Technical Test Backend

**Objetivo**

La prueba consistirá en crear un simple API REST para un único recurso, la prueba

se dividirá en 4 funcionalidades básicas y 4 funcionalidades adicionales (opcionales)

Se evaluará:

* Evaluar el conocimiento sobre la creación de un API REST en el lenguaje Java.
* Evaluar el conocimiento del framework SPRING BOOT
* Evaluar buenas prácticas en el código (imports, nombrado de clases y variables, comentarios que aporten valor al entendimiento del código, limpieza)
* Evaluar el modelo mental que se tiene para la organización de clases, funciones, módulos.

## Instrucciones

Algunas instrucciones a tener en cuenta:

* Lenguaje de programación: Java. **SÓLO escribir y enviar el código funcional que se necesite**.
* El código entregado debe representar código listo para desplegar en un ambiente productivo.
* Framework SPRING BOOT.
* **Utilizar como base de datos H2**, implementar consultas utilizando JPA contemplar data de prueba para soportar los casos de uso de la solución.
* Documentar adecuadamente el modelo mental (gráfico de modelo de base de datos y justificación de relaciones) y los casos de uso desarrollados con ejemplos en el archivo readme del Repositorio.
* Aplicar **uno o más patrones de diseño** en su solución, indicar donde se aplicó y justificar su utilización en la documentación.
* Crear los casos de uso adicionales necesarios para cubrir el modelo mental de la solución planteada.
* Request y response serializadas utilizando JSON
* Crear una colección postman para validar el funcionamiento de su solución

## Funcionalidades básicas Usuario y Autenticación

Se desea que el API sea restringido mediante json web tokens **(JWT) Endpoints para Notas**

Se desea crear endpoints para poder administrar **notas** (crear y listar), los datos se guardarán en una base de datos.

## Validación de datos y manejo de excepciones

Se desea validar los datos que se reciben vía POST y mostrar los errores adecuados al usuario (con información relevante)

## Pruebas Unitarias

Se desea que todos los casos de uso implementados tengan su set de pruebas unitarias que validen el funcionamiento optimo de la lógica

**Nota:** Los campos usados en la BD son irrelevantes y son propuestos por el evaluado

## Funcionalidades adicionales

Las siguientes funcionalidades se construirán sobre la funcionalidad base. que usa el API

# AUTORIZACION

Se desea asociar una **nota** con un **usuario**, de tal manera que cada usuario solo pueda ver sus propias notas.

# SWAGGER

Documentar todos los endpoints utilizando swagger.

# PRUEBAS DE ACEPTACION

Se desea tener una prueba de aceptación por cada endpoint contemplado en su solución, que valide su funcionamiento correcto.

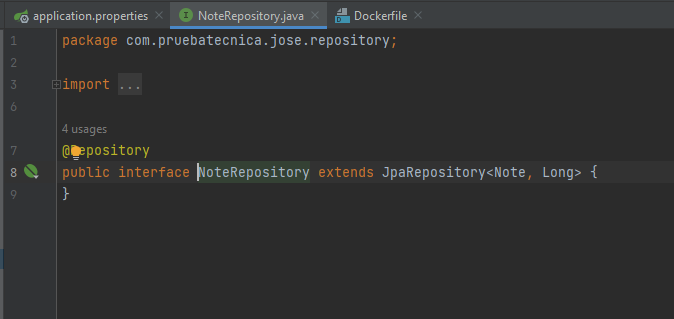
# DESPLIEGUE UTILIZANDO DOCKER

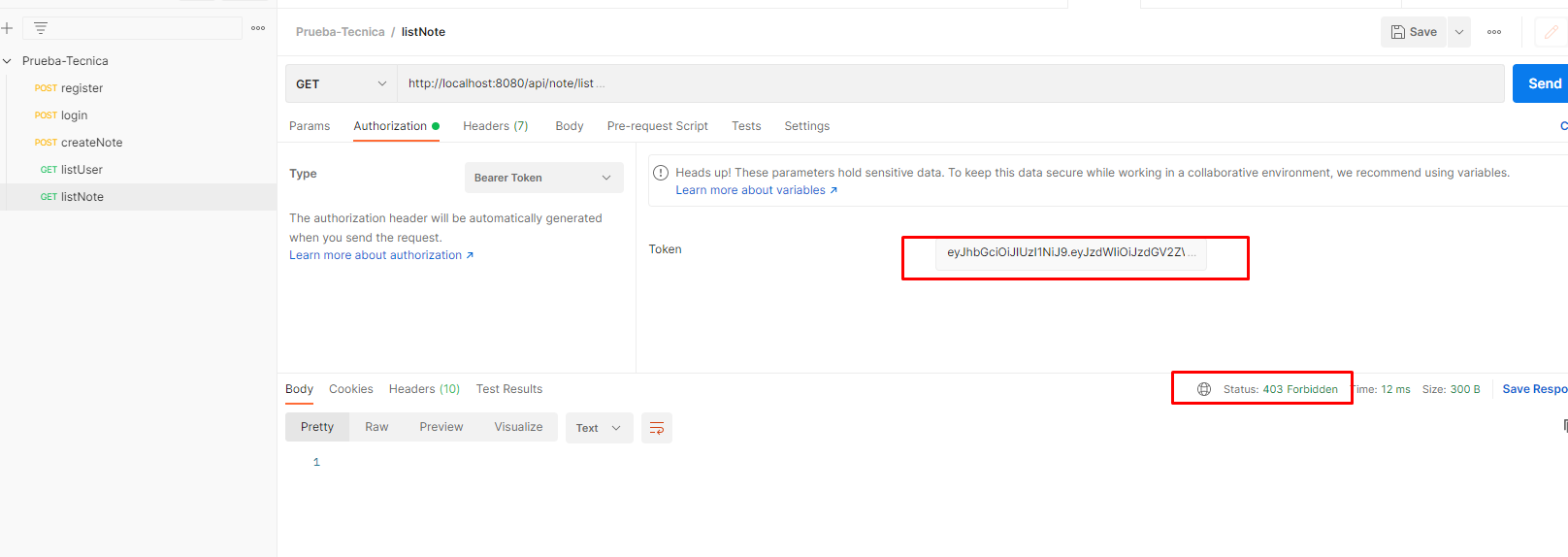
Crear y alojar un contenedor de la solución implementada utilizando

**D**ocker Hub (Documentar puertos y variables de entorno necesarias)

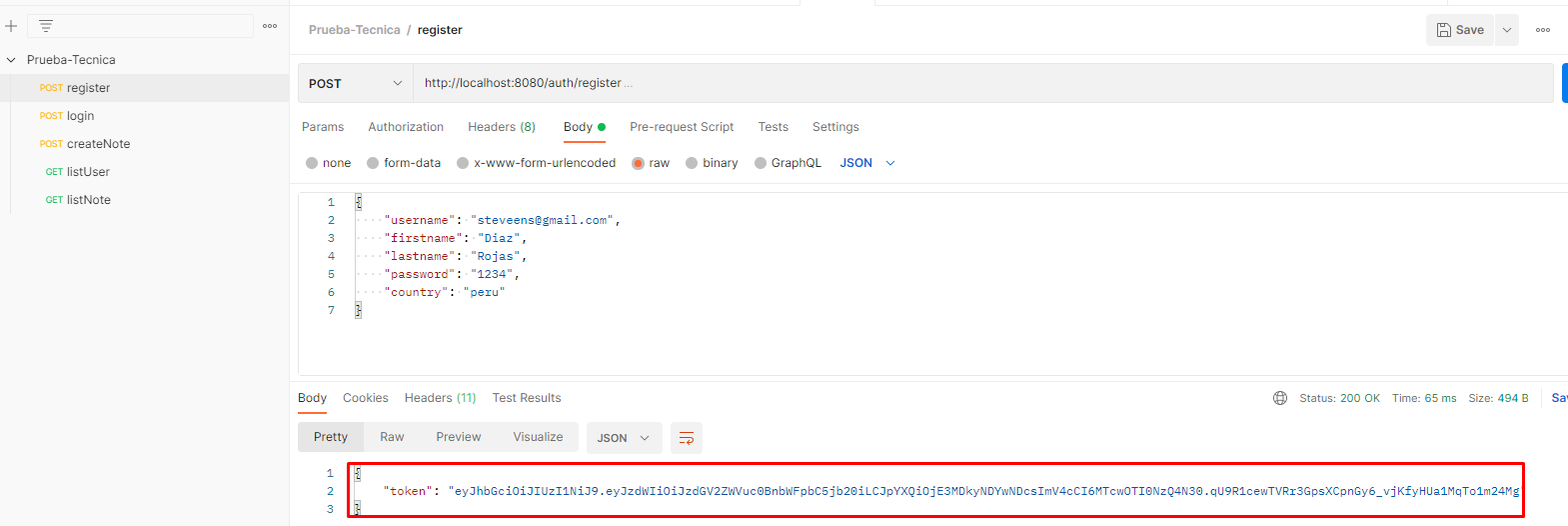
SOLUCION

-Se utilizó una base de datos H2, implementando consultas JPA

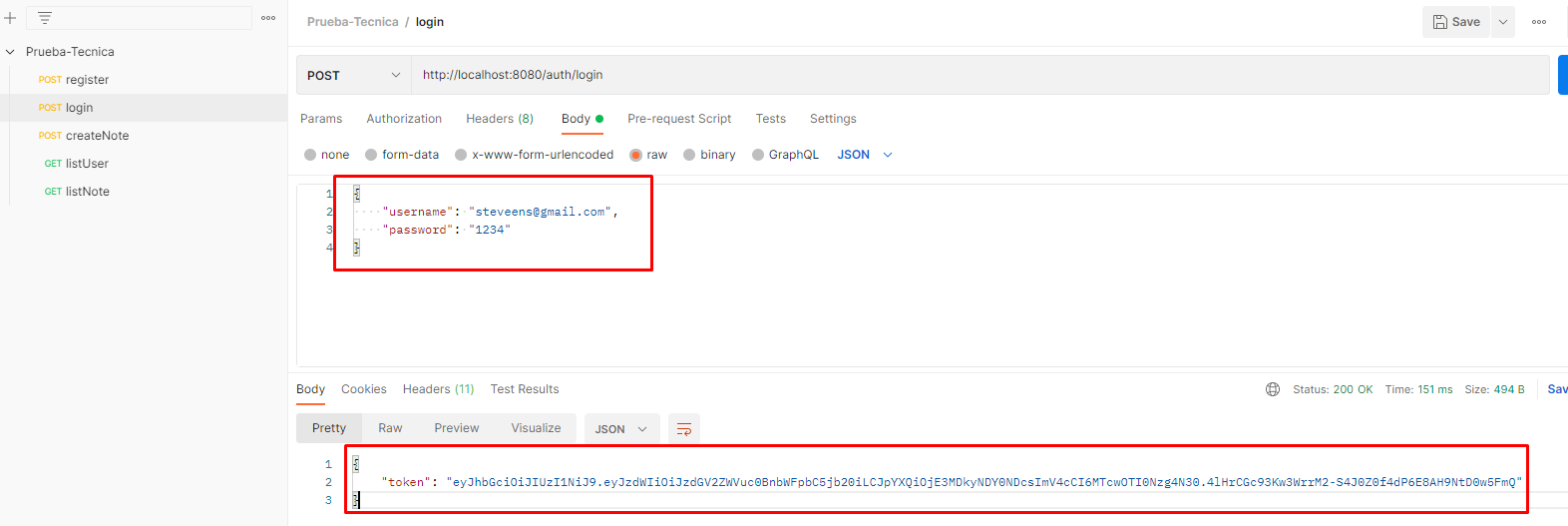


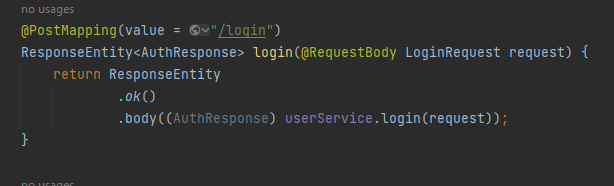
-Se creó JSON web Token (Jwt) para los usuarios dependiendo del Rol que tengan. La authorization solo se filtrará para los los endpoints de Notas hasta no tener el Token autorizado

**FLUJO PARA OBTENER EL TOKEN**

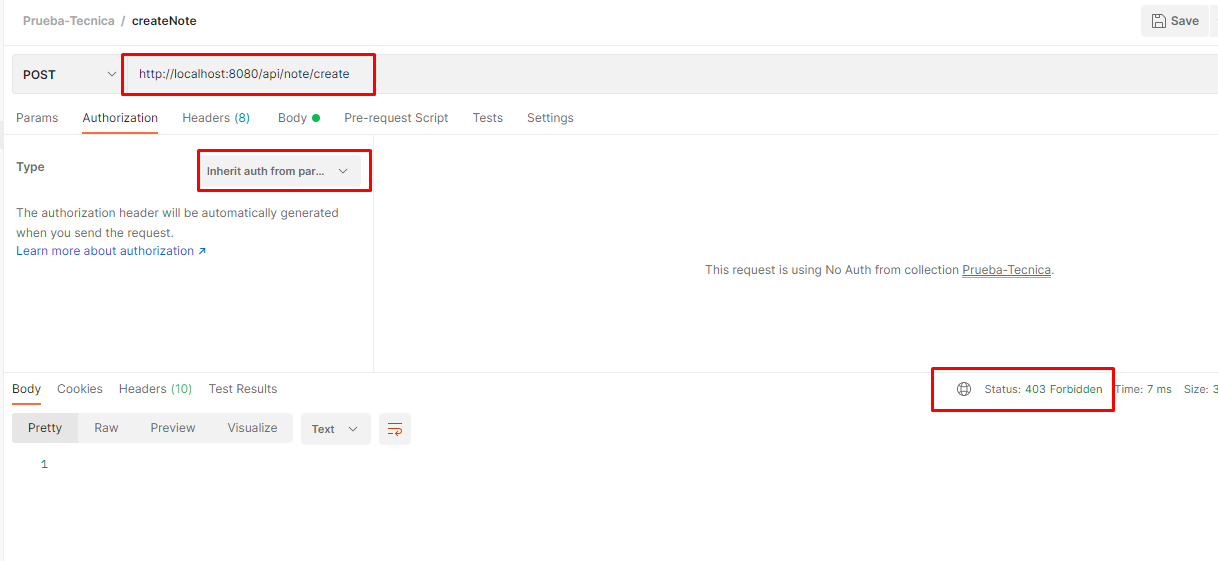
1. Creamos un endpoint register POST en la cual crearemos los usuarios cada vez que corremos el proyecto por el motivo que se está usando la base de datos h2;

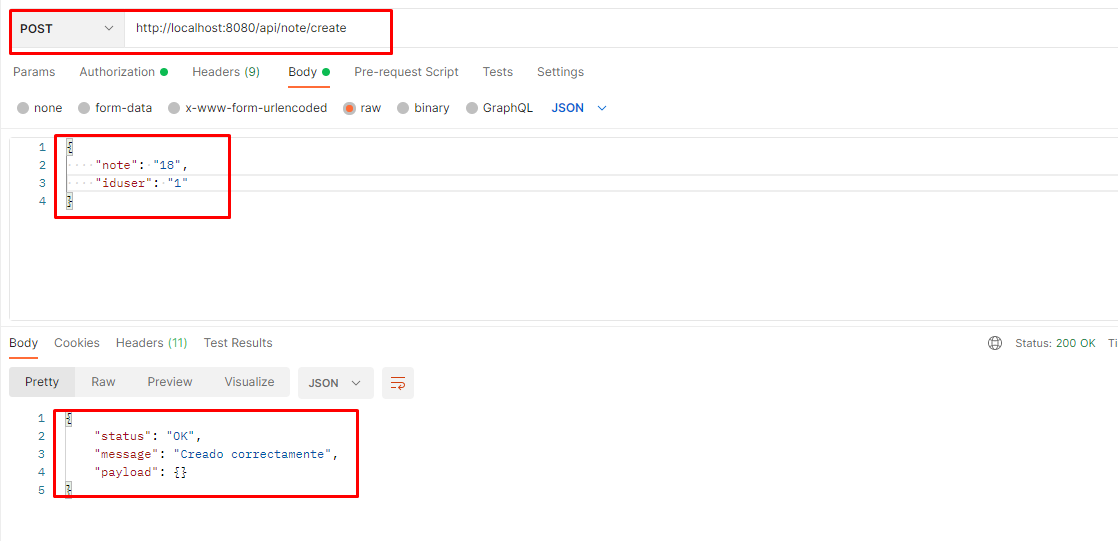


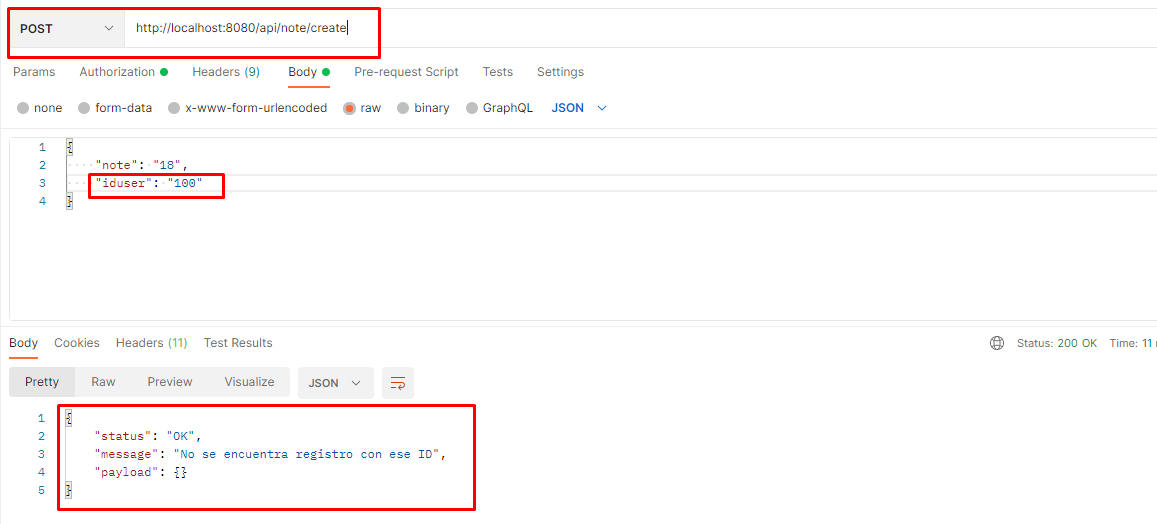
1. Creamos un endpoint login POST, en la cual tenemos que loguear y comparar si existe el registro creado anteriormente y esté devolverá un token en la cual lo usaremos en los siguientes endpoints de Notas

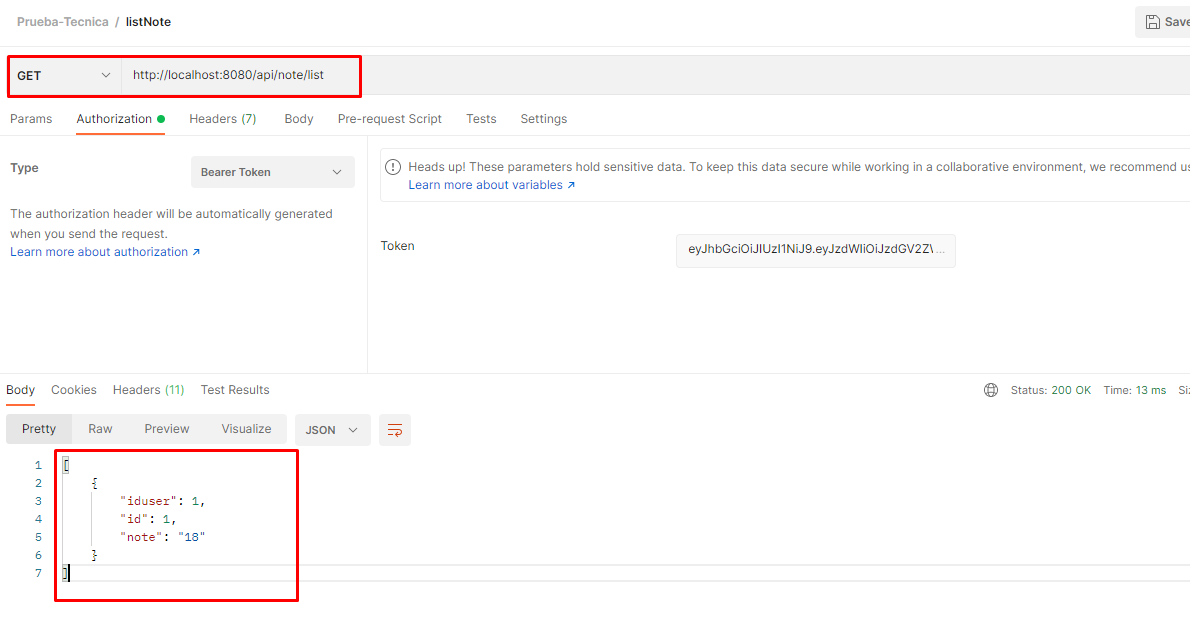


-Usaremos el endpoint crear Notas sin Token Authorization

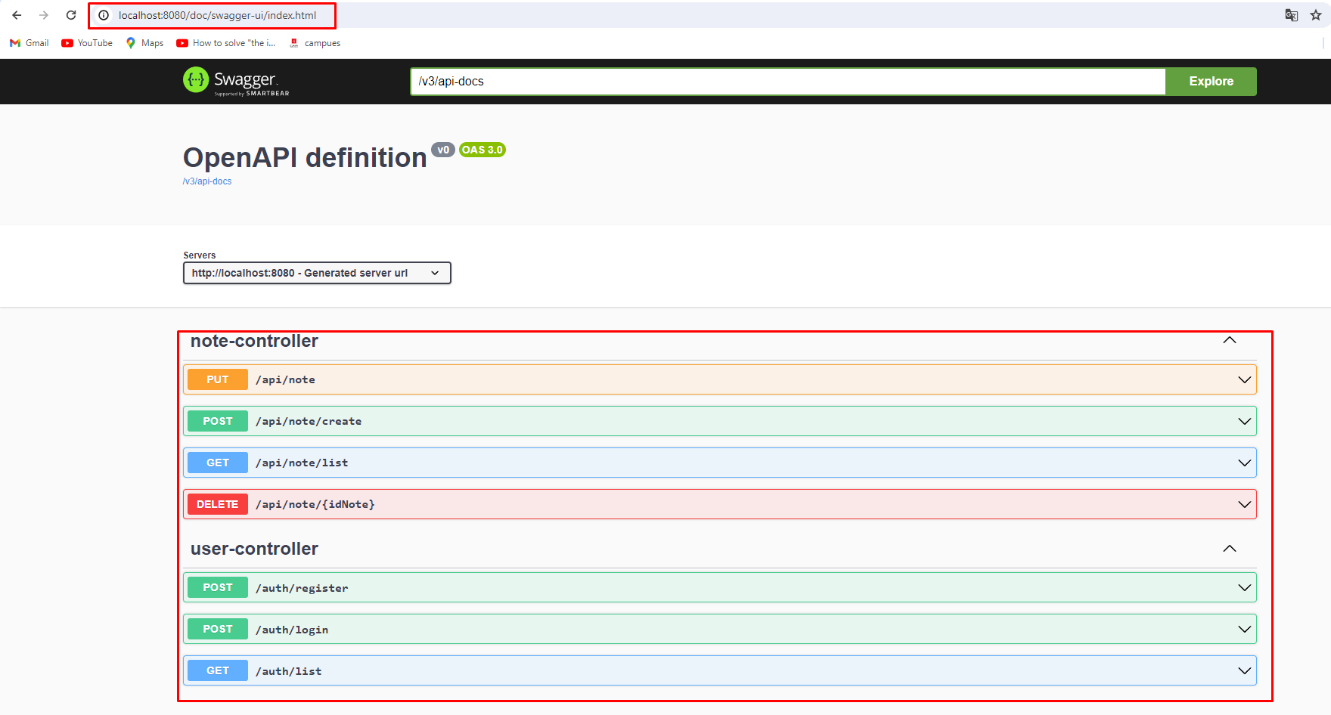


-Una vez obtenido el Token e ingresado el token podemos crear un registro en la tabla Notas que tiene como request : iduser(id del usuario ingresado en este caso es el ID 1) y notas

-En el caso que haya ingresado un ID de usuario que no existe

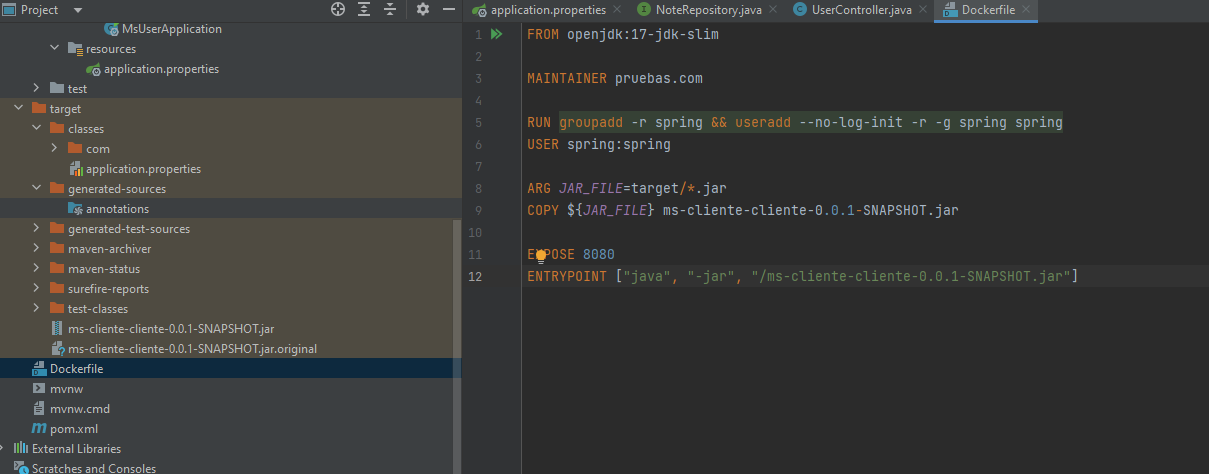
-Listamos el endpoint GET para notas y podemos el registro de notas relacionado con el id de usuario creado anteriormente

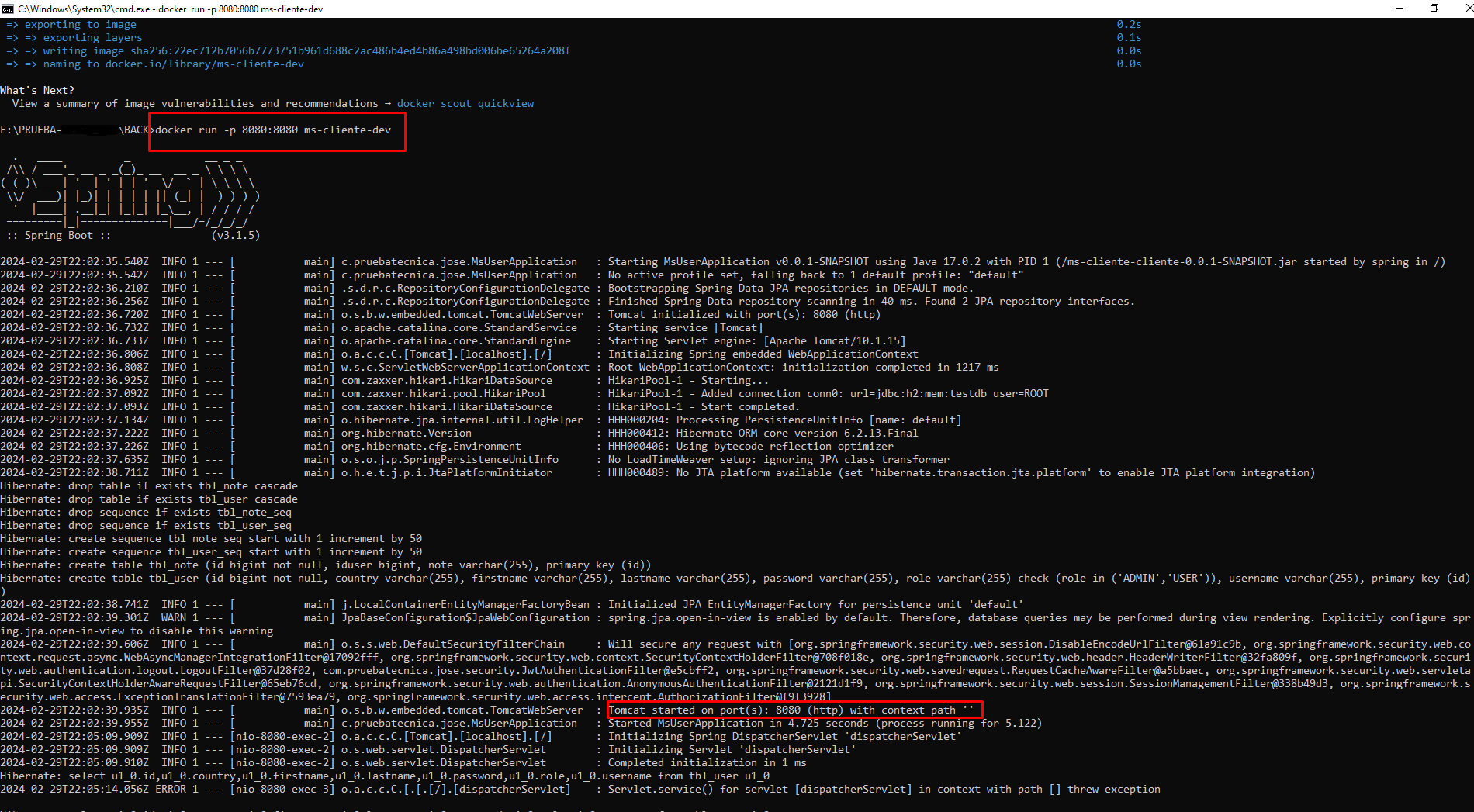
**SWAGGER**

-Se utilizó dependencias para la documentación de los endpoins

**DOCKER**

-Se implementó el Dockerfile



-Se corrió el proyecto mediante Docker

