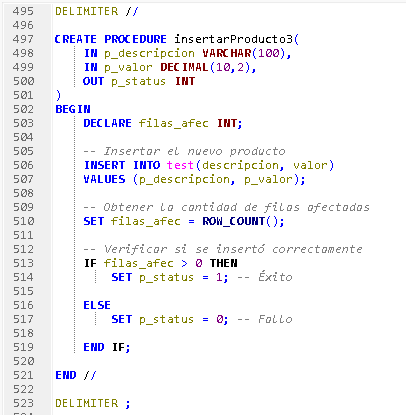
# GUIA PARA PROGRAMAR PROCEDIMIENTOS ALMACENADOS EN MARIADB Y EJECUTARLOS EN PROYECTO DE SPRINGBOOT

# 1.CREAR EL PROCEDIMIENTO ALMACENADO, EN ESTE CASO UN INSERT



# Ejemplo código procedimiento:

DELIMITER //

**CREATE** **PROCEDURE** insertarProducto3(

**IN** p\_descripcion **VARCHAR**(100),

**IN** p\_valor **DECIMAL**(10,2),

**OUT** p\_status **INT**

)

**BEGIN**

**DECLARE** filas\_afec **INT**;

*-- Insertar el nuevo producto*

**INSERT** **INTO** test(descripcion, valor)

**VALUES** (p\_descripcion, p\_valor);

*-- Obtener la cantidad de filas afectadas*

**SET** filas\_afec = **ROW\_COUNT**();

*-- Verificar si se insertó correctamente*

**IF** filas\_afec > 0 **THEN**

**SET** p\_status = 1; *-- Éxito*

**ELSE**

**SET** p\_status = 0; *-- Fallo*

**END** **IF**;

**END** //

DELIMITER ;

# Ejemplo código de los otros procedimientos:

CREATE TABLE TEST(

id INT AUTO\_INCREMENT PRIMARY KEY,

descripcion VARCHAR(100),

valor DECIMAL(10,2)

)

INSERT INTO invent.test(descripcion, valor)

VALUES

('zapatos botas brahma', 155000),

('camisa kosta azul', 125000),

('pantalon jean', 93000),

('medias de algodón t12', 30000),

('boxer tM hombre', 15000);

INSERT INTO invent.test(descripcion, valor)

VALUES

('producto prueba', 999999),

('producto prueba', 999999),

('producto prueba', 999999),

('producto prueba', 999999),

('producto prueba', 999999);

DELIMITER //

CREATE PROCEDURE obtenerProductoId(IN p\_id INT)

BEGIN

SELECT id, descripcion, valor

FROM test

WHERE id = p\_id;

END

//

CALL obtenerProductoId(1);

DELIMITER //

CREATE PROCEDURE obtenerProductoTodos()

BEGIN

SELECT id, descripcion, valor

FROM test;

END

//

CALL obtenerProductoTodos();

UPDATE test SET valor = 0 WHERE id IN (11,12);

DELIMITER //

CREATE PROCEDURE borrarProductoId(IN p\_id INT)

BEGIN

DELETE

FROM test

WHERE id = p\_id;

END

//

CALL borrarProductoId(10);

DELIMITER //

CREATE PROCEDURE modificarProducto(IN p\_id INT, p\_descripcion VARCHAR(100), p\_valor DECIMAL(10,2) )

BEGIN

UPDATE test

SET descripcion = p\_descripcion, valor = p\_valor

WHERE id = p\_id;

END

//

DELIMITER //

CREATE PROCEDURE modificarProducto2(

IN p\_id INT,

IN p\_descripcion VARCHAR(100),

IN p\_valor DECIMAL(10,2),

OUT p\_status INT

)

BEGIN

DECLARE filas\_afec INT;

-- Ejecutar la actualización

UPDATE test

SET descripcion = p\_descripcion, valor = p\_valor

WHERE id = p\_id;

-- Obtener la cantidad de filas afectadas

SET filas\_afec = ROW\_COUNT();

-- Verificar si se actualizó algún registro

IF filas\_afec > 0 THEN

SET p\_status = 1; -- Éxito

ELSE

SET p\_status = 0; -- No se encontró el registro

END IF;

END //

DELIMITER ;

CALL modificarProducto(9,'zapato de prueba', 10000);

DROP PROCEDURE insertarProducto;

DELIMITER //

CREATE PROCEDURE insertarProducto(p\_descripcion VARCHAR(100), p\_valor DECIMAL(10,2) )

BEGIN

INSERT INTO test(descripcion, valor)

VALUES (p\_descripcion, p\_valor);

END

//

CALL insertarProducto('producto de prueba', 15550);

DELIMITER //

CREATE PROCEDURE insertarProducto2(

IN p\_descripcion VARCHAR(100),

IN p\_valor DECIMAL(10,2),

OUT p\_status INT,

OUT p\_new\_id INT

)

BEGIN

DECLARE filas\_afec INT;

-- Insertar el nuevo producto

INSERT INTO test(descripcion, valor)

VALUES (p\_descripcion, p\_valor);

-- Obtener la cantidad de filas afectadas

SET filas\_afec = ROW\_COUNT();

-- Verificar si se insertó correctamente

IF filas\_afec > 0 THEN

SET p\_status = 1; -- Éxito

SET p\_new\_id = LAST\_INSERT\_ID(); -- Obtener el ID generado

ELSE

SET p\_status = 0; -- Fallo

SET p\_new\_id = NULL;

END IF;

END //

DELIMITER ;

CALL insertarProducto2('Producto de Prueba', 150.50, @status, @new\_id);

SELECT @status AS estado, @new\_id AS id\_insertado;

DELIMITER //

CREATE PROCEDURE insertarProducto3(

IN p\_descripcion VARCHAR(100),

IN p\_valor DECIMAL(10,2),

OUT p\_status INT

)

BEGIN

DECLARE filas\_afec INT;

-- Insertar el nuevo producto

INSERT INTO test(descripcion, valor)

VALUES (p\_descripcion, p\_valor);

-- Obtener la cantidad de filas afectadas

SET filas\_afec = ROW\_COUNT();

-- Verificar si se insertó correctamente

IF filas\_afec > 0 THEN

SET p\_status = 1; -- Éxito

ELSE

SET p\_status = 0; -- Fallo

END IF;

END //

DELIMITER ;

CALL insertarProducto3('Producto de Prueba', 250.5, @status);

SELECT @status AS estado;

DELIMITER //

CREATE PROCEDURE borrarProductoId2(

IN p\_id INT,

OUT p\_status INT

)

BEGIN

DECLARE filas\_afec INT;

-- Insertar el nuevo producto

DELETE FROM test

WHERE id = p\_id;

-- Obtener la cantidad de filas afectadas

SET filas\_afec = ROW\_COUNT();

-- Verificar si se insertó correctamente

IF filas\_afec > 0 THEN

SET p\_status = 1; -- Éxito

ELSE

SET p\_status = 0; -- Fallo

END IF;

END //

DELIMITER ;

SELECT \* FROM test;

CALL borrarProductoId2borrarProductoId2(16, @status);

SELECT @status AS estado;

DELIMITER //

CREATE PROCEDURE borrarProductoId3(

IN p\_id INT,

OUT p\_status INT

)

BEGIN

DECLARE filas\_afec INT;

-- Insertar el nuevo producto

DELETE FROM test

WHERE id = p\_id;

-- Obtener la cantidad de filas afectadas

SET filas\_afec = ROW\_COUNT();

-- Verificar si se insertó correctamente

IF filas\_afec > 0 THEN

SET p\_status = 1; -- Éxito

ELSE

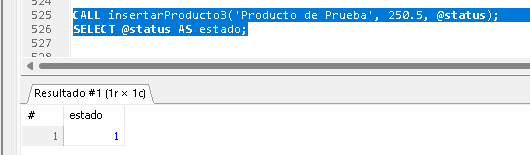
SET p\_status = 0; -- Fallo

END IF;

END //

DELIMITER ;

# 2.Se valida que ejecute correctamente en la base de datos

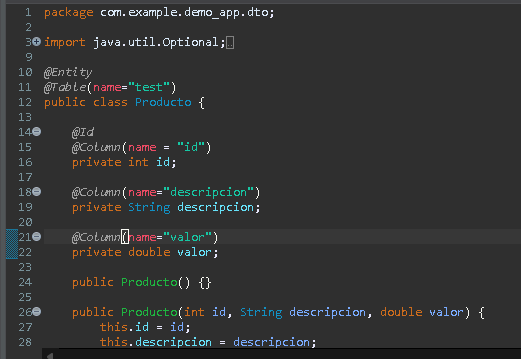


**CALL** insertarProducto3('Producto de Prueba', 250.5, @**status**);

**SELECT** @**status** **AS** estado;

# 3.Se crea el modelo de datos en la clase “Producto”.





# Código completo clase producto:

package com.example.demo\_app.dto;

import java.util.Optional;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table(name="test")

public class Producto {

@Id

@Column(name = "id")

private int id;

@Column(name="descripcion")

private String descripcion;

@Column(name="valor")

private double valor;

public Producto() {}

public Producto(int id, String descripcion, double valor) {

this.id = id;

this.descripcion = descripcion;

this.valor = valor;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getDescripcion() {

return descripcion;

}

public void setDescripcion(String descripcion) {

this.descripcion = descripcion;

}

public double getValor() {

return valor;

}

public void setValor(double valor) {

this.valor = valor;

}

@Override

public String toString() {

return "Producto [id=" + id + ", descripcion=" + descripcion + ", valor=" + valor + "]";

}

public Optional<Producto> map(Object object) {

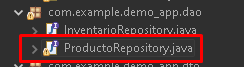
// TODO Auto-generated method stub

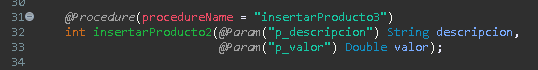
return null;

}

}

# 4.Se crea el método en la interfaz Repository





# Código completo interfaz repository:

package com.example.demo\_app.dao;

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

import org.springframework.data.jpa.repository.query.Procedure;

import org.springframework.data.repository.query.Param;

import org.springframework.stereotype.Repository;

import com.example.demo\_app.dto.Producto;

import jakarta.persistence.EntityManager;

import jakarta.persistence.ParameterMode;

import jakarta.persistence.PersistenceContext;

import jakarta.persistence.StoredProcedureQuery;

@Repository

public interface ProductoRepository extends JpaRepository<Producto, Integer> {

@Procedure(name = "obtenerProductoPorId")

Producto obtenerProductoId(@Param("p\_id") int id);

@Procedure(procedureName = "modificarProducto")

void modificarProducto(@Param("p\_id") int id,

@Param("p\_descripcion") String descripcion,

@Param("p\_valor") Double valor);

@Procedure(procedureName = "modificarProducto2")

int modificarProducto2(@Param("p\_id") int id,

@Param("p\_descripcion") String descripcion,

@Param("p\_valor") Double valor);

@Procedure(procedureName = "insertarProducto3")

int insertarProducto2(@Param("p\_descripcion") String descripcion,

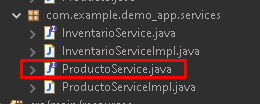
@Param("p\_valor") Double valor);

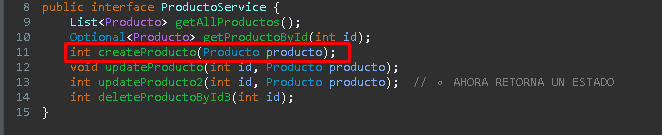
@Procedure(name = "borrarProductoId3")

int borrarProductoId3(@Param("p\_id") int id);

}

# 5.Se crea el método en la interfaz de service





# Código completo interfaz service:

package com.example.demo\_app.services;

import java.util.List;

import java.util.Optional;

import com.example.demo\_app.dto.Producto;

public interface ProductoService {

List<Producto> getAllProductos();

Optional<Producto> getProductoById(int id);

int createProducto(Producto producto);

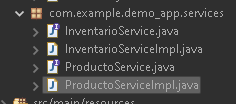
void updateProducto(int id, Producto producto);

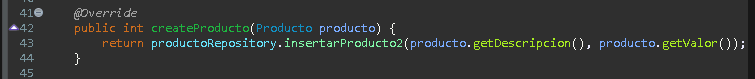
int updateProducto2(int id, Producto producto); // 🔹 AHORA RETORNA UN ESTADO

int deleteProductoById3(int id);

}

# 6.Se crea el método en el service implementado





# Código completo clase service implementado:

package com.example.demo\_app.services;

import java.util.List;

import java.util.Optional;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.example.demo\_app.dao.ProductoRepository;

import com.example.demo\_app.dto.Producto;

@Service

public class ProductoServiceImpl implements ProductoService {

@Autowired

public ProductoRepository productoRepository;

Logger logger = LoggerFactory.getLogger(InventarioServiceImpl.class);

@Override

public List<Producto> getAllProductos() {

return productoRepository.findAll();

}

/\*

@Override

public Optional<Producto> getProductoById(int id) {

return Optional.ofNullable(productoRepository.obtenerProductoId(id));

}

}\*/

@Override

@Transactional(readOnly = true) // 🔹 ¡AÑADIDO AQUÍ!

public Optional<Producto> getProductoById(int id) {

return Optional.ofNullable(productoRepository.obtenerProductoId(id));

}

@Override

public int createProducto(Producto producto) {

return productoRepository.insertarProducto2(producto.getDescripcion(), producto.getValor());

}

/\*

@Override

public Producto updateProducto(int id, Producto producto) {

return productoRepository.obtenerProductoId(id).map(existingProducto -> {

existingProducto.setDescripcion(producto.getDescripcion());

existingProducto.setValor(producto.getValor());

return productoRepository.save(existingProducto);

}).orElseThrow(() -> new RuntimeException("Producto no encontrado"));

}\*/

@Override

@Transactional // 🔹 REQUERIDO PARA ACTUALIZAR

public void updateProducto(int id, Producto producto) {

productoRepository.modificarProducto(id, producto.getDescripcion(), producto.getValor());

}

@Override

@Transactional // 🔹 REQUERIDO PARA ACTUALIZAR

public int updateProducto2(int id, Producto producto) {

return productoRepository.modificarProducto2(id, producto.getDescripcion(), producto.getValor());

}

@Override

@Transactional // 🔹 REQUERIDO PARA ACTUALIZAR

public int deleteProductoById3(int id) {

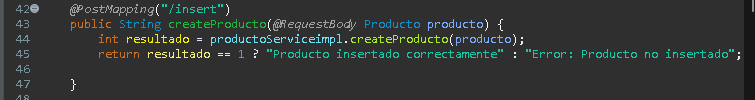
return productoRepository.borrarProductoId3(id);

}

}

# 7.Se crea el método en la clase controller.





# Código completo de la clase controller:

package com.example.demo\_app.controllers;

import java.util.List;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;

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

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

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

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

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

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

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

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

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

import com.example.demo\_app.dto.Producto;

import com.example.demo\_app.services.ProductoServiceImpl;

@RestController

@RequestMapping("/productos")

@CrossOrigin(origins = "\*")

public class ProductoController {

@Autowired

public ProductoServiceImpl productoServiceimpl;

@GetMapping("/all")

public List<Producto> getAllProductos() {

return productoServiceimpl.getAllProductos();

}

@GetMapping("/{id}")

public Optional<Producto> getProductoById(@PathVariable int id) {

return productoServiceimpl.getProductoById(id);

}

@PostMapping("/insert")

public String createProducto(@RequestBody Producto producto) {

int resultado = productoServiceimpl.createProducto(producto);

return resultado == 1 ? "Producto insertado correctamente" : "Error: Producto no insertado";

}

/\*

@PutMapping("/update/{id}")

public void updateProducto(@PathVariable int id, @RequestBody Producto producto) {

productoServiceimpl.updateProducto(id, producto);

}

\*/

@PutMapping("/update2/{id}")

public String updateProducto2(@PathVariable int id, @RequestBody Producto producto) {

int resultado = productoServiceimpl.updateProducto2(id, producto);

return resultado == 1 ? "Producto actualizado correctamente" : "Error: Producto no encontrado";

}

@PutMapping("/delete/{id}") // 🔹 Cambia de GET a DELETE

public String deleteProducto(@PathVariable int id) {

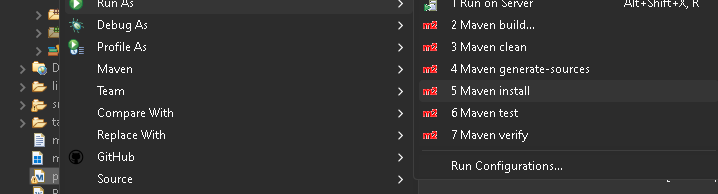
int resultado = productoServiceimpl.deleteProductoById3(id);

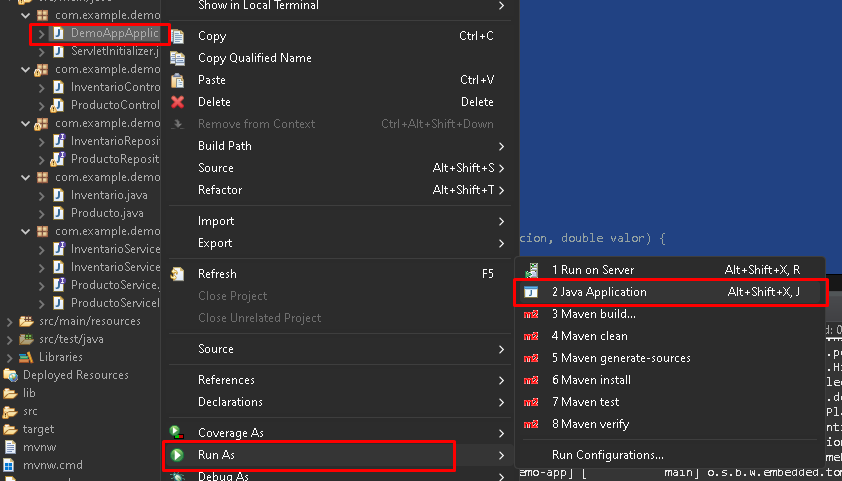
return resultado == 1 ? "Producto eliminado correctamente" : "Error: Producto no eliminado";

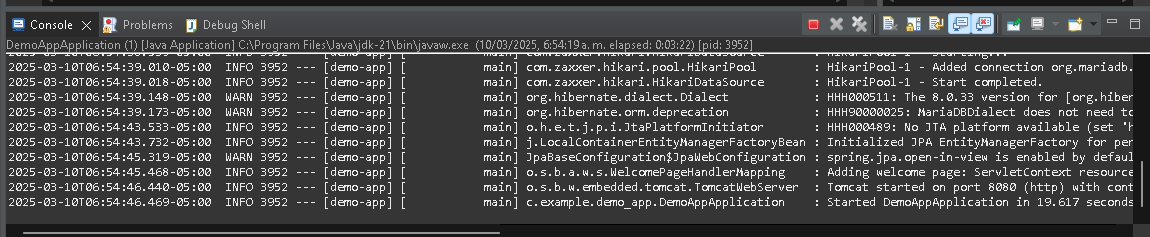
}

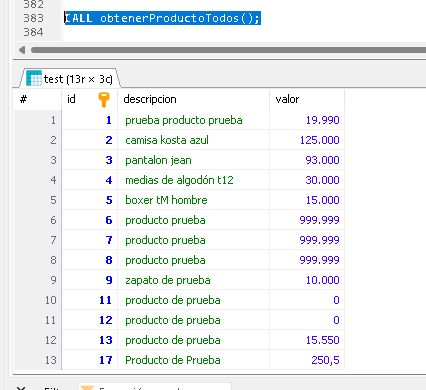
}

# 7.Se compila con maven clean, Maven install, se ejecuta la clase app y se prueba con postman

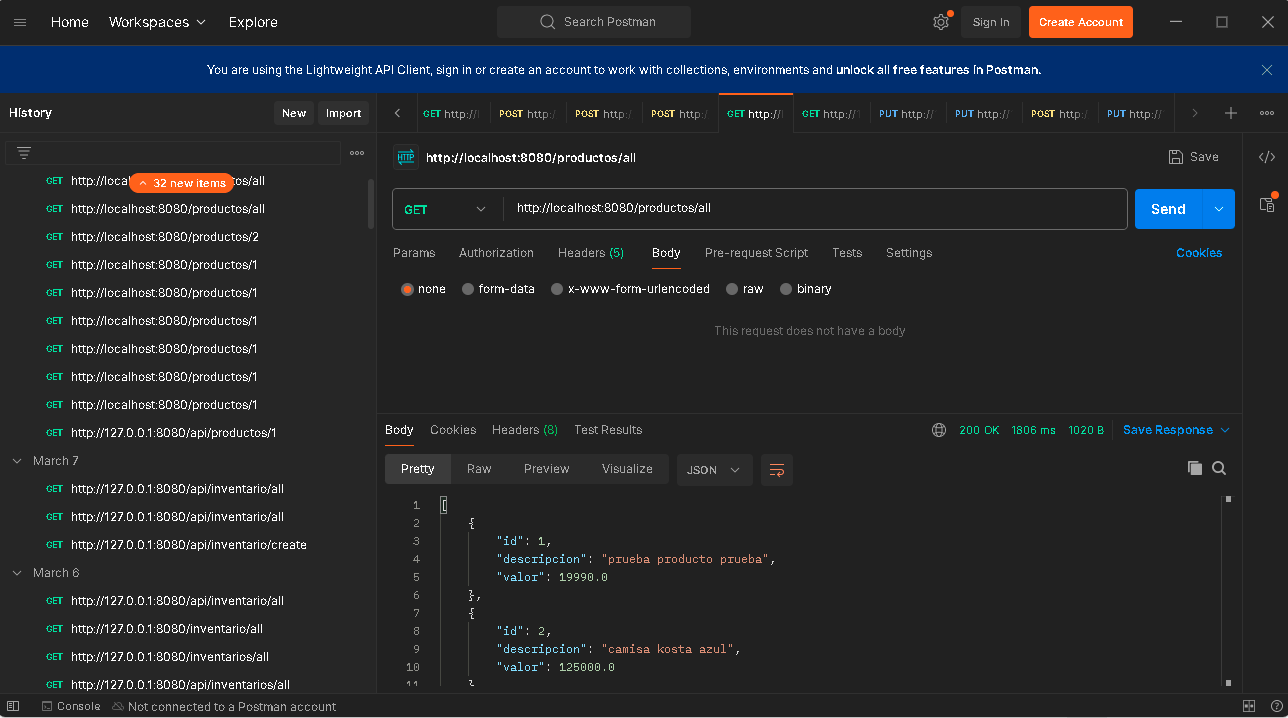




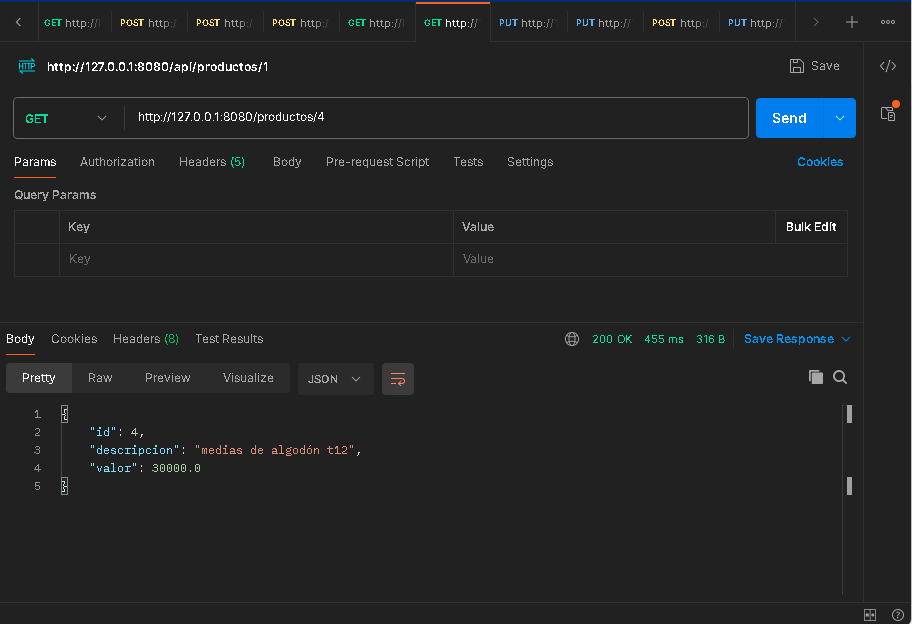




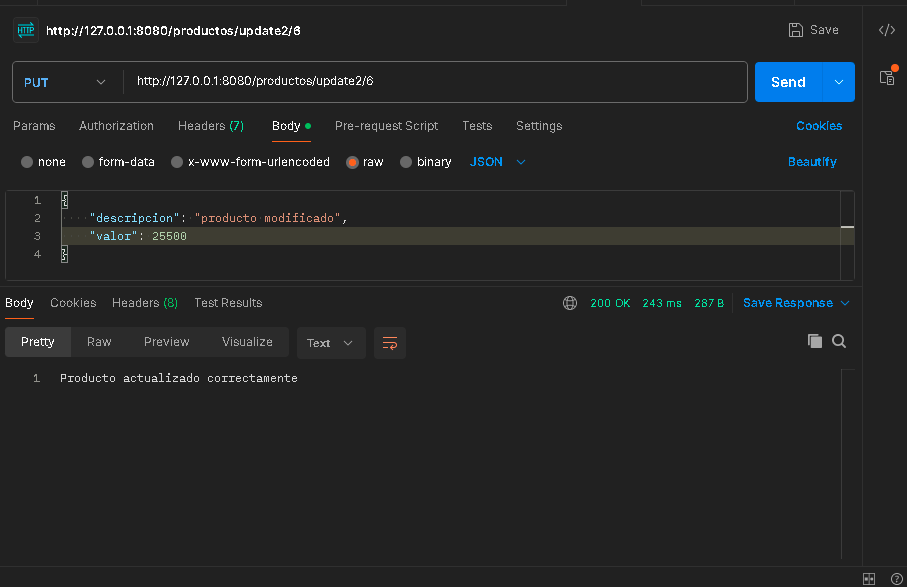
# Ejecución procedimiento para traer todos los registros de la tabla:

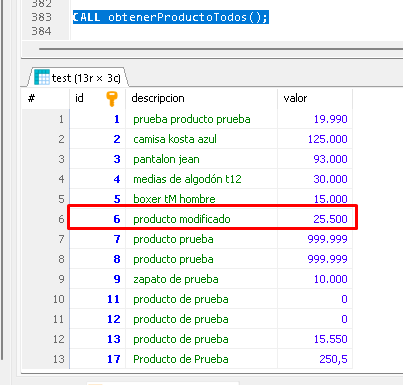


# Ejecución procedimiento para traer un registro de la tabla:

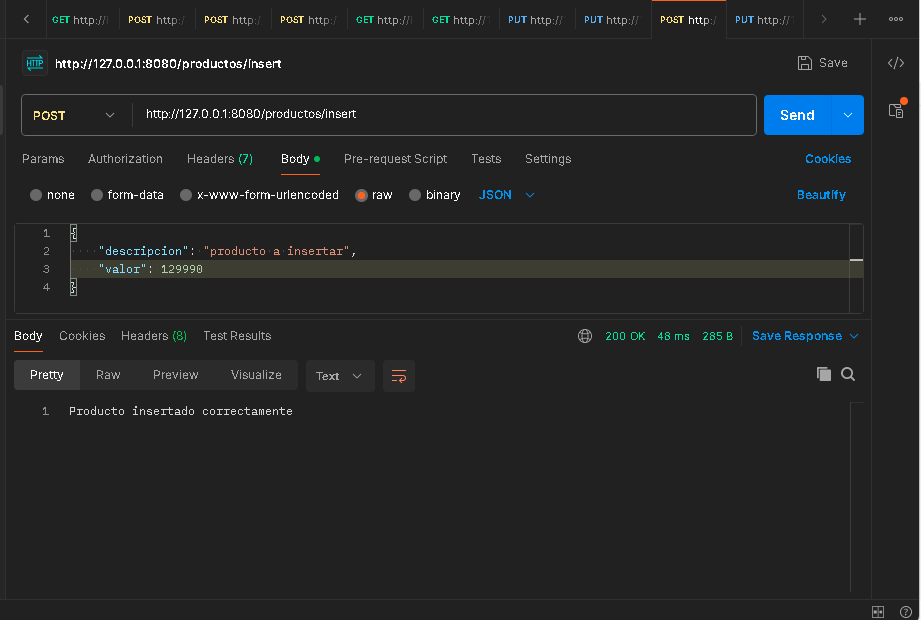


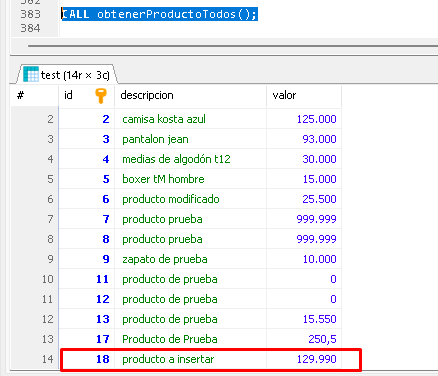
# Ejecución postman para modificar un registro:





# Ejecución postman para insertar un registro:





Ejecución postman para eliminar un registro:

