

INFORME DE TRABAJO AUTÓNOMO

1. Datos Generales:

Carrera:	Tecnología Superior en Desarrollo de Software
Período académico:	Mayo – Octubre 2022
Asignatura:	Tendencias Actuales de Programación
Unidad N°:	3
Tema:	Despliegue de una Api-Rest en cloud Atlas de MongoDB
Ciclo-Paralelo:	M5B
Estudiante:	Mayra Peñañiel
Docente:	Ing. Diego Cale
Fecha:	05/09/2022

2. Contenido:

2.1 Introducción:

El gestor de base de datos MongoDB se lo puede asociar a un conjunto de gestores de bases de datos que no tienen como lenguaje principal el SQL para su manipulación. Los gestores de bases de datos NoSQL no requieren estructuras fijas como tablas, normalmente no soportan operaciones join y presentan como gran ventaja que pueden escalar en forma sencilla. Heroku es una solución de Plataforma como Servicio (PaaS) basada en la nube para que el cliente solo se preocupe de desarrollar su aplicación mientras Heroku se encarga de la infraestructura que hay detrás. Para proporcionar este servicio se dispone de unos contenedores virtuales que son los encargados de mantener y ejecutar las aplicaciones. Estos contenedores virtuales son totalmente escalables bajo demanda. Tanto en número como en capacidades. Una ventaja de elegir Heroku es su capacidad de soportar múltiples lenguajes de programación. Los principales a utilizar son: Node.js, Ruby, Python, Java, PHP, Go, Scala y Clojure. Aunque esta cantidad de lenguajes puede aumentar en el caso de utilizar Heroku Buildpacks, que permiten compilar las aplicaciones en multitud de ellos más

2.2 Objetivo:

Entender el manejo de una base de datos NoSQL y como funciona mediante el gestor de datos MongoDB

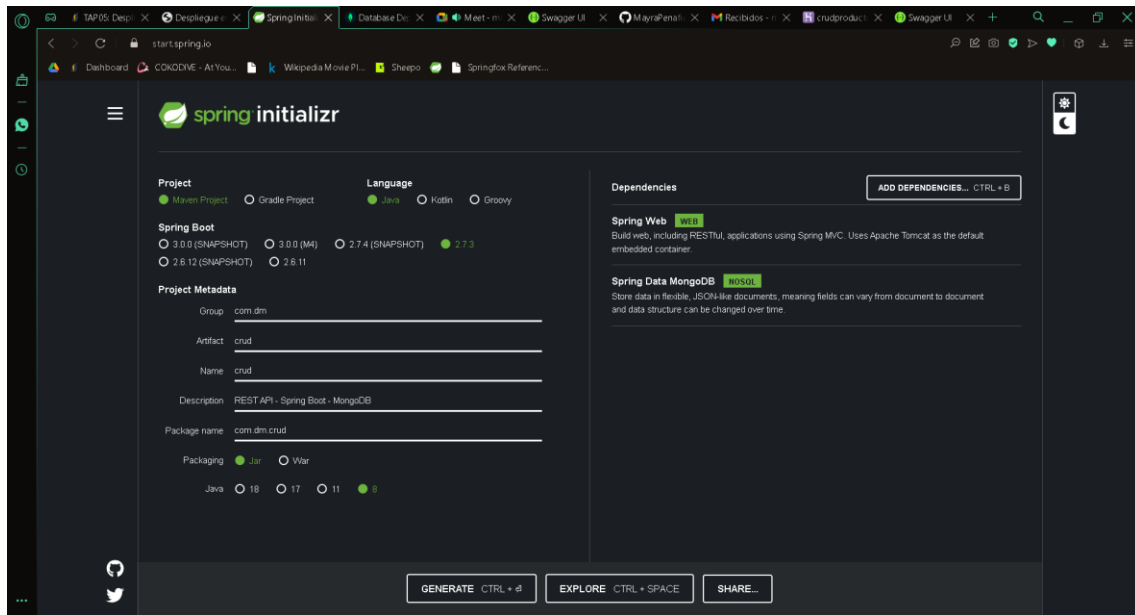
2.3 Materiales, herramientas, equipos y software:

Computador personal, IDE NetBeans, Lenguaje de Programación Java, MongoDB, Spring Initializer, Heroku, Swagger, MongoDB Compass.

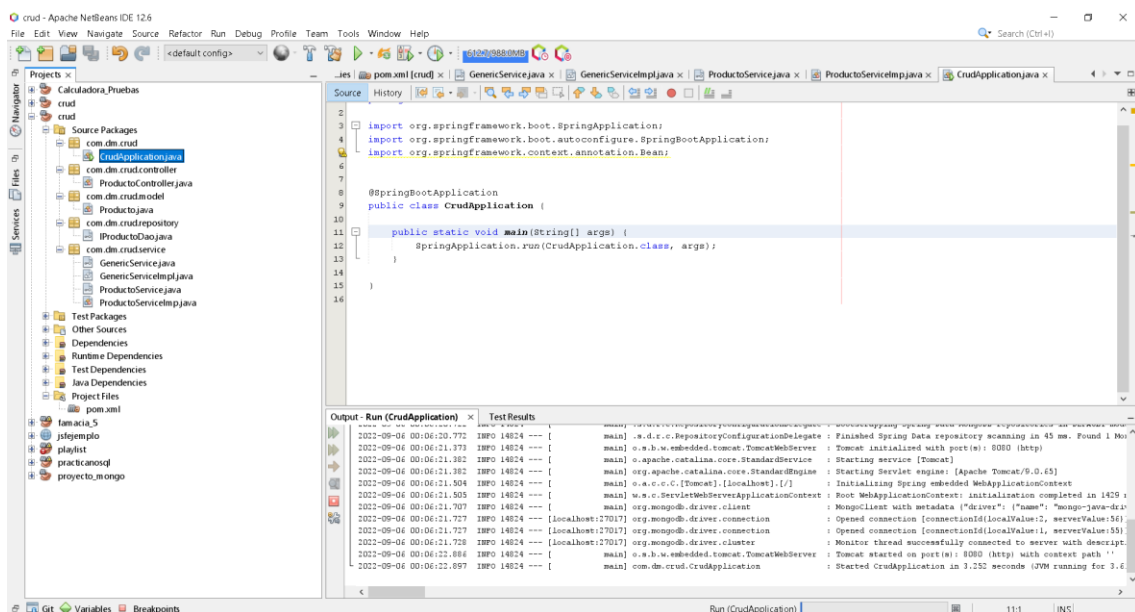
2.4 Procedimiento

- Creación de la Api:

Mediante la herramienta: <https://start.spring.io> creamos el proyecto en Maven y lo descargamos.

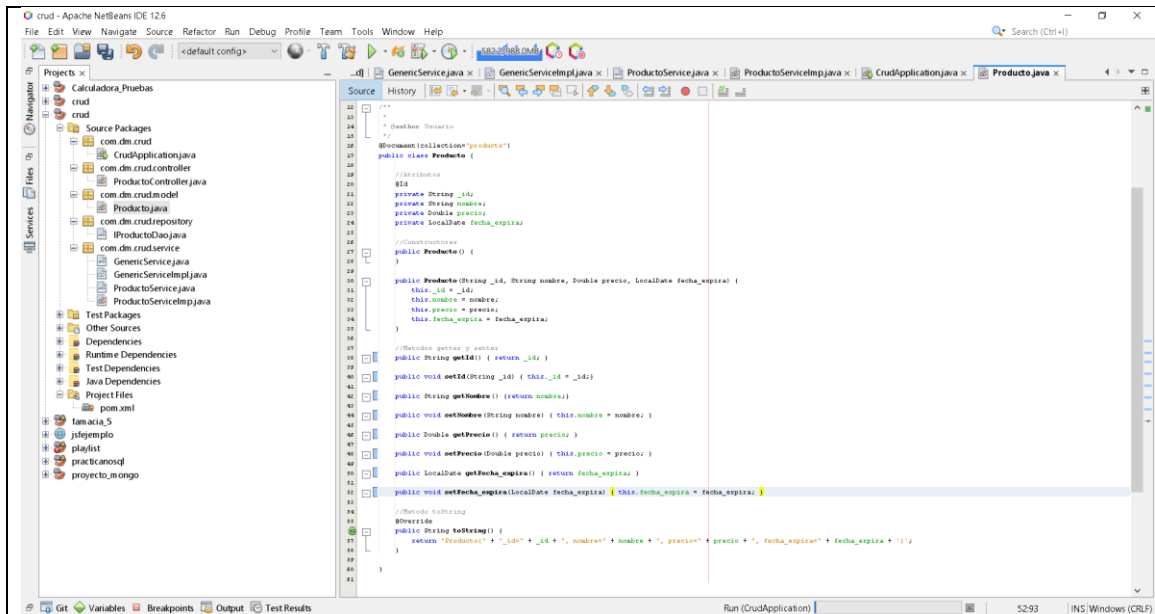


Una vez creado el proyecto abrirlo en su IDE de preferencia, en este caso NetBeans, para este caso se ha creado una Api de Productos con todas las capas necesarias: Controlador, Repository, Service y Model.



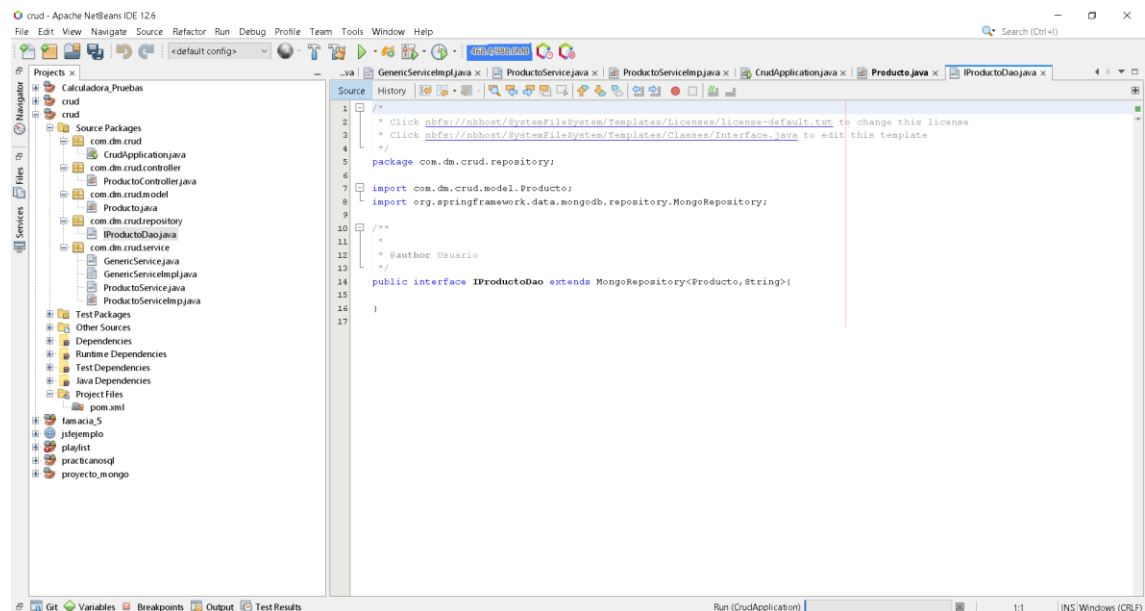
Paquete Modelo:

Clase Producto:



Paquete Repository:

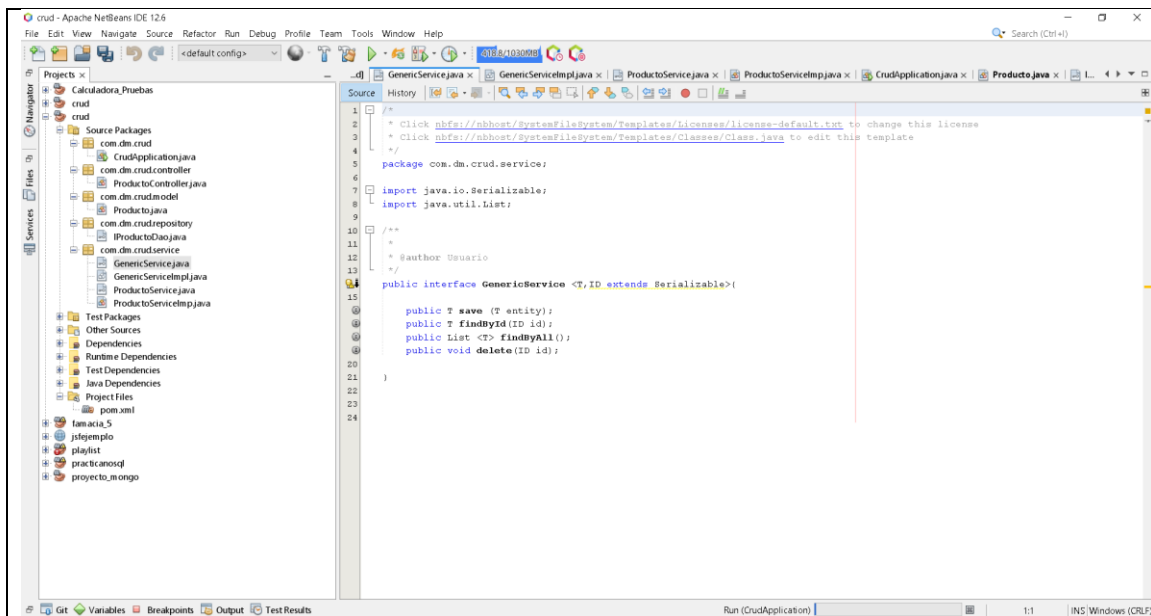
Clase IProductoDao:



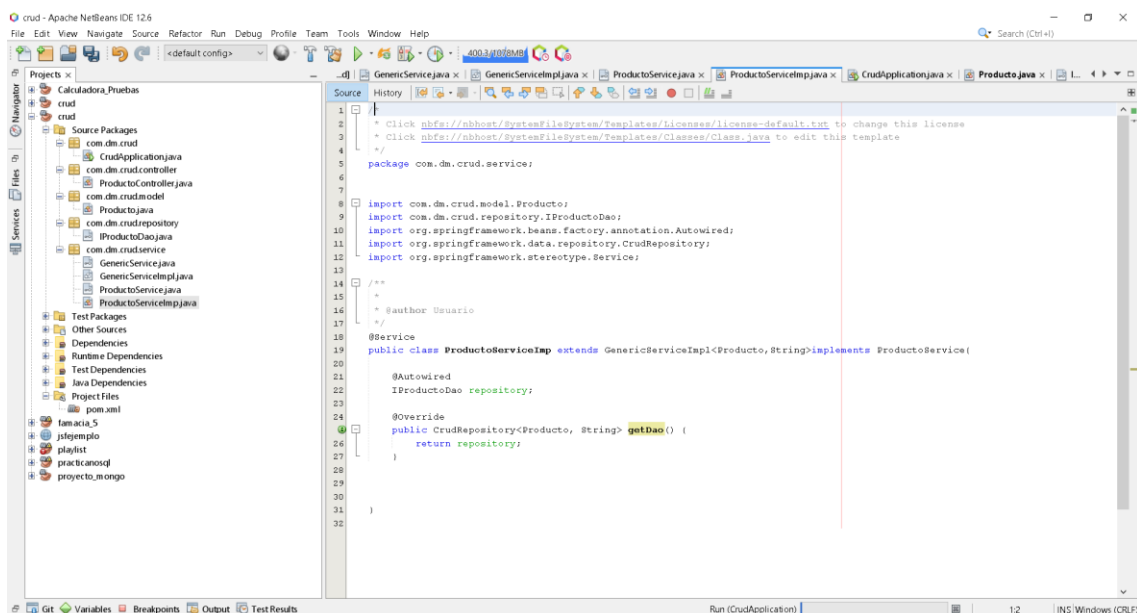
Package Service:

Classes:

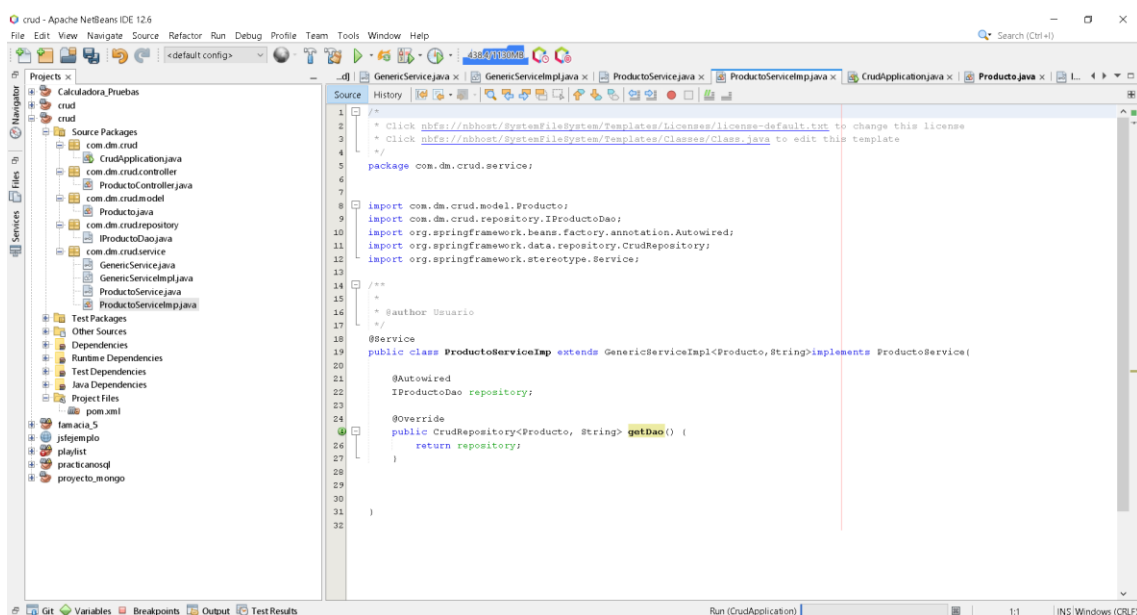
- GenericService:



- GenericServiceImpl:



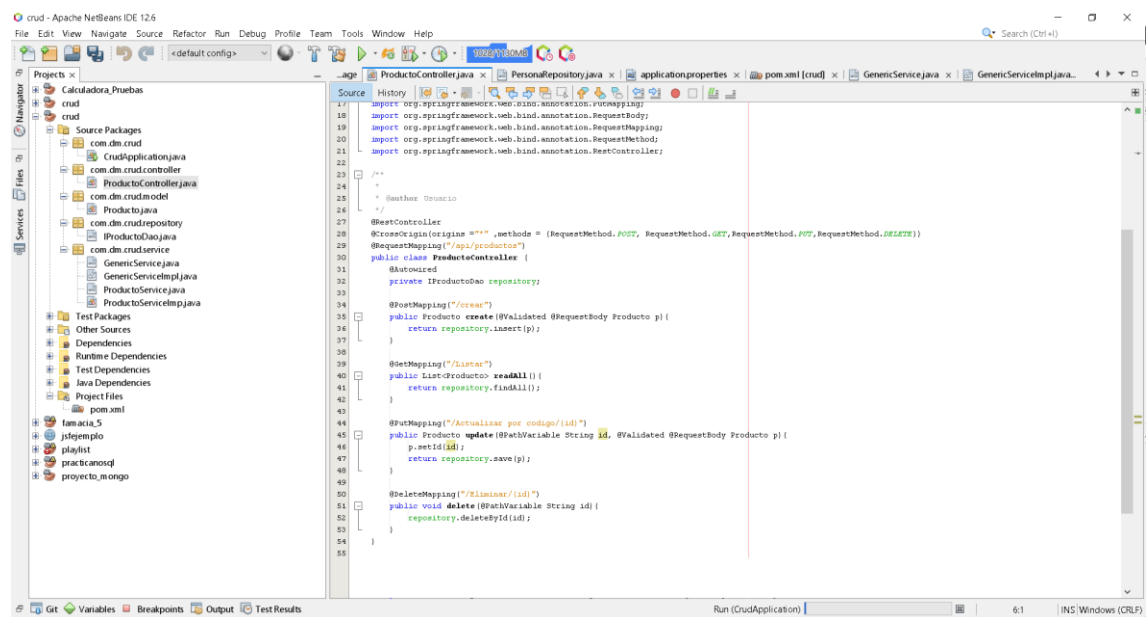
- ProductoServiceImpl:



The screenshot displays the Apache NetBeans IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, and Help. The toolbar contains icons for various development actions. The left sidebar shows the 'Projects' and 'Files' views. The 'Projects' view lists a project named 'crud' with a source package 'com.dm.crud'. The 'Files' view shows the project structure, including source packages, test packages, and other files. The main editor area displays the source code of the 'ProductService.java' file. The code defines a package 'com.dm.crud.service', imports 'com.dm.crud.model.Producto', and declares a public interface 'ProductService' that extends 'GenericService<Producto, String>'. The interface includes a comment for the author 'Uguario'.

```
1  *  
2  * Click https://nhoat/bytes/filesystem/templates/licenses/license-default.txt to change this license  
3  * Click https://nhoat/bytes/filesystem/templates/classes/class.java to edit this template  
4  */  
5  package com.dm.crud.service;  
6  
7  import com.dm.crud.model.Producto;  
8  
9  
10  
11  /**  
12   * @author Uguario  
13   */  
14  @public interface ProductService extends GenericService<Producto, String> {  
15  
16  
17  }  
18
```

Clase ProductoController:



- **Dependencias:**

```
<dependency>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-compiler-plugin</artifactId>
  <version>3.10.1</version>
</dependency>
```

- **Propiedades:**

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

- **Plugins:**

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

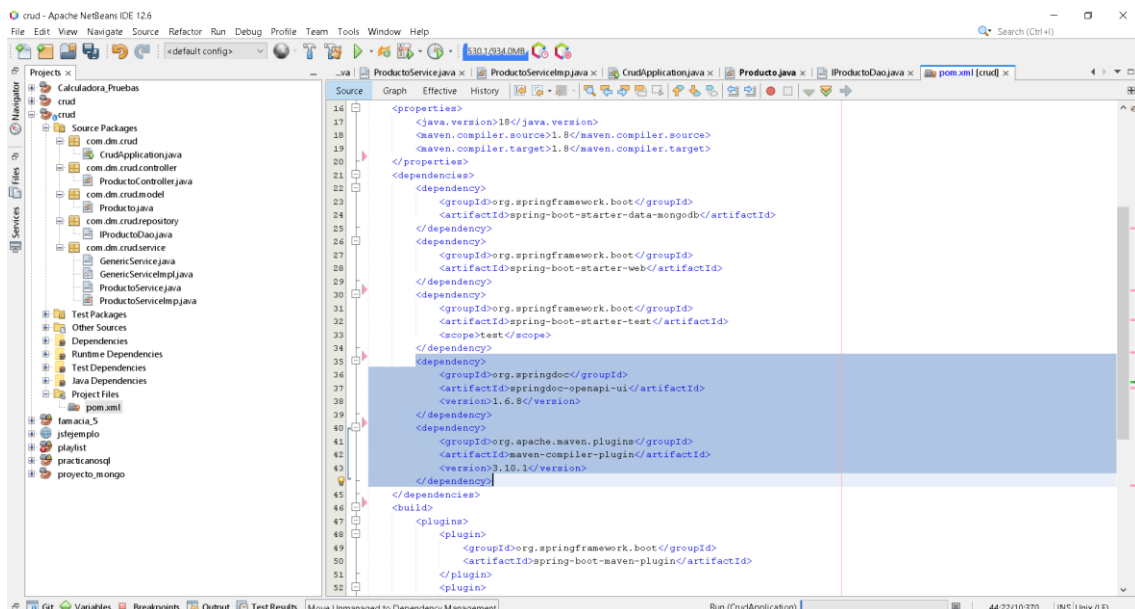
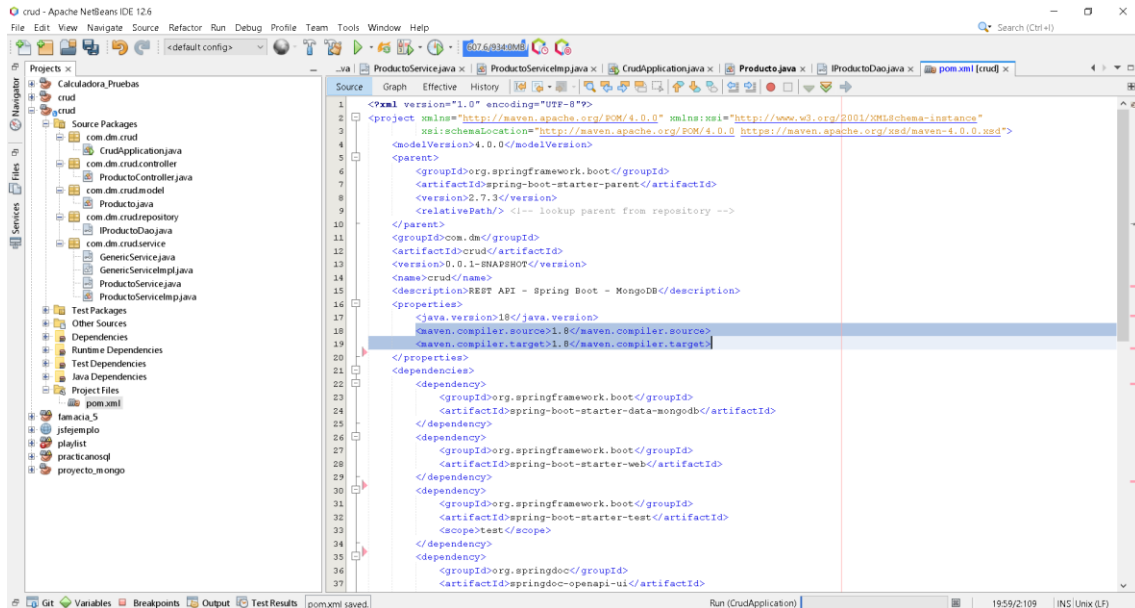
<configuration>

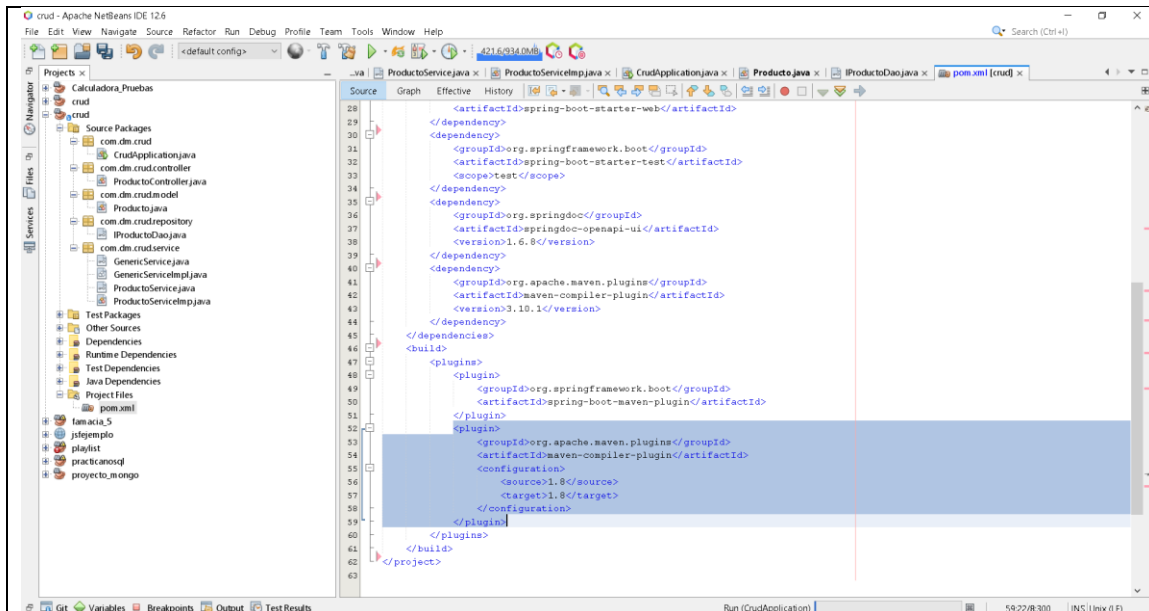
<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

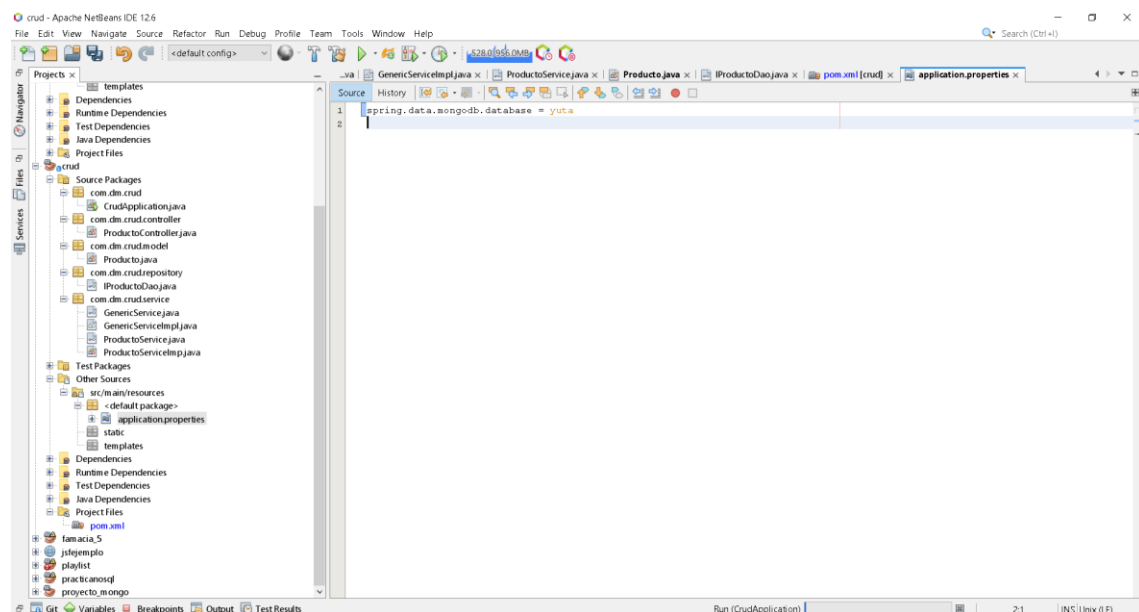


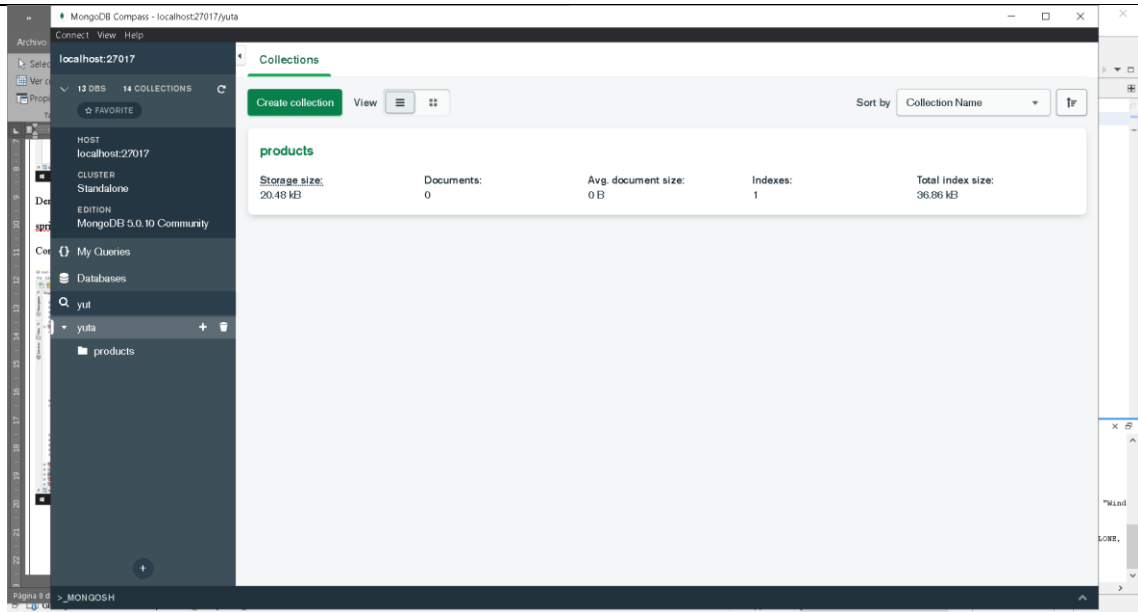


Dentro del archivo application.properties colocar:

spring.data.mongodb.database = NombreBaseDatos

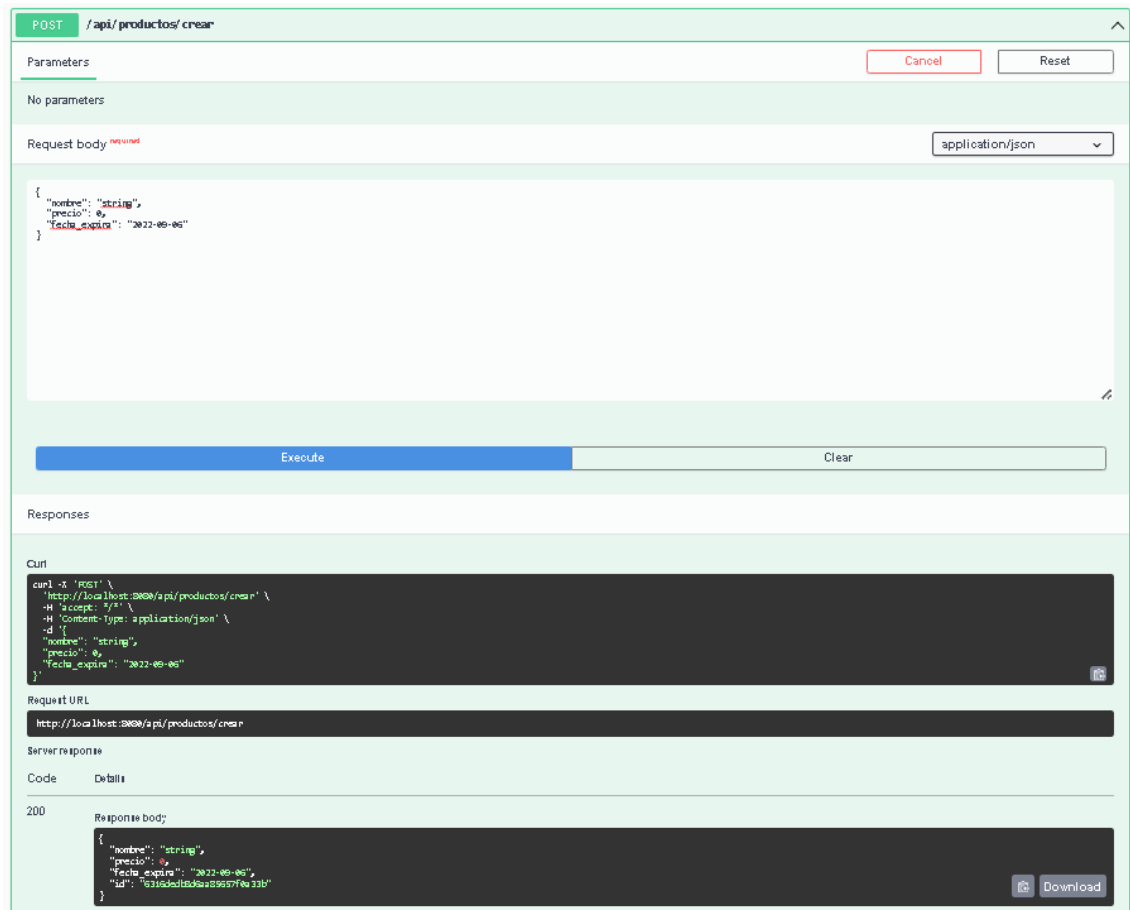
Corremos el programa y comprobamos que se haya creado la base “yuta” y la colección “Products” y Abrimos en Swagger para comprobar el funcionamiento del controlador

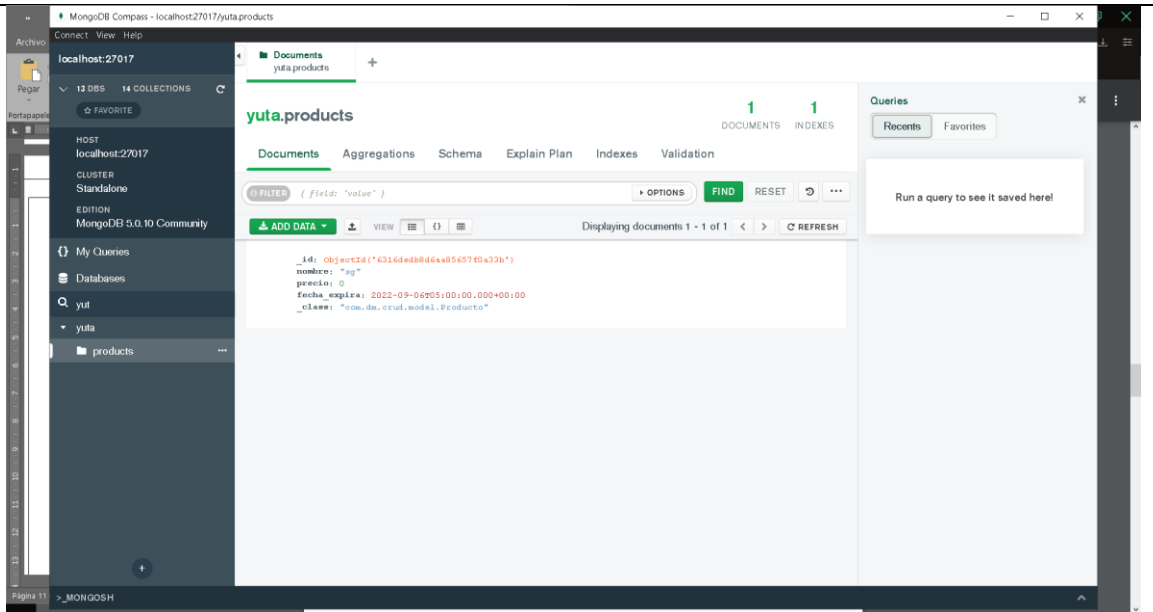




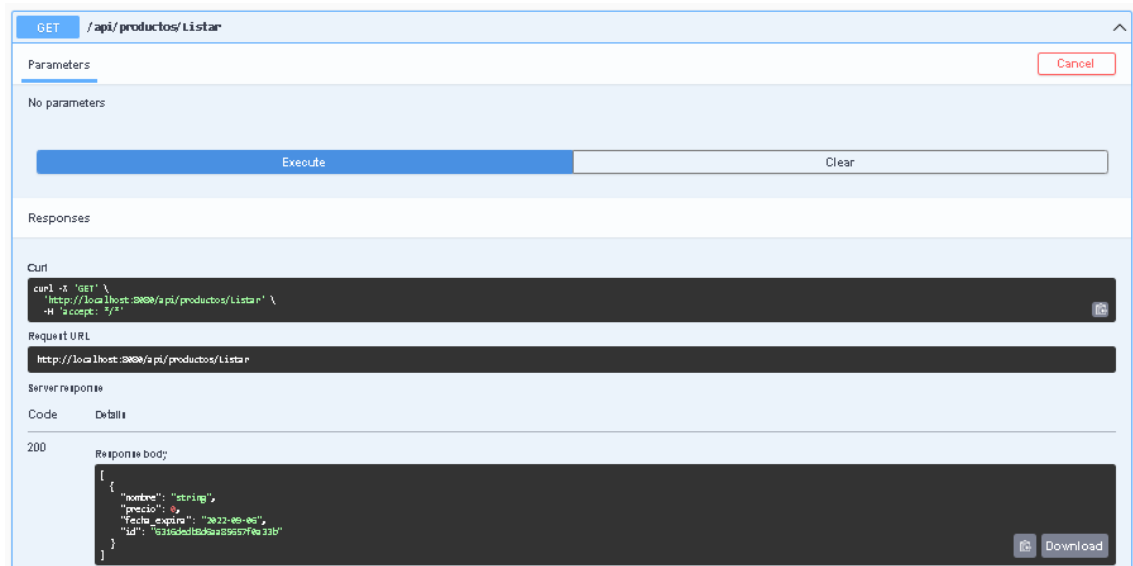
Metodos en Swagger:

- Crear:





- Listar:



- Metodo Actualizar:

PUT /api/productos/Actualizar por codigo/{id}

Parameters

Cancel Reset

Name	Description
id <small>required</small>	
string (path)	6316dedb8d6aa85657f0a33b

Request body required

application/json

```
{  "nombre": "ag",  "precio": 0,  "fecha_expira": "2022-06-06"}
```

Execute Clear

Responses

Curl

```
curl -X 'PUT' \  'http://localhost:8080/api/productos/Actualizar por codigo/6316dedb8d6aa85657f0a33b' \  -H 'accept: */*' \  -H 'Content-Type: application/json' \  -d '{  "nombre": "ag",  "precio": 0,  "fecha_expira": "2022-06-06"  }'
```

Request URL

http://localhost:8080/api/productos/Actualizar por codigo/6316dedb8d6aa85657f0a33b

Server response

Code	Details
200	Response body: <pre>{ "nombre": "ag", "precio": 0, "fecha_expira": "2022-06-06", "id": "6316dedb8d6aa85657f0a33b"}</pre>

Download

GET /api/productos/Listar

Parameters

Cancel

No parameters

Execute Clear

Responses

Curl

```
curl -X 'GET' \  'http://localhost:8080/api/productos/Listar' \  -H 'accept: */*'
```

Request URL

http://localhost:8080/api/productos/Listar

Server response

Code	Details
200	Response body: <pre>{ "nombre": "ag", "precio": 0, "fecha_expira": "2022-06-06", "id": "6316dedb8d6aa85657f0a33b"}</pre>

Download

- Eliminar

DELETE /api/productos/eliminar/{id}

Parameters

Cancel

Name	Description
id required string (path)	6316dedb8d6aa85657f0a33b

Execute

Clear

Responses

Curl

```
curl -X 'DELETE' \
  'http://localhost:8080/api/productos/eliminar/6316dedb8d6aa85657f0a33b' \
  -H 'accept: */*'
```

Request URL

http://localhost:8080/api/productos/eliminar/6316dedb8d6aa85657f0a33b

Server response

Code

Details

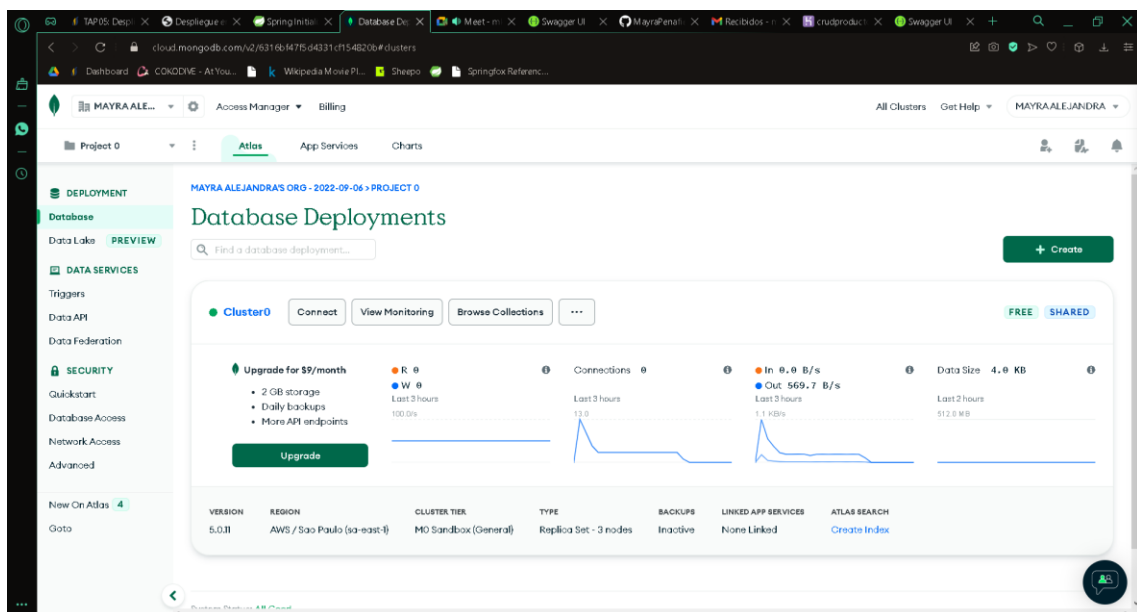
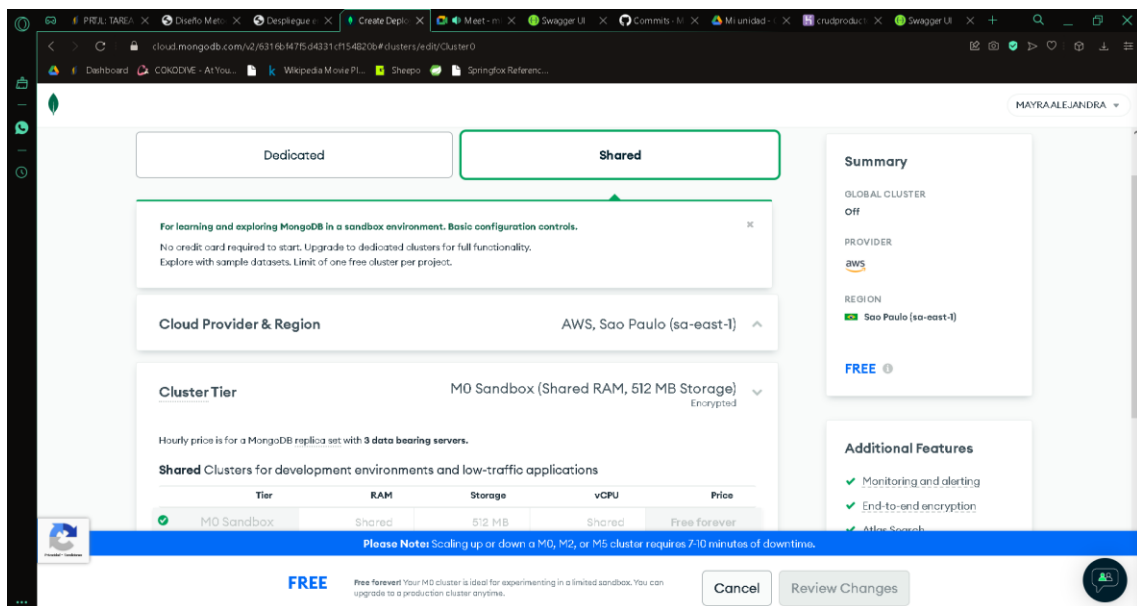
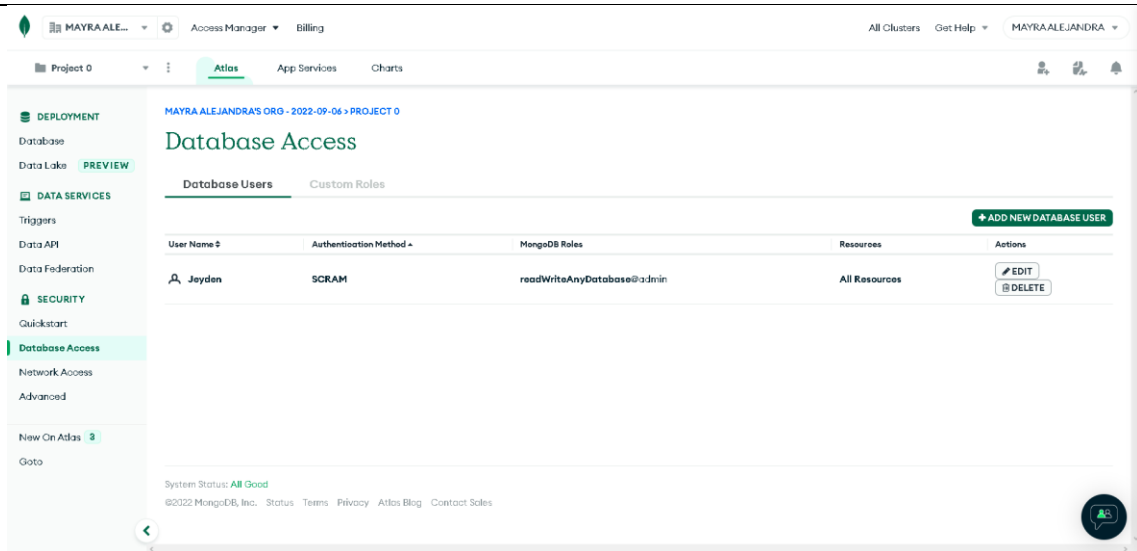
200

Response headers

```
connection: keep-alive
content-length: 0
date: Tue, 06 Sep 2022 00:02:27 GMT
keep-alive: timeout=60
vary: Origin,Access-Control-Request-Method,Access-Control-Request-Headers
```

Creamos un repositorio en github y subimos el proyecto a este en este caso se Llamara:
“SpringwMongoDB”

Luego Conectamos a MongoDB Atlas después de Registrar una cuenta, en este caso la base se llamara **Cluster 0** que será una base de datos Shared y la conectamos con la base creada en MongoDB anteriormente **“Yuta”** y creamos un usuario para gestionar la base de datos:

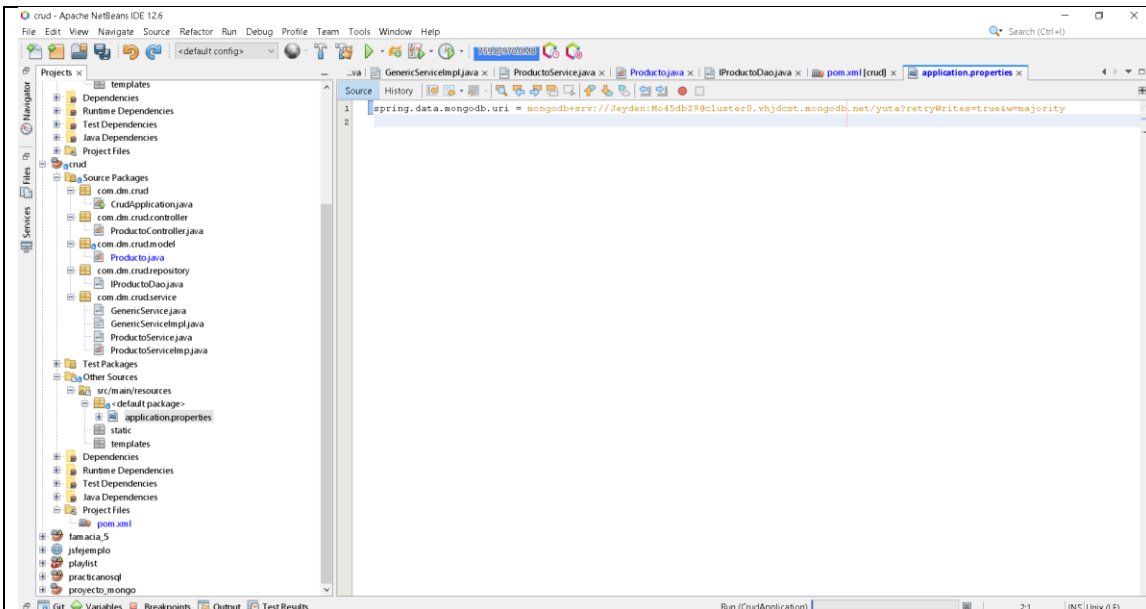


Luego Creamos una conexión en Heroku después de crear una cuenta y conectamos al repositorio en Github, y corremos desde Heroku con el botón “Deploy Branch” y en el botón “View” y nos presentara un error.

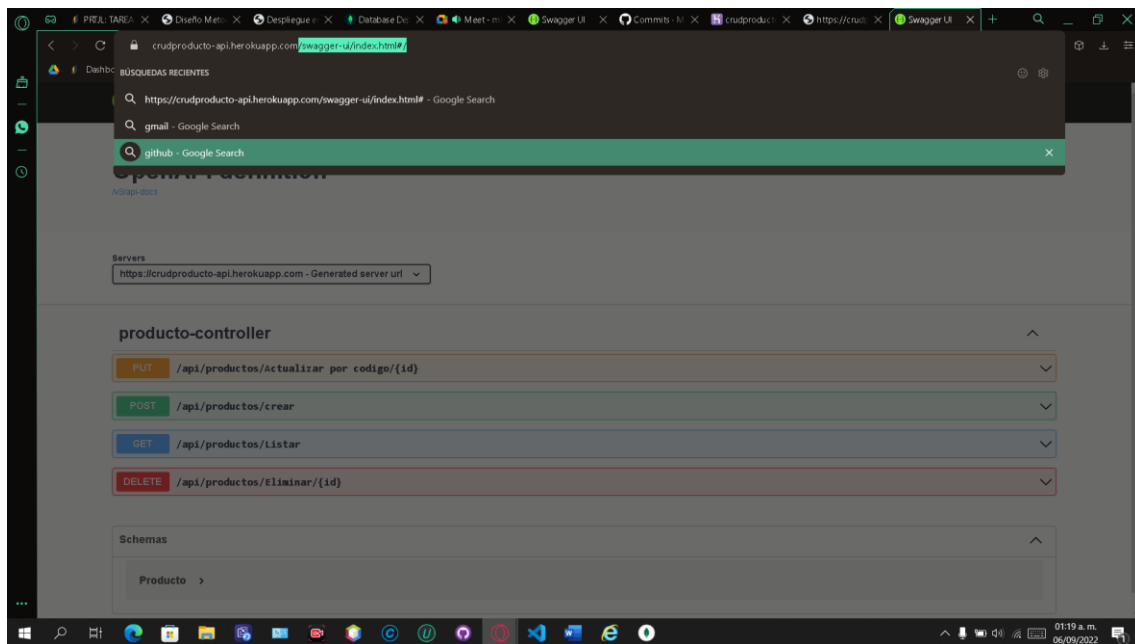
The image displays three sequential screenshots of the Heroku dashboard, illustrating the deployment workflow:

- Top Screenshot:** Shows the 'Deployment method' section with 'Heroku Git' and 'GitHub' as options. The 'App connected to GitHub' section shows the app is connected to the repository 'MayraPenafiel/SpringMongoDB'. The 'Automatic deploys' section is visible, with a note about changing the main deploy branch from 'master' to 'main'.
- Middle Screenshot:** Shows the 'Enable Automatic Deploys' button. Below it, the 'Manual deploy' section is active, showing the 'Deploy a GitHub branch' process. The 'Choose a branch to deploy' dropdown is set to 'main', and the 'Deploy Branch' button is visible.
- Bottom Screenshot:** Shows the successful deployment status. The 'Deploy to Heroku' step is marked with a green checkmark. A message states 'Your app was successfully deployed.' with a 'View' button. Below the deployment steps, a 'Whitelabel Error Page' is shown, indicating a fallback error (404) due to an unexpected error (type=Not Found, status=404).

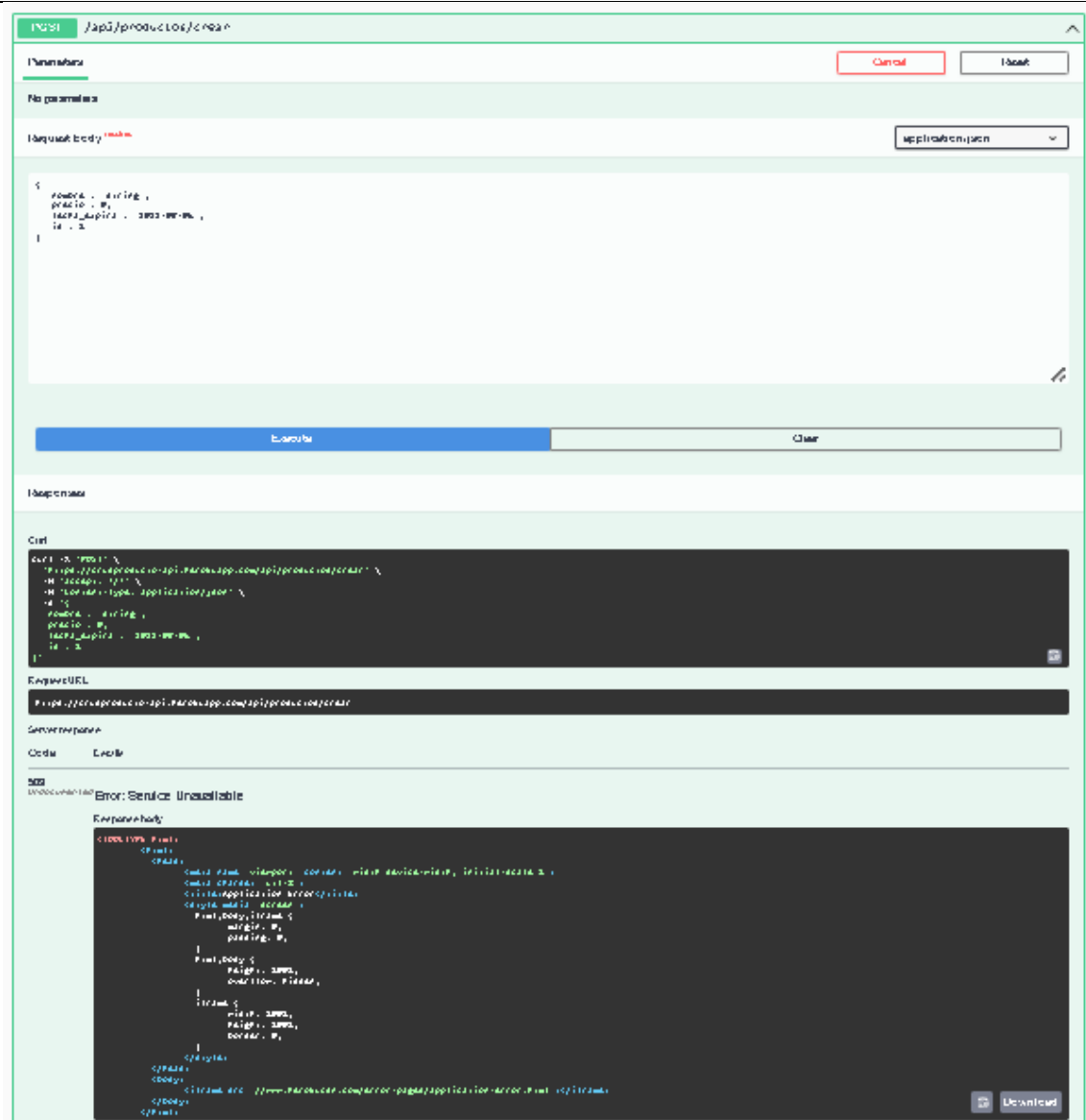
Para corregir este error dentro de propiedades en NetBeans colocar la siguiente ruta:



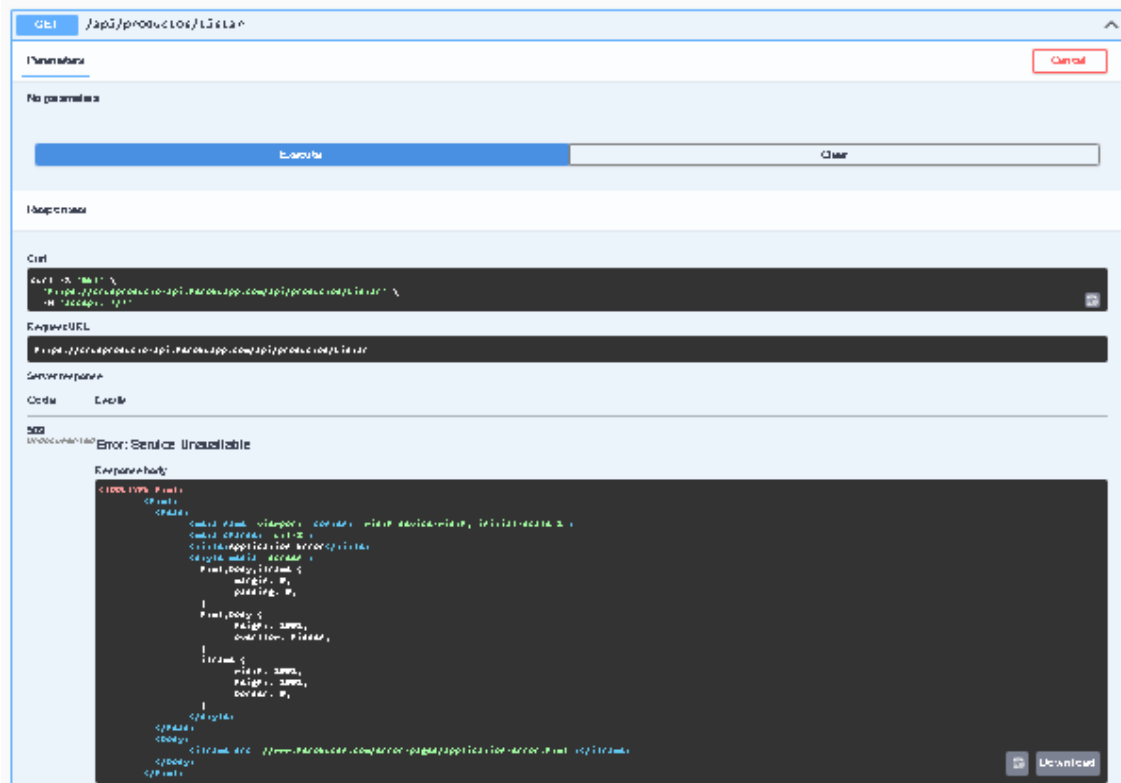
Guardamos los cambios en el git y volvemos a ejecutar con heroku y al final del enlce colocamos: **/swagger-ui/index.html#/** y procedmos a probar los métodos de los cuales se presenta un error que no se pudo encontrar una solución ya que no permite crear documentos y por ello tampoco se puede realizar los demás métodos.



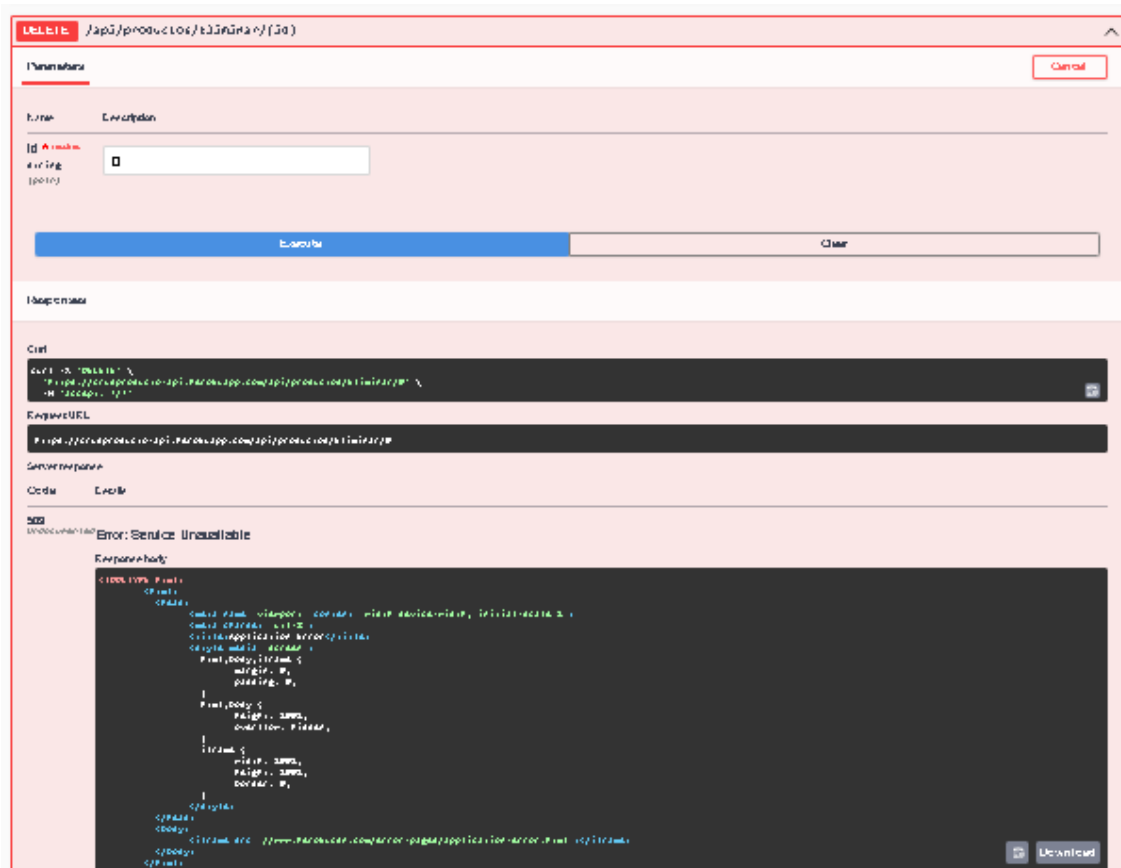
- Crear:



- Listar:



- Elimiar:




- Actualizar:

[illegible]

2.5 Conclusiones

Las consultas en una base NoSql resultan más sencillas una vez que llegamos a conocer los distintos operadores y sentencias que se pueden realizar y obtener las consultas de información que necesitemos al igual que sucede en una base de datos relacional solo que ahora con menor complejidad en comparación.

3 Firmas de Responsabilidad

ESTUDIANTE	DOCENTE
 Firma	Nombre: Firma
Fecha: 06/09/2022	Fecha: