flux;
import reactor.test.StepVerifier;


@SpringBootTest
public class ProductoControllerTest {

    @Autowired
    private ApplicationContext context;

    private ProductoController productoController;

    @BeforeEach
    void setUp() {
        productoController = context.getBean(ProductoController.class);
    }

    @Test
    public void testListaProductos() {
        Flux<Producto> productos = productoController.listarProductos();

        StepVerifier.create(productos)
                .expectNextMatches(p -> "Laptop".equals(p.getNombre()) && p.getPrecio() == 1200.0)
                .expectNextMatches(p -> "Mouse".equals(p.getNombre()) && p.getPrecio() == 25.0)
                .expectNextMatches(p -> "Teclado".equals(p.getNombre()) && p.getPrecio() == 45.0)
                .verifyComplete();
    }
}
```

POM

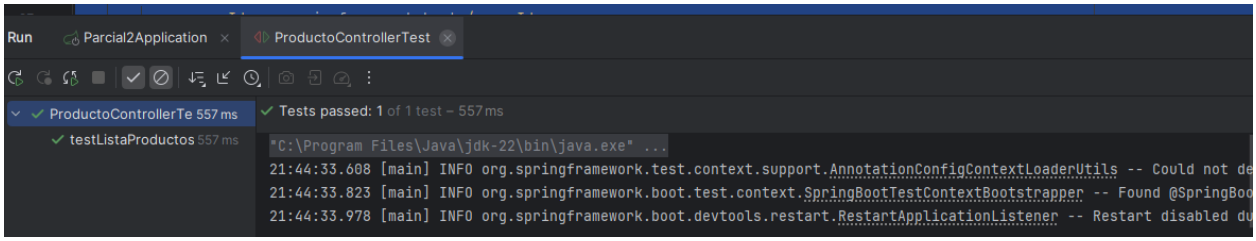
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
```

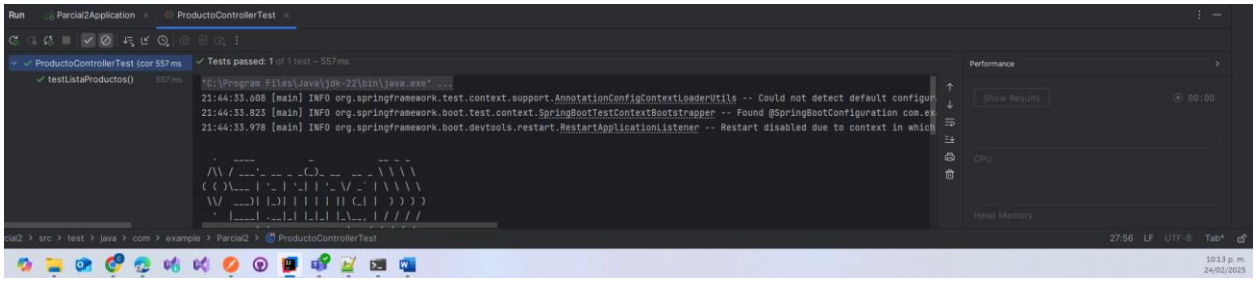
```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>3.4.3</version>
  <relativePath/>
</parent>

<groupId>com.example</groupId>
<artifactId>Parcial2</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>Parcial2</name>
<description>Demo project for Spring Boot</description>

<properties>
  <java.version>17</java.version>
  <mockito.version>5.7.0</mockito.version> <!-- Centralización de la versión de Mockito -->
</properties>

<dependencies>
  <!-- Dependencias principales -->
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-webflux</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
  </dependency>

  <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <scope>provided</scope>
  </dependency>

  <!-- Dependencias de pruebas -->
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>

  <dependency>
    <groupId>io.projectreactor</groupId>
```

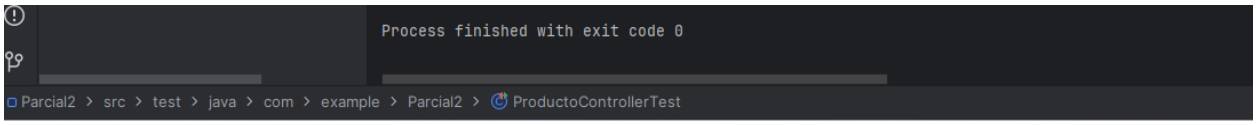
```
    <artifactId>reactor-test</artifactId>
    <scope>test</scope>
  </dependency>

  <!-- Mockito para JUnit 5 -->
  <dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-junit-jupiter</artifactId>
    <version>${mockito.version}</version>
    <scope>test</scope>
  </dependency>

  <!-- Mockito Inline (para evitar advertencias del mock-maker) -->
  <dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-inline</artifactId>
    <version>4.11.0</version>  <!-- Última versión estable -->
    <scope>test</scope>
  </dependency>
</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <configuration>
        <annotationProcessorPaths>
          <path>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
          </path>
        </annotationProcessorPaths>
      </configuration>
    </plugin>

    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
      <configuration>
        <excludes>
          <exclude>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
          </exclude>
        </excludes>
      </configuration>
    </plugin>

    <!-- Plugin para evitar la advertencia de Mockito y permitir el uso de agentes -->
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-plugin</artifactId>
```

```
            <version>3.0.0-M9</version>
            <configuration>
              <argLine>-javaagent:${settings.localRepository}/org/mockito/mockito-inline/4.11.0/mockito-inline-
4.11.0.jar -XX:+EnableDynamicAgentLoading</argLine>
            </configuration>
          </plugin>
        </plugins>
    </build>
</project>
```

Resultados punto 5







El proceso terminó con exit code 0, lo que significa que **las pruebas pasaron exitosamente**. 🚀