***API REST Registro de Usuarios***

Esta aplicación proporciona una API RESTl para el registro de usuarios. El servicio acepta y retorna solamente data en formato JSON, incluyendo mensajes de error. La aplicación está construida como se solicitó utilizando Java 17, Spring Boot, y utiliza JWT para la generación de tokens de acceso.

**TECNOLOGÍAS UTILIZADAS PARA LA SOLUCIÓN**

* Java 17
* Spring Boot
* Maven
* H2 Database (base de datos en memoria)
* JPA (Hibernate)
* JWT
* Pruebas Unitarias con Junit y Mockito
* Swagger para la documentación de la API

**BASE DE DATOS**

Esta aplicación utiliza H2, una base de datos en memoria que se carga automáticamente al iniciar la aplicación. Para acceder a la consola de H2:

* URL: http://localhost:8080/h2-console
* JDBC URL: jdbc:h2:mem:testdb
* Usuario: sa
* Contraseña: (vacío)

**SCRIPT DE CREACIÓN DE BASE DE DATOS**

La base de datos H2 se genera automáticamente con JPA y no requiere scripts manuales. Sin embargo, se pueden consultar las entidades definidas en el código fuente para entender su estructura puesto que la secuencia, el trigger y la tabla son creadas desde el proyecto.

**JWT**

El token JWT se genera durante el registro de usuario y se devuelve en la respuesta, lo que indica que el usuario fue registrado y autenticado por medio de dicho token.

**SWAGGER**

La documentación de la API está disponible en Swagger, siempre y cuando la aplicación se encuentre instalada y ejecutando URL Swagger: <http://localhost:8080/swagger-ui/index.html>

**PRUEBAS**

Las pruebas unitarias están incluidas en el proyecto y se realizaron con Junit y Mockito

**DISEÑO**

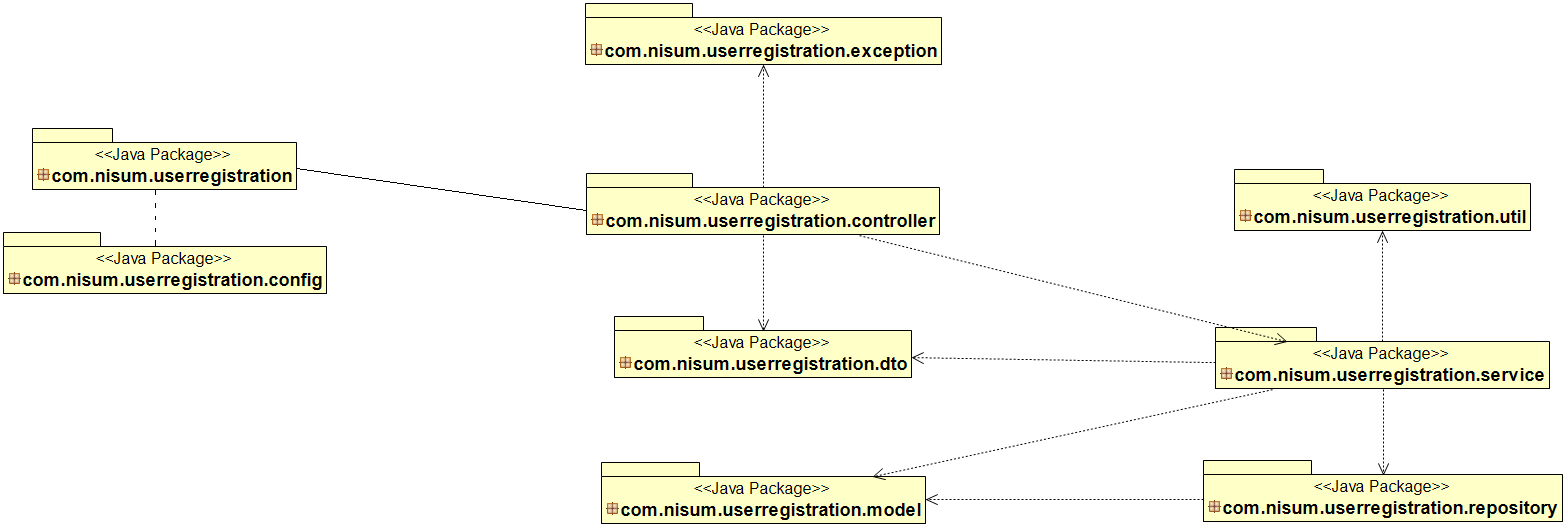
El diseño de la aplicación sigue los principios de separación de responsabilidades, utilizando patrones de diseño y buenas prácticas:

* DTOs para la transferencia de datos entre las capas.
* Servicios para la lógica de negocio.
* Controladores REST para gestionar las solicitudes HTTP.
* Repository Pattern para la interacción con la base de datos.
* Inyección de dependencias para gestionar las dependencias entre componentes.
* Manejo de errores estandarizado con respuestas en formato JSON.

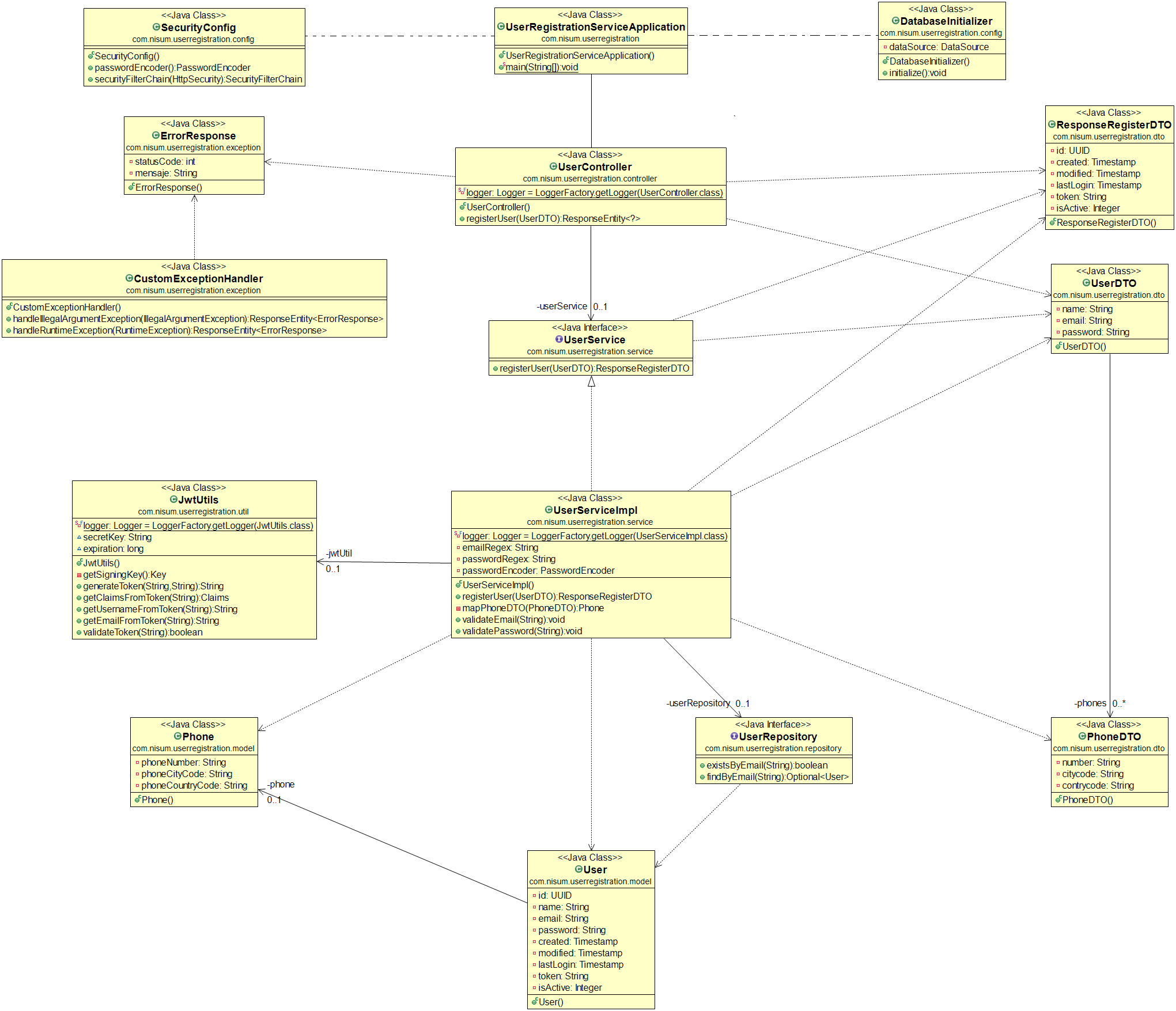
**DIAGRAMA DE PAQUETES**

A continuación se muestra una serie de Diagramas generados para el mayor entendimiento de la aplicación y como esta estructurada

* ***DIAGRAMA DE PAQUETES***

******

* ***DIAGRAMA DE CLASES***

******

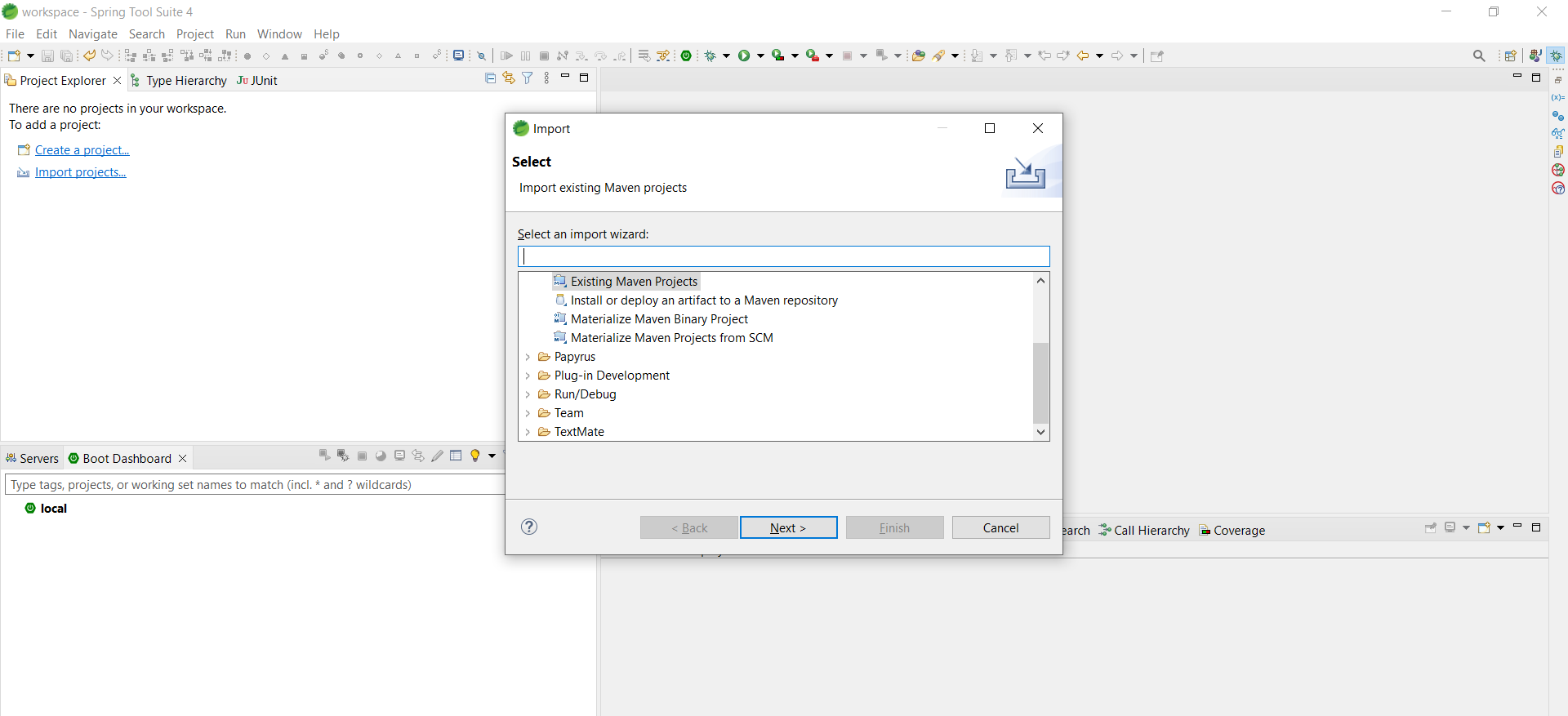
**INSTALACIÓN Y PRUEBA DEL PROYECTO**

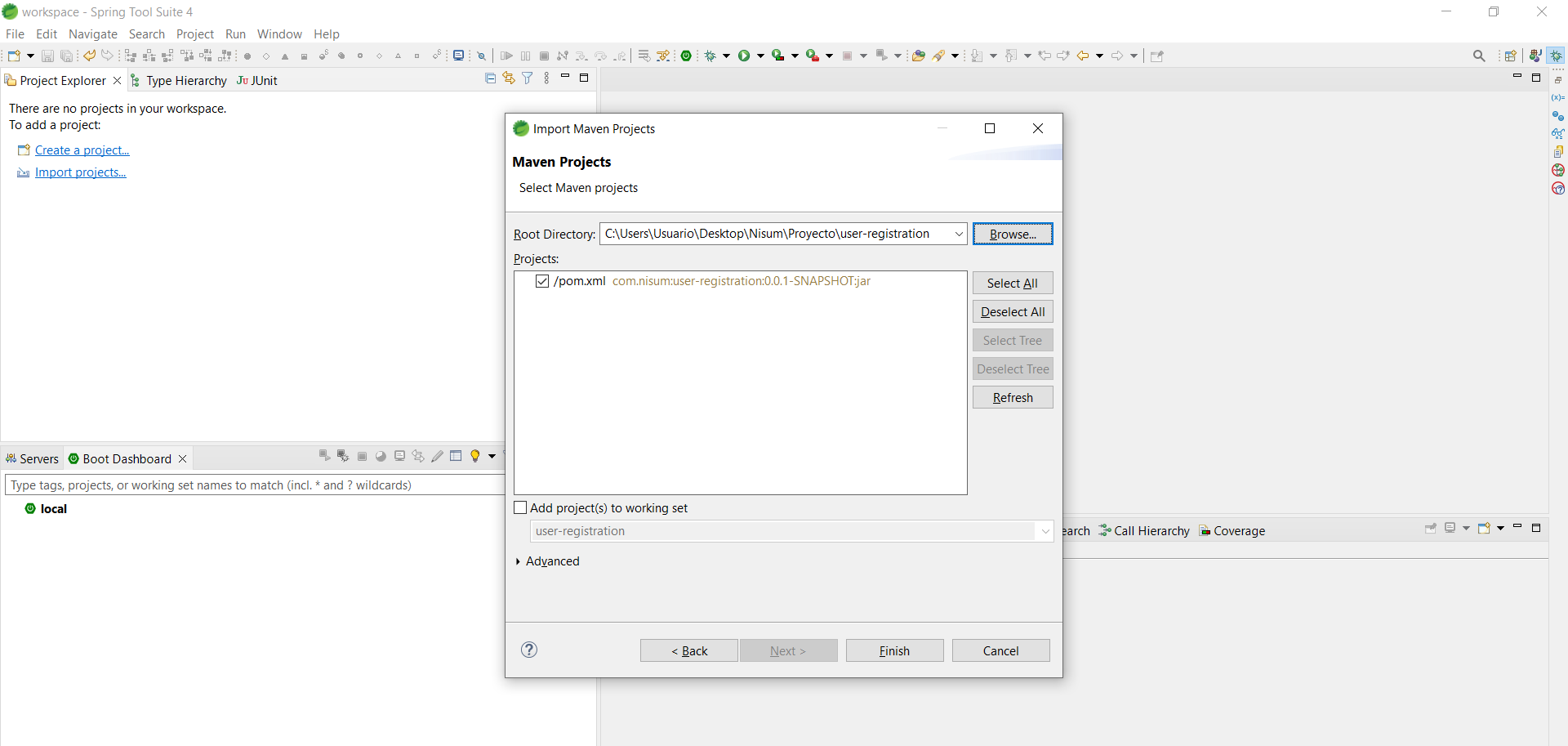
1. **CLONAR EL REPOSITORIO**

Se debe clonar el repositorio de GitHub Ubicado en la ruta <https://github.com/JhonnGomezTrabajos/Nisum.git>

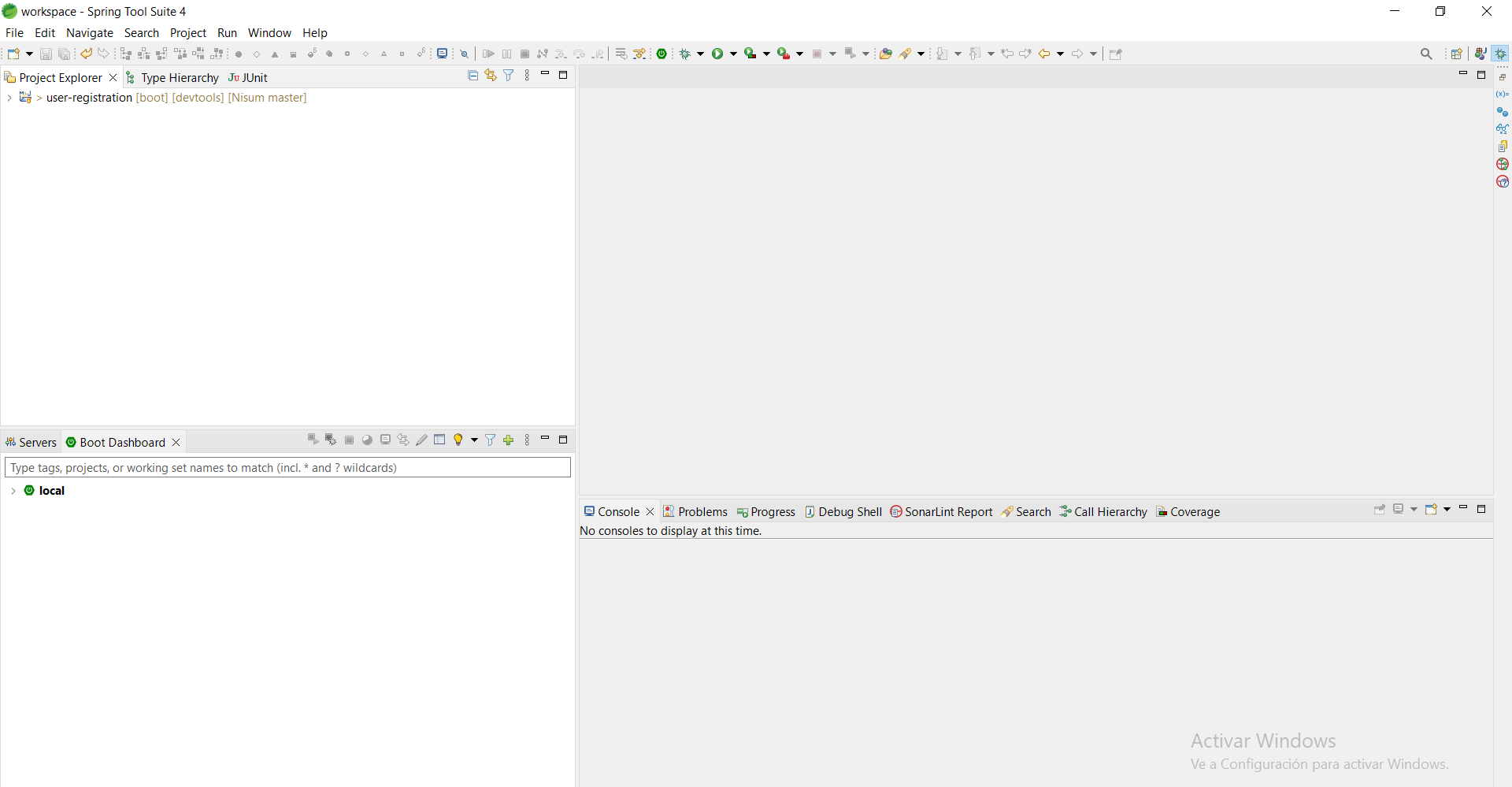
El proyecto java se encuentra en la carpeta Proyecto/user-registration

1. **ABRIR EL PROYECTO DESDE UN IDE(EN ESTE CASO YO UTILICE SPRING TOOLS SUITE)**



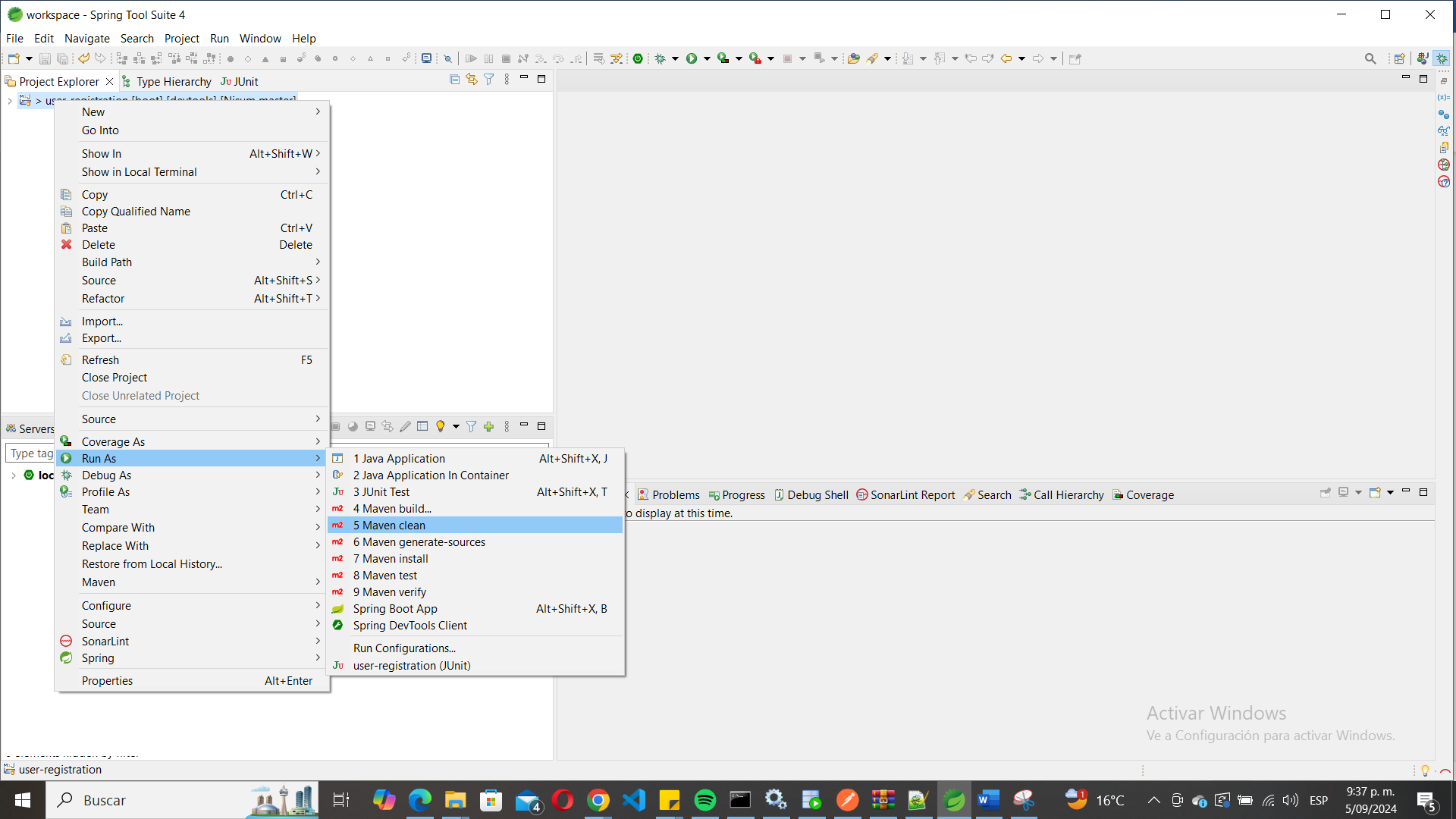


Como vemos en la siguiente imagen el proyecto es importado correctamente

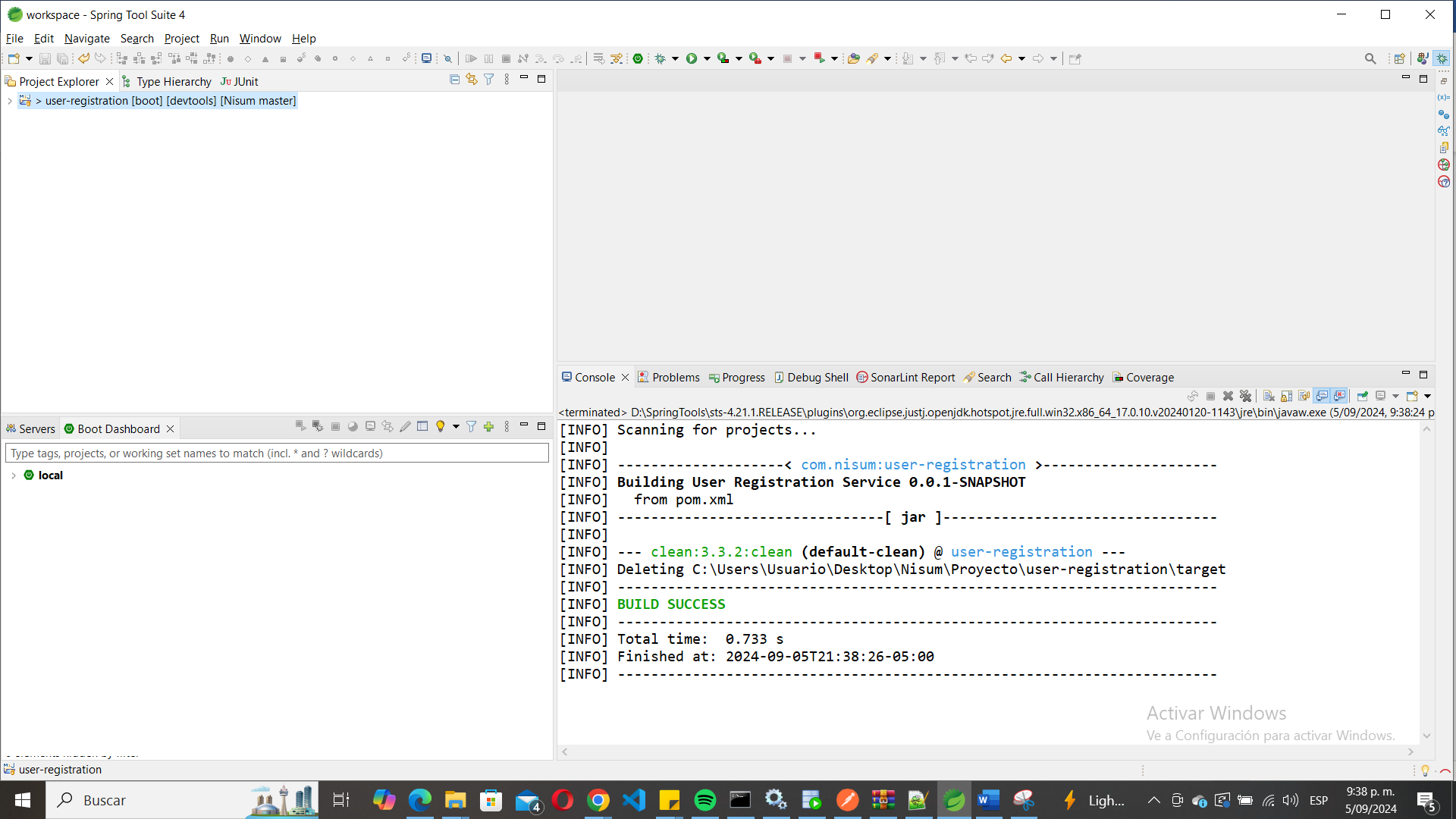


1. **COMPILAR EL PROYECTO CON MVN**

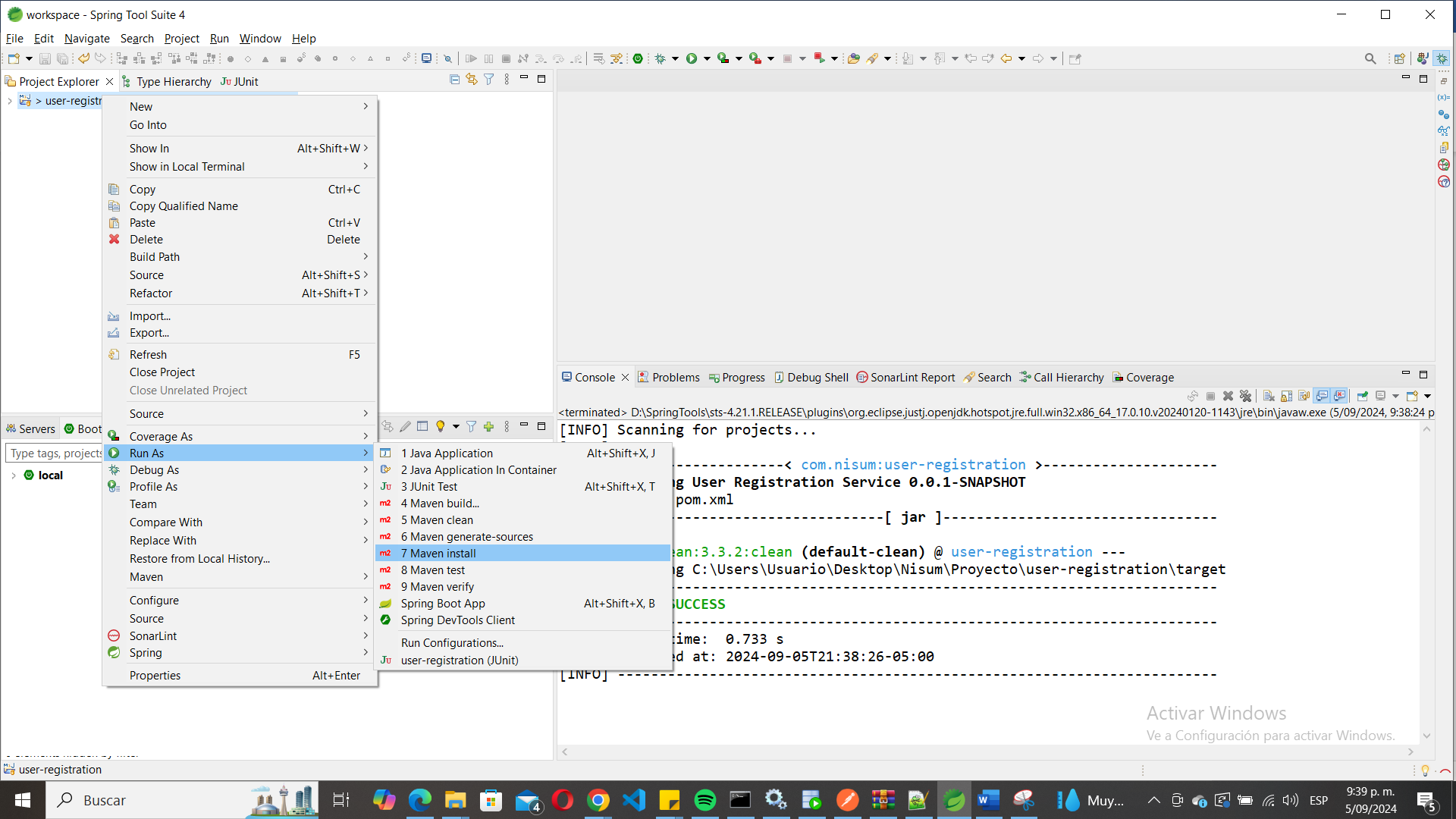
Realizamos la compilación del proyecto



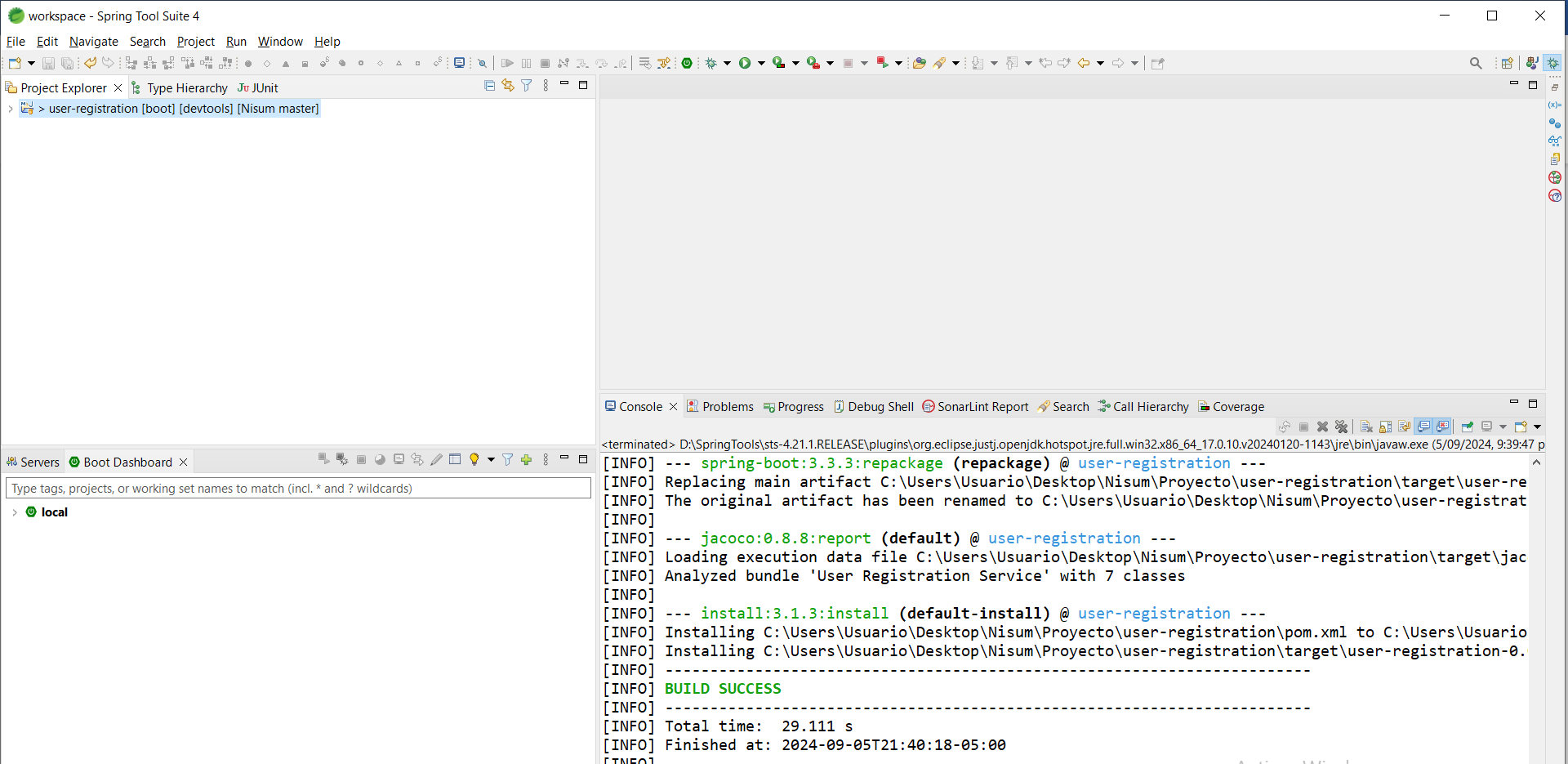
Como se evidencia el proyecto ejecuta el comando clean de manera correcta



Ahora vamos a ejecutar el comando install

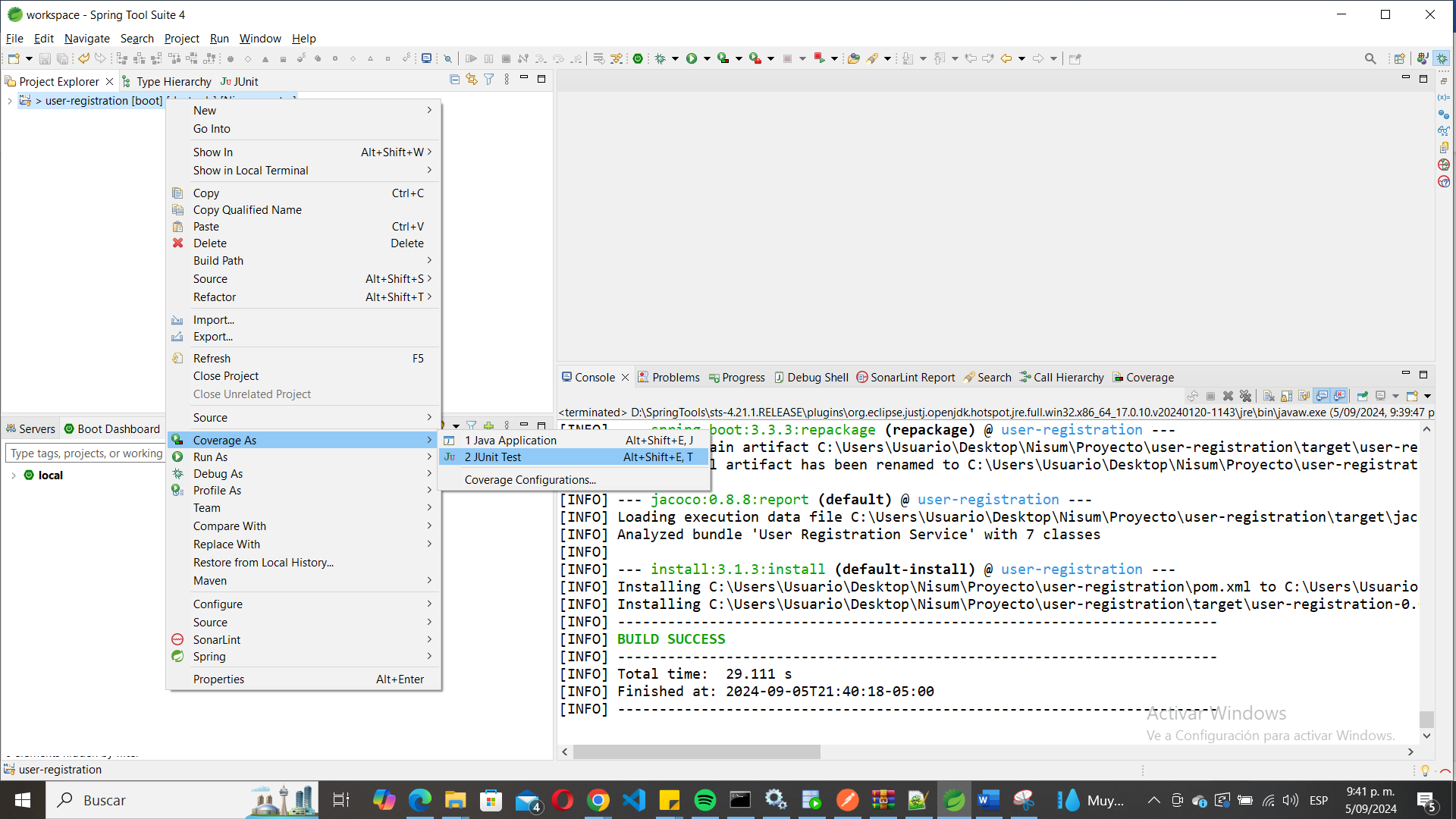


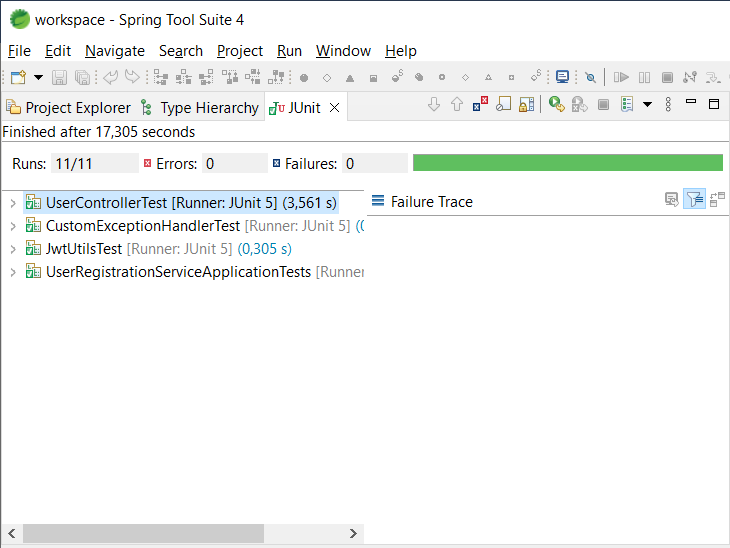
El proyecto se instala de manera correcta



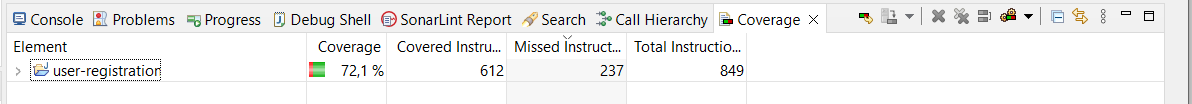
1. **VALIDAR COBERTURA DE CÓDIGO CON JUNIT PARA GARANTIZAR LAS PRUEBAS UNITARIAS A LAS CLASES CREADAS EN EL DESARROLLO**

Ejecutamos la cobertura de código con junit de la siguiente manera



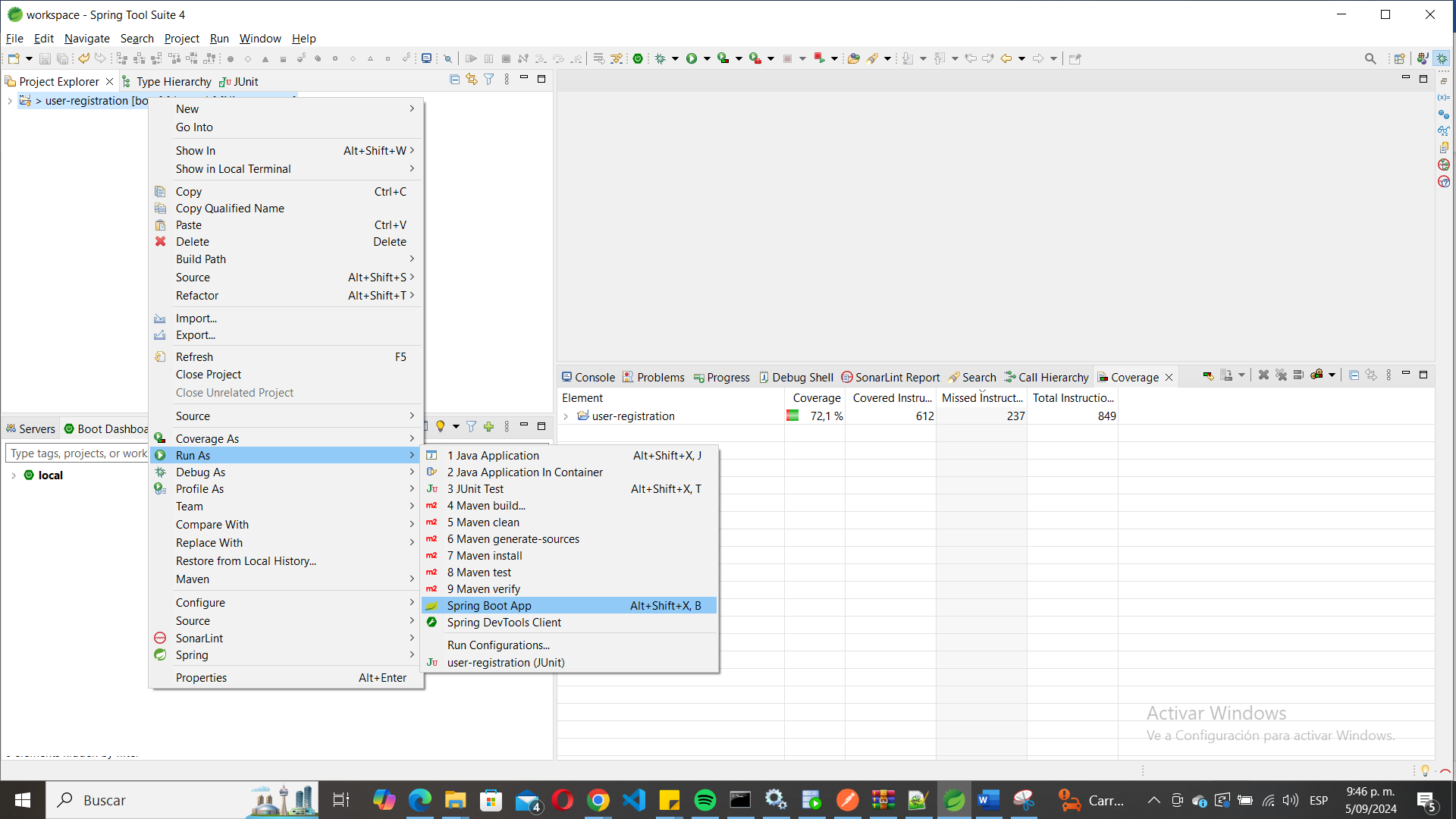


Se ejecutan 11 pruebas unitarias de manera correcta como se evidencia, lo cual representa 72,1% en cobertura de código

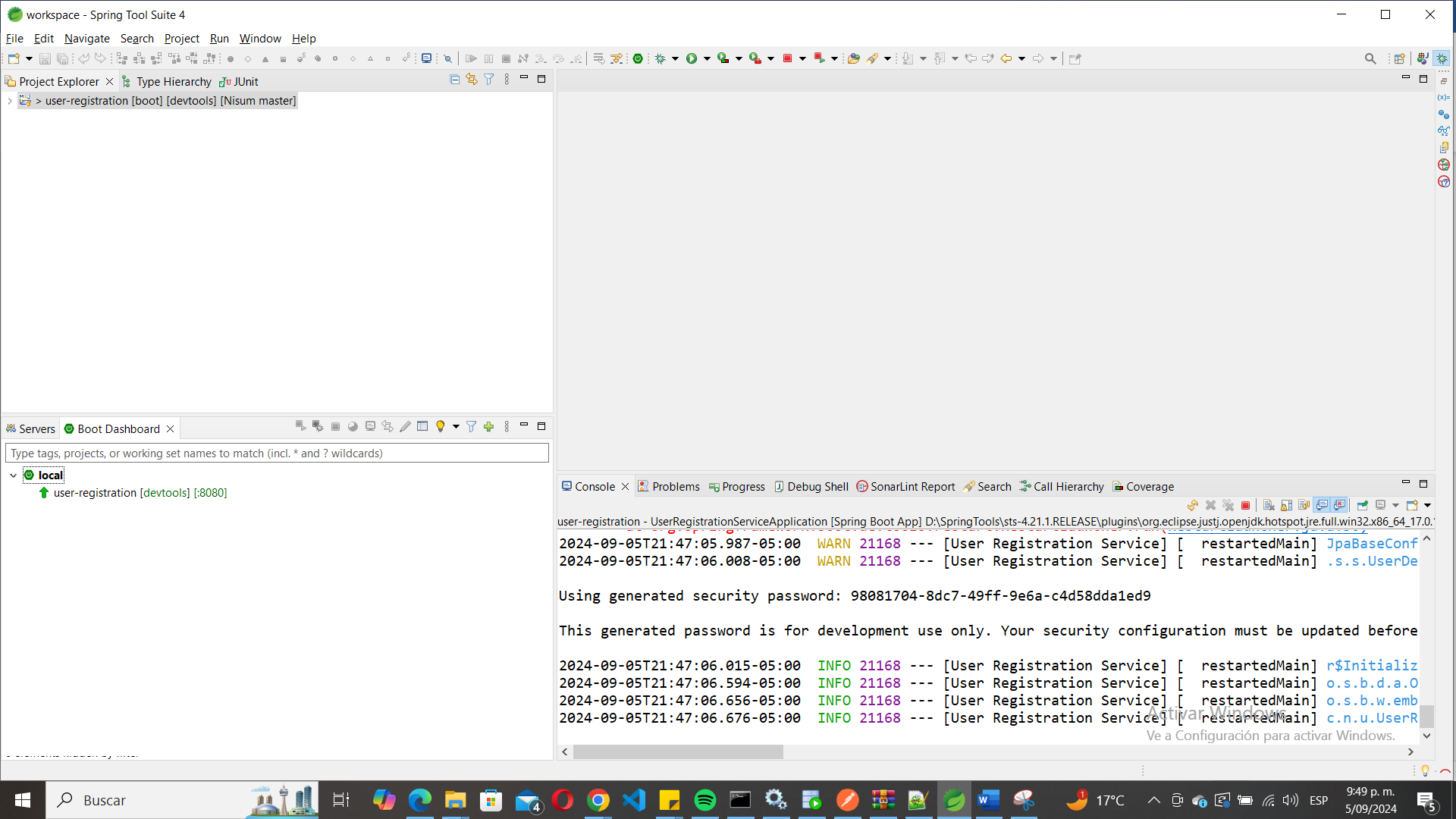


1. **EJECUTAR LA APLICACIÓN**

Ejecutamos la aplicación de la siguiente manera



La aplicación se ejecutará en <http://localhost:8080> como se evidencia en la siguiente imagen



1. **ENDPOINT DE LA API**

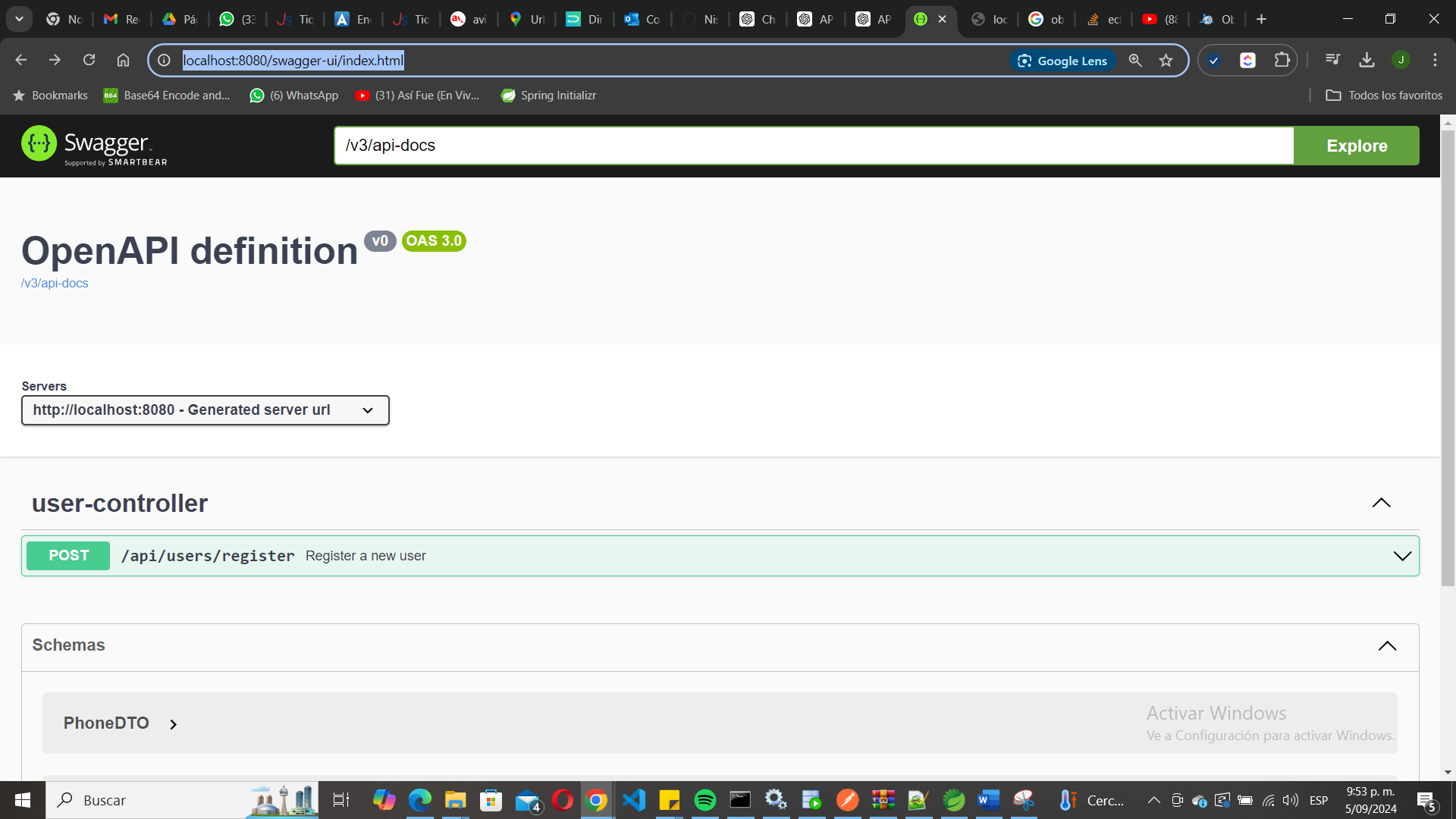
El endpoint del api de Registro de usuario Nisum es el siguiente

URL: <http://localhost:8080/api/users/register>

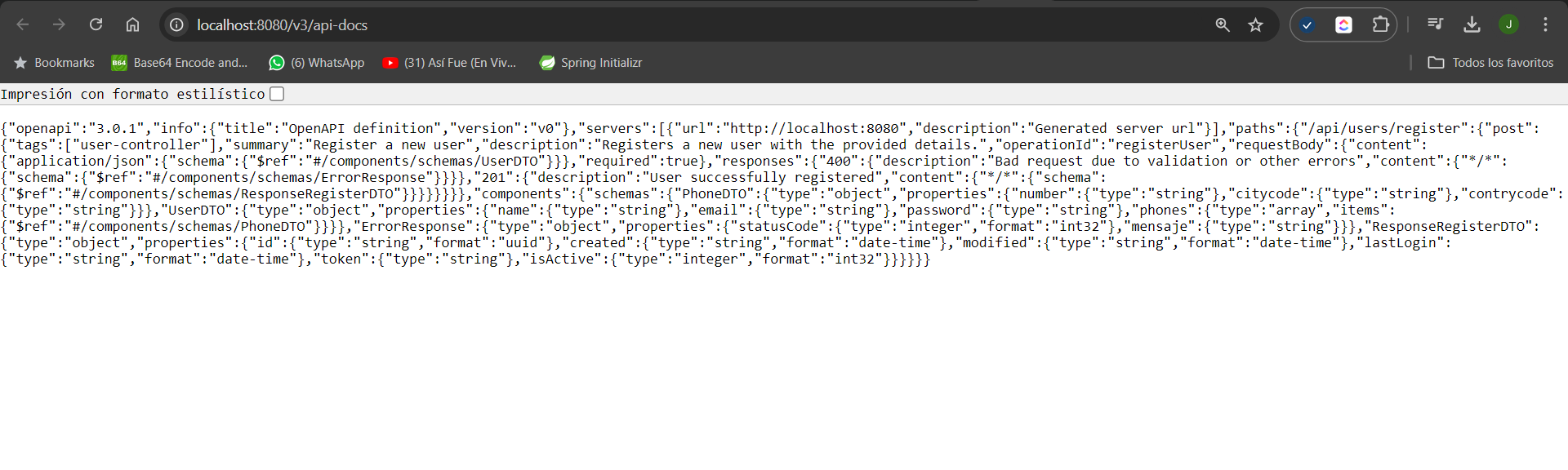
Método HTTP: POST

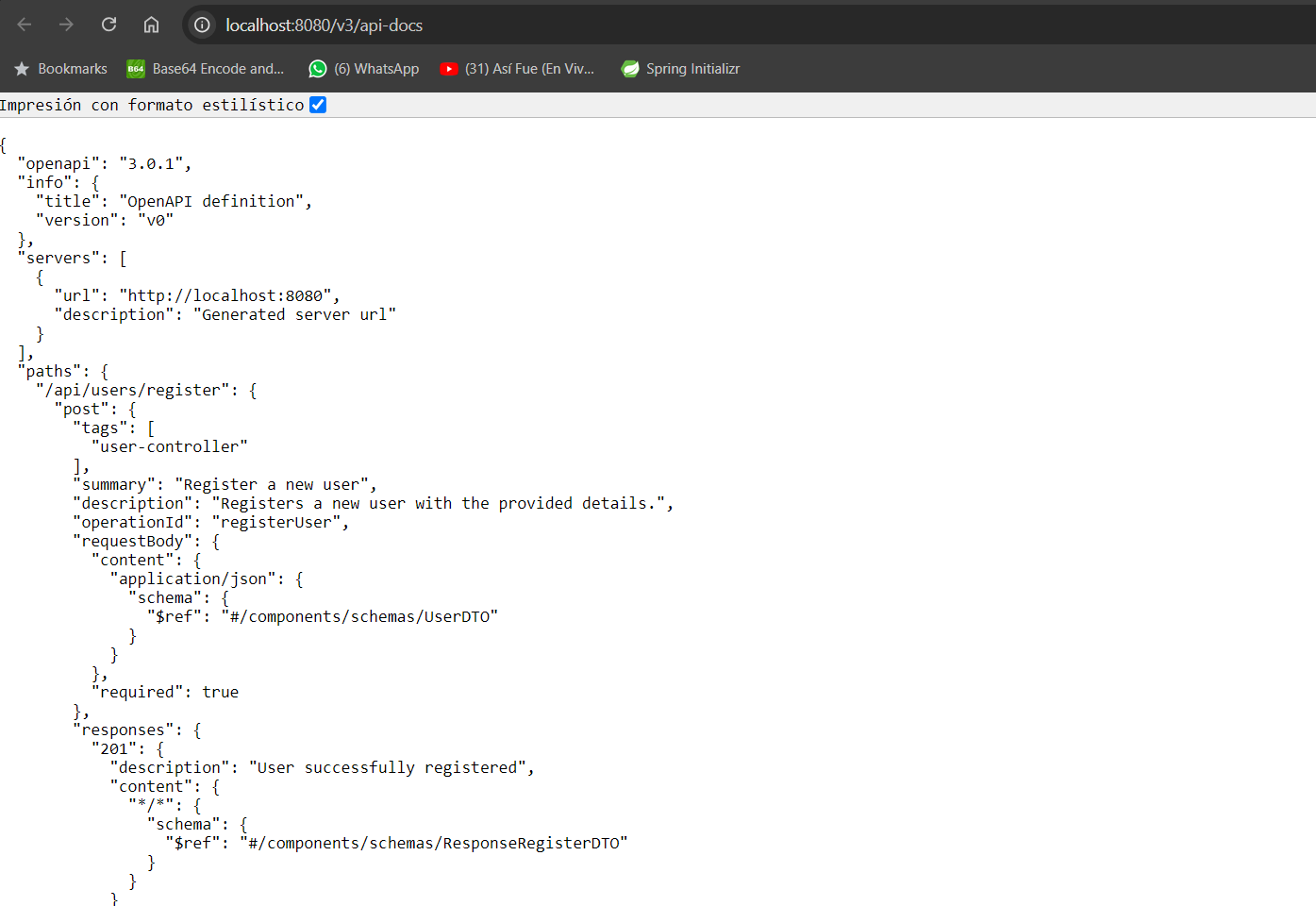
1. **VALIDAR SWAGGER DEL SERVICIO**

Realizamos la validación del swagger en la siguiente ruta <http://localhost:8080/swagger-ui/index.html> ,para ver la documentación del servicio



Validamos la documentación del api docs del servicio en la siguiente ruta <http://localhost:8080/v3/api-docs>





1. **FORMATO DE REQUEST**

{

  "name": "Pedro Mesa",

  "email": "usuario@dominio.cl",

  "password": "Password/12",

  "phones": [

    {

      "number": "123456789",

      "citycode": "01",

      "contrycode": "+56"

    }

  ]

}

1. **FORMATO DE RESPONSE**

{

    "id": "0cae411a-01c4-4419-93d7-180b661ca6c4",

    "created": "2024-09-05T23:31:25.218+00:00",

    "modified": "2024-09-05T23:31:25.218+00:00",

    "lastLogin": "2024-09-05T23:31:25.218+00:00",

    "token": "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJQZWRybyBNZXNhIiwiZW1haWwiOiJ1c3VhcmlvQGRvbWluaW8uY2wiLCJleHAiOjE3MjU1ODI2ODV9.yccgM47Ls13rid-jY68CBy1B\_-jfumkTl9hVPzMdb6dBYzSMgXPaJXso-IvtSCF8wjgOySo8ZgYiqZ7c6Uopng",

    "isActive": 1

}

1. **FORMATO DE ERROR**

{

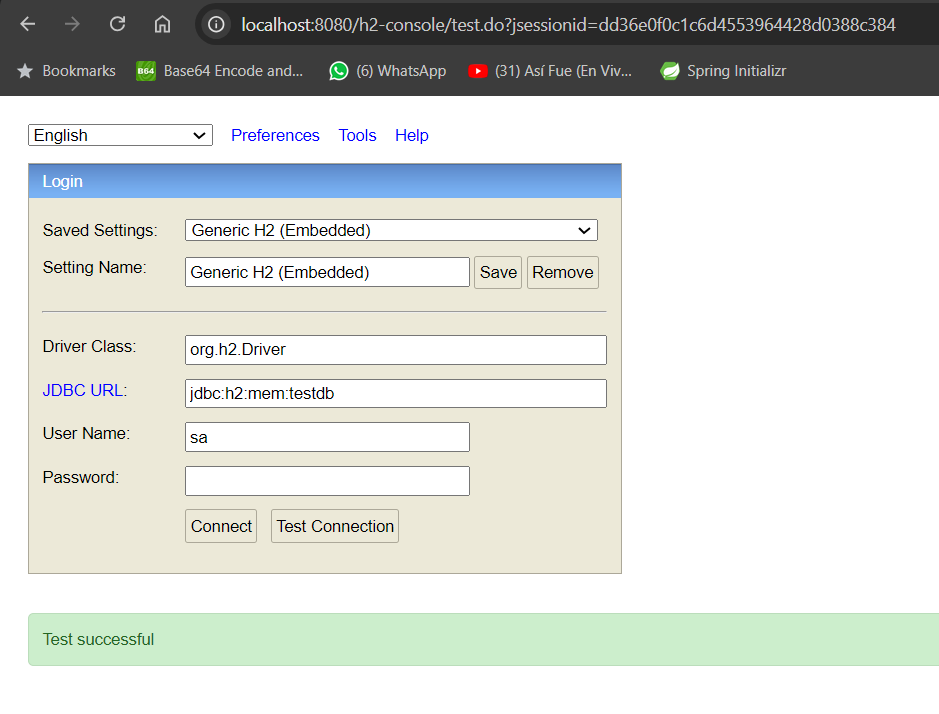
    "statusCode": 400,

    "mensaje": "Error, Formato de email inválido. Debe seguir el formato aaaaaaa@dominio.cl"

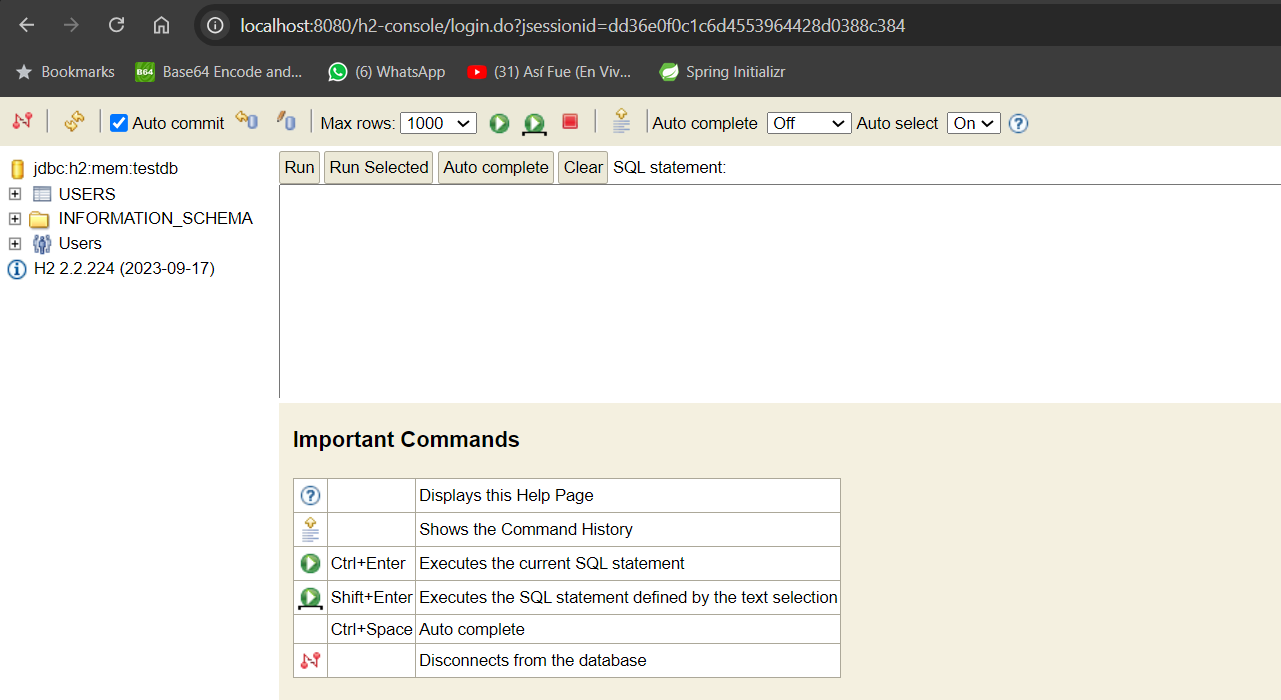
}

1. **ABRIR LA CONSOLA DE LA BD H2**

Procedemos a abrir la consola de la BD h2 en la siguiente url <http://localhost:8080/h2-console> para conectarnos a la base de datos donde se almacenará la información del registro de usuario, damos clic en test connection para validar la conexión a la BD



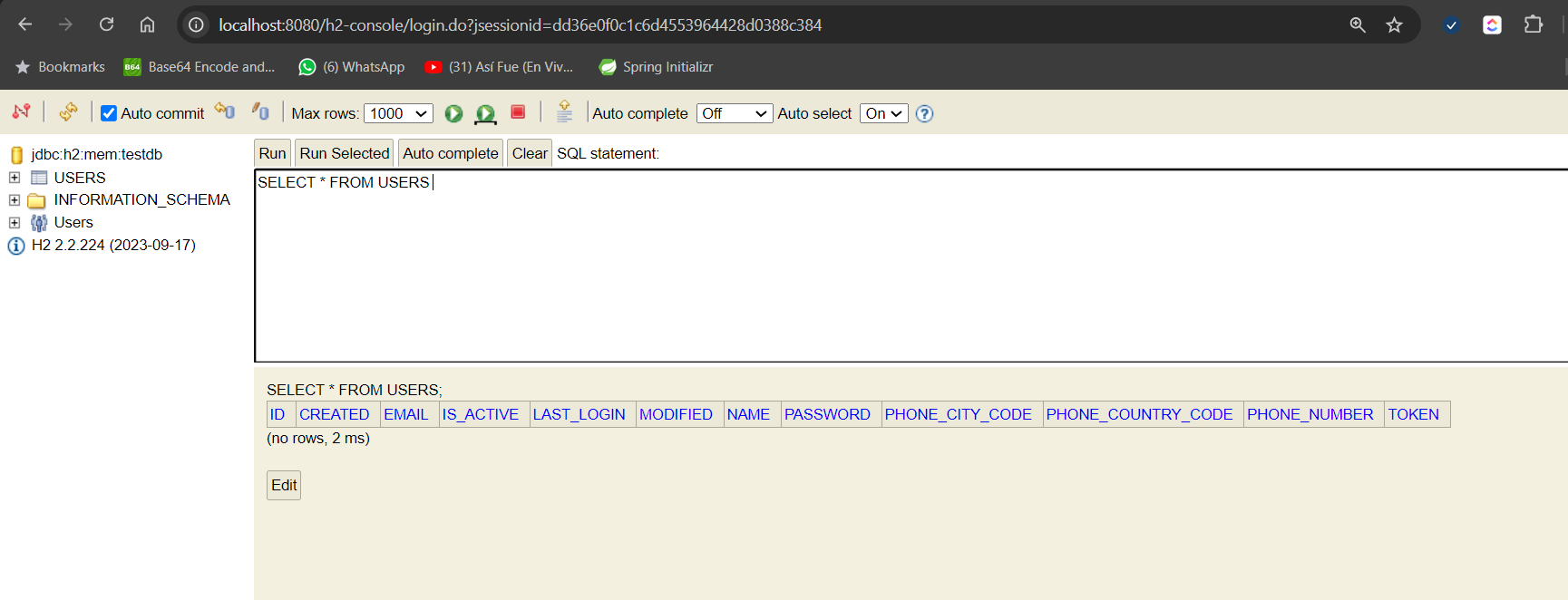
Como el test salió correcto damos clic en connect para conectarnos a la BD



Evidenciamos que la BD esta arriba y tiene creada por JPA la tabla USERS donde se almacenara la información

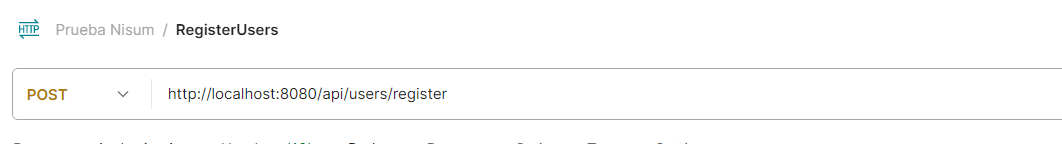
1. **SE CONSULTA LA TABLA PARA VALIDAR QUE ESTA VACÍA**

Se realiza la consulta a la tabla para evidenciar que se encuentra vacía



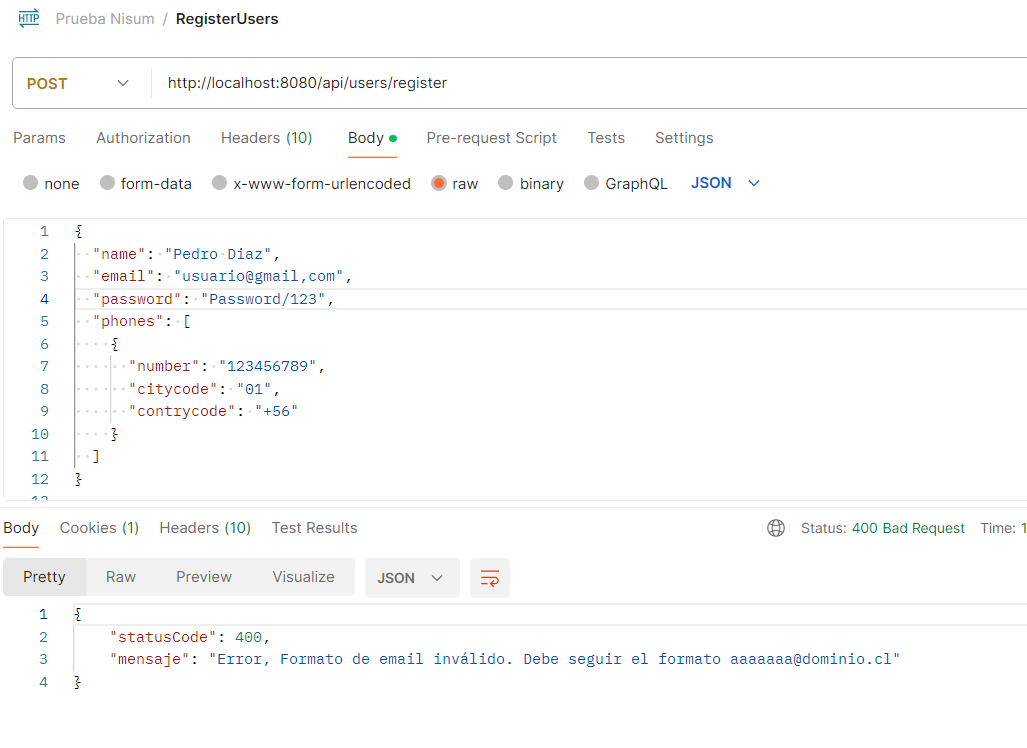
1. **IMPORTAR LA COLECCIÓN DE POSTMAN QUE SE ADJUNTÓ EN EL REPOSITORIO**

En el repositorio se encuentra la colección postman la cual se debe importar o crear para probar el servicio

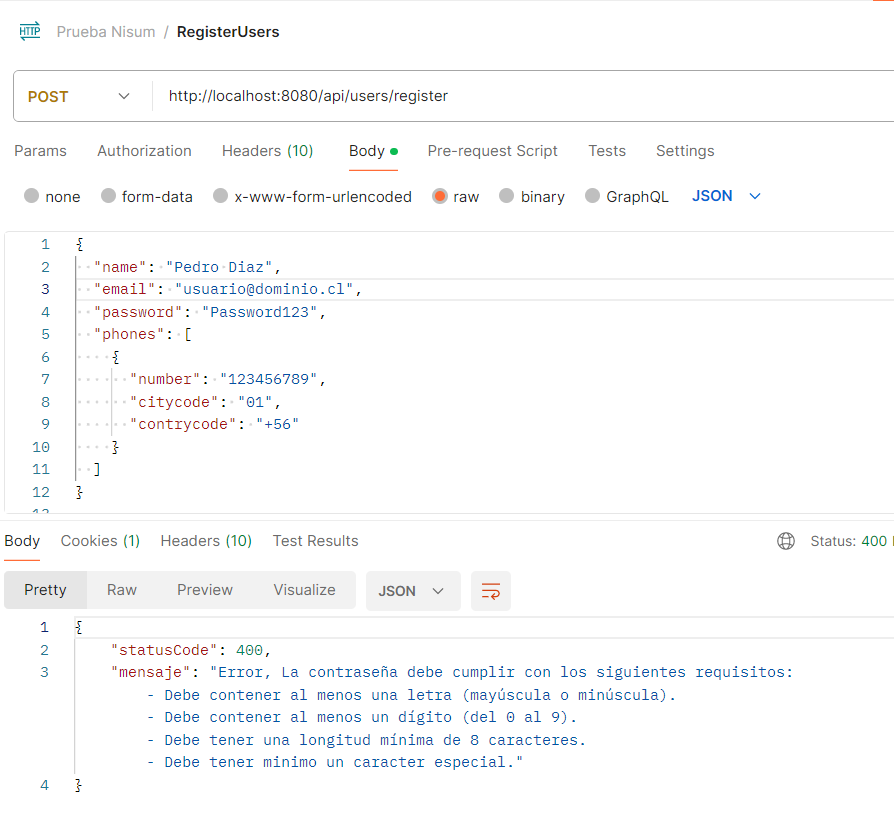


1. **EJECUTAR PRUEBAS DESDE POSTMAN**

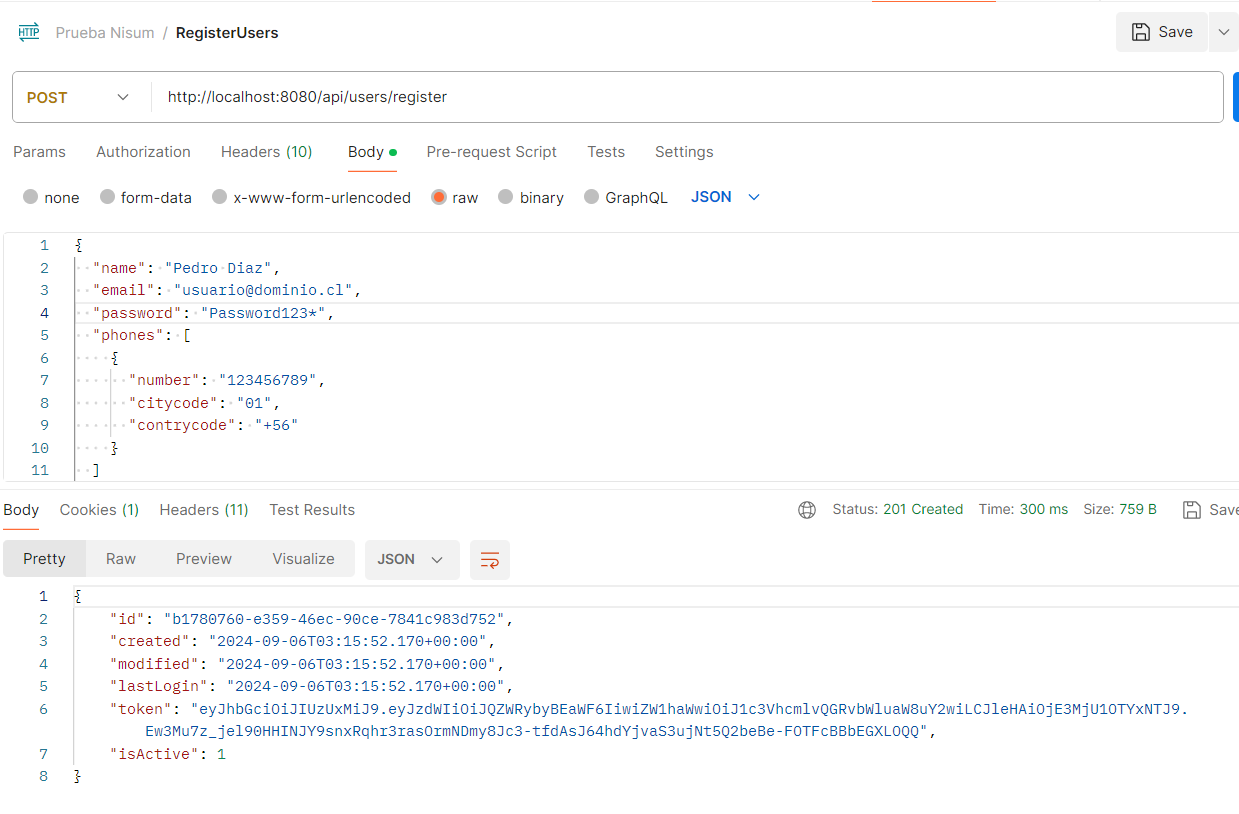
* COMO PRIMERA MEDIDA REALIZAREMOS LA PRUEBA DEL FORMATO DE CORREO



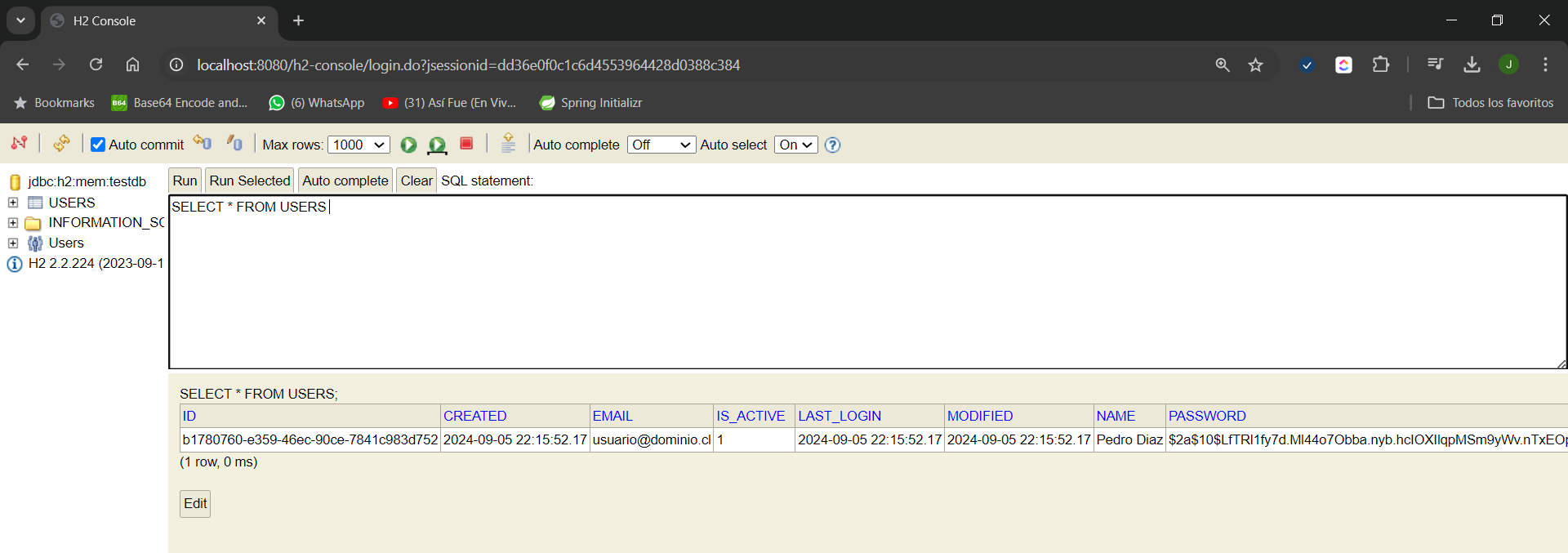
* AHORA PROBAREMOS EL FORMATO DE LA CONTRASEÑA



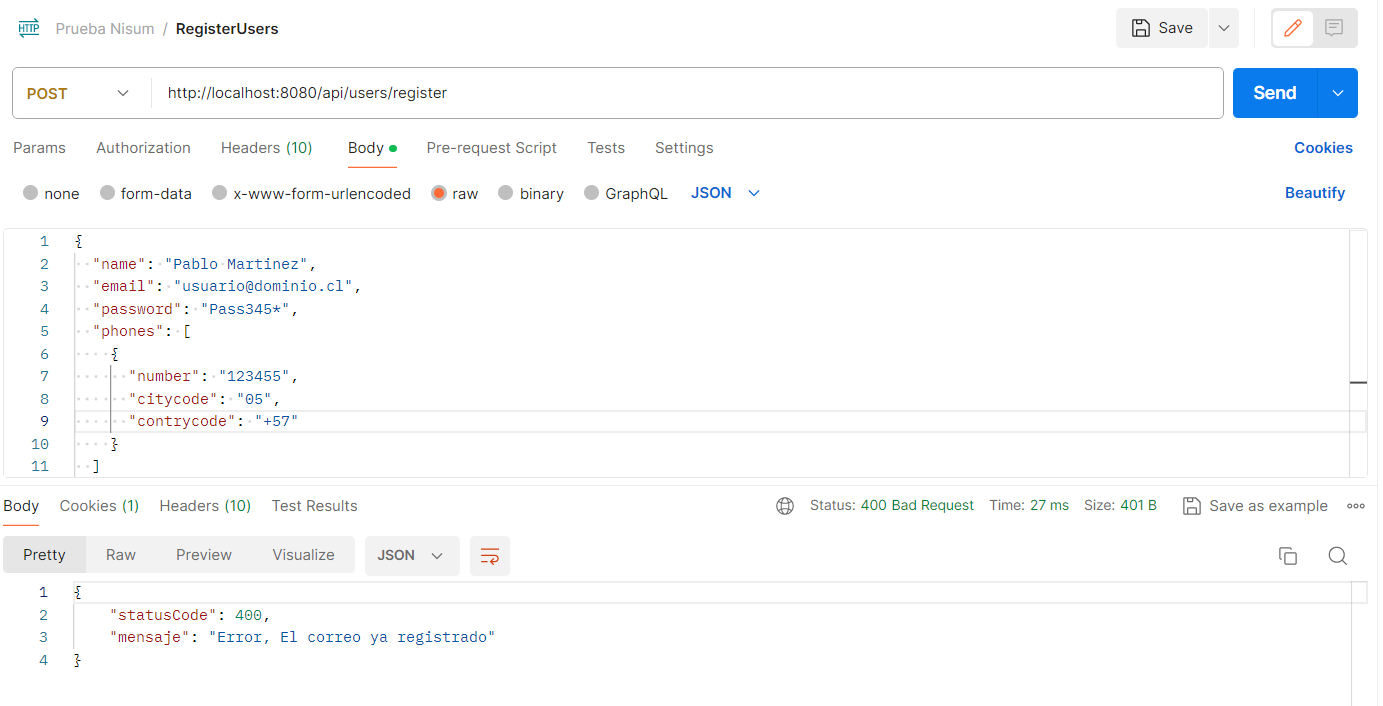
* AHORA HAREMOS UN REGISTRO CORRECTO DEL USUARIO



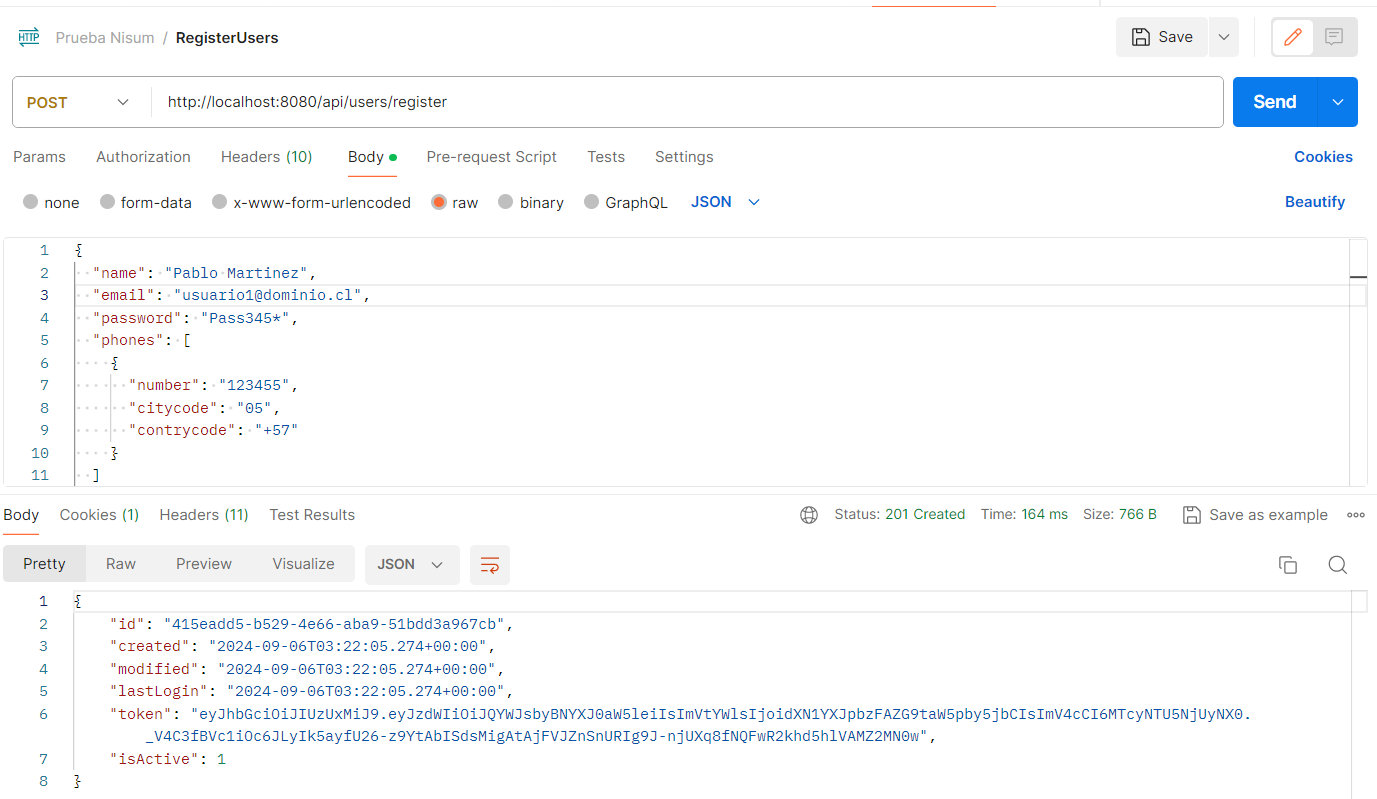
* VALIDAMOS EL REGISTRO DEL USUARIO EN LA BD



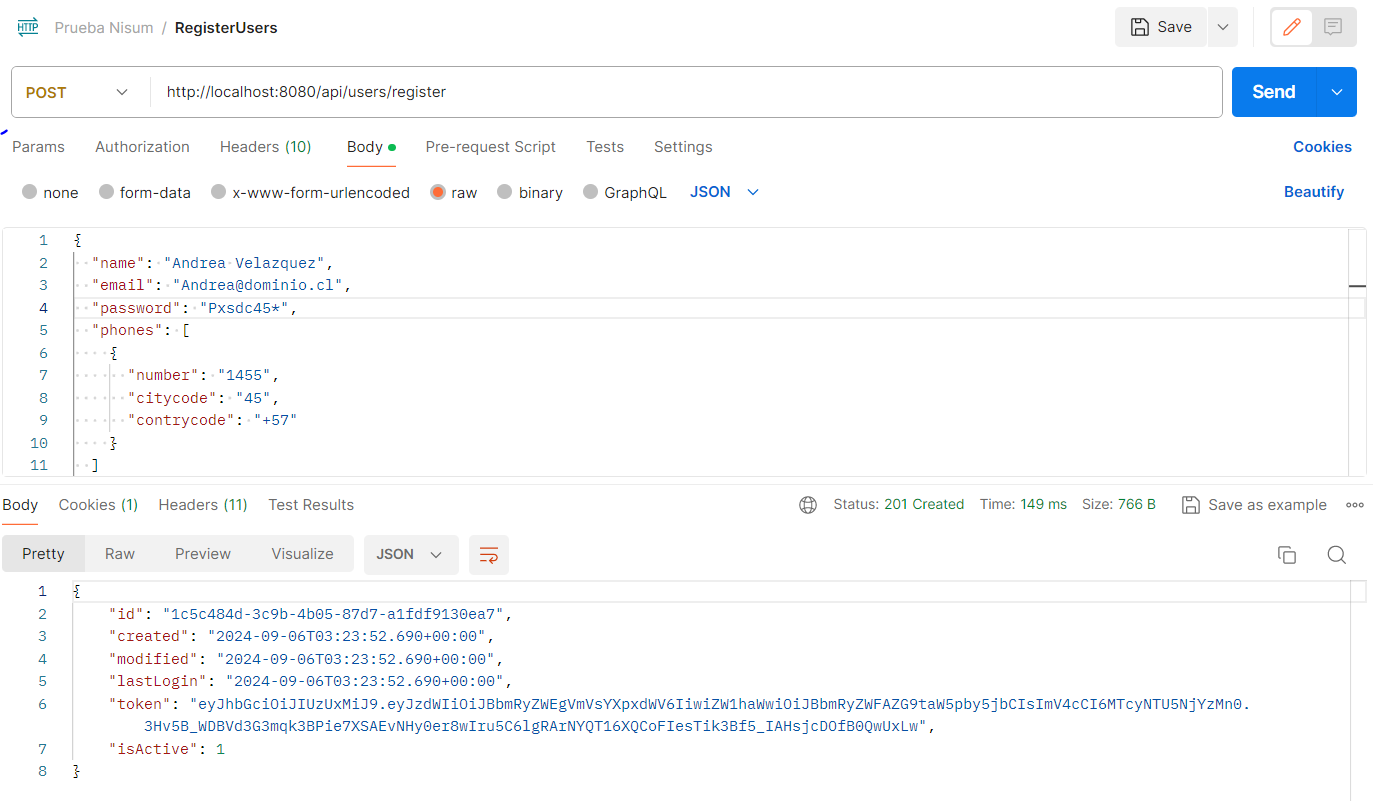
* AHORA PROBAREMOS REGISTRAR UN USUARIO CON UN CORREO YA EXISTENTE



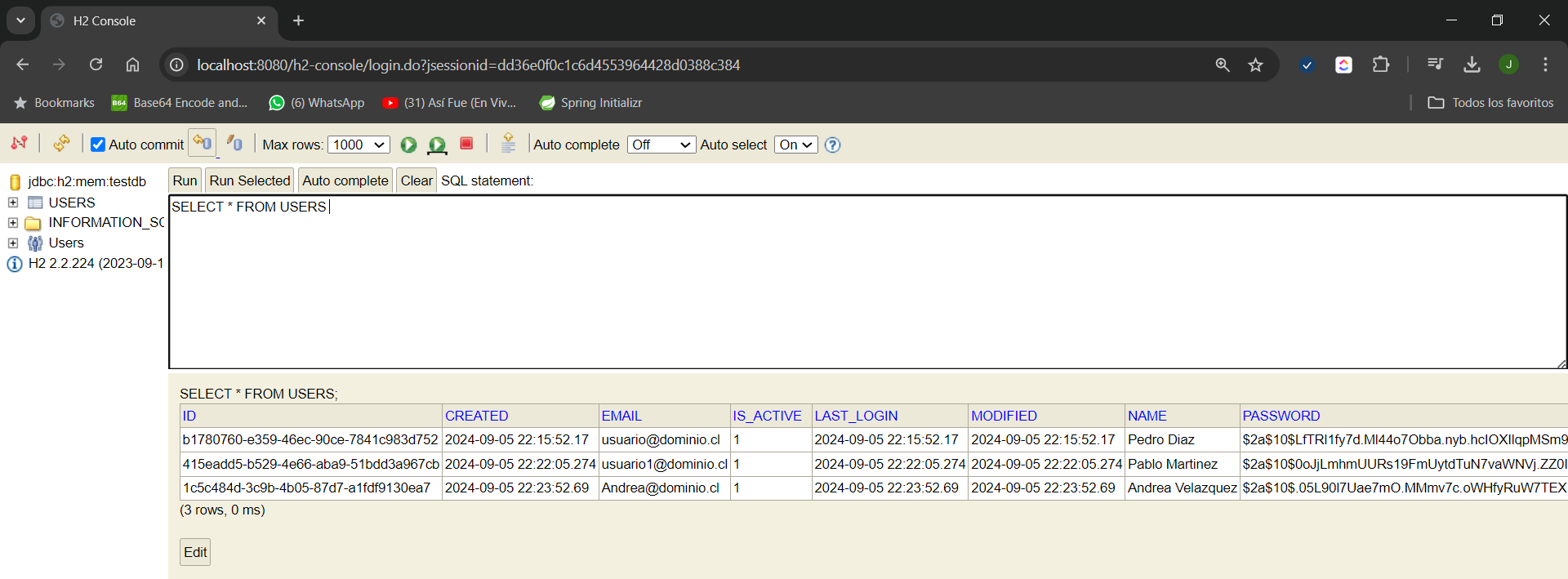
* AHORA REGISTRAREMOS VARIOS USUARIOS



* CREAMOS OTRO USUARIO



* VALIDAMOS EN LA BD QUE LOS USUARIOS HAYAN QUEDADO REGISTRADOS CORRECTAMENTE



* POR ÚLTIMO, VAMOS A VALIDAR QUE LOS JWT GENERADOS ESTEN CON LAS FIRMAS CORRECTAS EN CUALQUIER VALIDADOR DE JWT ONLINE

