



# CRUD REST MongoDB

Spring Boot & MongoDB



Academia Java

Entregable

Natalia Esquivel Ocadiz

12 de septiembre 2025

## Descripción del proyecto:

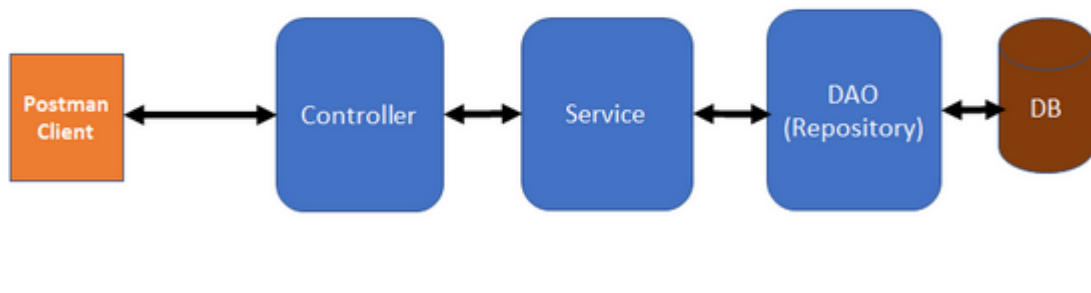
Este proyecto implementa las operaciones CRUD (Create, Read, Update, Delete) utilizando SpringBoot como framework backend y MongoDB como base de datos NoSQL. Además MongoDB se encuentra en un “contenedor” gracias a la plataforma Docker. La aplicación permite gestionar productos de una tienda, la entidad de productos tiene atributos como nombre, descripción, precio y cantidad disponible en stock, a través de una API rest.

## Tecnologías utilizadas

- Java 17
- SpringBoot 3.5.5
- Spring Web
- Spring Boot Dev Tools
- Spring Data MongoDB
- MongoDB
- Maven
- IDE: Eclipse
- Docker

## Arquitectura de aplicación REST

### Spring Data REST Application Architecture



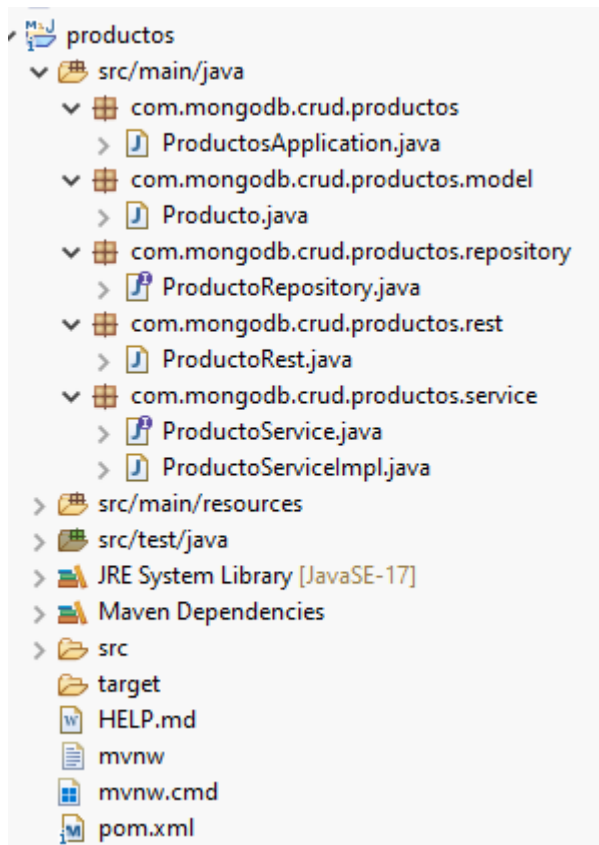
## Desarrollo del proyecto

1. Se crea una base de datos en MongoDB el cual se encuentra en un contenedor de la plataforma Docker, ayudando a que el proyecto sea portable, la Base de datos se llamará Productos, y la colección (collection) tendrá el mismo nombre (productos).
2. Se crea el proyecto Maven con Spring Initializr agregando las dependencias necesarias como Spring Data MongoDB para conectar la base de datos, Spring Web para construir las aplicaciones REST, y Spring Boot Dev Tools para proveer una mejor experiencia en el desarrollo de la aplicación. Se genera e importa el proyecto a la IDE.

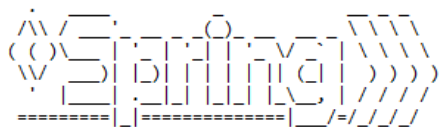
3. Se agrega la configuración MongoDB al documento application.properties
4. La estructura del proyecto incluye capas para ayudar a mantener el código más limpio y modular:
  - a. Model: Representa la colección y los atributos de la colección como una clase Java, las anotaciones utilizadas son @Document (indica la colección), @Id (el ObjectId único en MongoDB), y @Indexed (para la búsqueda de campos).
  - b. Repository: Se define la interfaz que extiende a MongoRepository, al extender este repositorio, Spring Boot nos proporciona automáticamente todos los métodos CRUD básicos sin necesidad de escribir código adicional.
  - c. Service: Se define la lógica del negocio con una interface ProductoService el cual define las operaciones y una clase ProductoServiceImpl donde se define cómo se implementarán las operaciones definidas en la interface Service.
  - d. RestController: Se define la clase que actúa como punto de entrada de la API, el @RestController mediante Spring Boot nos permite el mapeo de peticiones HTTP a métodos Java y la conversión de objetos a JSON para nuestra base de datos
5. Se realizan las pruebas de que las operaciones CRUD se lleven a cabo correctamente. También se revisa que la conexión a la base de datos de MongoDB haya sido exitosa.

### **Capturas de pantalla**

Proyecto en Eclipse: Se muestra la estructura del proyecto mostrando los packages y archivos principales



Ejecución de la Aplicación: Se muestra la consola mostrando que la aplicación corrió exitosamente



:: Spring Boot :: (v3.5.5)

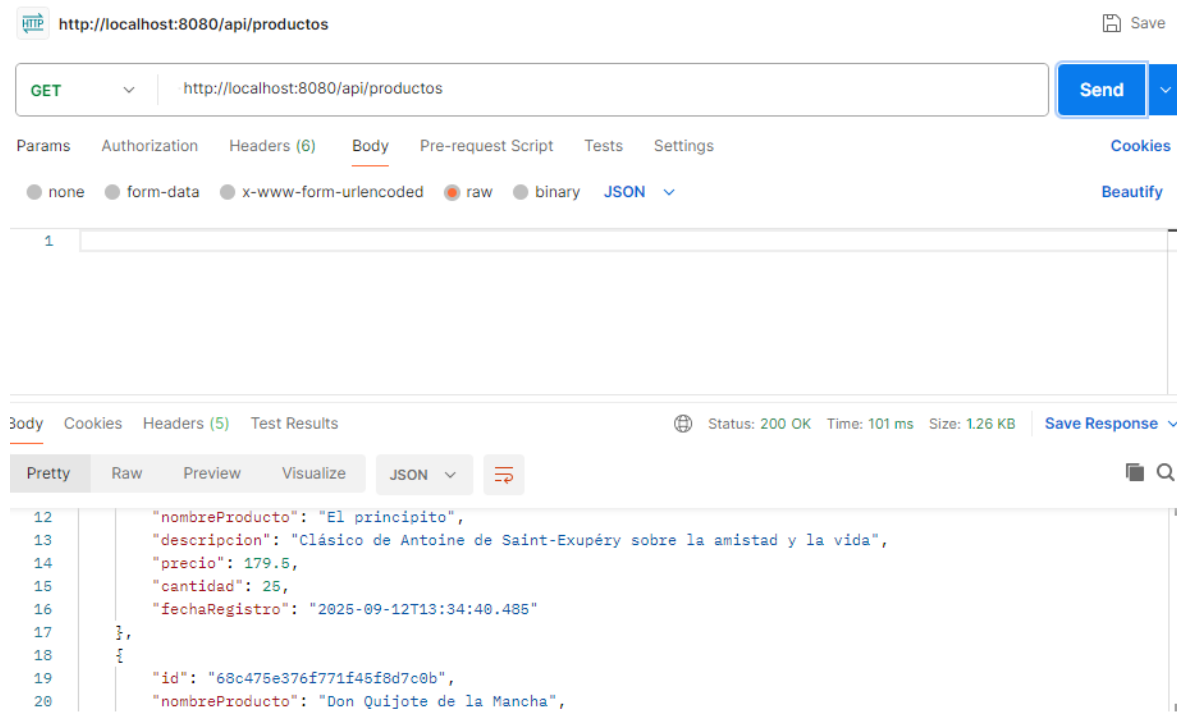
```

2025-09-12T12:29:56.213-06:00 INFO 33084 --- [productos] [ restartedMain] c.m.crud.productos
2025-09-12T12:29:56.217-06:00 INFO 33084 --- [productos] [ restartedMain] c.m.crud.productos
2025-09-12T12:29:56.296-06:00 INFO 33084 --- [productos] [ restartedMain] .e.DevToolsPropert
2025-09-12T12:29:56.296-06:00 INFO 33084 --- [productos] [ restartedMain] .e.DevToolsPropert
2025-09-12T12:29:57.272-06:00 INFO 33084 --- [productos] [ restartedMain] .s.d.r.c.Repositor
2025-09-12T12:29:57.389-06:00 INFO 33084 --- [productos] [ restartedMain] .s.d.r.c.Repositor
2025-09-12T12:29:58.014-06:00 INFO 33084 --- [productos] [ restartedMain] o.s.b.w.embedded.t
2025-09-12T12:29:58.055-06:00 INFO 33084 --- [productos] [ restartedMain] o.apache.catalina.
2025-09-12T12:29:58.055-06:00 INFO 33084 --- [productos] [ restartedMain] o.apache.catalina.
2025-09-12T12:29:58.105-06:00 INFO 33084 --- [productos] [ restartedMain] o.a.c.c.C.[Tomcat]
2025-09-12T12:29:58.106-06:00 INFO 33084 --- [productos] [ restartedMain] w.s.c.ServletWebSe
2025-09-12T12:29:58.399-06:00 INFO 33084 --- [productos] [ restartedMain] org.mongodb.driver
2025-09-12T12:29:58.431-06:00 INFO 33084 --- [productos] [localhost:27017] org.mongodb.driver
2025-09-12T12:29:58.591-06:00 INFO 33084 --- [productos] [ restartedMain] o.s.b.d.a.Optional
2025-09-12T12:29:59.190-06:00 INFO 33084 --- [productos] [ restartedMain] o.s.b.w.embedded.t
2025-09-12T12:29:59.200-06:00 INFO 33084 --- [productos] [ restartedMain] c.m.crud.productos

```

## Pruebas de API en Postman: Se muestra las operaciones CRUD funcionando

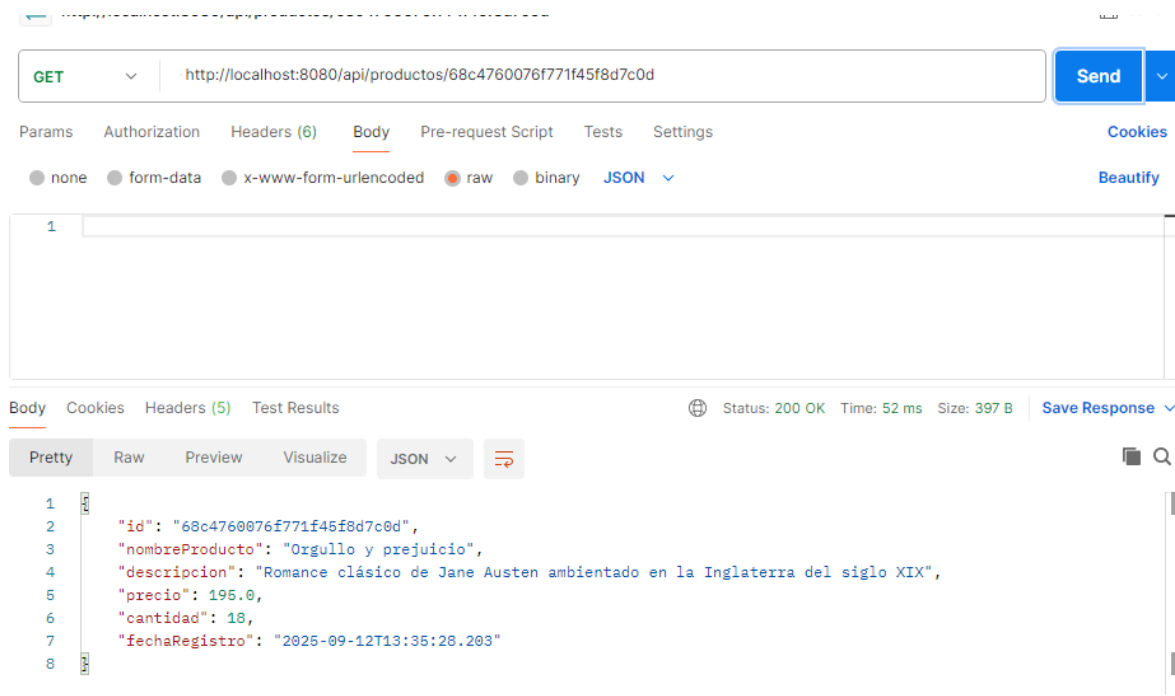
### - Get (Read)



A screenshot of the Postman interface showing a GET request to `http://localhost:8080/api/productos`. The request is successful, returning a 200 OK status. The response body is displayed in JSON format, showing an array of two product objects. The first object is for 'El principito' and the second is for 'Don Quijote de la Mancha'.

```
1 [{"nombreProducto": "El principito",  
  "descripcion": "Clásico de Antoine de Saint-Exupéry sobre la amistad y la vida",  
  "precio": 179.5,  
  "cantidad": 25,  
  "fechaRegistro": "2025-09-12T13:34:40.485"},  
  {"id": "68c475e376f771f45f8d7c0b",  
  "nombreProducto": "Don Quijote de la Mancha",
```

### - GetOne (Read)



A screenshot of the Postman interface showing a GET request to `http://localhost:8080/api/productos/68c4760076f771f45f8d7c0d`. The request is successful, returning a 200 OK status. The response body is displayed in JSON format, showing a single product object for 'Orgullo y prejuicio'.

```
1 {"id": "68c4760076f771f45f8d7c0d",  
2   "nombreProducto": "Orgullo y prejuicio",  
3   "descripcion": "Romance clásico de Jane Austen ambientado en la Inglaterra del siglo XIX",  
4   "precio": 195.0,  
5   "cantidad": 18,  
6   "fechaRegistro": "2025-09-12T13:35:28.203"}
```

## - Post (Create)

HTTP **POST** http://localhost:8080/api/productos Save

**POST** http://localhost:8080/api/productos Send

Params Authorization Headers (7) **Body** Pre-request Script Tests Settings Cookies Beautify

● none ● form-data ● x-www-form-urlencoded ● raw ● binary **JSON** ▼

```
1
```

Body Cookies Headers (5) Test Results 201 Created 86 ms 405 B Save Response ▼

Pretty Raw Preview Visualize **JSON** ▼

```
1 {
2   "id": "68c4753476f771f45f8d7c09",
3   "nombreProducto": "Cien años de soledad",
4   "descripcion": "Novela de Gabriel García Márquez, obra maestra del realismo mágico",
5   "precio": 299.99,
6   "cantidad": 15,
7   "fechaRegistro": "2025-09-12T13:32:04.1733012"
8 }
```

## - Put (Update)

**PUT** http://localhost:8080/api/productos/68c4760076f771f45f8d7c0d Send

Params Authorization Headers (10) **Body** Pre-request Script Tests Settings Cookies Beautify

● none ● form-data ● x-www-form-urlencoded ● raw ● binary **JSON** ▼

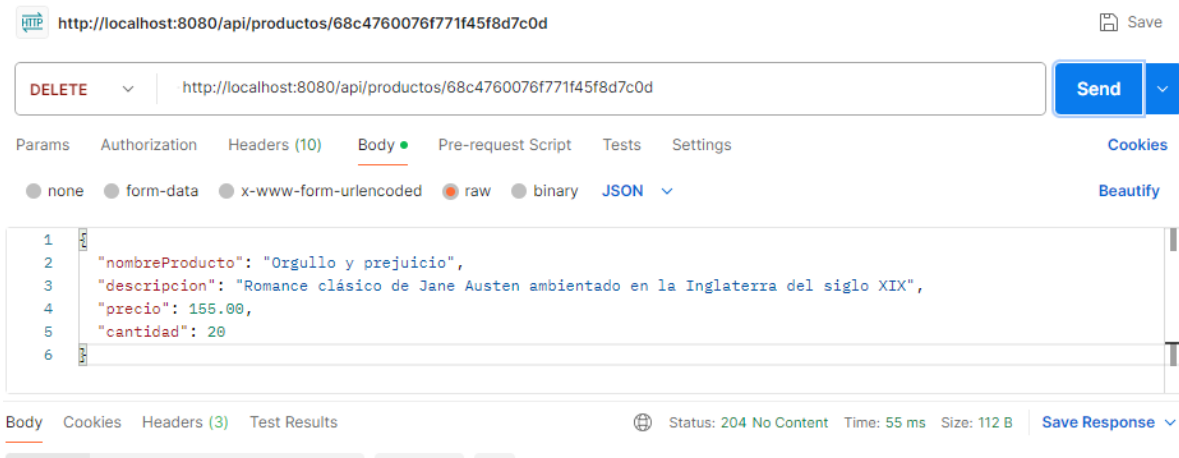
```
1
2 "nombreProducto": "Orgullo y prejuicio",
3 "descripcion": "Romance clásico de Jane Austen ambientado en la Inglaterra del siglo XIX",
4 "precio": 155.00,
5 "cantidad": 20
6
```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 39 ms Size: 397 B Save Response ▼

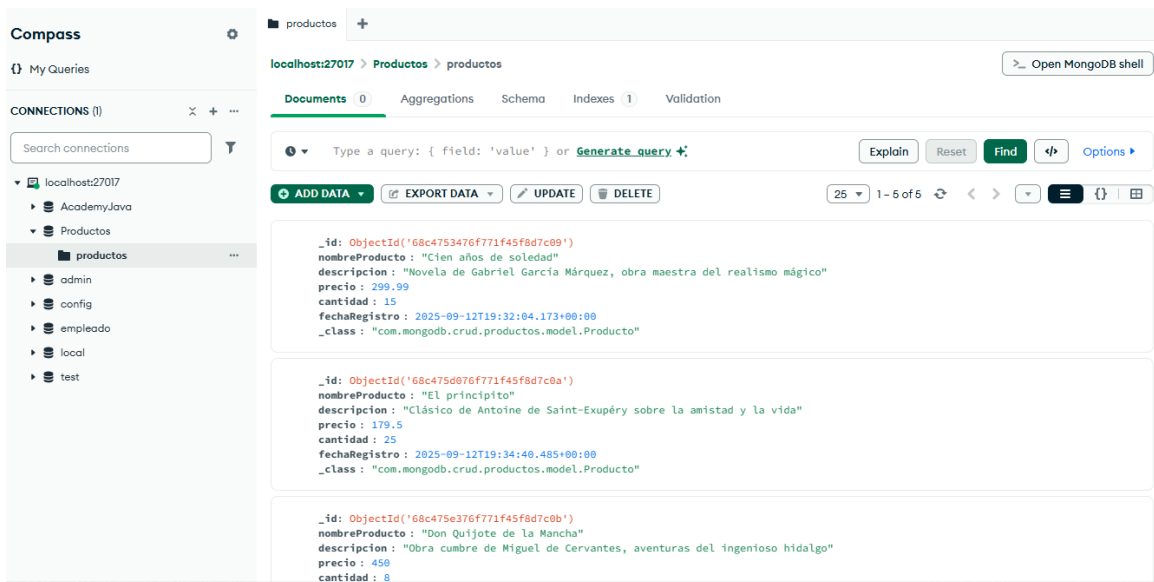
Pretty Raw Preview Visualize **JSON** ▼

```
1 {
2   "id": "68c4760076f771f45f8d7c0d",
3   "nombreProducto": "Orgullo y prejuicio",
4   "descripcion": "Romance clásico de Jane Austen ambientado en la Inglaterra del siglo XIX",
5   "precio": 155.0,
6   "cantidad": 20,
7   "fechaRegistro": "2025-09-12T13:35:28.203"
8 }
```

## - Delete (Delete)



## Base de datos en MongoDB



## Funcionalidades Implementadas

- Create: Crea nuevos productos
- Read: Consulta todos los productos y por Id
- Update: Actualiza los productos existentes
- Delete: Elimina productos

## Endpoints de la API

Método	Endpoint	Descripción
GET	/api/productos	Obtiene todos los productos
GET	/api/productos/{id}	Obtiene producto por Id

POST	/api/productos	Crea un nuevo producto
PUT	/api/productos/{id}	Actualiza producto
DELETE	/api/productos/{id}	Elimina producto

## Conclusiones

El proyecto demuestra la implementación exitosa de: Integración Spring Boot con MongoDB, Integración de APO REST siguiendo buenas prácticas, y operaciones CRUD completas.