PRUEBA SPRING BOOT

Nombre: Kheiner Ivan Murillo Mosquera

NOTA: Para ejecutar la aplicación, Ejecutar el comando “docker compose up” en la ruta relativa del repositorio, donde se encuentra ubicado el archivo “compose.yml”

Repositorio:

- https://github.com/kimmi1206/santander-prueba-backend

API Endpoints:

- */api/productos/by-id/{productoId}* (Get Single Product)

- */api/productos/by-nombre*  (Find Product By Name)

- */api/productos/all*  (Get All Products)

- */api/productos/save* (Save/Update Product)

- */api/productos/delete/{productoId}* (Delete Product)

OpenAPI Endpoint:

- http://localhost:8080/swagger-ui/index.html

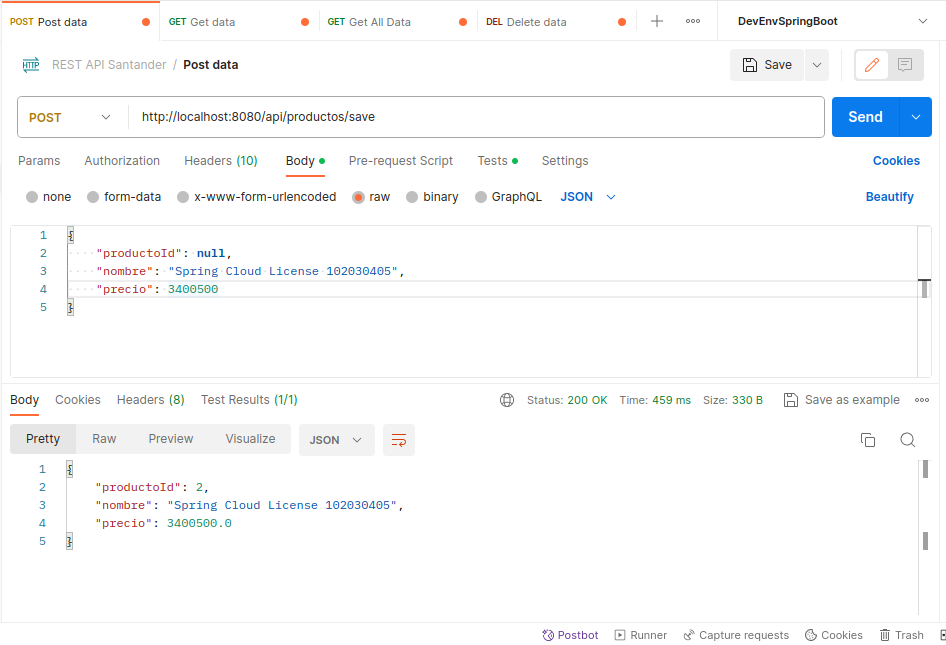
Explicación Código Fuente:

En el **Modelo**, se define la entidad “**Producto**” con los campos:

* ***productoId*** (Autogenerado)
* ***nombre*** (Unico, Entre 3 y 100 caracteres, NO nulo, vacio o en blanco)
* ***precio*** (Mayor o Igual a 0)

En el **Controlador** se definen las plantillas URL del CRUD (**ProductoController**):

* ***Obtener Por ID*** *(****getById****)*: Recibe la variable de ruta “*productoId*” y se invoca el **Servicio** **ProductoService** con el parametro recibido (*productoId*). El Servicio *ProductoService* busca el producto con el id recibido, si existe, retorna el objeto, si no existe, arroja una excepción indicando que el producto no ha sido encontrado (*ProductNotFoundException*) con codigo Http 404 Not Found.
* ***Obtener Todos los Productos*** *(****getAll****)*: Retorna una lista con los productos encontrados, o una lista vacia si no se encontraron productos.
* ***Guardar Producto*** *(****save****):* Recibe un objeto Producto en formato Json y se valida que cumpla con las restricciones definidas en la entidad *Producto*, si no cumple con las restricciones, se arroja una excepción “*ConstraintViolationException*” con codigo Http 400 Bad Request y se retorna una respuesta que incluye un arreglo con los errores generados. Si cumple con las restricciones, se invoca el servicio *ProductoService* con el objeto recibido en la peticion (*WebRequest*). Si el producto ya existe, se arroja una excepción “*ProductoAlreadyExistsException*” con código Http 409 Conflict, Si no existe, se guarda el producto en el repositorio y se devuelve el objeto. Si ocurre un error al guardar el objeto en el repositorio, se arroja la excepción “*ProductoNotSavedException*” con codigo http 500 Internal Server Error.



* ***Eliminar Producto*** *(****delete****):* Recibe el id del producto a eliminar y llama el metodo “*delete*” del servicio *ProductoService* con el parametro recibido en la peticion (*request*). Si el producto no existe, se arroja una excepcion (*ProductNotFoundException*) con codigo de respuesta Http 404 Not Found. Si el producto existe, se arroja una excepcion *ProductoNotDeletedException* con codigo http 500 Internal Server Error. Si el producto fue eliminado, seretorna un booleano (*productDeleted*) en formato Json.