Manual de Usuario Prueba Técnica Nisum

1. Resumen

Este esquema de directorios y archivos refleja una estructura típica de proyecto Java con Spring Boot, organizada por capas (controladores, servicios, repositorios) y componentes específicos para la autenticación JWT, gestión de usuarios y seguridad. Cada directorio y archivo juega un papel crucial en la arquitectura y funcionalidad general del proyecto.

2. Tecnologias

- Java 11(SpringBoot) - Maven
- H2 (DataBase)
- Docker
- Tomcat

3. Diseño Arquitectura

4. Directorios y Contenido

1. **src/**
    - **main/**
        - **java/**
            - **com/**
                - **example/**

- **demo/**
  - **controller_jwt/**: Contiene controladores relacionados con la autenticación y autorización JWT.
    - **UserController.java**: Controlador para manejar las operaciones relacionadas con los usuarios.
  - **dto/**: DTOs (Data Transfer Objects) utilizados para representar datos en las solicitudes y respuestas.
    - **JwtDto.java**: DTO para el token JWT.
    - **NewUser.java**: DTO para la creación de nuevos usuarios.
    - **loginUser.java**: DTO para el login de usuarios.
    - **Mensaje.java**: DTO para mensajes de respuesta.
  - **user/**: Contiene las entidades relacionadas con los usuarios.
    - **entity/**
      - **Rol.java**: Entidad que representa el rol de usuario.
      - **User.java**: Entidad principal que representa a un usuario.
      - **Phone.java**: Entidad que representa un teléfono asociado a un usuario.
    - **repository/**: Repositorios JPA para acceder a datos.
      - **RolRepository.java**: Repositorio para la entidad Rol.
      - **UserRepository.java**: Repositorio para la entidad User.
      - **PhoneRepository.java**: Repositorio para la entidad Phone.
    - **service/**: Contiene interfaces y servicios relacionados con la lógica de negocio.
      - **RolService.java**: Interfaz para el servicio relacionado con roles.

        - **UserService.java**: Interfaz para el servicio relacionado con usuarios.
        - **PhoneService.java**: Interfaz para el servicio relacionado con teléfonos.
- **serviceImpl/**: Implementaciones concretas de los servicios.
    - **UserServiceImpl.java**: Implementación del servicio UserService.
- **login/**: Contiene clases relacionadas con la autenticación.
    - **RolName.java**: Enumeración que define los nombres de los roles.
- **jwt/**: Clases relacionadas con la gestión de tokens JWT para autenticación.
    - **JwtEntityPoint.java**: Punto de entrada para la configuración de JWT.
    - **JwtProvider.java**: Proveedor JWT para la generación y validación de tokens.
    - **JwtTokenFilter.java**: Filtro para interceptar y procesar tokens JWT en las solicitudes.
- **util/**: Utilidades varias utilizadas en el proyecto.
    - **CreateRoles.java**: Clase utilitaria para la creación inicial de roles.
    - **UserPrincipal.java**: Clase que representa al principal del usuario autenticado.
    - **ValidEmail.java**: Validador para direcciones de correo electrónico.
- **security/**: Configuraciones de seguridad de la aplicación.

- **MainSecurity.java**: Configuración principal de seguridad.
- **ControllerSecurity/**: Controladores relacionados con la seguridad.
  - **AuthController.java**: Controlador para manejar las operaciones de autenticación.

- **resources/**
  - **application.properties**: Archivo de configuración principal de la aplicación, que incluye configuraciones de base de datos, etc.

- **test/**
  - **java/**
    - **com/**
      - **example/**
        - **demo/**
          - **tests/**: Contiene las pruebas unitarias y de integración del proyecto.
            - **CreateRolesTest.java**: Pruebas para la clase CreateRoles.
            - **JwtDtoTest.java**: Pruebas para el DTO JwtDto.
            - **JwtEntityPointTest.java**: Pruebas para la configuración de JwtEntityPoint.
            - **JwtEntityProviderTest.java**: Pruebas para el proveedor de entidad JWT.
            - **JwtProviderTest.java**: Pruebas para el proveedor JWT.
            - **PhoneServiceTest.java**: Pruebas para el servicio PhoneService.
            - **RolServiceTest.java**: Pruebas para el servicio RolService.
            - **UserServiceImplTest.java**: Pruebas para la implementación UserServiceImpl.
            - **UserControllerTest.java**: Pruebas para el controlador UserController.
    - **resources/**

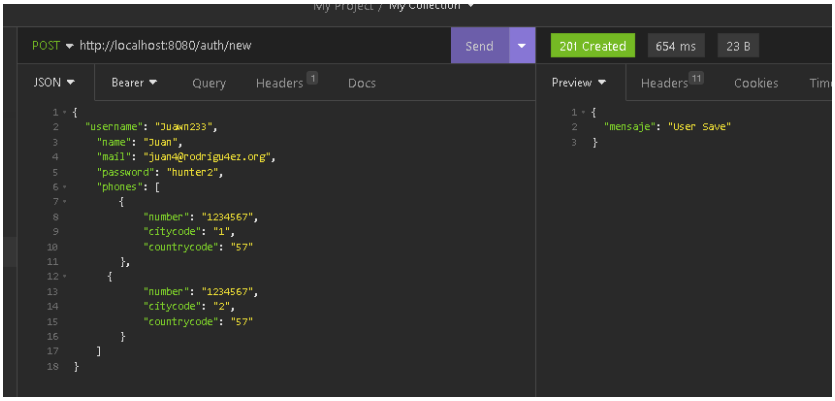
        - **test-application.properties**: Archivo de configuración específico para pruebas.
2. **pom.xml**: Archivo de configuración de Maven que especifica las dependencias del proyecto y otros detalles.
3. **Manual Usuario Pruebas Endpoints Nisum.pdf**: Documentación del manual de usuario y pruebas de los endpoints.

```
├── src/
│   ├── main/
│   │   ├── java/
│   │   │   └── com/
│   │   │       └── example/
│   │   │           └── demo/
│   │   │               ├── controller_jwt/
│   │   │               │   └── UserController.java
│   │   │               ├── dto/
│   │   │               │   ├── JwtDto.java
│   │   │               │   ├── NewUser.java
│   │   │               │   ├── loginUser.java
│   │   │               │   └── Mensaje.java
│   │   │               └── user/
│   │   │                   ├── entity/
│   │   │                   │   ├── Rol.java
│   │   │                   │   ├── User.java
│   │   │                   │   └── Phone.java
│   │   │                   ├── repository/
│   │   │                   │   ├── RolRepository.java
│   │   │                   │   ├── UserRepository.java
│   │   │                   │   └── PhoneRepository.java
│   │   │                   ├── service/
│   │   │                   │   ├── RolService.java
│   │   │                   │   ├── UserService.java
│   │   │                   │   └── PhoneService.java
│   │   │                   ├── serviceImpl/
│   │   │                   │   └── UserServiceImpl.java
│   │   │                   ├── login/
│   │   │                   │   └── RolName.java
│   │   │                   ├── jwt/
│   │   │                   │   ├── JwtEntityPoint.java
│   │   │                   │   ├── JwtProvider.java
│   │   │                   │   └── JwtTokenFilter.java
```

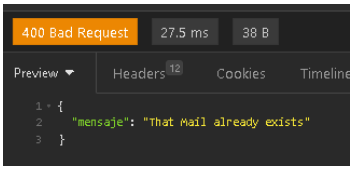
```
39. |   |   |               ├── util/
40. |   |   |               |   ├── CreateRoles.java
41. |   |   |               |   ├── UserPrincipal.java
42. |   |   |               |   └── ValidEmail.java
43. |   |   |               └── security/
44. |   |   |                   ├── MainSecurity.java
45. |   |   |                   └── ControllerSecurity/
46. |   |   |                       └── AuthController.java
47. |   |   └── resources/
48. |   |       └── application.properties
49. |   └── test/
50. |       ├── java/
51. |       |   └── com/
52. |       |       └── example/
53. |       |           └── demo/
54. |       |               └── tests/
55. |       |                   ├── CreateRolesTest.java
56. |       |                   ├── JwtDtoTest.java
57. |       |                   ├── JwtEntityPointTest.java
58. |       |                   ├── JwtEntityProviderTest.java
59. |       |                   ├── JwtProviderTest.java
60. |       |                   ├── PhoneServiceTest.java
61. |       |                   ├── RolServiceTest.java
62. |       |                   ├── UserServiceImplTest.java
63. |       |                   └── UserControllerTest.java
64. |       └── resources/
65. |           └── test-application.properties
66. ├── pom.xml
67. └── Manual Usuario Pruebas Endpoints Nisum.pdf
```

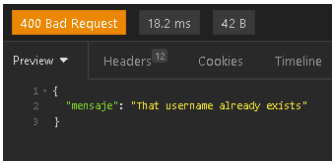
5. **EndPoints**
    1. New User (PostMethod)
        - http://localhost:8080/auth/new
        - No necesita token de autentificacion
        - Crear nuevo usuario
        - Json:

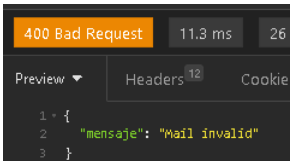
```
{
        "username": "juan123",
        "name": "Juan Rodriguez",
        "mail": "juan@rodriguez.org",
        "password": "hunter2",
        "phones": [
                {
                        "number": "1234567",
                        "citycode": "1",
                        "contrycode": "57"
                }
               ]

}
```



- Username duplicado tendrá su respectiva validación
- Mail duplicado tendrá su respectiva Validación
- Mail mal formado tendrá su respectiva Validación

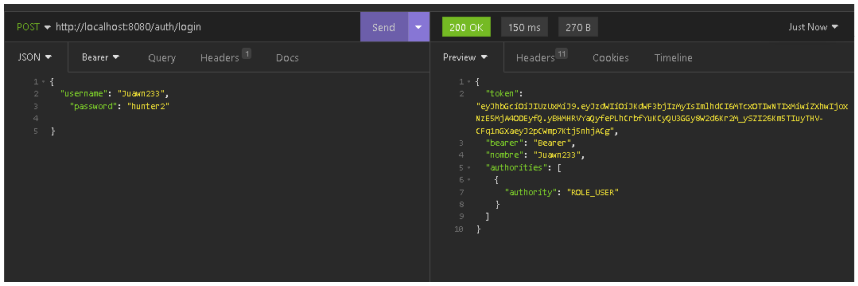






2. Login (PostMethod)
    - http://localhost:8080/auth/login
    - Acceder usando username y password, generará los datos de acceso y un token
    - Json:

```
{
  "username": "Juawn233",
  "password": "hunter2"
}
```

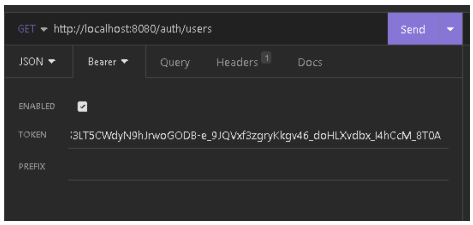


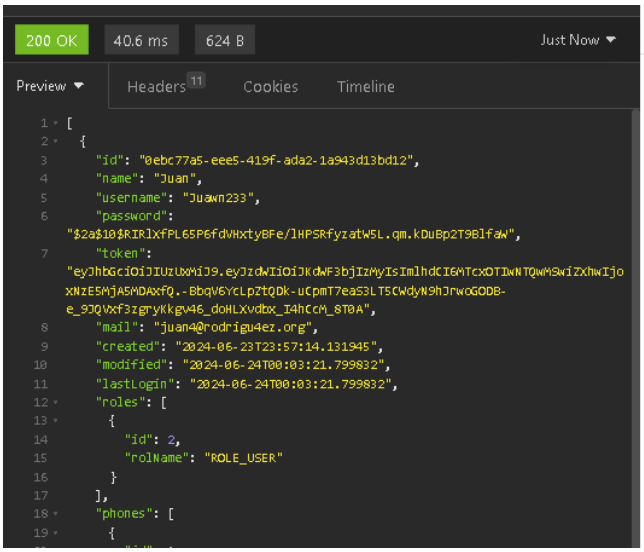
- En caso contrario arrojará Unathorized (Sin acceso)

```
{
  "timestamp": "2024-06-24T05:02:26.165+00:00",
  "status": 401,
  "error": "Unauthorized",
  "trace":
"org.springframework.security.authentication.InternalAu
eption: No value present\r\n\tat
org.springframework.security.authentication.dao.DaoAuth
rieveUser(DaoAuthenticationProvider.java:108)\r\n\tat
```


3 GetUsers (GetMethod)

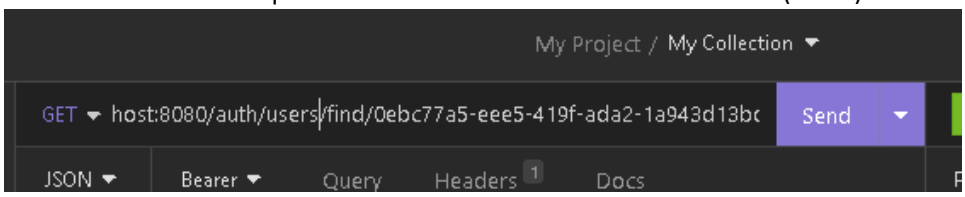
2. http://localhost:8080/auth/users
3. Se deberá usar Bearer como método de acceso mediante token y registrar el token perteneciente a la sesión iniciada para poder así acceder y evidenciar los usuarios guardados/registrados previamente.





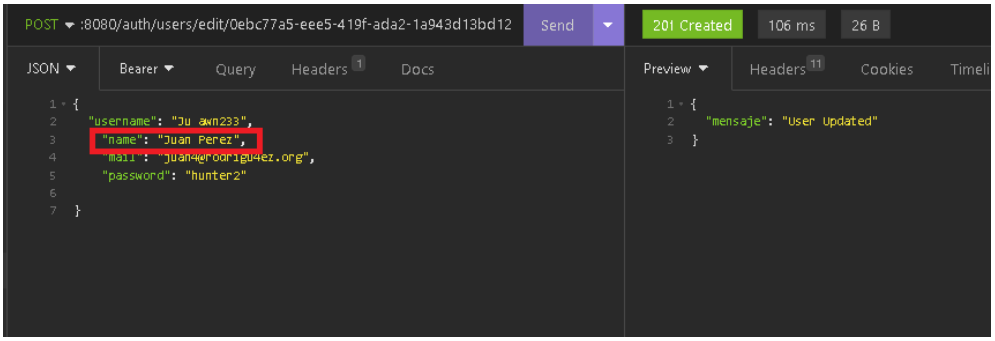
4. FindById (GetMethod)

- http://localhost:8080/auth/users/find/{id}
- Se deberá usar Bearer como método de acceso mediante token y registrar el token perteneciente a la sesión iniciada para poder así acceder y poder realizar la búsqueda.
- Se realiza búsqueda de los datos del usuario mediante id (UUID)

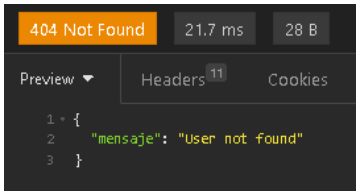


```
{
  "id": "0ebc77a5-eee5-419f-ada2-1a943d13bd12",
  "name": "Juan",
  "username": "Juawn233",
  "password": "$2a$10$RIRlXfPL65P6fdVHxtyBFe/lHPSRfyzatW5L.qm.kDuBp2T9BlfaW",
  "token":
"eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJKdWF3bjIzMyIsImlhdCI6MTcxOTIwNTQwMSwiZXhwIjox
NzE5MjA5MDAxfQ.-BbqV6YcLpZtQDk-uCpmT7eaS3LT5CWdyN9hJrwoGODB-
e_9JQVxf3zgryKkgv46_doHLXvdbx_I4hCcM_8T0A",
  "mail": "juan4@rodrigu4ez.org",
  "created": "2024-06-23T23:57:14.131945",
  "modified": "2024-06-24T00:03:21.799832",
  "lastLogin": "2024-06-24T00:03:21.799832",
  "roles": [
    {
      "id": 2,
      "rolName": "ROLE_USER"
    }
```


5. UpdateById (PostMethod)
   - http://localhost:8080/auth/users/edit/{id}
   - Se deberá usar Bearer como método de acceso mediante token y registrar el token perteneciente a la sesión iniciada para poder así acceder y realizar la edición.
   - Se editará usuario mediante el uso del PathVariable para búsqueda del registro, y se realizar modificaciones del objeto según sea la necesidad
   - Datos básicos accesibles a modificaciones (name, username, password, mail), usando el método GetUsers, se podrá evidenciar el cambio en los datos.

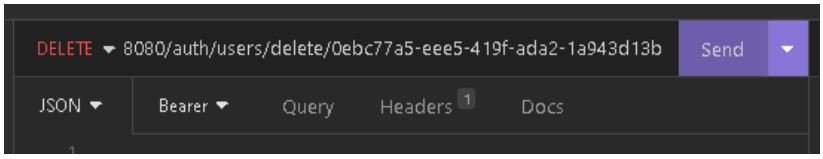


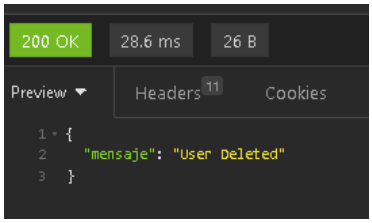
- En caso de no encontrar el usuario, mostrará mensaje de User Not Found



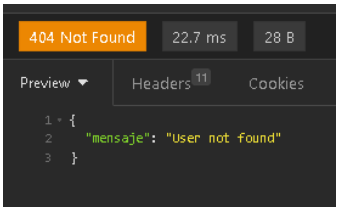
6. DeleteUserById (DeletMethod)

- http://localhost:8080/auth/users/delete/{id}
- Se deberá usar Bearer como método de acceso mediante token y registrar el token perteneciente a la sesión iniciada para poder así acceder y realizar la eliminación.
- Se realiza búsqueda de los datos del usuario a eliminar mediante id (UUID)





- En caso de no encontrar el respectivo usuario a eliminar, se mostrará un mensaje de User Not Found.



6. **DockerFile**

- Archivo de configuración Docker con el propósito de crear una imagen y poder ser desplegada en el servidor Tomcat (Configuracion de preferencia)