Proyecto Final Nest JS

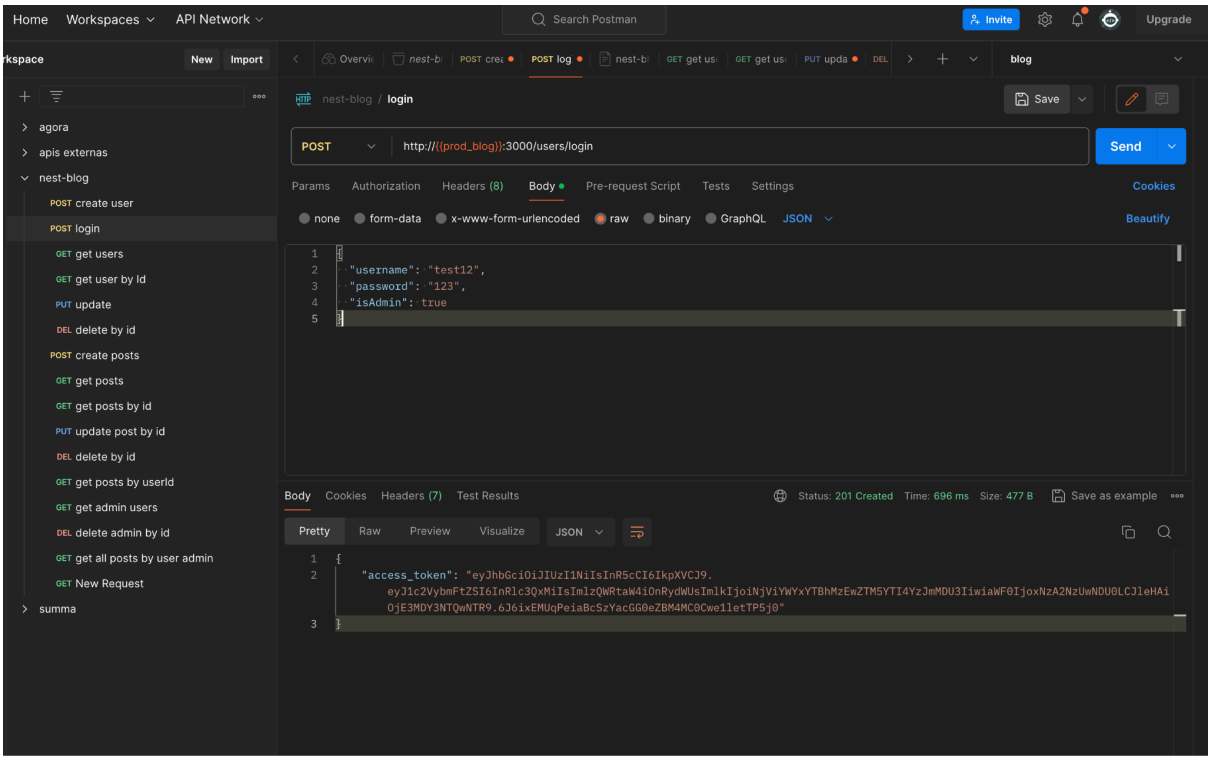
1. Objetivo

El proyecto consiste en crear nuestro propio blog para esto tenemos definido una serie de endpoints RESTful (https://iamdoomling.notion.site/iamdoomling/Trabajo-pr-ctico-final-f366a1dab34245ae83726bb31fb59a25) para gestionar usuarios, posts y autenticación usando Nest.js. Considerando la utilización de MongoDB y un middleware de autorización para usuarios administradores.

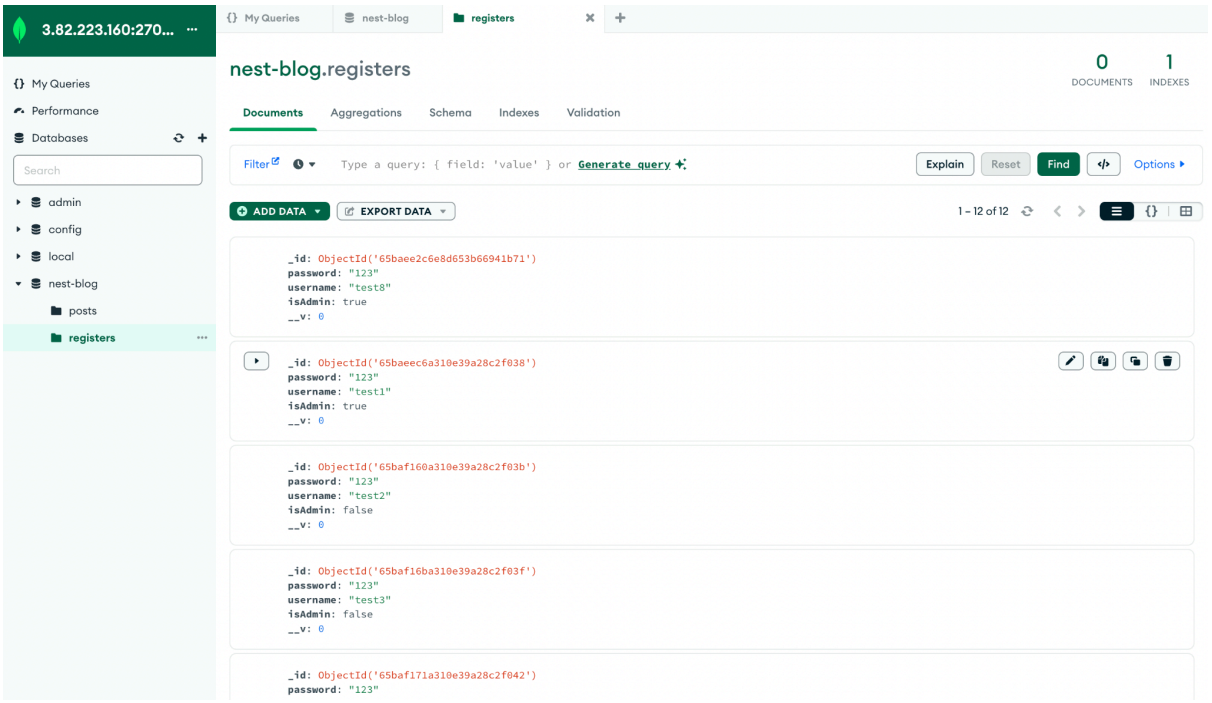
2. Autenticación
   Para la autenticación usamos Password (jwt) y obtendremos un *access_token*, el cual enviaremos en Authorization al consumir las apis.

ejm:


```
{
  "access_token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6InRlc3QxMiIsImlzQWRtaW4iOnRydWUsImlkIjoiNjViYWYxYTBhMzEwZTM5YTI4YzJmMDU3IiwiaWF0IjoxNzA2NzUwNDU0LCJleHAiOjE3MDY3NTQwNTR9.6J6ixEMUqPeiaBcSzYacGG0eZBM4MC0Cwe1letTP5j0"
}
```

3. Base de Datos
   Para la base de datos usaremos Mongodb y podremos ver en *mongodb://admin:password1@3.82.223.160:27017/*



4. Apis

   4.1. Usuarios

   Para el endpoint `DELETE /users/{id}` (solo administradores) se usó un middleware **AdminMiddleware** para validar si el usuario isAdmin.

Ambiente Local: *http://localhost:3000/*

	Endpoint	Descripción	Swagger
`POST /users`	METHOD: POST http://localhost:3000/users	Registrar nuevos usuarios. Cada usuario debe tener username, password y isAdmin	http://localhost:3000/swagger#/Users/AuthController_userRegister
`POST /users/login`	METHOD: POST http://localhost:3000/users/login	Inicio de sesión para usuarios.	http://localhost:3000/swagger#/Users/AuthController_login

GET /users	METHOD: GET http://localhost:3000/users	Listado de usuarios	http://localhost:3000/swagger#/Users/AuthController_getUsers
GET /users/{id}	METHOD: GET http://localhost:3000/users/65b52ccb3c90c7a8ad033933	Obtener detalle de un usuario específico pasando la query el userId.	http://localhost:3000/swagger#/Users/AuthController_getUserById
PUT /users/{id}	METHOD: PUT http://localhost:3000/users/65b52f4cda03af7bbe5c5105	Actualizar un usuario específico (sólo su propio perfil o si es administrador).	http://localhost:3000/swagger#/Users/AuthController_updateUserById
DELETE /users/{id}	METHOD: DELETE http://localhost:3000/users/delete/65b59ef8b1c3574110fd1109	Eliminar un usuario (solo administradores).	http://localhost:3000/swagger#/Users/AuthController_deleteById

4.2. Posts

	Endpoint	Descripción	Swagger
POST /posts	METHOD: POST http://localhost:3000/posts	Crear un nuevo post (solo usuarios registrados). Los post tendrán id, título, autor, contenido y un array de categorías	http://localhost:3000/swagger#/Posts/PostsController_create
GET /posts	METHOD: GET http://localhost:3000/posts?rows=10&page=0	Listado de todos los posts. Debe admitir	http://localhost:3000/swagger#/Posts/PostsControll

		parámetros para paginar resultados (el default de resultados si no hay param será 10)	er_getAllPosts
`GET /posts/{id}`	METHOD: GET http://localhost:3000/posts/65b5e83cc4126a41e536f077	Ver detalles de un post específico.	http://localhost:3000/swagger#/Posts/PostsController_getPostById
`PUT /posts/{id}`	METHOD: PUT http://localhost:3000/posts/65b5f57e118534d63841b542	Actualizar un post (solo el autor o administradores).	http://localhost:3000/swagger#/Posts/PostsController_updatePostById
`DELETE /posts/{id}`	METHOD: DELETE http://localhost:3000/posts/65b606c98594dad2a4dd53fc	Eliminar un post (solo el autor o administradores).	http://localhost:3000/swagger#/Posts/PostsController_deleteById
`GET /posts/user/{userId}`	METHOD: GET http://localhost:3000/posts/user/65b59eb7b1c3574110fd1100	Ver todos los posts de un usuario específico.	http://localhost:3000/swagger#/Posts/PostsController_getPostByUser

4.3. Búsqueda y Filtrado

	Endpoint	Descripción	Swagger
`GET /posts/search`	Rehusamos la api `GET /posts` y adicionamos query título y contenido METHOD: GET http://localhost:3000/po	Buscar posts por título, contenido, etc. Debe admitir parámetros para paginar resultados (el default de	http://localhost:3000/swagger#/Posts/PostsController_getAllPosts

	sts?rows=10&title=posts19&content=welcome&page=0	resultados si no hay param será 10)	
`GET /posts/filter`	Rehusamos la api `GET /posts` y adicionamos query categoría y autor METHOD: GET http://localhost:3000/posts?rows=10&title=posts19&content=welcome&categories=travel,experience&author=test1&page=0	Endpoints adicionales para filtrar posts por categoría o autor	http://localhost:3000/swagger#/Posts/PostsController_getAllPosts

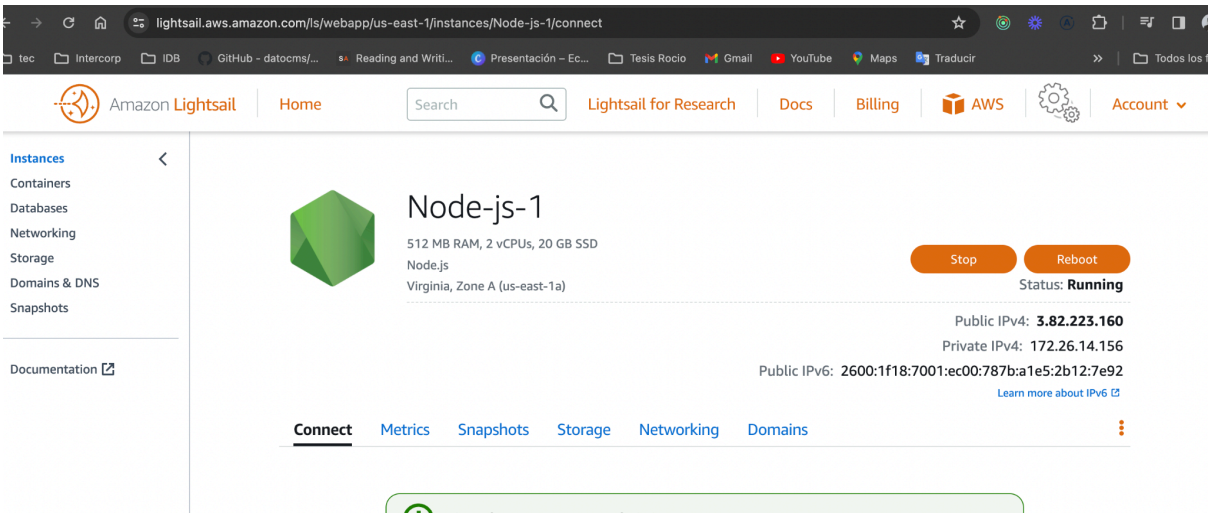
4.4. Administración

Para los endpoints se usó un middleware adicional **AdminMiddleware** para validar si el usuario isAdmin.

	Endpoints	Descripción	Swagger
`GET /admin/users`	METHOD: GET http://localhost:3000/posts/admin/users	Obtener todos los usuarios (solo administradores).	http://localhost:3000/swagger#/Posts/PostsController_getAllUser
`DELETE /admin/users/{id}`	METHOD: DELETE http://localhost:3000/posts/admin/users/65b99aa63685da61fdae0b7c	Eliminar usuarios (solo administradores).	http://localhost:3000/swagger#/Posts/PostsController_deleteAdminById
`GET /admin/posts`	METHOD: GET http://localhost:3000/posts/admin/posts	Obtener todos los posts con opciones de moderación (borrar o editar) (solo administradores).	http://localhost:3000/swagger#/Posts/PostsController_getAllPostsAdmin

5. Despliegue

Para el despliegue usaremos Lightsail aws
Crearemos una instancia y habilitaremos los puertos 3000 y 27017, nuestra ip estática será ***3.82.223.160***



6. Documentación de Apis

   Swagger: http://3.82.223.160:3000/swagger

7. Código fuente

   Github: https://github.com/Janetquispeu/nest-project (rama main)