Sprint 4: Microservicios

Grupo 4

Integrantes:
David Carvalho
Andrea Giraldo
Stivel Pinilla
Sebastián Fuentes
Oscar Arbelaez
Andres Agudelo

Misión TIC
Desarrollo Web
Universidad de Antioquia
15 de octubre de 2021

Metodología

Para el desarrollo de este Sprint utilizamos *Intellij IDEA* debido a que *Spring Tool Suite* nos presenta errores que no logramos solucionar, lo que no ocurrió con *Intellij* que nos funcionó bastante bien y rápido. Los archivos realizados en este Sprint se encuentran en el repositorio https://github.com/DarcanoS/Proyecto_Ciclo_IV_Grupo_1_Equipo_4 en la carpeta *microservice_project_mintic*.

Mostraremos los modelos que realizamos en el proyecto. Se hizo un modelo para los proyectos, realizamos dos modelos diferentes para los usuario estudiantes y para los usuarios investigadores, y además se creó un modelo general de credenciales. Cada uno de los modelos cuenta con su constructor y sus Getters y Setters, los cuales no son necesarios mostrarlos.

```
public class EstudiantesModel {

    @Id
    private String estudianteId;

    private String nombreEstudiante;
    private String carrera;
    private String celular;
    private Date fechaIngreso;
    private String cedula;
    private ArrayList<String> proyectosId;
    private String credencialeId;
```

Figura 1. Modelo Estudiante

```
public class InvestigadoresModel {

    @Id
    private String investigadorId;

    private String credencialId;
    private ArrayList<HashMap<String,String>> proyectos;
```

Figura 2. Modelo Investigador

```java
public class ProyectosModel {

    @Id
    private String proyectoId;

    private String nombreProyecto;
    private String descripcion;
    private String estado;
    private String obGeneral;
    private String obEspecificos;
    private String presupuesto;
    private String investigadorId;
    private Date fechaInicio;
    private Date fechaFinal;
    private ArrayList<String> avances;
```

Figura 3. Modelo Proyecto

```java
public class CredencialesModel {

    @Id
    private String credencialId;

    private String usuario;
    private String clave;
```

Figura 4. Modelo Credenciales

Realizamos la conexión a la base de Mango Atlas.

Nuestro proyecto con Spring Boot consiste en 3 carpetas: *controllers*, *models* y *repositories*. Ya se han mostrado todos los *models* que utilizamos, los *repositories* no son necesarios mostrarlos pues no tiene mayor cosa. la distribución de los archivos es de esta manera:

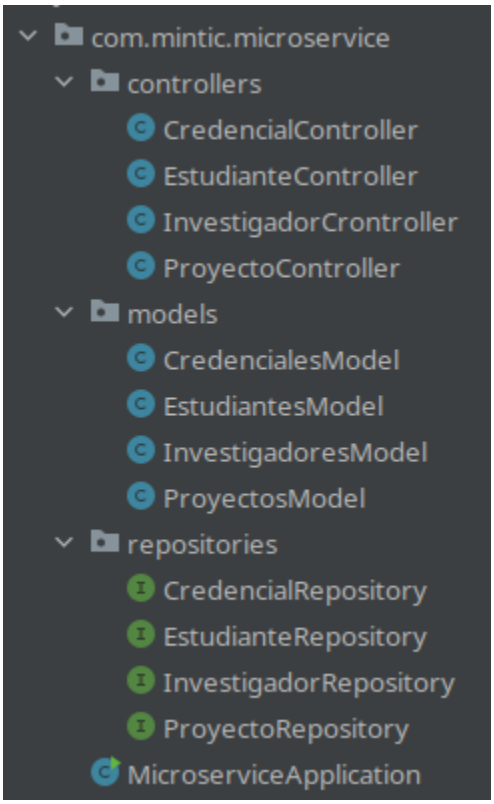

Figura 5. Distribucion del proyecto

Historia de usuario: HU_016

Se crea un endpoint para listar todos los proyectos.

La instanciación del endpoint para esta historia de usuario en nuestro archivo *ProyectoController* es:

```
@GetMapping("/proyecto/all")
List<ProyectosModel> getAllProyectos() { return proyectoRepository.findAll(); }
```

Figura 6. Instanciación del endpoint "/proyecto/all"

Con ello podemos listar todos los proyectos existentes. Un ejemplo de una consulta utilizando *Postman* a la lista de proyectos ya existentes en la base de datos:

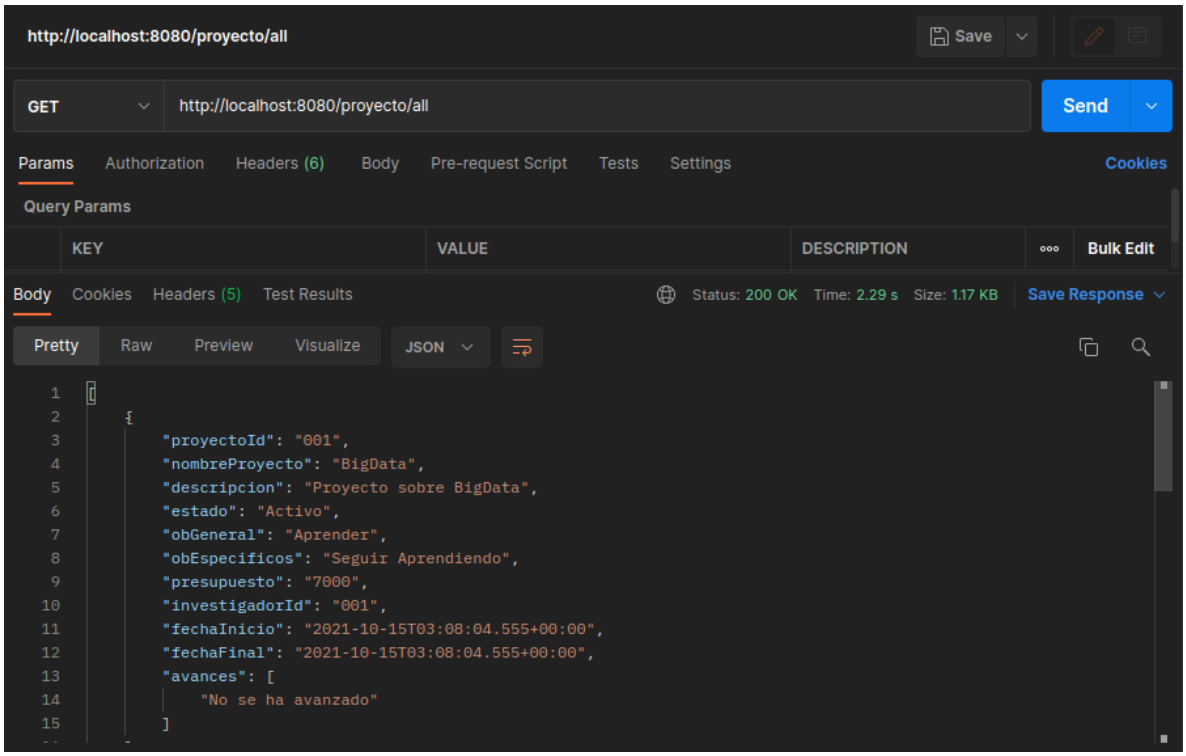

Figura 7.Consulta a "/proyecto/all"

Por cuestión de tamaño no se muestran todos los proyectos en la base de datos.

Historia de usuario: HU_017

Se crean endpoints para consultar a todos los participantes.

Debido a que tenemos dos modelos, listamos los modelos de forma independiente de acuerdo a su rol de investigador o estudiante.

Las instanciaciones para los endpoints que cumplen con la historia de usuario, ubicados en los archivos de *EstudianteCrontroller* e *InvestigadorController* son:

```
//Retorna todos los estudiantes en la DB
@GetMapping("/estudiante/all")
List<EstudiantesModel> getAllEstudiantes() { return estudianteRepository.findAll(); }
```

Figura 8. Instanciación del endpoint "/estudiante/all"

```
@GetMapping("/investigador/all")
List<InvestigadoresModel> getAllInvestigadores() { return investigadorRepository.findAll(); }
```

Figura 9. Instanciación del endpoint "/investigador/all"

Con ello podemos listar todos los estudiantes e investigadores existentes. Un ejemplo de una consulta utilizando *Postman* a la lista de estudiantes e investigadores ya existentes en la base de datos respectivamente:

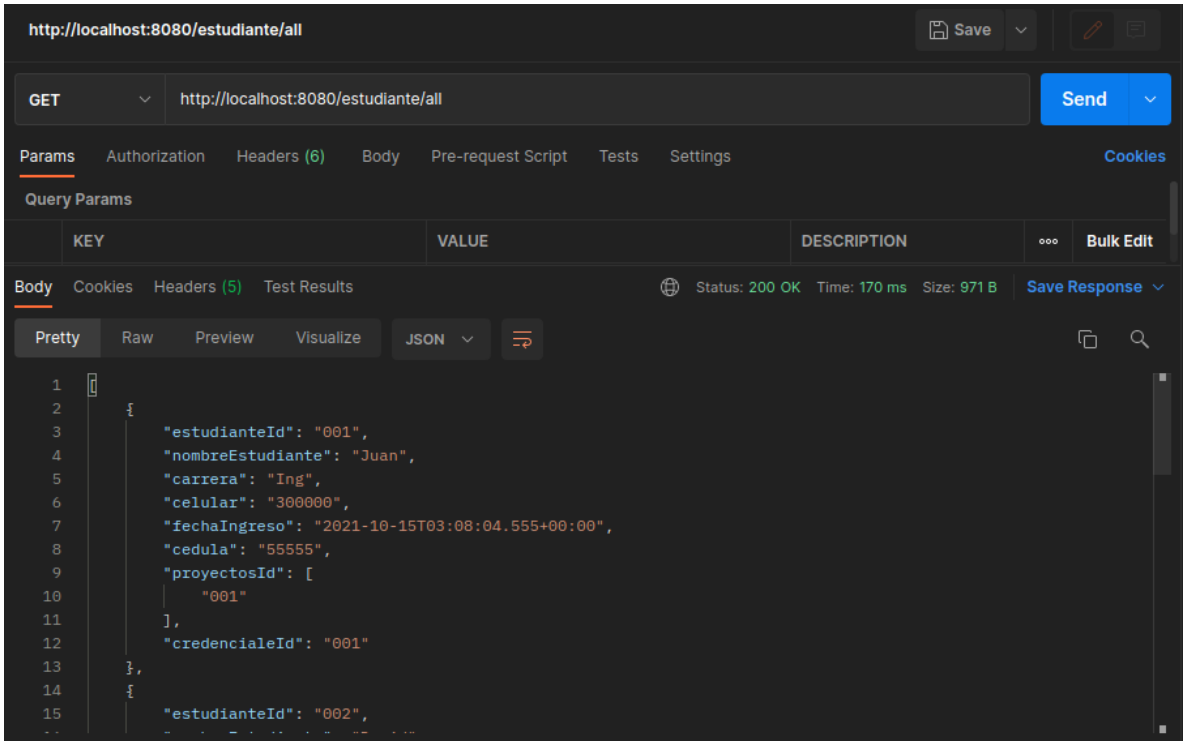

Figura 10.Consulta a "/estudiante/all"

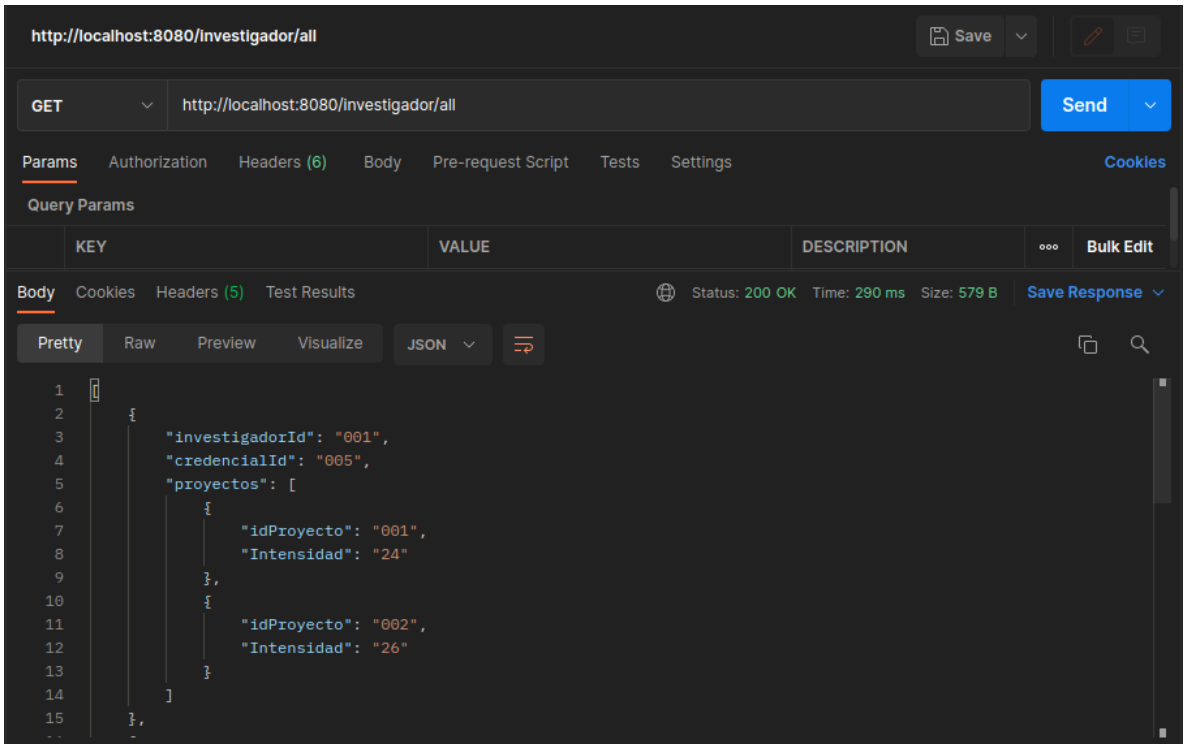

Figura 11.Consulta a "/investigador/all"

Historia de usuario: HU_018

Se crea un endpoint para consultar un proyecto en particular y para editarlo.

Las instanciaciones los endpoint que cumplen con la historia de usuario, ubicados en el archivo *ProyectoController* son:

```
@GetMapping("/proyecto/{proyectoId}")
ProyectosModel findProyecto(@PathVariable String proyectoId){
    return proyectoRepository.findById(proyectoId).get();
}
```

Figura 12. Instanciación del endpoint "proyecto/{proyectoId}"

Con ello podemos listar un proyecto en específico. Un ejemplo de una consulta utilizando *Postman*:

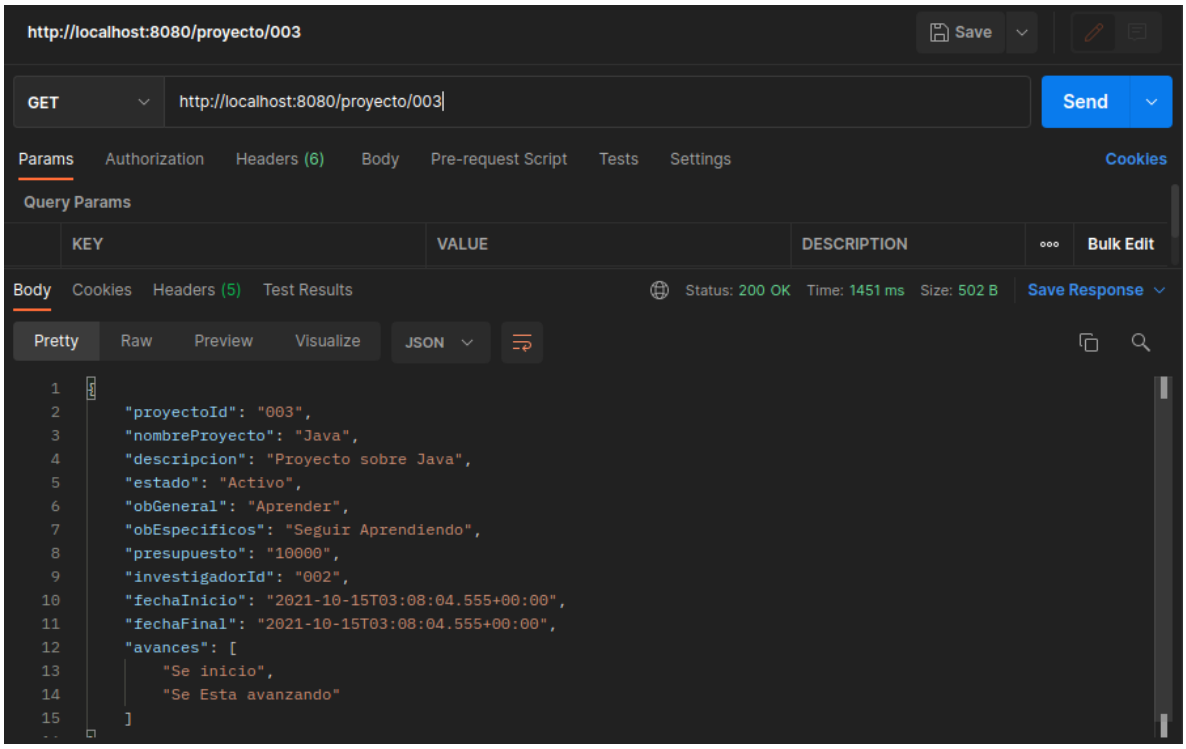

Figura 13.Consulta a "/proyecto/003"

Recordemos que *proyectoId* es el identificador único que nos permite encontrar el proyecto en la base de datos.

```
@PostMapping("/proyecto/add")
ProyectosModel addProyecto(@RequestBody ProyectosModel proyecto){

    return proyectoRepository.save(proyecto);

}
```

Figura 14. Instanciación del endpoint "proyecto/add"

Con ello podemos agregar o editar cualquier proyecto, utilizando *Postman* realizaremos el cambio de las fechas de inicio y fin para el proyecto que anteriormente consultamos:

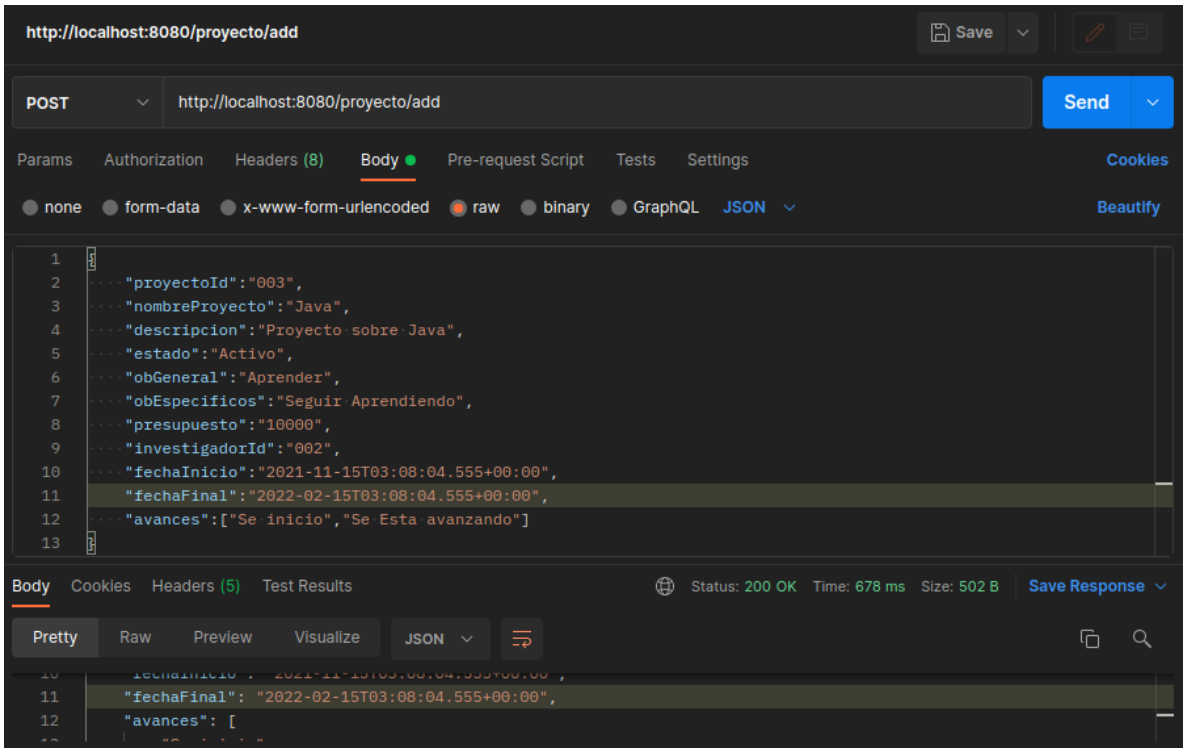

Figura 15.Consulta a "/proyecto/add"

Con ello nos devolvemos a verificar el cambio en la base de datos listando ese proyecto en específico con una consulta *GET*:

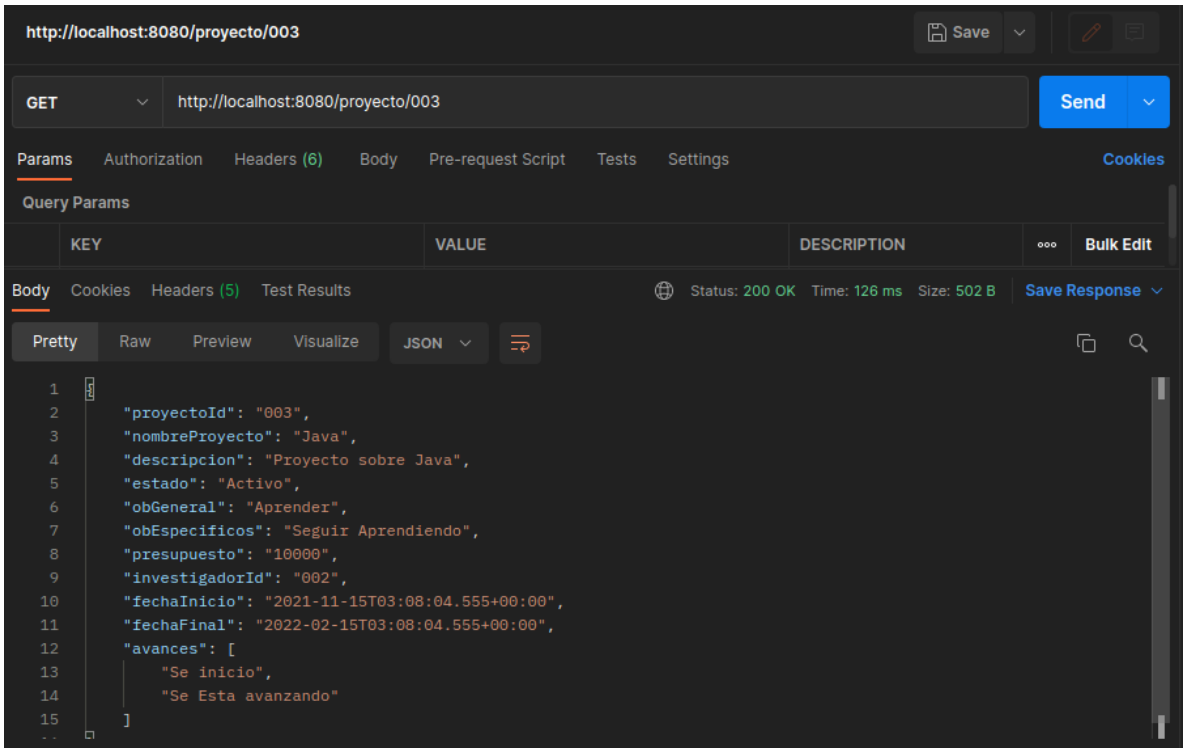

Figura 16.Consulta a "/proyecto/003"

Historia de usuario: HU_019

Se crean endpoints para cambiar los proyectos donde se involucran ciertos estudiantes e investigadores.

Para ellos realizamos estos cambios en cada uno de los usuarios, sean estudiantes o investigadores.

Las instanciaciones de los endpoints que cumplen con la historia de usuario, ubicados en los archivos *EstudianteController* y *InvestigadorController* son:

```
//Agrega o modifica un estudiante en la DB
@PostMapping("/estudiante/add")
EstudiantesModel addEstudiante(@RequestBody EstudiantesModel estudiante){
    estudiante.setFechaIngreso(new Date());
    return estudianteRepository.save(estudiante);
}
```

Figura 17. Instanciación del endpoint "estudiante/add"

Con ello podemos cambiar los proyectos a los que pertenece un estudiante en específico. Para ello cambiaremos los proyectos a los que pertenece el estudiante con id: 002. En primera instancia consultaremos este estudiante, realizaremos los cambios y volveremos a consultarlo:

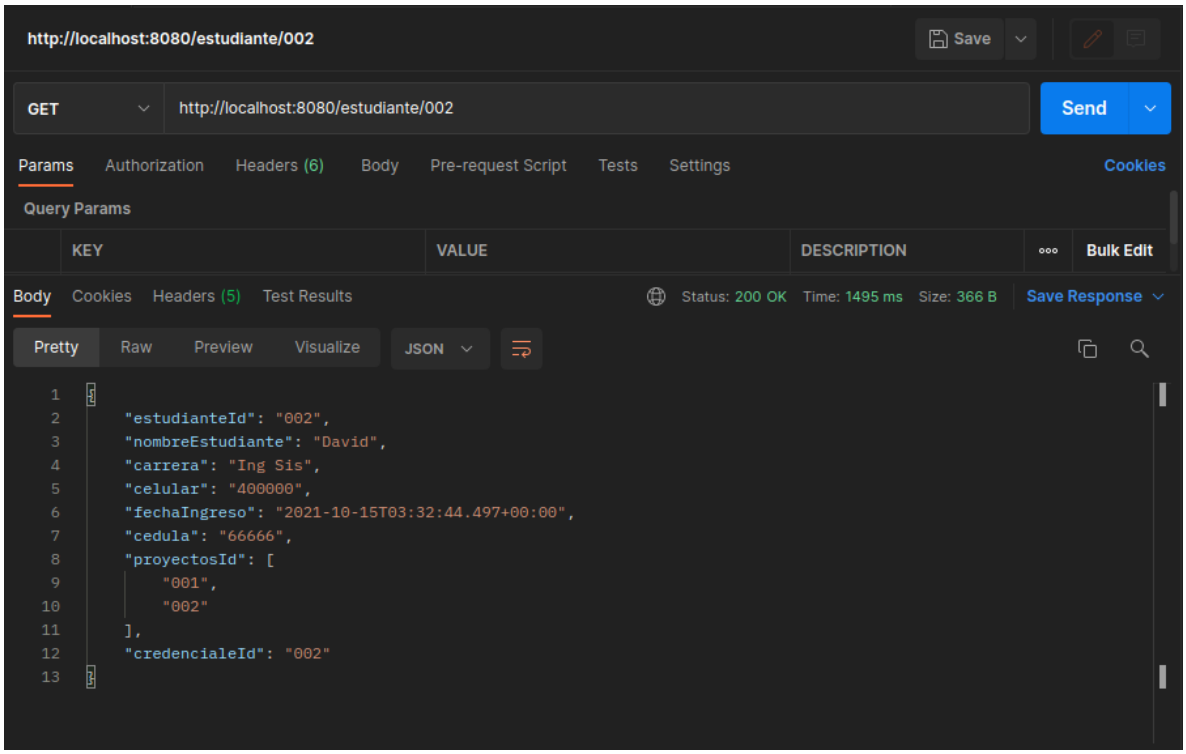

Figura 18.Consulta a "/estudiante/002"

Quitaremos el proyecto 001 de los proyectos del estudiante y agregaremos el 003:

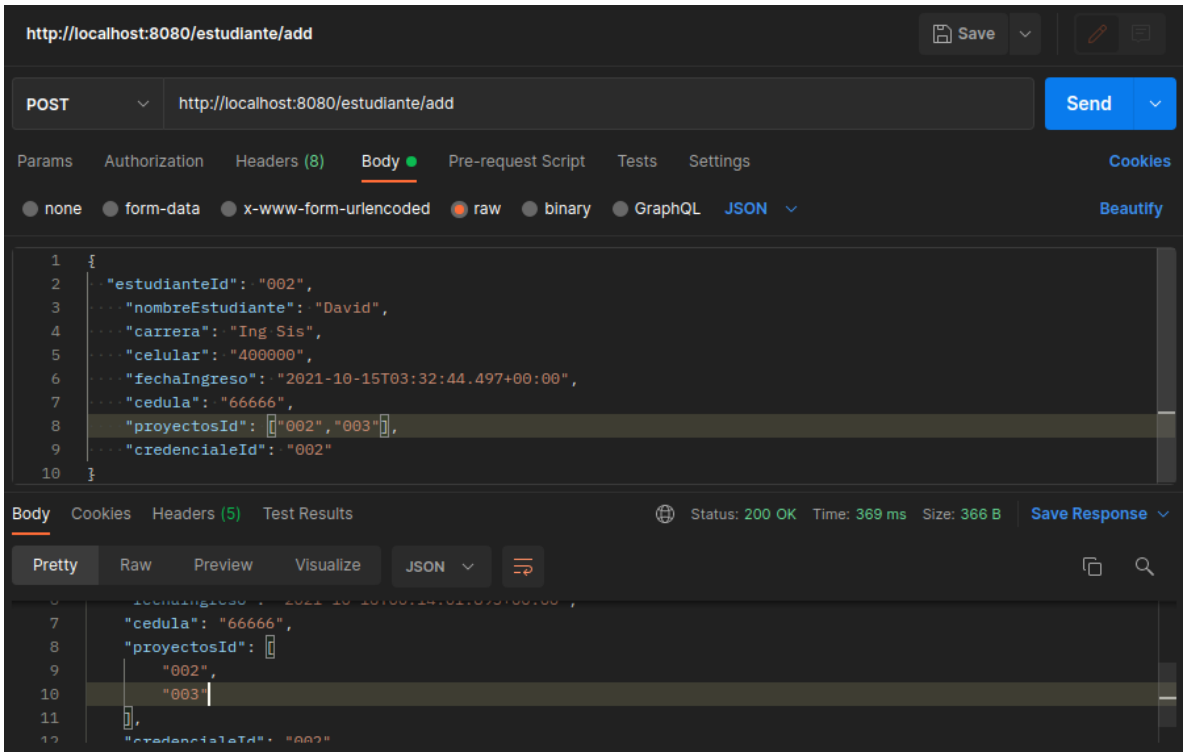

Figura 19.Consulta a "/estudiante/add"

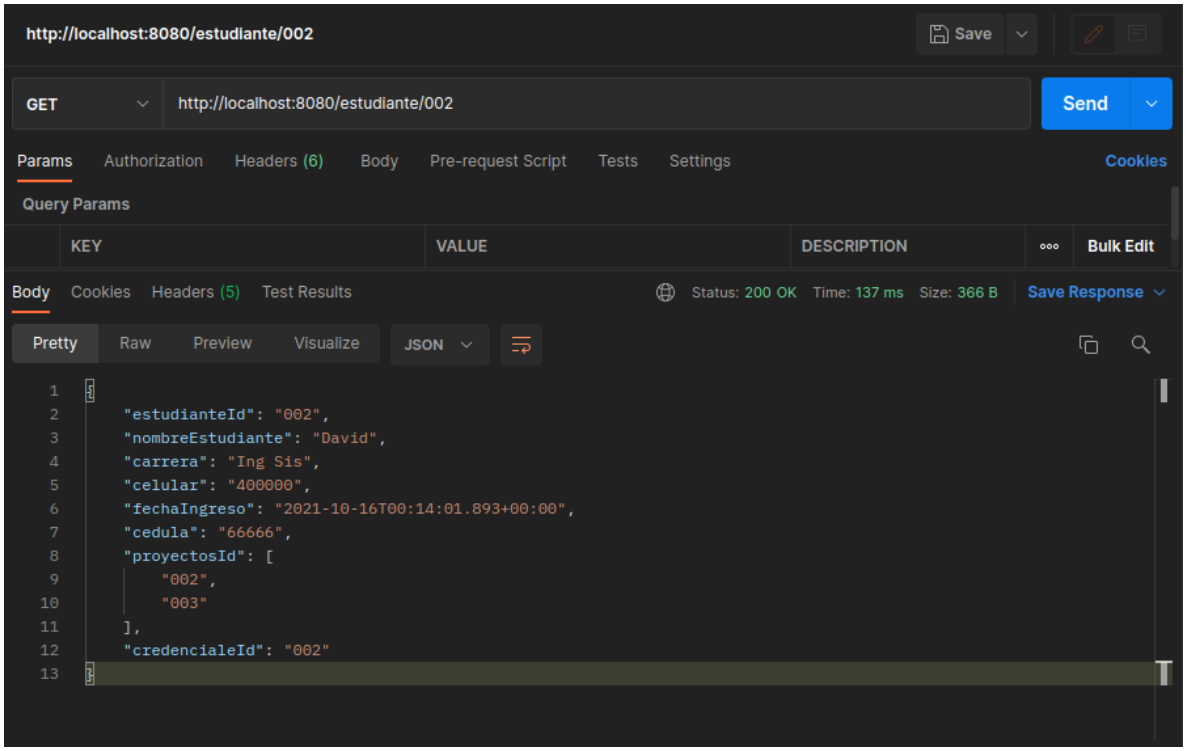

Figura 20.Consulta a "/estudiante/002"

La instanciación del endpoint para los investigadores es:

```
@PostMapping("/investigador/add")
InvestigadoresModel addInvestigador(@RequestBody InvestigadoresModel investigadoresModel){
    return investigadorRepository.save(investigadoresModel);
}
```

Figura 21. Instanciación del endpoint "investigador/add"

Historia de usuario: HU_020

Se crean endpoints para cambiar los datos de los estudiantes e investigadores.

Las instanciaciones de los endpoints que cumplen con la historia de usuario, ubicados en los archivos *EstudianteController* y *InvestigadorController* son:

```java
//Agrega o modifica un estudiante en la DB
@PostMapping("/estudiante/add")
EstudiantesModel addEstudiante(@RequestBody EstudiantesModel estudiante){
    estudiante.setFechaIngreso(new Date());
    return estudianteRepository.save(estudiante);
}
```

Figura 22. Instanciación del endpoint "estudiante/add"

```java
@PostMapping("/investigador/add")
InvestigadoresModel addInvestigador(@RequestBody InvestigadoresModel investigadoresModel){
    return investigadorRepository.save(investigadoresModel);
}
```

Figura 23. Instanciación del endpoint "investigador/add"

Como se demuestra en la historia de usuario anterior y en la Figura 19. Estos endpoints nos permiten realizar modificaciones a los estudiantes o investigadores. Teniendo en cuenta que el *id* de cada uno de ellos es el identificador que nos permite elegir a qué usuario realizamos las modificaciones, sea estudiante o investigador.

Historia de usuario: HU_021

Se crean endpoints para asignar proyectos a los estudiantes e investigadores.

De acuerdo con la historia de usuario anterior, podemos modificar la información de cada uno de los usuarios, sean estudiantes o investigadores, lo que igualmente permite agregar o eliminar a qué proyectos pertenecen.

Enlaces

Trello: https://trello.com/b/rxlH2t0x/mintic

GitHub: https://github.com/DarcanoS/Proyecto_Ciclo_IV_Grupo_1_Equipo_4