

Sprint 5: GraphQL, NodeJs y ReactJs y aplicación de la interfaz gráfica con React Js

Grupo 4

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

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

Metodología

Para realizar este Sprint utilizamos el mismo repositorio que llevamos en todo el transcurso del curso: https://github.com/DarcanoS/Proyecto_Ciclo_IV_Grupo_1_Equipo_4.

La ubicación de la implementación con *GraphQL* corresponde al proyecto inmediatamente presente:

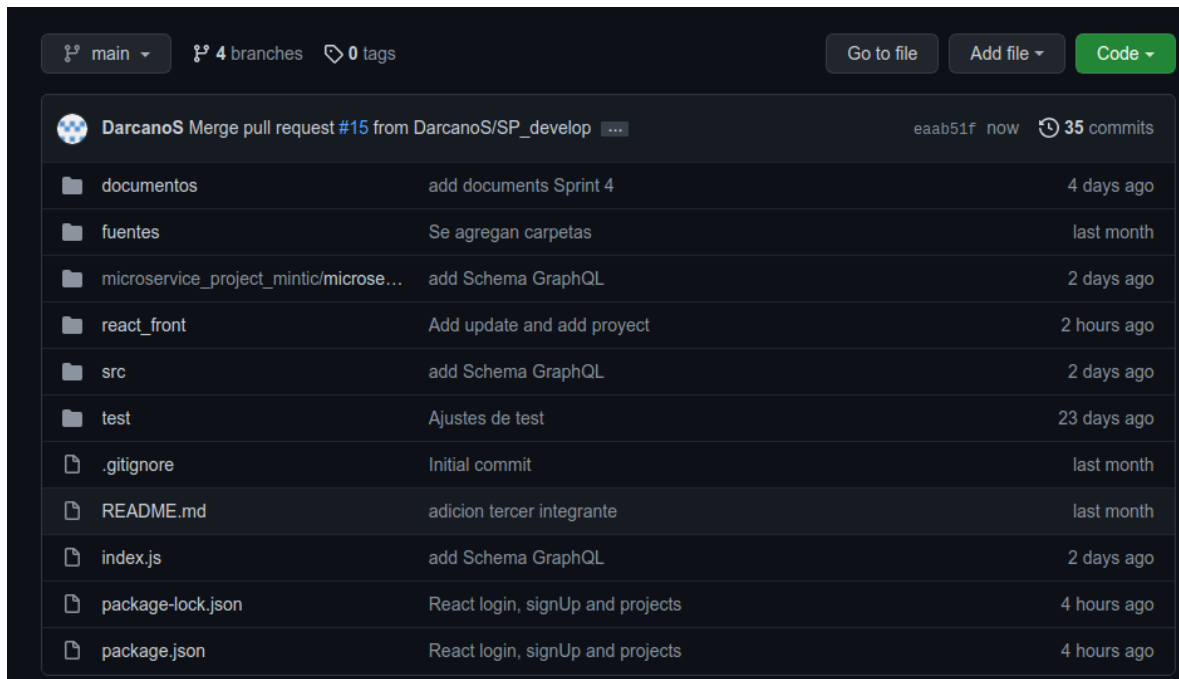


Figura 1. Instanciación del endpoint “/proyecto/all”

Como se puede observar, el archivo *index.js* corresponde a la raíz del proyecto implementando *GraphQL*. En la implementación del Front-End con React se encuentra en la carpeta *react_front*.

Cada uno se puede probar ejecutando con los comandos ***node index.js*** y ***npm start*** respectivamente. Hay que recordar que la base de datos debe tener permisos, por ende se aconseja cambiar las rutas de las bases de datos por una nueva.

GraphQL

Historia de usuario: HU_022

Se crea un Schema de *GraphQL* para consultar los proyectos.

Realizamos la creación del schema proyectos de la siguiente manera:

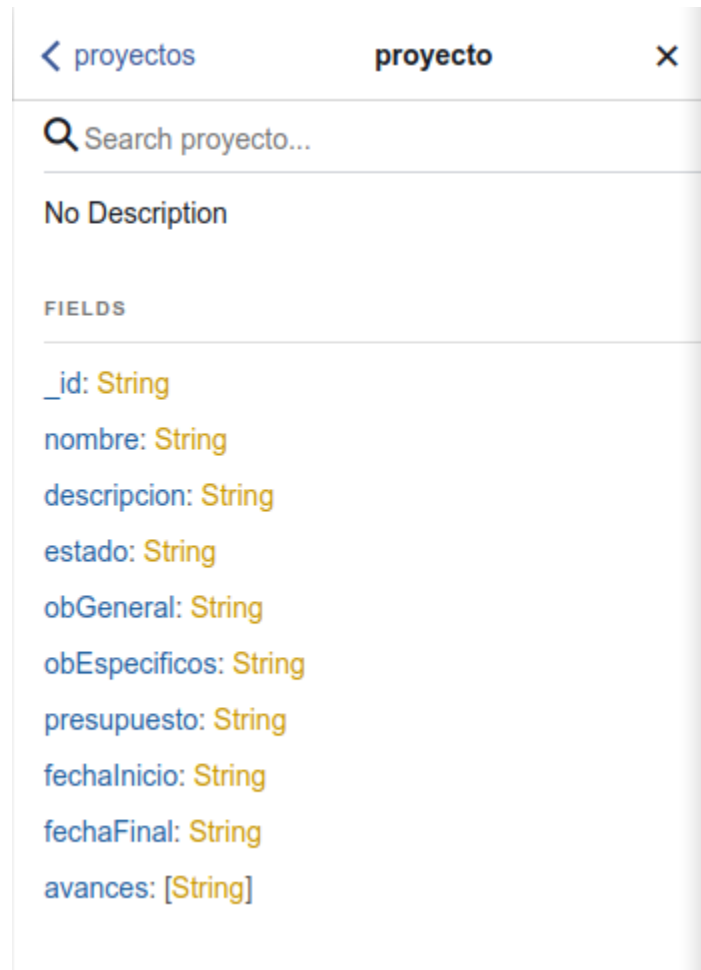


Figura 2.Schema proyecto

Con el fin de realizar la consulta para todos los proyectos :

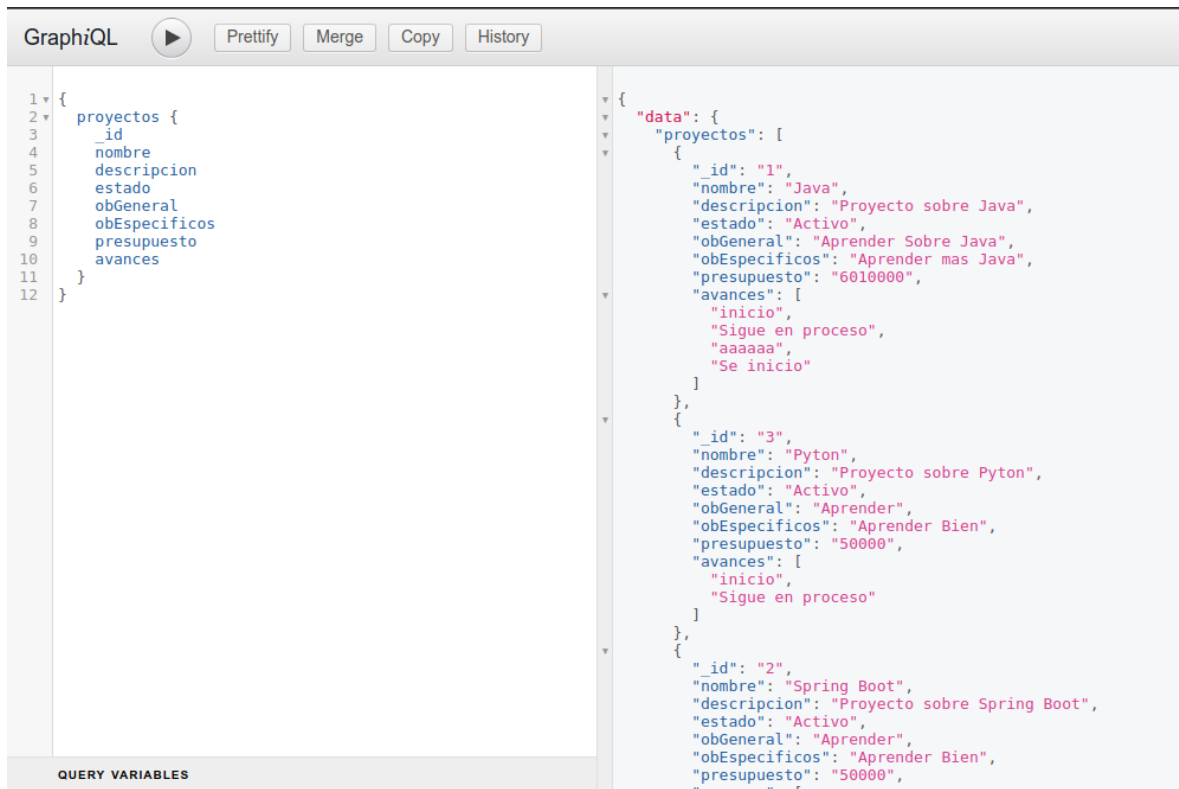


Figura 3. Consulta de proyectos con GraphQL

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

Historia de usuario: HU_023

Se crea un Schema de *GraphQL* para la consulta de un proyecto en específico.

Realizamos la creación del schema para resolver esta historia de usuario se puede ver de este manera::

proyecto(id: ID): proyecto

Figura 4. Instanciación del schema para obtener un proyecto en particular

Como se puede apreciar en la figura, el query recibe un parametro tipo *String* para relacionarlo con el *id* del proyecto:

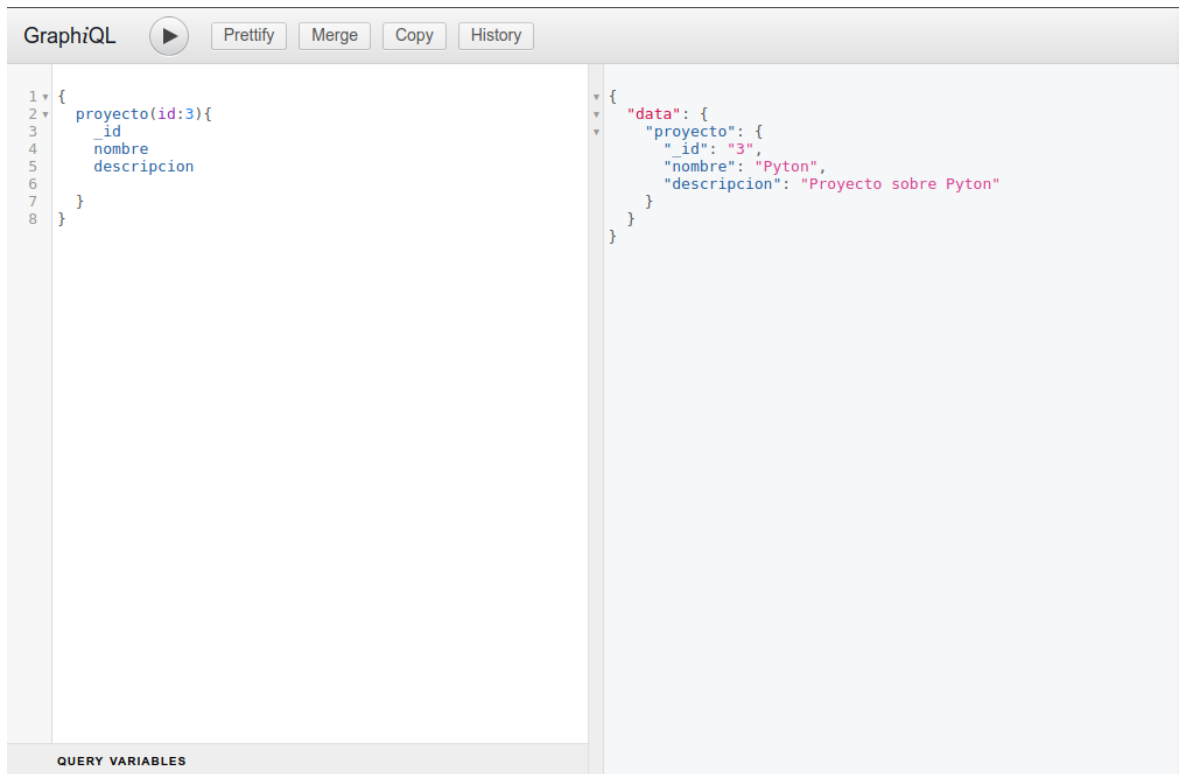


Figura 5. Consulta de un proyecto con GraphQL

Historia de usuario: HU_024

Se crea un Schema de *GraphQL* para consultar a los participantes. Recordando que con nuestra abstracción del problema se crearon dos colecciones de estudiantes e investigadores.

Realizamos la creación del schema los integrantes de la siguiente manera:

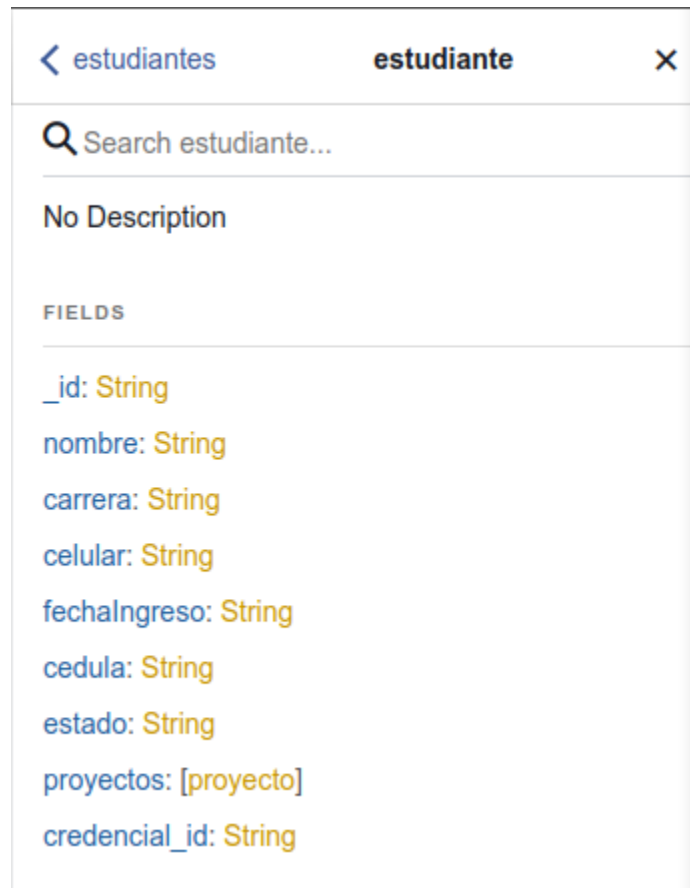


Figura 6. Schema de estudiante

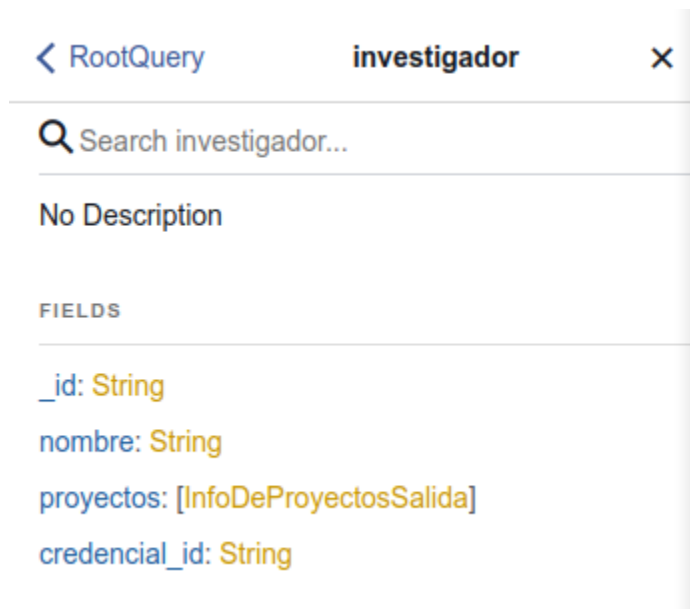
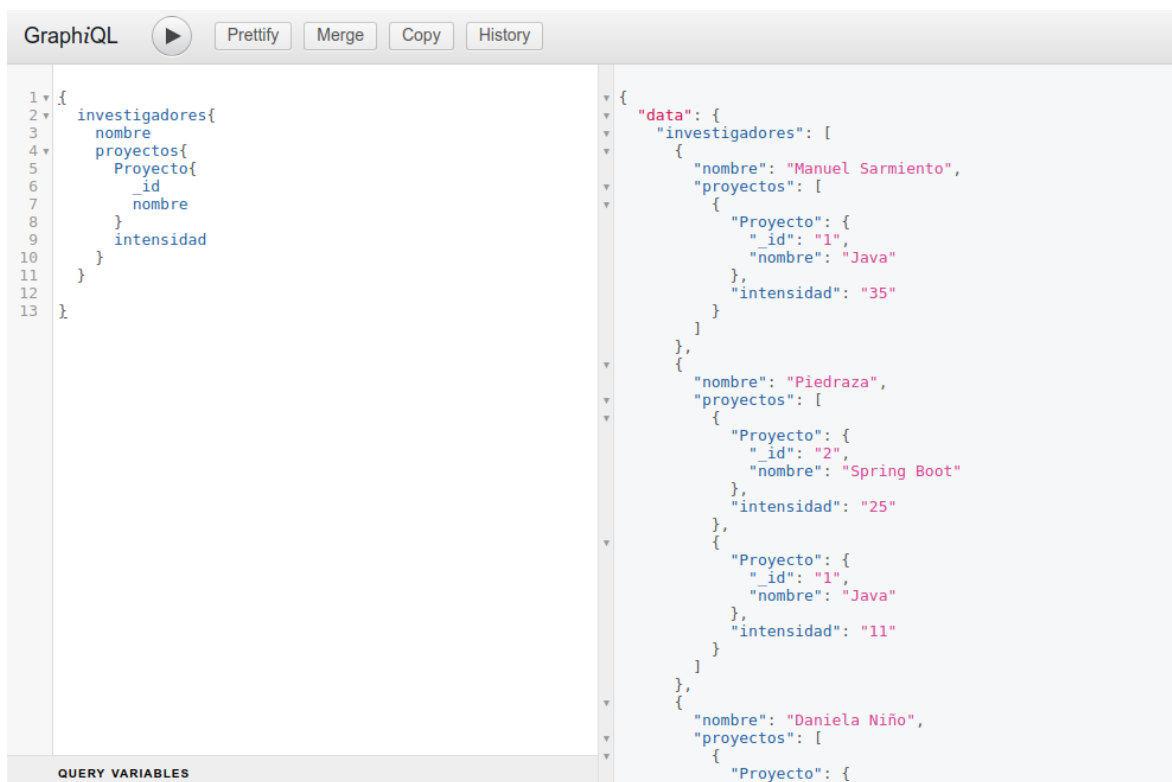


Figura 7. Schema Investigador

Con el fin de realizar la consulta para todos los participantes :



The image shows a GraphQL IDE interface with a query on the left and its JSON response on the right. The query is a nested selection set for 'investigadores', 'nombre', 'proyectos', and 'intensidad'. The response is a JSON object with a 'data' field containing an array of investigator objects. Each object has a 'nombre' and a 'proyectos' array. The projects array contains objects with '_id' and 'nombre' fields. The 'intensidad' field is a string value.

```
1 {
2   investigadores{
3     nombre
4     proyectos{
5       Proyecto{
6         _id
7         nombre
8       }
9     }
10   }
11 }
12 }
13 }
```

```
{
  "data": {
    "investigadores": [
      {
        "nombre": "Manuel Sarmiento",
        "proyectos": [
          {
            "Proyecto": {
              "_id": "1",
              "nombre": "Java"
            },
            "intensidad": "35"
          }
        ]
      },
      {
        "nombre": "Piedraza",
        "proyectos": [
          {
            "Proyecto": {
              "_id": "2",
              "nombre": "Spring Boot"
            },
            "intensidad": "25"
          },
          {
            "Proyecto": {
              "_id": "1",
              "nombre": "Java"
            },
            "intensidad": "11"
          }
        ]
      },
      {
        "nombre": "Daniela Niño",
        "proyectos": [
          {
            "Proyecto": {
              "_id": "1",
              "nombre": "Java"
            },
            "intensidad": "11"
          }
        ]
      }
    ]
  }
}
```

QUERY VARIABLES

Figura 8.Consulta de investigadores con GraphQL

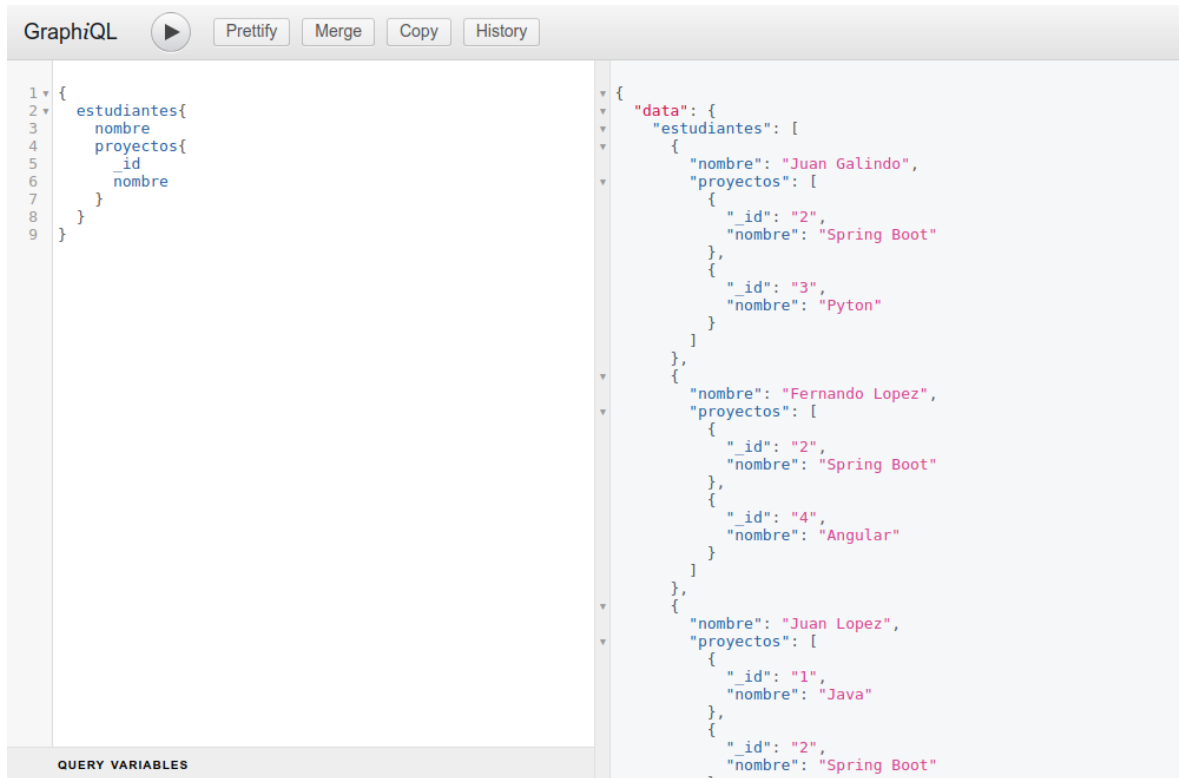


Figura 9. Consulta de estudiantes con GraphQL

Por cuestión de tamaño no se muestran todos los integrantes que se encuentran en la base de datos.

Historia de usuario: HU_025

Se crea un Schema de *GraphQL* para la consulta de un participante en específico.

Realizamos la creación del schema para resolver esta historia de usuario se puede ver de este manera:

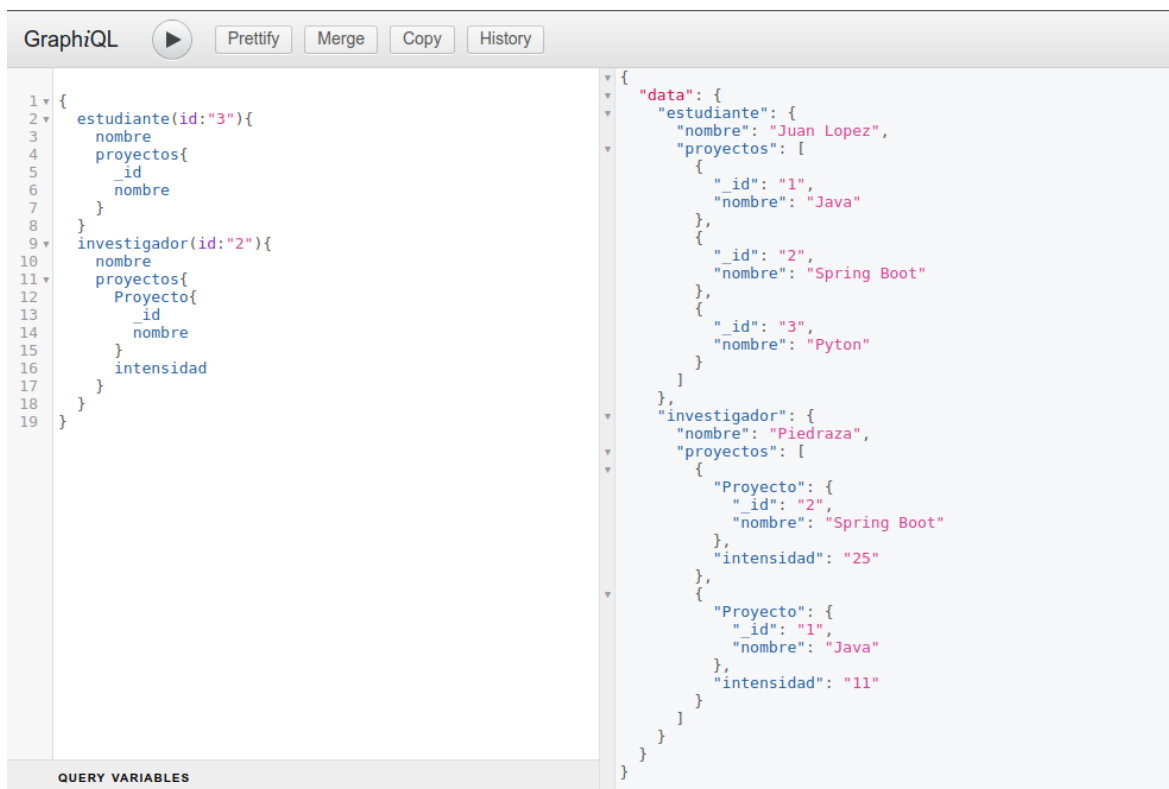
estudiante(id: ID): estudiante

Figura 10. Instanciación del schema para obtener un estudiante en particular

investigador(id: ID): investigador

Figura 11. Instanciación del schema para obtener un investigador en particular

Como se puede apreciar en la figura, el query recibe un parametro tipo *String* para relacionarlo con el *id* del participante, como se muestra la siguiente figura:



The screenshot shows the GraphQL IDE interface. On the left, the query is written in a monospaced font with line numbers 1 through 19. It queries for a student with id '3' and an investigator with id '2', including their names and associated projects. On the right, the JSON response is displayed, showing the data for the student (Juan Lopez) and investigator (Piedraza) along with their respective projects (Java, Spring Boot, Python) and the investigator's intensity (25). The interface includes a 'QUERY VARIABLES' section at the bottom left and buttons for 'Prettify', 'Merge', 'Copy', and 'History' at the top.

```
1 {
2   estudiante(id:"3"){
3     nombre
4     proyectos{
5       _id
6       nombre
7     }
8   }
9   investigador(id:"2"){
10    nombre
11    proyectos{
12      Proyecto{
13        _id
14        nombre
15      }
16      intensidad
17    }
18  }
19 }
```

```
{
  "data": {
    "estudiante": {
      "nombre": "Juan Lopez",
      "proyectos": [
        {
          "_id": "1",
          "nombre": "Java"
        },
        {
          "_id": "2",
          "nombre": "Spring Boot"
        },
        {
          "_id": "3",
          "nombre": "Python"
        }
      ]
    },
    "investigador": {
      "nombre": "Piedraza",
      "proyectos": [
        {
          "Proyecto": {
            "_id": "2",
            "nombre": "Spring Boot"
          },
          "intensidad": "25"
        },
        {
          "Proyecto": {
            "_id": "1",
            "nombre": "Java"
          },
          "intensidad": "11"
        }
      ]
    }
  }
}
```

QUERY VARIABLES

Figura 12.Consulta de un estudiante y un investigador con GraphQL

Historia de usuario: HU_026

Se crea un Schema de *GraphQL* para la creación de un proyecto.

Realizamos la creación del schema para resolver esta historia de usuario se puede ver de este manera:

```

addProyecto(
  _id: String!
  nombre: String!
  descripcion: String!
  estado: String!
  obGeneral: String!
  obEspecificos: String!
  presupuesto: String!
  fechaInicio: String!
  fechaFinal: String!
  avances: [String]!
): proyecto

```

Figura 13. Instanciación del schema para agregar un proyecto

Como se puede apreciar en la figura, todos los parámetros que recibe son obligatorios, y se realiza una mutación de este tipo en la siguiente figura:

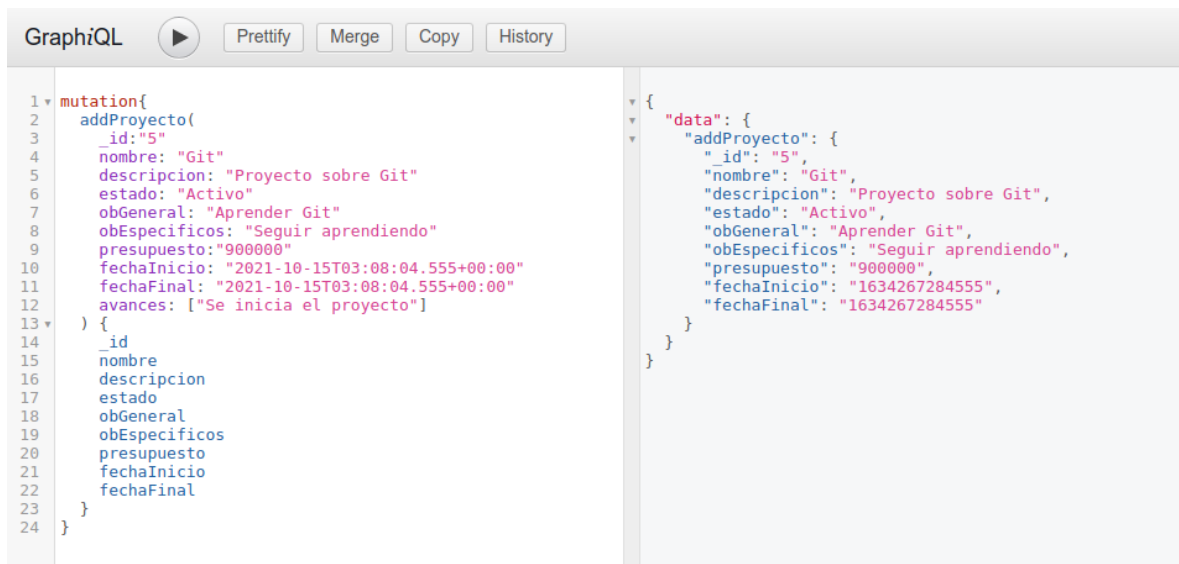


Figura 14. Mutación para crear un proyecto con GraphQL

Historia de usuario: HU_027

Se crea un Schema de *GraphQL* para la actualización de un proyecto.

Realizamos la creación del schema para resolver esta historia de usuario se puede ver de este manera:

```

updateProyecto(
  _id: String!
  nombre: String
  descripcion: String
  estado: String
  obGeneral: String
  obEspecificos: String
  presupuesto: String
  fechaInicio: String
  fechaFinal: String
  avances: [String]
): proyecto

```

Figura 15. Instanciación del schema para actualizar un proyecto

Como se puede apreciar en la figura, el unico parametro obligatoria es el id que permite identificar a qué proyecto se realizar la actualización y se colocan solo los valores que se van a modificar:

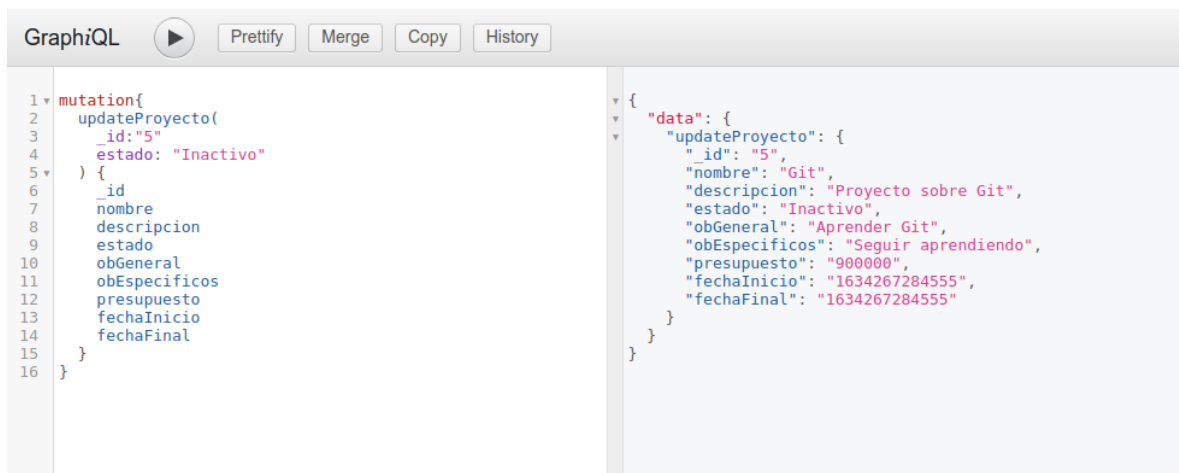


Figura 16. Mutación para actualizar un proyecto con GraphQL

Historia de usuario: HU_028

Se crea un Schema de *GraphQL* para la eliminacion de un proyecto.

Realizamos la creación del schema para resolver esta historia de usuario se puede ver de este manera:

```

removeProyecto(_id: String!): proyecto

```

Figura 17. Instanciación del schema para eliminar un proyecto

Como se puede apreciar en la figura, el parámetro id indica que proyecto va a ser eliminado:

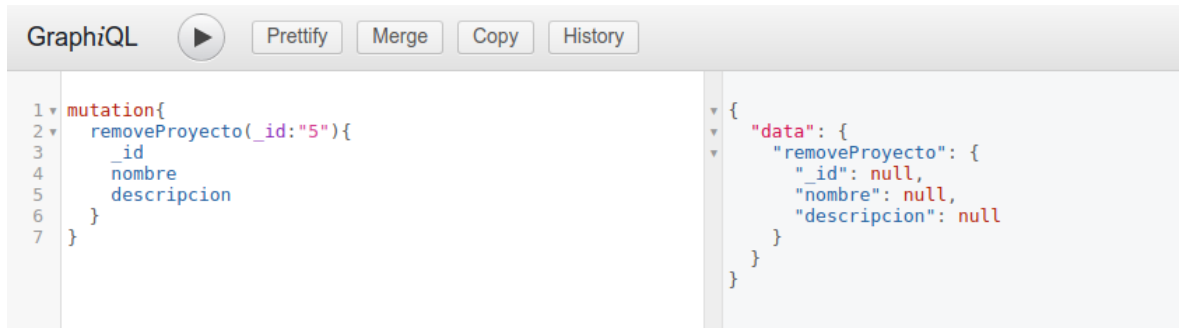


Figura 18. Mutación para eliminar un proyecto con GraphQL

Historia de usuario: HU_029

Se crea un Schema de *GraphQL* para la creación de un integrante.

Realizamos la creación del schema para resolver esta historia de usuario se puede ver de este manera:

```
addEstudiante(
  _id: String!
  nombre: String!
  carrera: String!
  celular: String!
  fechaIngreso: String!
  cedula: String!
  estado: String!
  proyectos: [String]!
  credencial_id: String!
): estudiante
```

Figura 19. Instanciación del schema para agregar un estudiante

```
addInvestigador(
  _id: String!
  nombre: String!
  proyectos: [InfoDeProyectos]!
  credencial_id: String!
): investigador
```

Figura 20. Instanciación del schema para agregar un investigador

Como se puede apreciar en la figura, todos los parámetros que recibe son obligatorios, y se realiza una mutación de este tipo en la siguiente figura:

```

1 mutation{
2   addEstudiante(
3     _id:"4"
4     nombre:"Mateo Suarez"
5     carrera: "Gastronomia"
6     celular: "001111111"
7     fechaIngreso: "2021-10-15T03:08:04.555+00:00"
8     cedula: "55555555"
9     estado: "Matriculado"
10    proyectos: ["1","2","3","4"]
11    credencial_id: "10"
12  ){
13    nombre
14    proyectos{
15      _id
16      nombre
17    }
18  }
19 }
20

```

```

{
  "data": {
    "addEstudiante": {
      "nombre": "Mateo Suarez",
      "proyectos": [
        {
          "_id": "1",
          "nombre": "Java"
        },
        {
          "_id": "2",
          "nombre": "Spring Boot"
        },
        {
          "_id": "3",
          "nombre": "Python"
        },
        {
          "_id": "4",
          "nombre": "Angular"
        }
      ]
    }
  }
}

```

Figura 21. Mutación para crear un estudiante con GraphQL

```

1 mutation{
2   addInvestigador(
3     _id: "5"
4     nombre: "Martin Lutero"
5     proyectos:[{
6       idProyecto:"1"
7       intensidad: "11"
8     },
9     {
10      idProyecto: "4"
11      intensidad: "9"
12    }]
13    credencial_id: "9"
14  ){
15    nombre
16    proyectos{
17      Proyecto{
18        _id
19        nombre
20        descripcion
21      }
22      intensidad
23    }
24  }
25 }

```

```

{
  "data": {
    "addInvestigador": {
      "nombre": "Martin Lutero",
      "proyectos": [
        {
          "Proyecto": {
            "_id": "1",
            "nombre": "Java",
            "descripcion": "Proyecto sobre Java"
          },
          "intensidad": "11"
        },
        {
          "Proyecto": {
            "_id": "4",
            "nombre": "Angular",
            "descripcion": "Proyecto sobre Angular"
          },
          "intensidad": "9"
        }
      ]
    }
  }
}

```

Figura 22. Mutación para crear un investigador con GraphQL

Se crea un Schema de *GraphQL* para la actualización de un integrante.

Realizamos la creación del schema para resolver esta historia de usuario se puede ver de este manera:

```
updateEstudiante(  
  _id: String!  
  nombre: String  
  carrera: String  
  celular: String  
  fechaIngreso: String  
  cedula: String  
  estado: String  
  proyectos: [String]  
  credencial_id: String  
): estudiante
```

Figura 23. Instanciación del schema para actualizar un estudiante

```
updateInvestigador(  
  _id: String!  
  nombre: String  
  proyectos: [InfoDeProyectos]  
  credencial_id: String  
): investigador
```

Figura 24. Instanciación del schema para actualizar un investigador

Como se puede apreciar en la figura, el unico parametro obligatoria es el id que permite identificar a qué integrante se realizar la actualización y se colocan solo los valores que se van a modificar:

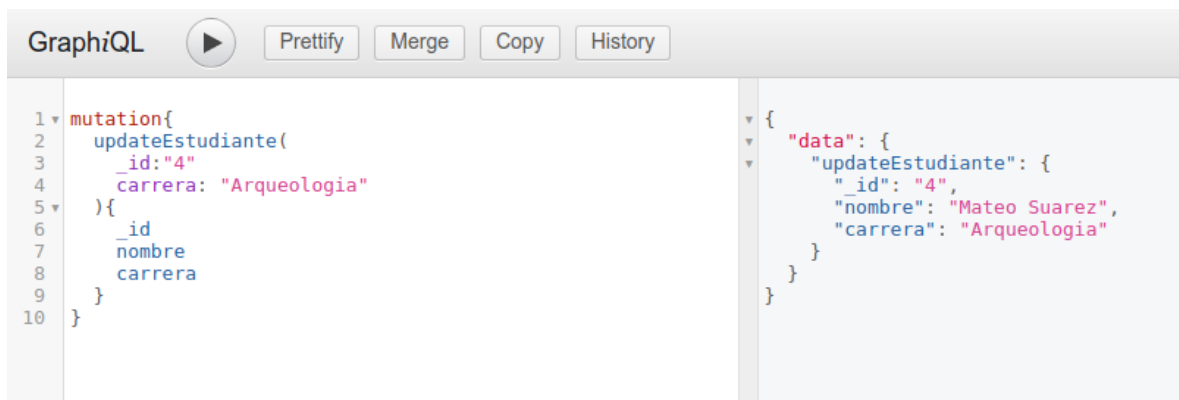


Figura 25. Mutación para actualizar un estudiante con GraphQL

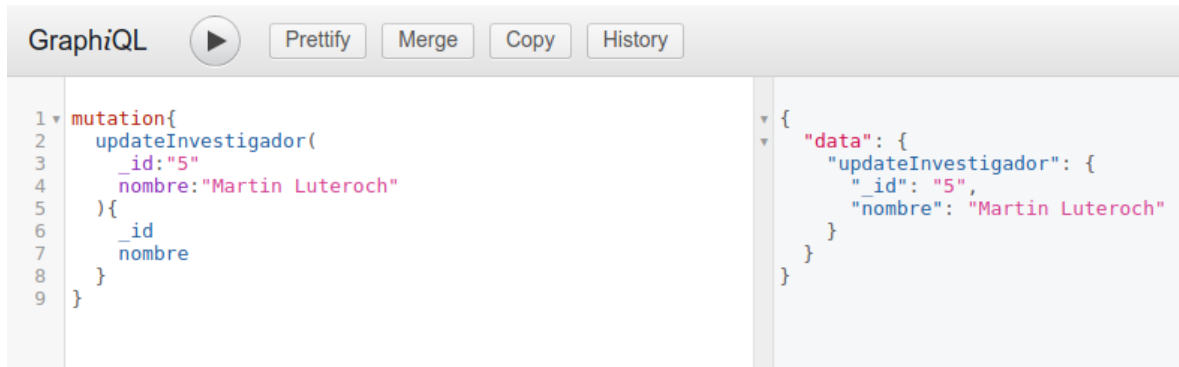


Figura 26. Mutación para actualizar un investigador con GraphQL

Historia de usuario: HU_031

Se crea un Schema de *GraphQL* para la eliminación de un integrante.

Realizamos la creación del schema para resolver esta historia de usuario se puede ver de este manera:

`removeEstudiante(_id: String!): estudiante`

Figura 27. Instanciación del schema para eliminar un estudiante

`removeInvestigador(_id: String!): investigador`

Figura 28. Instanciación del schema para eliminar un investigador

Como se puede apreciar en la figura, el parámetro id indica que proyecta va a ser eliminado:

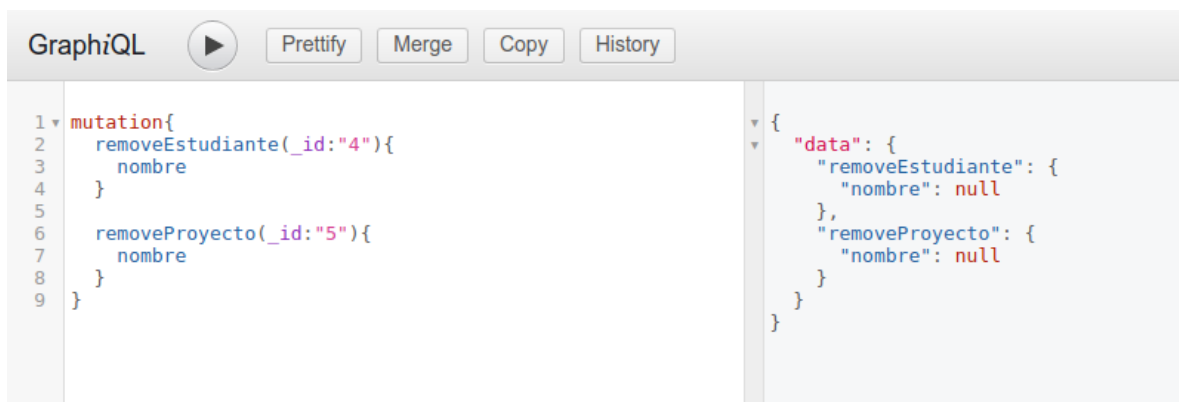


Figura 29. Mutación para eliminar un integrante con GraphQL

Historia de usuario: HU_032

Utilizando el esquema para actualizar a un integrante se puede agregar o quitar a que proyecto pertenece

Recordemos la estructura de la mutacion:

```
updateEstudiante(  
  _id: String!  
  nombre: String  
  carrera: String  
  celular: String  
  fechaIngreso: String  
  cedula: String  
  estado: String  
  proyectos: [String]  
  credencial_id: String  
): estudiante
```

Figura 30. Instanciación del schema para actualizar un estudiante

```
updateInvestigador(  
  _id: String!  
  nombre: String  
  proyectos: [InfoDeProyectos]  
  credencial_id: String  
): investigador
```

Figura 31. Instanciación del schema para actualizar un investigador

Y su implementación, en este caso solo se insertaría como valores los cambios pertenecientes a los proyectos con los cuales se encuentran vinculados:

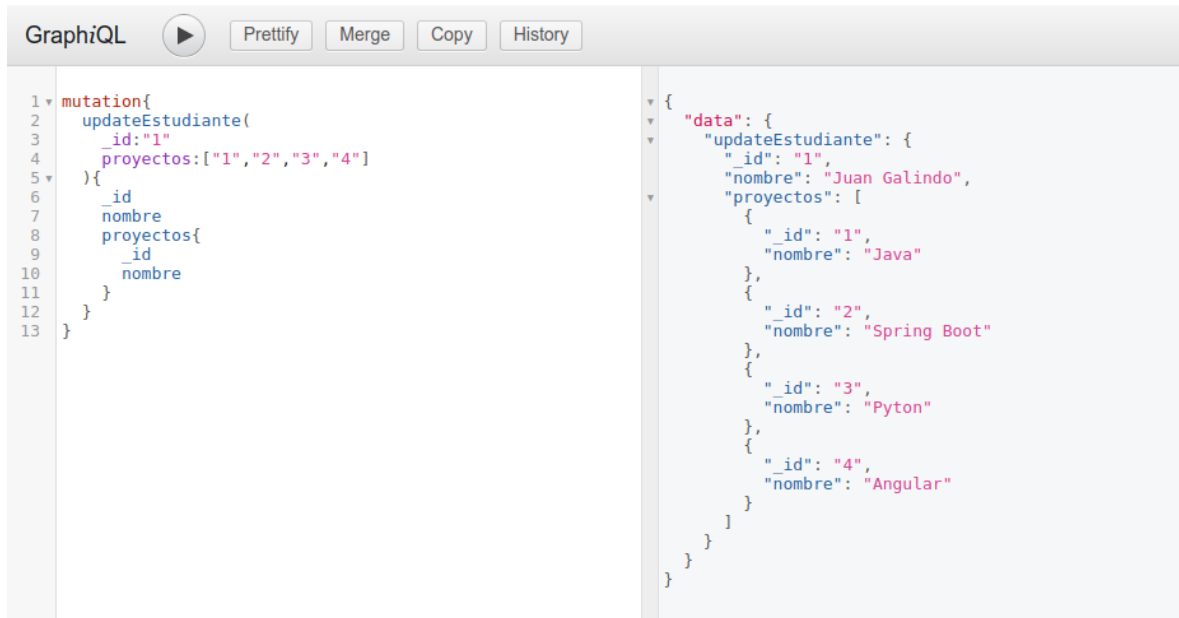


Figura 32. Mutación para actualizar los proyectos de un estudiante con GraphQL

Interfaz gráfica

Historia de usuario: HU_034 y HU_36

Nuestra interfaz grafica cuenta con un Login y un SignUp, que permite ingresar con tu usuario o crear uno nuevo para poder loguearse para obtener maor informacion.

Las vistas de Login y SignUp:

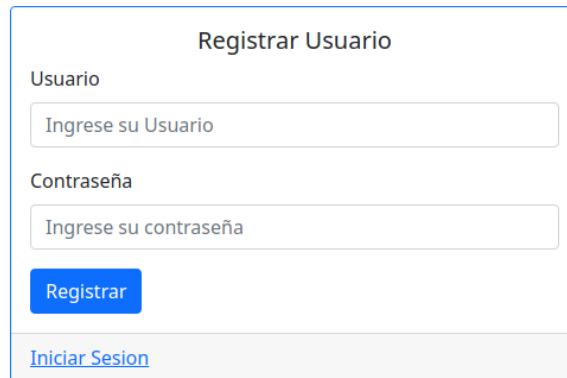
Iniciar Sesion

Usuario

Contraseña

[Registrar](#)

Figura 33. Vista para iniciar Sesion



El formulario 'Registrar Usuario' está contenido en un recuadro con un borde azul. En la parte superior, el título 'Registrar Usuario' está centrado. Debajo, el campo 'Usuario' incluye un input con el placeholder 'Ingrese su Usuario'. El campo 'Contraseña' incluye un input con el placeholder 'Ingrese su contraseña'. Un botón azul con el texto 'Registrar' está situado debajo de los campos. En la parte inferior del recuadro, el texto 'Iniciar Sesion' es un enlace azul subrayado.

Figura 34. Vista para registrar un usuario

Estas dos vistas nos permiten gestionar con la base de datos si nuestro usuario y contraseña existen. en caso contrario nos devolverá el error pertinente.

Iniciar Sesion

Usuario o contraseña incorrectos

Usuario

Stivel

Contraseña

.....

Iniciar Sesion

[Registrar](#)

Figura 35. Vista de inicio de sesion con un error de logueo

El caso más particular es aquel que nos dice que nuestro usuario o contraseña no fue encontrado en la base de datos. Estos dos datos tienen que corresponder a un mismo documento dentro de nuestra base de datos NoSQL.

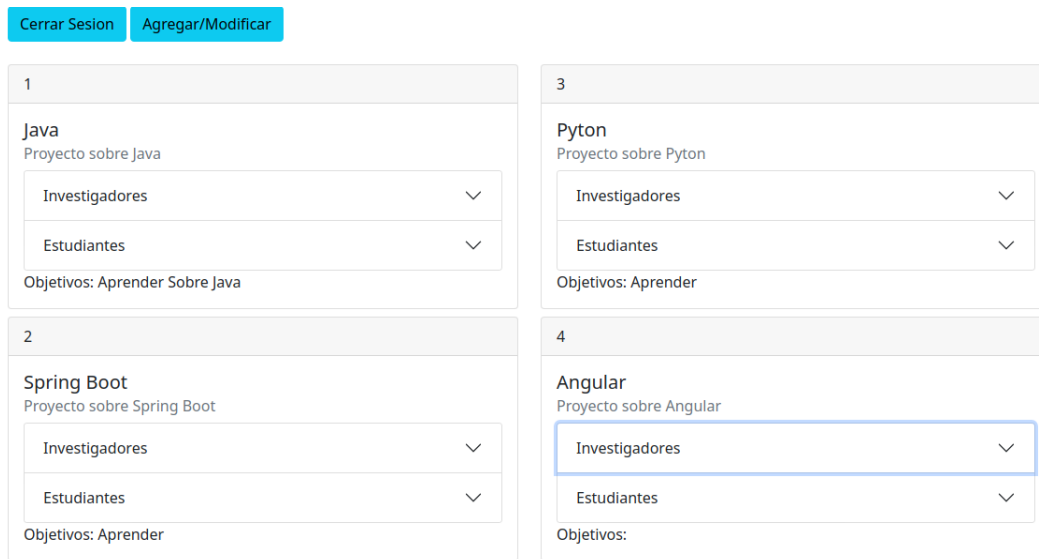


Figura 36. Vista general de los proyectos

Al iniciar sesión o haber creado un usuario nos redireccionará a una pestaña donde podremos ver parte de la información de los proyectos y sus integrantes al desplegar las pestañas de integrantes o estudiantes:

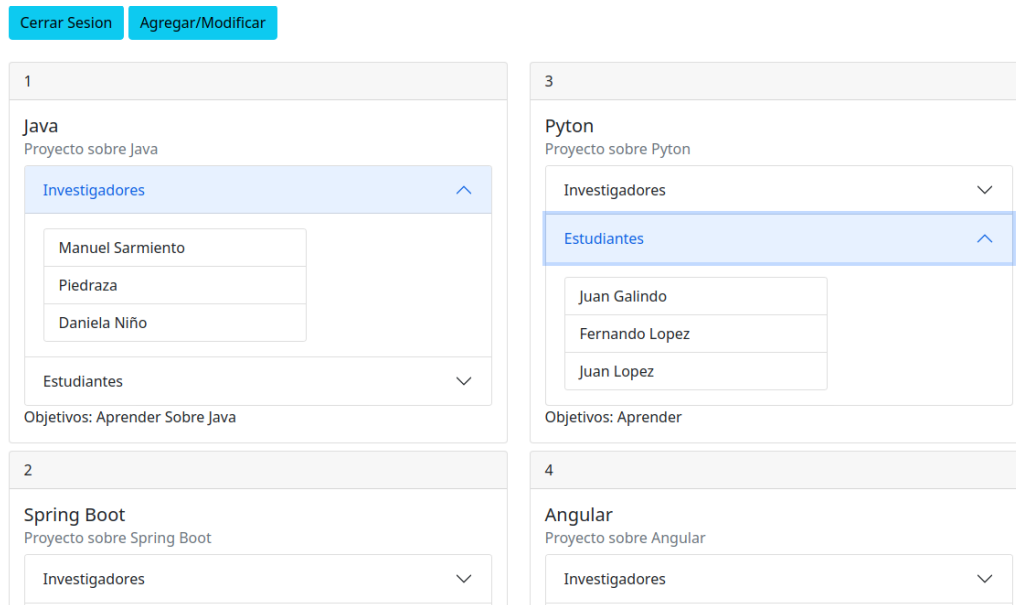


Figura 37. Despliegue de los integrantes en los proyectos

En la parte superior izquierda tenemos dos botones. El primero nos permite salir de la sesión en la que estamos, lo que conlleva a que necesitaríamos nuevamente logearnos para

poder visualizar los proyectos. El segundo botón nos redirecciona a un formulario en el que podemos crear un nuevo proyecto. El formulario no debe tener ninguna casilla sin llenar, teniendo en cuenta los Schemas que presentamos anteriormente:

Formulario para agregar un proyecto. El formulario está dividido en secciones con los siguientes campos:

- Nombre:** Campo de texto con el placeholder "Ingrese el nombre del Proyecto".
- Descripción:** Campo de texto con el placeholder "Descripción".
- Estado del Proyecto:** Campo de texto con el placeholder "Ingrese el estado del proyecto".
- Objetivos Generales:** Campo de texto con el placeholder "Objetivos Generales".
- Objetivos Específicos:** Campo de texto con el placeholder "Objetivos Específicos".
- Presupuesto:** Campo de texto con el placeholder "Ingrese el presupuesto".
- Fecha Inicio:** Campo de texto con el placeholder "Ingrese la fecha de inicio dle proyecto".
- Fecha Final:** Campo de texto con el placeholder "Ingrese la fecha de final dle proyecto".
- Avances:** Campo de texto con el placeholder "Avances".

En la parte inferior del formulario, hay un botón azul que dice "Agregar Proyecto" y un enlace azul que dice "Modificar Proyecto".

Figura 38. Vista para agregar un proyecto

Al seleccionar el botón de “*Modificar Proyecto*” nos redirecciona a un formulario similar que nos permitirá modificar un proyecto respecto a su id, este formulario debe llevar el id obligatoriamente, y solo los elementos que se deben cambiar son los que se deben llenar:

Agregar Proyecto

ID

Ingrese el ID del Proyecto

Nombre

Ingrese el nombre del Proyecto

Descripcion

Estado del Proyecto

Ingrese el estado del proyecto

Objetivos Generales

Objetivos Especificos

Presupuesto

Ingrese el presupuesto

Fecha Inicio

Ingrese la fecha de inicio dle proyecto

Fecha Finala

Ingrese la fecha de final dle proyecto

Avances

Modificar

[Agregar proyecto](#)

Figura 39. Vista para modificar un proyecto

Aporte de los integrantes

Debido a que por rendimiento para poder realizar la entrega de los sprints los roles que nos íbamos asignando podían variar bastante, ya que realizamos lo que estuviera en nuestras manos sin importar que no fuera parte de nuestro deber según nuestro rol y por ende cada uno realiza de todo, nos autocalificamos con el fin de expresar cómo nos desempeñamos dentro del grupo.

Auto-Calificación

Sebastián Fuentes	8/10	Como líder del grupo considero que mi experiencia aportó al equipo orden y conocimientos básicos para el trabajo en equipo.
Oscar Arbelaez	9/10	Mi participación constante en el grupo me permitió ofrecer mis habilidades para lograr la entrega de todos los Sprint.
David Carvalho	7/10	Mi tiempo no me permitía colaborar como me hubiera gustado, pero realice con éxito mis tareas respecto al grupo.
Andrea Giraldo	10/10	Considero muy grata la experiencia y estoy muy feliz de haber participado. De acuerdo a mi rendimiento lo considero bastante bueno, a pesar de mi falta de conocimiento, me encuentre a la par de las clases.
Stivel Pinilla	10/10	Con mi poca experiencia realice mi mayor esfuerzo para seguir el ritmo y trabajar al lado de mi equipo. La programación es de paciencia y constancia, y considero que aporte estas dos cosas en el desarrollo del curso.
Andres Agudelo	9/10	A pesar de lo apurado de las clases y el gran contenido de información de cada una de ellas, logre abstraer el

		conocimiento suficiente para aportar en el desarrollo de las actividades y el en apoyo a mis compañeros.
--	--	--

Enlaces

Trello: <https://trello.com/b/rxIH2t0x/mintic>

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