

**Universidad Don Bosco Ciclo II - 2025**

**Escuela de Computación - Técnico en Ingeniería en Desarrollo de Software**

Desarrollo de Aplicaciones con Web Frameworks DWF404 G03L

Docente: Ing. Juan Carlos Menjívar Ramírez

Actividad: Desafio Práctico III.

Estudiante:

-José Adrián López Medina LM242664

Fecha de entrega: 21/10/25

# Introducción al Sistema

El sistema desarrollado es una **API RESTful** construida con el framework **Spring Boot**, cuyo propósito principal es gestionar la **autenticación y autorización de usuarios** mediante el uso de **JSON Web Tokens (JWT)** y **OAuth2 con GitHub**. Este proyecto tiene como objetivo ofrecer una base sólida y segura para cualquier aplicación moderna que requiera manejo de usuarios, inicio de sesión y control de acceso a recursos protegidos.

En términos generales, el sistema permite que los usuarios puedan **registrarse, iniciar sesión y acceder a información protegida** según su rol o nivel de permisos. Para lograrlo, implementa una arquitectura centrada en la seguridad, la modularidad y la escalabilidad, siguiendo buenas prácticas del ecosistema Spring.

Durante el proceso de registro, los datos del usuario —como nombre, correo electrónico y contraseña— se almacenan en una base de datos **MySQL**. La contraseña no se guarda en texto plano; se cifra utilizando el algoritmo **BCrypt**, lo que garantiza que incluso si la base de datos fuera comprometida, las contraseñas seguirán protegidas. Al iniciar sesión, el sistema valida las credenciales y genera un **token JWT**, el cual actúa como una “llave digital” que permite al usuario acceder temporalmente a los recursos del sistema sin necesidad de autenticarse nuevamente en cada solicitud.

La autenticación mediante **OAuth2** ofrece una alternativa más moderna y práctica. A través de la integración con **GitHub**, los usuarios pueden iniciar sesión utilizando sus credenciales de esa plataforma sin necesidad de crear una cuenta adicional. Cuando el usuario inicia sesión por primera vez a través de GitHub, el sistema verifica si ya existe en la base de datos; de no ser así, se crea un nuevo registro automáticamente. Una vez autenticado, el sistema genera también un token JWT que el cliente puede usar para acceder a los endpoints protegidos.

# Estructura General del Proyecto y Explicación de los Paquetes.

El proyecto está desarrollado en **Spring Boot**, siguiendo una arquitectura modular basada en la separación de responsabilidades. Su estructura principal se organiza en paquetes que agrupan las distintas capas del sistema:

* **controller**: Contiene las clases que exponen los endpoints REST al cliente. Se encargan de recibir las peticiones HTTP, procesarlas y devolver las respuestas adecuadas. Ejemplo: AuthController gestiona el registro y login de usuarios, mientras que UserController maneja operaciones relacionadas con los datos del usuario.
* **service**: En esta capa se encuentra la lógica de negocio. Aquí se definen los servicios encargados de procesar la información, generar los tokens JWT, validar credenciales y aplicar reglas de seguridad. Ejemplo: JwtService crea y valida los tokens, y CustomUserDetailsService carga los usuarios desde la base de datos.
* **repository**: Contiene las interfaces que interactúan con la base de datos usando **Spring Data JPA**. Por ejemplo, UserRepository permite acceder y manipular la entidad User sin necesidad de escribir consultas SQL explícitas.
* **model**: Define las **entidades JPA**, es decir, las clases que representan las tablas de la base de datos. User.java contiene los atributos y anotaciones necesarias para mapear la tabla de usuarios.
* **config**: Incluye la configuración de seguridad y beans globales. SecurityConfig.java establece las reglas de autenticación, autorización y filtros. ApplicationConfig.java define beans como el PasswordEncoder o el AuthenticationManager.
* **dto**: Contiene los **Data Transfer Objects**, clases usadas para transferir datos entre el cliente y el servidor sin exponer información sensible. Por ejemplo, AuthRequest, AuthResponse o RegisterRequest.

# Flujo de autenticación con JWT y OAuth2

El sistema implementa dos métodos de autenticación: JWT tradicional y OAuth2 con GitHub.

**Autenticación con JWT**

* El usuario se registra enviando un POST /api/auth/register con su nombre, correo y contraseña.
* La contraseña se encripta con BCrypt y se guarda en la base de datos.
* Al hacer POST /api/auth/login, el sistema valida las credenciales.
* Si son correctas, se genera un JWT con información del usuario y una fecha de expiración.
* El cliente recibe el token y lo envía en cada petición protegida mediante el header Authorization: Bearer <token>.
* Un filtro de autenticación (JwtAuthenticationFilter) valida el token y autoriza la solicitud.

**Autenticación con OAuth2 (GitHub)**

* El usuario selecciona “Iniciar sesión con GitHub”.
* Es redirigido al sitio de GitHub para autorizar la aplicación.
* GitHub devuelve los datos del usuario al backend.
* Si el usuario no existe, se crea automáticamente en la base de datos.
* Se genera un JWT y se devuelve al frontend (por redirección o JSON).
* A partir de ese momento, el flujo es igual al de JWT: el token se usa para acceder a los endpoints protegidos.

# Seguridad con roles y uso de @PreAuthorize

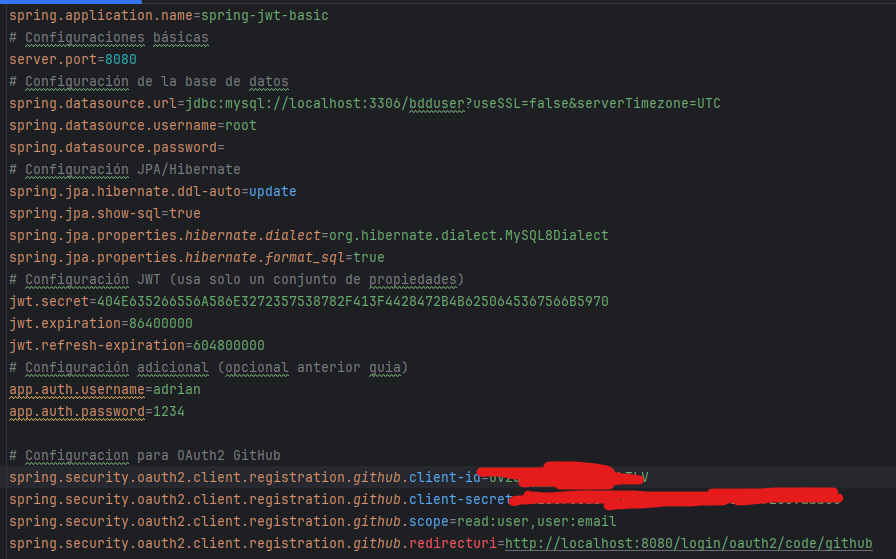
El proyecto utiliza Spring Security para definir roles y permisos.

Cada usuario posee un rol, como ROLE\_USER o ROLE\_ADMIN, almacenado en su entidad.



Esto indica que solo los usuarios con rol ADMIN pueden acceder a esa ruta. Spring Security evalúa el token JWT, obtiene los roles del usuario autenticado y decide si la solicitud es válida o no.

# Explicación del archivo application.properties



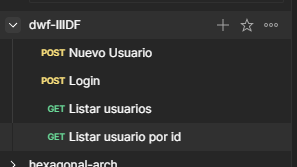
El archivo application.properties es el punto central de configuración del proyecto, donde se definen los parámetros necesarios para que la aplicación funcione correctamente. En este archivo se especifican tanto los datos de conexión con la base de datos como la configuración de seguridad, los valores asociados al uso de JWT y la integración con OAuth2 a través de GitHub.

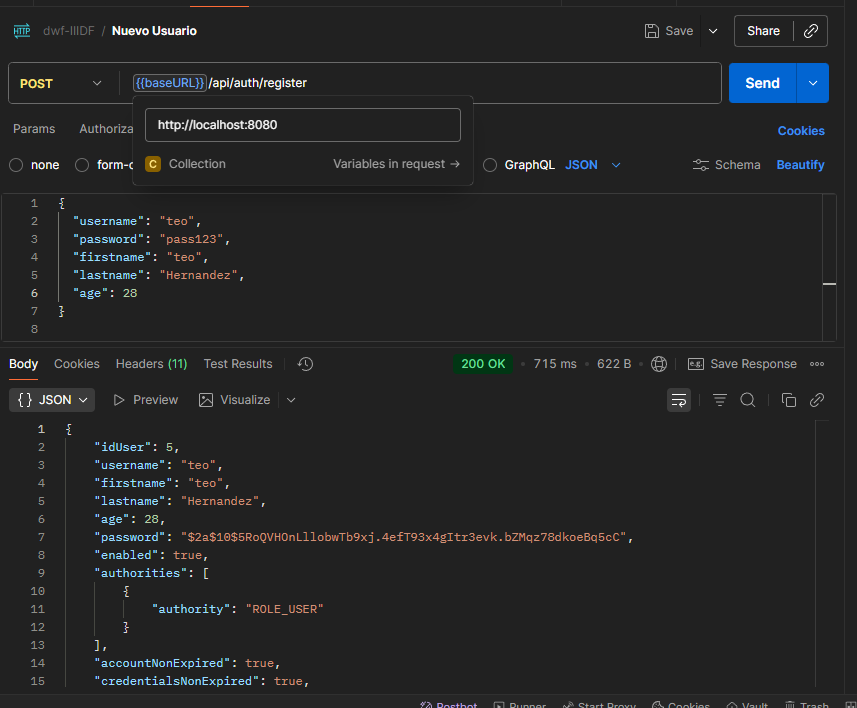
Se configuran los parámetros de conexión a la base de datos MySQL, indicando la URL del servidor, el nombre de la base de datos, el usuario y la contraseña. Esto permite que el sistema establezca comunicación con la base de datos y almacene la información de los usuarios registrados.

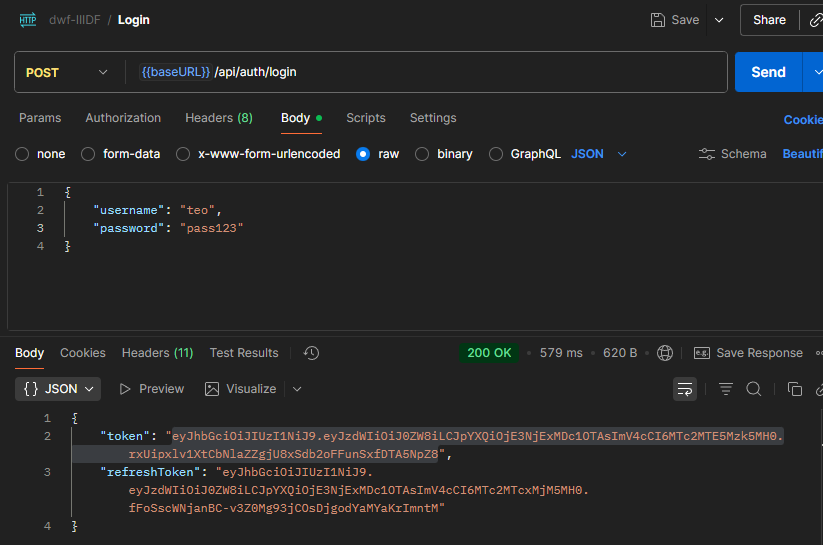
En la sección de seguridad y autenticación, se definen los parámetros utilizados para generar y validar los JSON Web Tokens (JWT). Entre ellos se encuentra el secreto criptográfico (jwt.secret), que es una cadena de texto empleada para firmar digitalmente los tokens y garantizar que no sean alterados. Por motivos de seguridad, este valor no debe escribirse directamente en el archivo, sino que se recomienda almacenarlo como una variable de entorno para evitar su exposición en el repositorio. También se establecen los tiempos de expiración del token principal y del token de refresco, definidos en milisegundos, que determinan cuánto tiempo puede permanecer válido un token antes de necesitar renovarse.

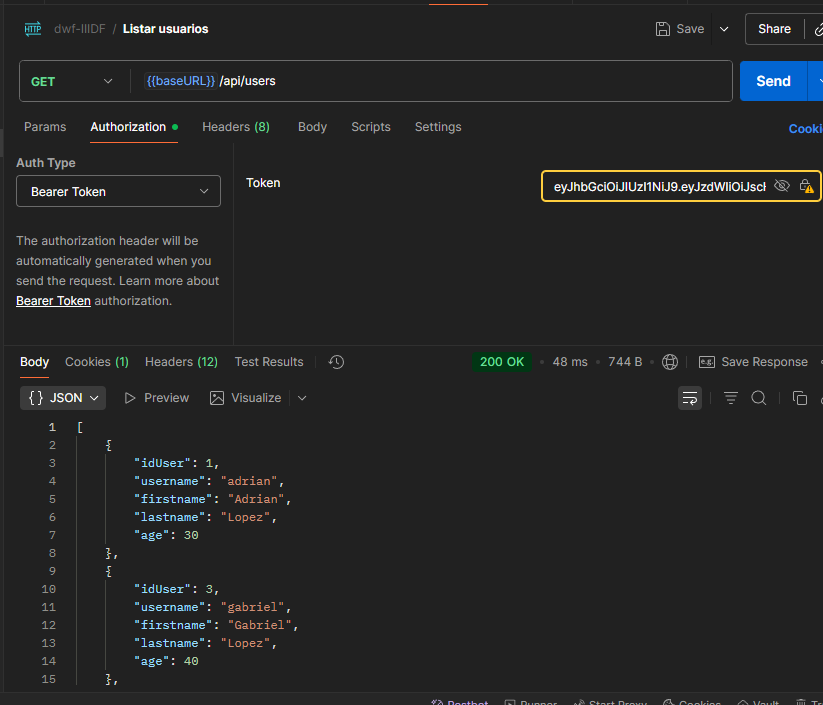
El archivo también contiene la configuración necesaria para la autenticación mediante OAuth2 con GitHub. En esta parte se incluyen el identificador de cliente y el secreto de cliente que GitHub proporciona al registrar la aplicación. Estos valores permiten que la API se comunique de forma segura con GitHub y gestione el flujo de inicio de sesión de los usuarios externos. Al igual que con el JWT secret, estas credenciales deben mantenerse fuera del código fuente y gestionarse mediante variables de entorno (GITHUB\_CLIENT\_ID y GITHUB\_CLIENT\_SECRET) para evitar riesgos de seguridad.

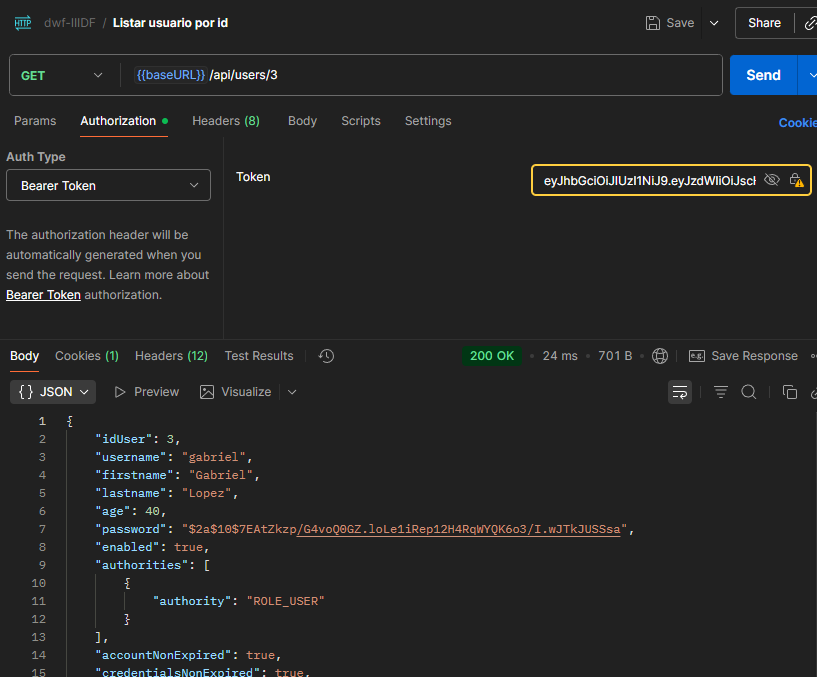
# Ejemplo de pruebas (Postman)

Collection realizada:  










# Conclusiones personales sobre la implementación

La realización de este sistema fue una experiencia muy valiosa, porque me permitió poner en práctica muchos de los conocimientos que aprendí durante la carrera y en las guías de laboratorio. A lo largo del desarrollo pude ver cómo todos los conceptos que había estudiado —como autenticación, seguridad, manejo de usuarios y conexión con bases de datos— se unen para formar un proyecto completo y funcional.

Trabajar con Spring Boot me ayudó a entender mejor cómo se organiza una aplicación real, con sus diferentes capas y responsabilidades. También aprendí la importancia de mantener un código limpio y ordenado, separando las partes que manejan la lógica, la seguridad y la base de datos. Al principio resultó un poco desafiante configurar la seguridad con JWT y OAuth2, pero al final logré comprender cómo funciona todo el proceso de inicio de sesión y validación de usuarios.

El uso de JWT fue una de las partes que más me gustó, ya que pude ver cómo un simple token puede reemplazar las sesiones tradicionales y hacer que el sistema sea más rápido y seguro. Por otro lado, la integración con GitHub mediante OAuth2 fue interesante porque muestra cómo los usuarios pueden ingresar a una aplicación usando cuentas externas, algo muy común en los sistemas actuales.