SPRINT 4. MICROSERVICIOS

Marcela Peñaranda Ortega – C.C: 1091670131

Brandom Mosquera Cadena– C.C: 1013674995

Sergio González Betin -C.C: 92529506

Miguel González Rodríguez– C.C: 1010005478

Alejandra Rosero Maya – C.C: 1233194367

MISION TIC 2022 – DESARROLLO WEB

UNIVERSIDAD DE ANTIOQUIA

10 DE OCTUBRE DE 2021

**Objetivo Grupal**

* Crear los microservicios correspondientes a nuestro proyecto basado en el marco de trabajo Spring Boot, que permita interactuar con la base de datos y cumplir con las necesidades del negocio identificadas para el proyecto.

**Proceso para alcanzar los objetivos y requerimientos del sprint.**

Para poder llevar a cabo la ejecución del Sprint 4, cada uno de los miembros del equipo escogieron una tarea específica con el propósito de cumplir la entrega.

Con el propósito de garantizar la trazabilidad del proyecto se implementó el uso de GitHub y Trello, pues en Github se puede registrar los cambios del proyecto (como agregación del proyecto y pruebas), mientras que en Trello se especifican las tareas de cada miembro:

1. Dentro de Trello, se gestionaron las tareas del sprint 4 para cada miembro, en donde se incluían el desarrollo de los endpoints necesarios para realizar tareas de gestión de datos en la base de datos (CRUD) que permitan el funcionamiento de la app.
2. Para la gestión de la actualización del proyecto, cada uno de los miembros se encargó de subir los archivos desarrollados con el propósito de hacer un pull request una vez estén completados.
3. Para crear los endpoints se hizo seguimiento de la tutoría dada en clase a fin de garantizar un buen desarrollo de las mismas.
4. Cada integrante del equipo realizó la descarga de IDE Spring Tool Suite como herramienta para trabajar con el framework Spring Boot.

De nuevo, para actualizar el repositorio de Git se hizo uso de las ramas iniciales y se modificaron las carpetas de fuentes y bases de datos para indicar las instrucciones que se utilizaron para crear los endpoints de microservicios.

**Rutas**

**GitHub:**   <https://github.com/AlejandraRo/ciclo4-proyectofinal>

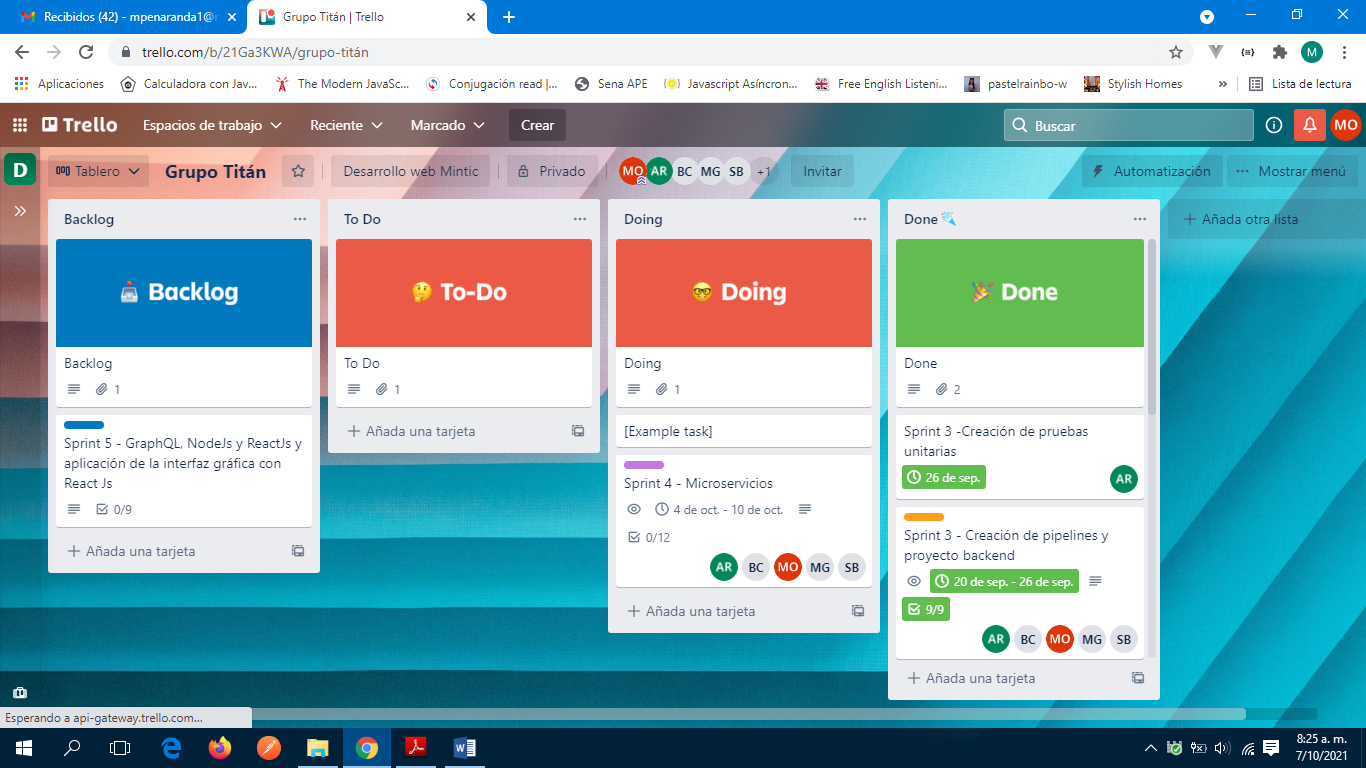
(Dentro de la rama Development en la carpeta Microservicios)

**Trello:** <https://trello.com/b/21Ga3KWA/grupo-tit%C3%A1n>

**MongoDB:** <https://cloud.mongodb.com/v2/613e16830a992558f04f17cc#access>

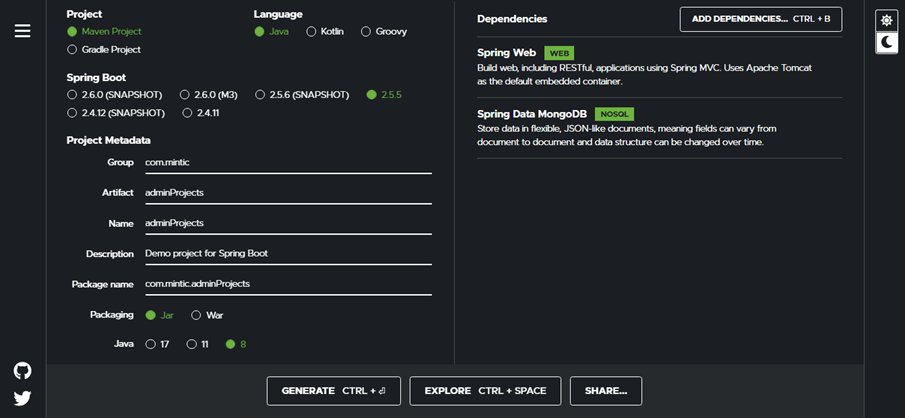
**Azure:** <https://dev.azure.com/ssgonzalezb/manage_proyects/_build>

**EVIDENCIAS**



**INICIALIZACIÓN DE PROYECTO SPRING BOOT**

Inicializamos el proyecto usando la herramienta Spring Initializr para crear la estructura básica del proyecto, añadiendo las configuraciones y dependencias necesarias para el mismo.



**ESTRUCTURA DEL PROYECTO**

**ENTIDADES**

* **ProjectEntity – UserEntity - ReportEntity:** Modelamos los campos definidos en la base de datos para la colección proyectos y usuarios con sus respectivos tipos. Generamos el constructor de mismo y sus métodos getter y setter para obtener y editar los datos. Para el campo reporte\_avance dentro de proyectos, al ser un documento embebido en proyectos definimos que el tipo de dato sea ReportEntity, del cual modelamos aparte.

**REPOSITORIOS**

* **ProjectRepository – UserRepository - ReportRepository:** Creamos las interfaces para las tres entidades y definimos como parámetros de cada repositorio, las mismas entidades seguido del tipo de dato usado para el Id. Cada repositorio extiende de MongoRepository, que permite indicar a Spring Boot que se trata de un repositorio específico de MongoDB y hereda todos los métodos disponibles. Dentro, creamos los métodos que serán usados por los controladores para obtener, editar y eliminar información de la base de datos.

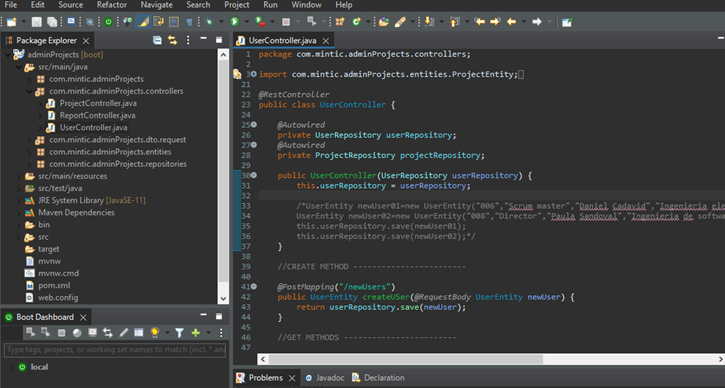
**CONTROLADORES**

* **ProjectController- UserController - ReportController:** Importamos los métodos extendidos de los repositorios usando la anotación @Autowired para usarlos en los métodos HTTP. Usamos la anotación @GetMapping, @PostMapping, @PutMapping, y definimos dentro de los mismos los endopint que son las urls que responden a las peticiones. Definimos el método que permitiera manejas las solicitudes HTTP. Usamos la anotación @PathVariable para tomar las variables pasadas en la URL y @RequestBody que nos permite recuperar el cuerpo de la petición.

**HISTORIAS DE USUARIOS**

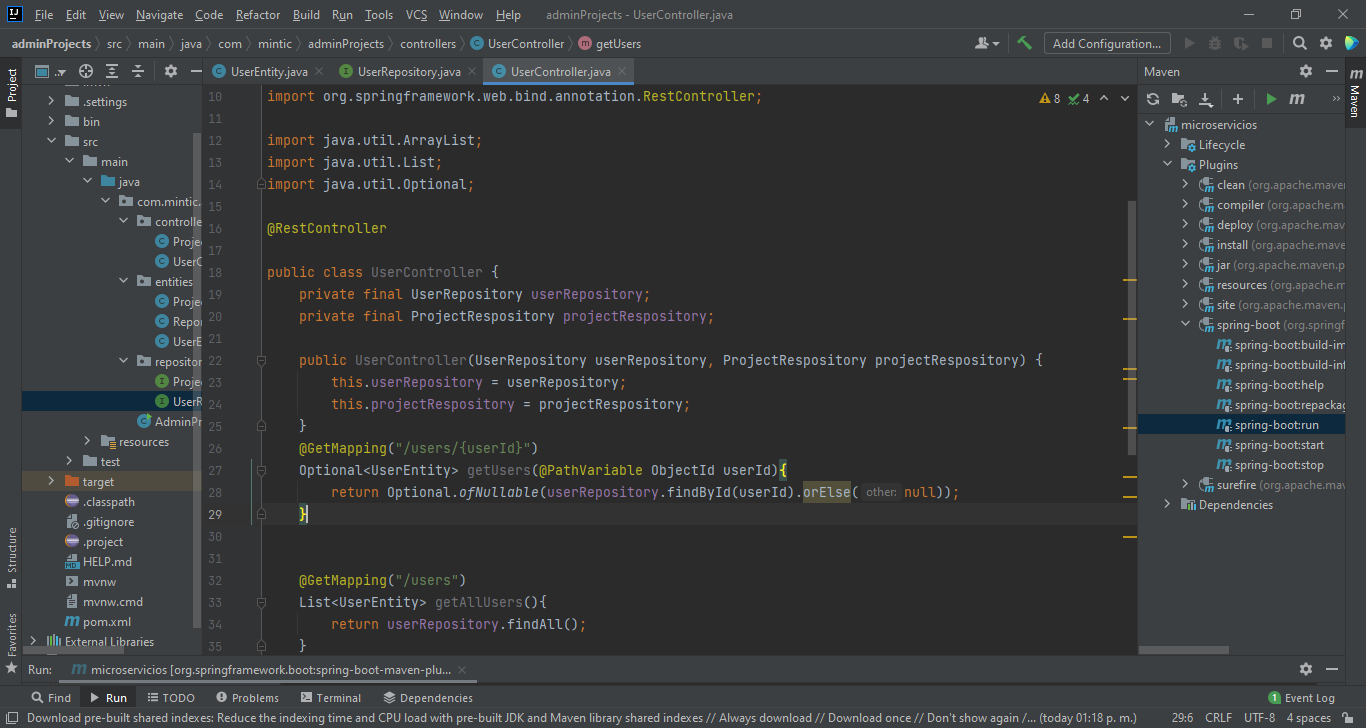
* **Crear nuevos participantes**

Para crear nuevos participantes usamos la ruta “/newUsers” y usamos el método save() extendido en el UserRepository y pasamos el parámetro que contiene el cuerpo de la petición:

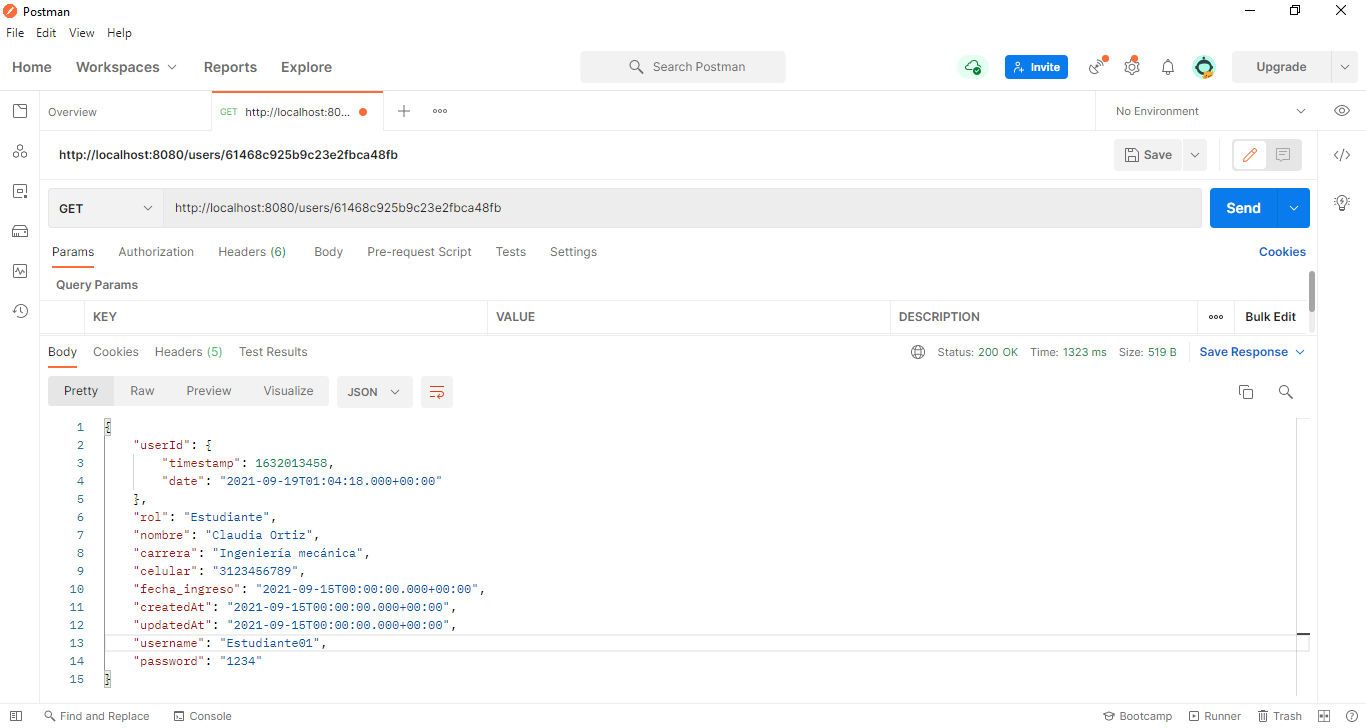
****

* **Consultando un usuario en particular**

Para consultar un usuario en particular, se usó la ruta "/users/{userId}" como se muestra a continuación:



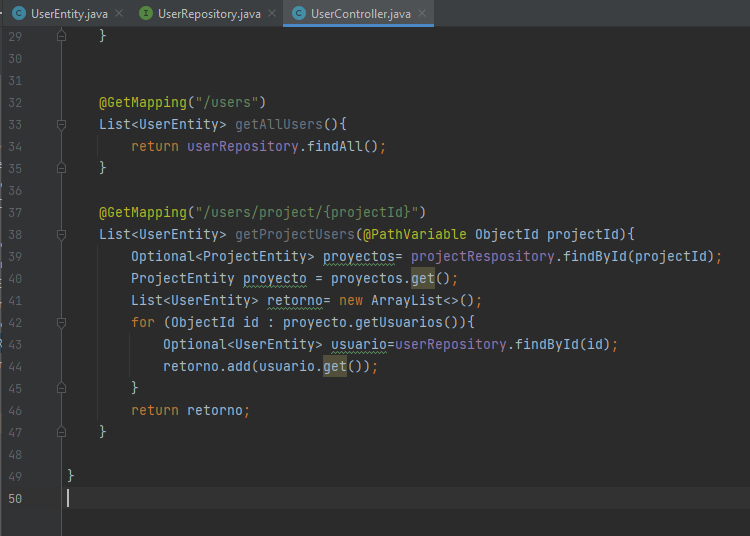
Con esta utilizamos el id proporcionado por la ruta para buscarlo en la base de datos, usando Postman podemos observar el resultado:



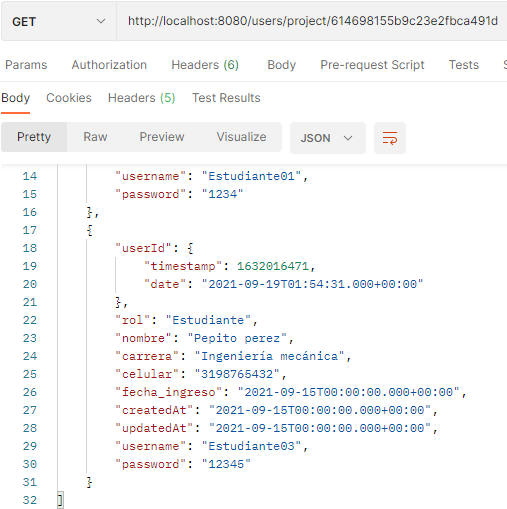
Con lo cual podemos asegurar que el proceso de buscar un usuario en particular se implementó con éxito.

* **Consultando los usuarios de un proyecto en especifico**

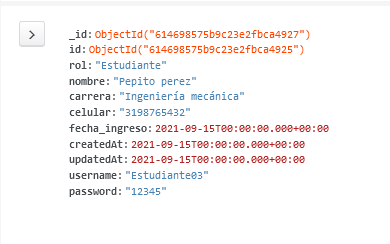
Para consultar un usuario en específico se usó la ruta "/users/project/{projectId}" como se muestra en la siguiente imagen:



Esta función trae todos los id de los usuarios asignados a un proyecto, busca dichos usuarios en la base de datos y los retorna. Usando Postman podemos observar el resultado:



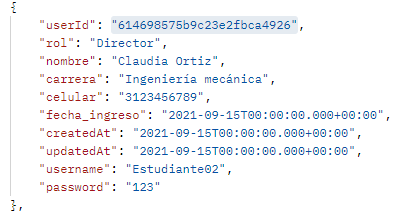
Los cuales coinciden con los datos almacenados en la base de datos



* **Asignación de participantes a los proyectos con sus respectivos roles**

1. Asignar un líder al proyecto:

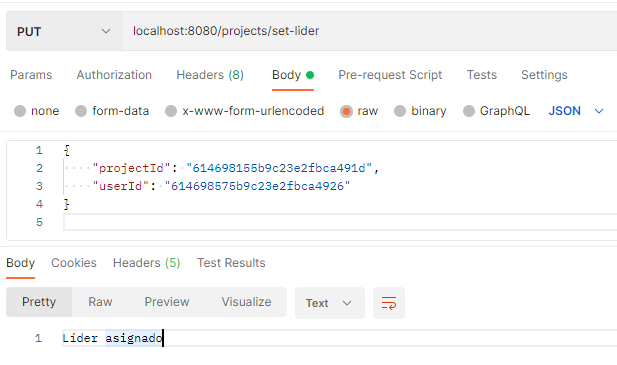
Líder que será asignado



Proyecto al que será asignado el líder



Petición HTTP para asignar el líder. Se envía un objeto con el id del proyecto y el del líder.

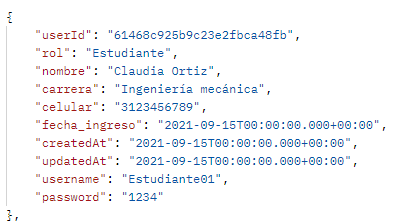


Estado del proyecto después de la asignación



1. Asignar un estudiante al proyecto

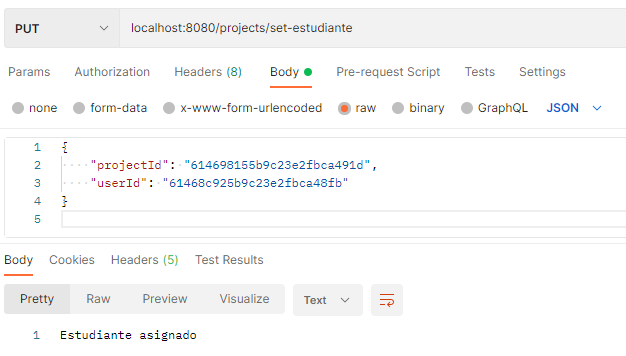
Estudiante que será asignado al proyecto



Proyecto al que será asignado el estudiante



Proceso de asignación del estudiante. Se efectúa similar a la asignación del líder.

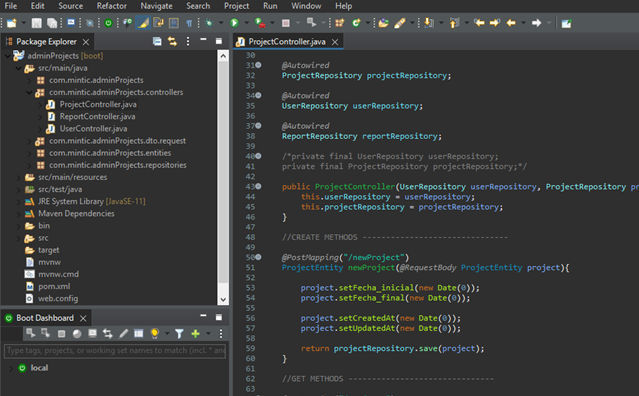


Estado del proyecto después de la asignación



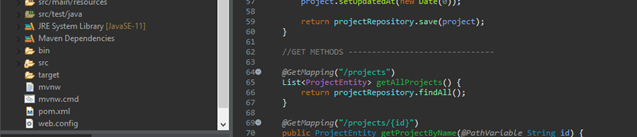
* **Crear nuevos proyectos**

Para crear nuevos proyectos usamos la ruta “/newProject” y usamos el método save() extendido en el ProjectRepository y pasamos el parámetro que contiene el cuerpo de la petición gracias a @RequestBody. Utilizamos los métodos set de fecha\_inicial, de createdAt y UpdatedAt para establecer una fecha actual pasando new Date().



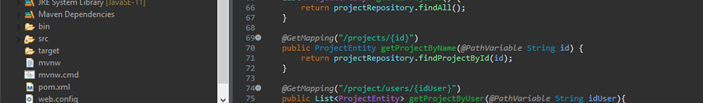
* **Obtener todos los proyectos**

Para obtener todos los proyectos usamos la ruta “/projects” y creamos un método que retorne una lista con la estructura definida en ProjectEntity, usando el método findAll().

****

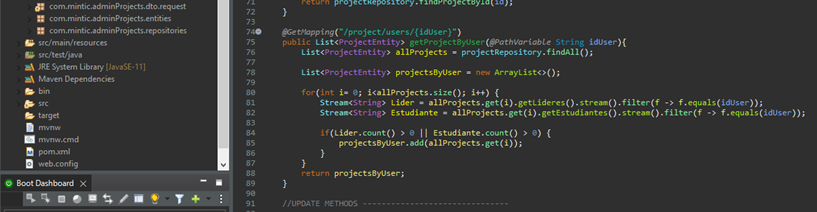
* **Obtener un proyecto específico por su id**

Para obtener un proyecto en particular usamos la ruta “/projects/{id}” y usamos la anotación @PathVariable para obtener el id pasado en la url.

****

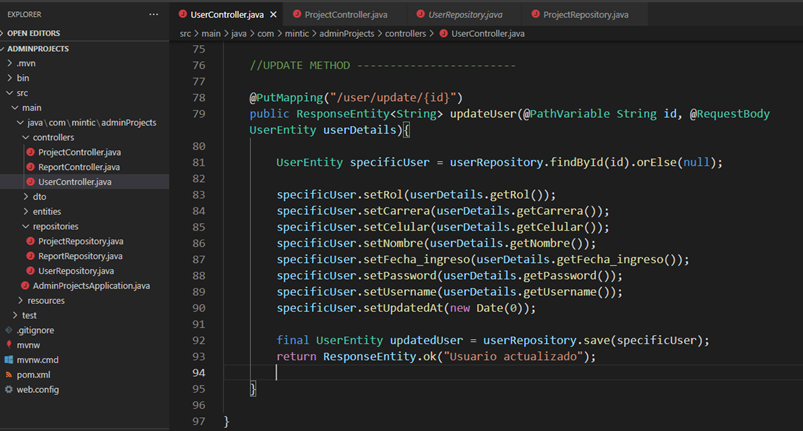
* **Obtener un proyecto por el id del usuario**

Para obtener el proyecto a partir de un usuario usamos la ruta “/Project/users/{idUser}”, dentro de una variable guardamos la lista de los proyectos, iteramos sobre ellos para obtener los usuarios (líderes y estudiantes) y verificamos que el idUser pasado en la url se encontrara las listas de líderes o estudiantes, al ser verdadero retornamos dicho proyecto.

****

* **Actualizando usuario por su id**

Para la actualización usamos la ruta “/user/update/{id} y usamos los métodos setter definidos en UserEntity.

****

* **Creando reporte de avance del proyecto**

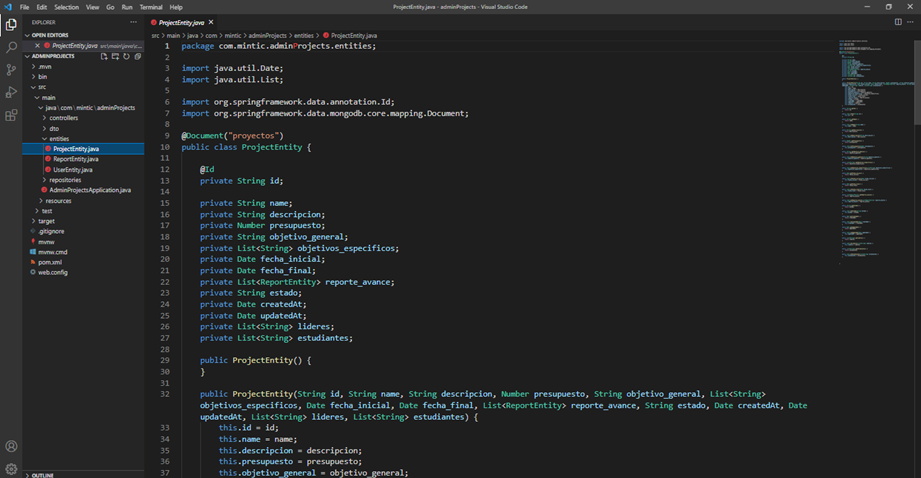
Para crear un reporte de avance se usó la ruta "/projects/reporte-avance/{projectId}" como se muestra en la siguiente imagen:

Texto

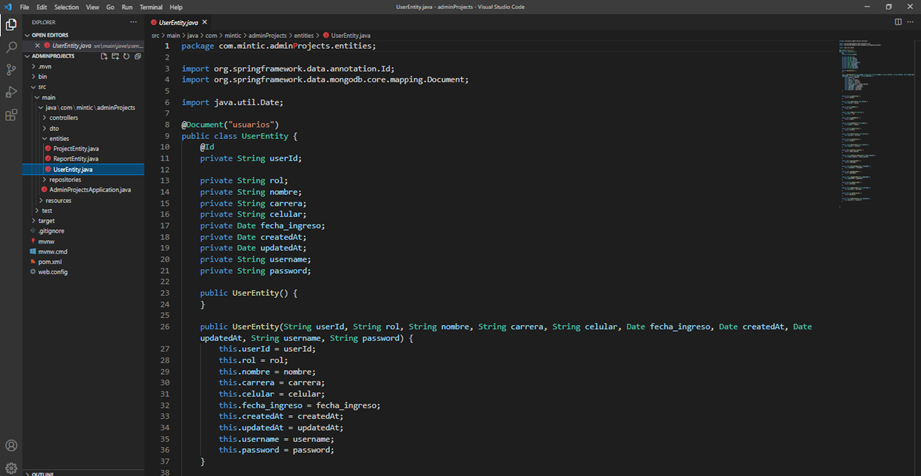
Descripción generada automáticamente

**ENTIDADES:**

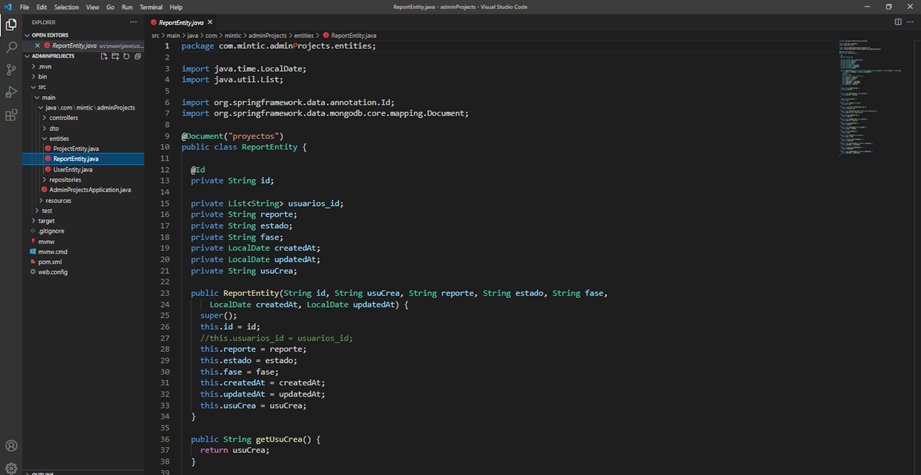
**ProjectEntity:**

****

**UserEntity:**

****

**ReportEntity:**

****