**4 Ciclo FullStackDevs TIC**

**SPRINT 3**

**CREACIÓN DE UN REPOSITORIO EN GITHUB**

**DIANA CAROLINA FLOREZ BLANCO**

**DANIEL FELIPE LUBO**

**BRAHAN ANDRES ACOSTA GALINDO**

**SANTIAGO VILLA ROMERO**

**SERGIO ANDRÉS RUIZ**

**MISION TIC 2022 – UNIVERSIDAD DE ANTIOQUÍA**

**BOGOTA D.C**

**05 / 12 / 2021**

1. **Descripción del proceso que se llevó a cabo para cumplir con los requerimientos de este sprint.**

* **Comandos ingresados para la creación del Backend:**

Para el desarrollo de este sprint se hizo uso del IDE Visual Studio Code, inicialmente para ingresar los comandos, clic en la opción terminal:



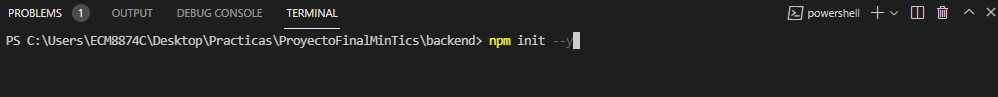
Luego de esto New terminal.

Captura de pantalla de computadora

Descripción generada automáticamente

Ya estando en la consola ingresamos los siguientes comandos:

npm init --y -- Para crear el proyecto inicial de node.js, crea el pakage.json



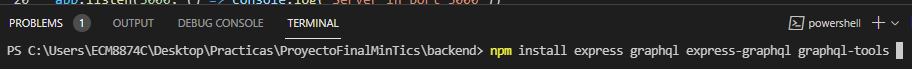
npm i nodemon -D --- Instala nodemon, Para que luego podamos modificar el json en el pakage.json, para que el servidor se mantenga arriba y se actualice automáticamente.



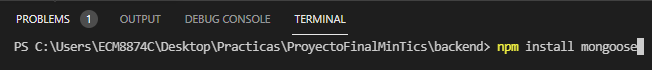
Texto

Descripción generada automáticamente

npm install express graphql express-graphql graphql-tools --instala graphql para hacer uso del freamework creado por Google, también se instala express para la creación de servidor y se instala express-graphql para integrar las dos tecnologías.



npm i mongoose -- Por ultimo instalamos mongoose Para usar los modelos y los esquemas (model first), además de integrar nuestra aplicación desarrollada con node.js. express, graphQL con mongodb.



* **Creación de conexión con la base de datos.**

Para este caso usamos el marco de trabajo mongoose instalado anteriormente, creamos un archivo llamado database.js en la carpeta Backend y usaremos la cadena de conexión de la base de datos alojada en el servidor de la nube Atlas. El código implementado se ilustra a continuación:

Texto

Descripción generada automáticamente

* **Creación y configuración del servidor.**

Inicialmente creamos un archivo de JavaScript llamado index.js en la siguiente ubicación: Backend/src/index.js.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Para la creación y configuración del servidor se hace uso del marco de trabajo express, en la línea uno se requiere express, en la línea 8 se inicializa la variabe app instanciando el servidor y en la línea 20 se indica que el servidor va a escuchar en el puerto 3000.

En este documento se indica que se van a usar dos servicios web Api Rest Full usando la tecnología Graphql:

* Servicio Web User: Se desarrolla de la línea 10 a 13.
* Servicio Web Proyect: Se desarrolla de la línea 15 a 18.

En este documento se importan dos archivos de la carpeta Shemas, los esquemas utilizados en los servicios REST, esta importación se realiza en las líneas 3 y 4 y se describirán en el documento actual con más detalle.

Texto

Descripción generada automáticamente

* **Creación de los esquemas utilizados por los servicios en graphQL.**

Para que el código tenga bajo acoplamiento se decide dejar los esquemas utilizados en una carpeta independiente se crea una carpeta shemas en la ubicación Backend/src/shemas/. En esta carpeta se crean dos archivos cada uno con el esquema pertinente para el caso, los archivos son projectShemas.js y userShemas.js.

Texto

Descripción generada automáticamente

1. **Evidencia de la funcionalidad de aplicación.**

Para evidenciar el funcionamiento de la aplicación se enunciará cada una de las historias de usuario, con su(s) debido(s) request y response.

* **Historia de usuario: HU\_001**

**Como** usuario

**Dado** que voy a ingresar al sistema de gestión de proyectos

**Cuando** necesite registrarme en el sistema

**Entonces** podré ingresar los datos de registro (Incluyendo elegir el rol al que aspiro)

Para cumplir con los criterios de aceptación en esta historia de usuario se crea usuario (Incluyendo elegir el rol al que aspira).

**Request:**

Texto

Descripción generada automáticamente

Se ingresan los datos básicos del usuario, como respuesta le indicamos que muestre todos los datos del usuario, el usuario puede elegir el rol al que aspira en este caso Estudiante.

**Response:**

Texto

Descripción generada automáticamente

Como respuesta del método créateUser muestra todos los datos del usuario, incluyendo el estado que por defecto es false indicándonos que el usuario aun no ha sido autorizado.

**Prueba de funcionamiento desde GraphQL:**

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

En la imagen se evidencia que se pueden ingresar los datos del registro de un usuario (Incluyendo elegir el rol al que aspiro). Comprobando solución de la HU\_001.

* **Historia de usuario: HU\_002**

**Como** usuario autorizado

**Dado** que voy a ingresar al sistema de gestión de proyectos

**Cuando** necesite autenticarme en el sistema

**Entonces** podré ingresar mi correo y contraseña para ser validados

Para cumplir con los criterios de aceptación en esta historia de usuario se realiza un método en el servicio usuario el cual valida inicialmente que el usuario este autorizado, para luego validar correo y contraseña, de cumplir con las tres condiciones a cabalidad devolverá los datos del usuario de no ser así devolverá un null.

**Autenticación errada (Correo Correcto, Contraseña correcta, Sin autorizar).**

Para este escenario de prueba utilizaremos el usuario que creamos para demostrar la funcionalidad del **HU\_001**.

Texto

Descripción generada automáticamente

**Request:**

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Ingresamos el correo y contraseña verídicos del usuario, en el método Login el cual se encarga de validar la autenticación.

**Response:**

Texto

Descripción generada automáticamente con confianza media

El método nos devuelve un null, no se puede validar usuario satisfactoriamente esto sucede ya que el usuario no está autorizado (estado: false)

**Prueba de funcionamiento desde GraphQL:**

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

En la imagen se evidencia que si en la autenticación se falla con el correo o con a contraseña o con la autorización no retorna datos ,Comprobando solución de la HU\_002.

**Autenticación correcta (Correo Correcto, Contraseña correcta, Autorizado).**

Para este escenario de prueba utilizaremos un usuario autorizado (Estado: true)

Texto, Carta

Descripción generada automáticamente

**Request:**

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Ingresamos el correo y contraseña verídicos del usuario, en el método Login el cual se encarga de validar la autenticación.

**Response:**

Texto

Descripción generada automáticamente

El método nos devuelve un objeto de tipo User, Valida el usuario satisfactoriamente esto sucede ya que el usuario está autorizado (estado: true)

**Prueba de funcionamiento desde GraphQL:**

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

En la imagen se evidencia que si la autenticación es correcta esto quiere decir con el correo correcto y con la contraseña correcta y con la autorización en verdadero, retorna el objeto del usuario , Comprobando solución de la HU\_002.

* **Historia de usuario: HU\_003**

**Como** usuario.

**Dado** que ingresé al sistema de gestión de proyectos.

**Cuando** necesite actualizar la información personal.

**Entonces** podré ingresar los datos que deseo actualizar.

Para este escenario de prueba utilizaremos el usuario que creamos para demostrar la funcionalidad del **HU\_001**.

Texto

Descripción generada automáticamente

**Request:**

Texto

Descripción generada automáticamente

En el método updateUser e cual recibe dos parámetros, el primero el id del usuario y como segundo parámetro un objeto de tipo usuario donde se le puede enviar los datos que deseo edita. En este caso vamos a editar el nombre, correo y contraseña.

**Response:**

Texto

Descripción generada automáticamente

Como respuesta el servicio nos retorna el nuevo usuario modificado.

**Prueba de funcionamiento desde GraphQL:**

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

En la imagen se evidencia que se puede actualizar a información persona de usuario ingresando los datos que deseo actualizar. Comprobando solución de la **HU\_003**.

* **Historia de usuario: HU\_004**

**Como** administrador

**Dado** que ingresé al sistema de gestión de proyectos

**Cuando** requiera ver la lista de los usuarios registrados en la plataforma (Tanto autorizados como no autorizados)

**Entonces** podré ver la información de los usuarios registrados en la plataforma.

**Request:**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Para que el administrador consulte toda la lista de usuarios, se hará uso del método Users, que como datos de salida en este caso tendrá todos los datos del usuario.

**Response:**

Texto

Descripción generada automáticamente

Como respuesta se puede ver la información de los usuarios registrados en la plataforma (Tanto autorizados como no autorizados), en nuestro caso son 5 usuarios.

**Prueba de funcionamiento desde GraphQL:**

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

En la imagen se evidencia que se puede ver la información de los usuarios registrados en la plataforma (Tanto autorizados como no autorizados). Comprobando solución de la **HU\_004**.

* **Historia de usuario: HU\_005**

**Como** administrador

**Dado** que estoy viendo la lista de los usuarios registrados en la plataforma

**Cuando** requiera aceptar un usuario en la plataforma

**Entonces** podré cambiar el estado del usuario

Para este escenario de prueba utilizaremos un usuario no autorizado (Estado: false)

Texto

Descripción generada automáticamente

**Request:**

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Para que el administrador pueda aceptar un usuario en la plataforma debe hacer uso del método updateUser enviándole en el estado: true.

**Response:**

Texto

Descripción generada automáticamente

Como respuesta obtenemos e usuario modificado en su atributo estado: true, dejándolo como aceptado por parte del administrador.

**Prueba de funcionamiento desde GraphQL:**

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

En la imagen se evidencia que se puede aceptar un usuario en la plataforma, cambiando el estado del usuario. Comprobando solución de la **HU\_005**.

* **Historia de usuario: HU\_006**

**Como** administrador

**Dado** que ingresé al sistema de gestión de proyectos

**Cuando** requiera ver la lista de proyectos registrados en la plataforma

**Entonces** podré ver la lista de los proyectos registrados en la plataforma

**Request:**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Para ver la lista de los proyectos se debe hacer uso del método Projects del api encargada de los proyectos <http://localhost:3000/Project>.

**Response:**

Texto

Descripción generada automáticamente con confianza media

Texto

Descripción generada automáticamente

En el reponse se evidencian dos proyectos creados, Se evidencian todos los proyectos creados en la BD.

**Prueba de funcionamiento desde GraphQL:**

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

En la imagen se evidencia que el administrador puede ver la lista de los proyectos registrados en la plataforma. Comprobando solución de la **HU\_006**.

* **Historia de usuario: HU\_007**

**Como** administrador

**Dado** que estoy viendo la lista de los proyectos registrados en la plataforma

**Cuando** requiera aprobar la creación de un proyecto

**Entonces** podré actualizar el estado del proyecto

Para este escenario de prueba utilizaremos un proyecto no aprobado (Estado: false)

Texto

Descripción generada automáticamente

**Request:**

Texto

Descripción generada automáticamente

Para modificar el estado del proyecto se hará uso del método updateProject, este método recibe dos parámetros el primero el \_id del Project el segundo es el modelo del proyecto con los datos que se quieren modificar en este caso el estado.

**Response:**

Texto

Descripción generada automáticamente

En el response se observa el estado del proyecto en true.

**Prueba de funcionamiento desde GraphQL:**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

En la imagen se observa que el administrador si puede aprobar la creación de un proyecto actualizando el estado. Comprobando solución de la **HU\_007**.

* **Historia de usuario: HU\_008**

**Como** administrador

**Dado** que estoy viendo la lista de los proyectos registrados en la plataforma

**Cuando** requiera activar o inactivar un proyecto

**Entonces** podré actualizar el estado del proyecto

Para este escenario de prueba utilizaremos un proyecto aprobado (Estado: true)

Texto

Descripción generada automáticamente

**Request:**

Texto

Descripción generada automáticamente

Para modificar el estado del proyecto se hará uso del método updateProject, este método recibe dos parámetros el primero el \_id del Project el segundo es el modelo del proyecto con los datos que se quieren modificar en este caso el estado.

**Response:**

Texto

Descripción generada automáticamente

En el response se observa el estado del proyecto en false.

**Prueba de funcionamiento desde GraphQL:**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

En la imagen se observa que el administrador si puede actualizar el estado del proyecto. Comprobando solución de la **HU\_008**.

* **Historia de usuario: HU\_009**

**Como** administrador

**Dado** que estoy viendo la lista de los proyectos registrados en la plataforma

**Cuando** requiera cambiar la fase de un proyecto de “En desarrollo” a “Terminado”

**Entonces** podré actualizar la fase del proyecto.

Para este escenario de prueba utilizaremos un proyecto en fase desarrollo (fase: "En desarrollo")

Texto

Descripción generada automáticamente

**Request:**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Para modificar la fase del proyecto se hará uso del método updateProject, este método recibe dos parámetros el primero el \_id del Project el segundo es el modelo del proyecto con los datos que se quieren modificar en este caso el estado.

**Response:**

Texto

Descripción generada automáticamente

En el response se observa el estado del proyecto en terminado.

**Prueba de funcionamiento desde GraphQL:**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

En la imagen se observa que el administrador puede cambiar la fase del proyecto como por ejemplo cambiar la fase de un proyecto de “En desarrollo” a “Terminado”. Comprobando solución de la **HU\_009**.

* **Historia de usuario: HU\_010**

**Como** líder

**Dado** que ingresé al sistema de gestión de proyectos

**Cuando** requiera ver la lista de los estudiantes registrados en la plataforma (Tanto autorizados como no autorizados)

**Entonces** podré ver la información de los estudiantes registrados en la plataforma

**Request:**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Para poder ver la información de los estudiantes registrados en la plataforma usamos el método userByType, este método recibe un string tipoUsuario.

**Response:**

Texto

Descripción generada automáticamente

Como respuesta el servicio nos arroja dos registros. Los dos estudiantes existentes entre los usuarios.

**Prueba de funcionamiento desde GraphQL:**

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

En la imagen se observa que el líder puede ver la lista de los estudiantes registrados en la plataforma (Tanto autorizados como no autorizados). Comprobando solución de la **HU\_010**.

* **Historia de usuario: HU\_011**

**Como** líder

**Dado** que estoy viendo la lista de los estudiantes registrados en la plataforma

**Cuando** requiera aceptar un estudiante en la plataforma

**Entonces** podré cambiar el estado del estudiante de “Pendiente” a “Autorizado”.

Para este escenario de prueba utilizaremos un usuario en estado pendiente y tipo estudiante (Estado: false)

Texto

Descripción generada automáticamente

**Request:**

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Para que el líder pueda autorizar un estudiante en la plataforma debe hacer uso del método updateUser enviándole en el estado: true.

**Response:**

Texto

Descripción generada automáticamente

Como respuesta obtenemos al estudiante modificado en su atributo estado: true, dejándolo como autorizado por parte del lider.

**Prueba de funcionamiento desde GraphQL:**

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

En la imagen se evidencia que el líder puede aceptar un estudiante en la plataforma, cambiando el estado del usuario. Comprobando solución de la **HU\_011**.

* **Historia de usuario: HU\_012**

**Como** líder

**Dado** que ingresé al sistema de gestión de proyectos

**Cuando** requiera liderar un nuevo proyecto

**Entonces** podré crear un nuevo proyecto

Para este escenario de prueba utilizaremos un usuario de tipo Líder (tipousuario: Lider)

Texto

Descripción generada automáticamente

**Request:**

Texto

Descripción generada automáticamente

Para que el líder pueda crear un proyecto debe hacer uso del método createProject, en los usuarios inscritos ingresamos al lider con su respectivo ID.

**Response:**

Texto

Descripción generada automáticamente

Como resultado tenemos el usuario creado en la base de datos.

**Prueba de funcionamiento desde GraphQL:**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

En la imagen se evidencia que el líder podrá crear un nuevo proyecto, quedando como líder de este. Comprobando solución de la **HU\_012**.

* **Historia de usuario: HU\_013**

**Como** líder

**Dado** que ingresé al sistema de gestión de proyectos

**Cuando** requiera ver los proyectos que lidero (Tanto los que están en estado “Activo” como “Inactivo”).

**Entonces** sólo podré listar los proyectos que tengo a cargo

Para este caso de prueba usaremos el siguiente usuario con rol de Líder (tipoUsuario: lider).

Texto

Descripción generada automáticamente

**Request:**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Para que el líder pueda listar los proyectos que tenga a cargo, debe hacerse uso de método projectByIdUser enviando el ID del usuario.

**Response:**

Imagen que contiene Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

En el response se evidencia que el usuario tiene dos proyectos donde el usuario se encuentra como líder de estos.

**Prueba de funcionamiento desde GraphQL:**

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

En la imagen se evidencia que el líder podrá ver los proyectos que lidera (Tanto los que están en estado “Activo” como “Inactivo”). Comprobando solución de la **HU\_013**.

* **Historia de usuario: HU\_014**

**Como** líder

**Dado** que estoy viendo la lista de los proyectos que he registrado en la plataforma

**Cuando** requiera actualizar los datos de uno de los proyectos que lidero y que está en estado “Activo”

**Entonces** podré editar la información relacionada al proyecto cuya información necesito actualizar (Solamente puede actualizar: Nombre del proyecto, los objetivos generales, específicos y el presupuesto).

Para la edición del proyecto se usará el siguiente proyecto el cual se encuentra en estado activo (estado: true)

Texto

Descripción generada automáticamente

**Request:**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Para que el líder pueda modificar sus proyectos en los campos: Nombre del proyecto, los objetivos generales, específicos y el presupuesto, debe hacer uso del método UpdateProject como se evidencia en el request.

**Response:**

Imagen que contiene Tabla

Descripción generada automáticamente

Se evidencia que como response el servicio obtiene el proyecto modificado con los campos ingresados en el request.

**Prueba de funcionamiento desde GraphQL:**

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

En la imagen se evidencia que el líder podrá editar la información relacionada al proyecto cuya información necesito actualizar (Solamente puede actualizar: Nombre del proyecto, los objetivos generales, específicos y el presupuesto). Comprobando solución de la **HU\_014**.

* **Historia de usuario: HU\_015**

**Como** líder

**Dado** que ingresé al sistema de gestión de proyectos

**Cuando** requiera revisar las solicitudes pendientes por aceptar o rechazar de estudiantes de inscripción a mis proyectos

**Entonces** podré listar las solicitudes realizadas por los estudiantes.

Para este caso de prueba usaremos el siguiente usuario el cual tiene rol de líder y además de eso está activo.

Texto

Descripción generada automáticamente

**Request:**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Para revisar las solicitudes pendientes por aceptar o rechazar de estudiantes de inscripción a mis proyectos, haremos uso del método projectByIdUser.

**Response:**

Texto

Descripción generada automáticamente con confianza baja

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente con confianza media

Imagen que contiene Tabla

Descripción generada automáticamente

En el response se observa que el líder tiene 5 cursos a su cargo de los cuales solo uno tiene solicitudes de estudiantes para inscripción en el proyecto, este curso es el último, la lista de solicitudes al curso se observa en la parte inscritos, las solicitudes se distinguen por que el rol del inscrito es estudiante, si el estado es false es porque el estudiante aun no ha sido aceptado en el curso.

**Prueba de funcionamiento desde GraphQL:**

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente con confianza media

En la imagen se evidencia que el líder podrá revisar las solicitudes pendientes por aceptar o rechazar de estudiantes de inscripción a sus proyectos. Comprobando solución de la **HU\_015**.

* **Historia de usuario: HU\_016**

**Como** líder

**Dado** que estoy viendo la lista de solicitudes de inscripción a los proyectos que lidero

**Cuando** requiera cambiar el estado a las solicitudes

**Entonces** podré aceptar o rechazar sus inscripciones

Para este caso de prueba haremos uso de un proyecto que tiene dos solicitudes de inscripción de los estudiantes, este proyecto se ilustra a continuación.

Imagen que contiene Tabla

Descripción generada automáticamente

**Request:**

Texto

Descripción generada automáticamente

Para que el líder pueda aceptar o rechazar inscripciones a los cursos por parte de los estudiantes, debe hacer uso del método updateProject, en su objeto inscritos, en este caso acepta dos estudiantes.

**Response:**

Imagen que contiene Tabla

Descripción generada automáticamente

En el response del proyecto se observa que el líder acepto dos estudiantes a su curso.

**Prueba de funcionamiento desde GraphQL:**

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

En la imagen se evidencia que el líder podrá aceptar o rechazar las inscripciones de los estudiantes. Comprobando solución de la **HU\_016**.

* **Historia de usuario: HU\_017**

**Como** líder

**Dado** que estoy viendo la lista de los proyectos que he registrado en la plataforma

**Cuando** necesite realizar una revisión a uno de mis proyectos

**Entonces** podré listar la información relacionada al proyecto que deseo revisar (Incluyendo los avances).

**Request:**

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Para poder listar la información relacionada al proyecto que el líder desee revisar se deberá hacer uso del método projectbyId, como dato de entrada recibe el id del proyecto.

**Response:**

Imagen que contiene Tabla

Descripción generada automáticamente

En el response se observa la información del proyecto, incluyendo sus avances.

**Prueba de funcionamiento desde GraphQL:**

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

En la imagen se evidencia que el líder podrá listar la información relacionada al proyecto que desea revisar (Incluyendo los avances). Comprobando solución de la **HU\_017**.

* **Historia de usuario: HU\_018**

**Como** líder

**Dado** que estoy viendo la lista de los avances registrados en uno de los proyectos que lidero

**Cuando** necesite agregar observaciones a un avance en uno de mis proyectos

**Entonces** podré actualizar el campo de observaciones del avance seleccionado.

Para modificar una observación de un avance de un proyecto por parte del líder se ara uso del siguiente proyecto.

Imagen que contiene Tabla

Descripción generada automáticamente

**Request:**

Texto

Descripción generada automáticamente

Para que el líder pueda modificar una observación de un avance en un proyecto se debe hacer uso de método updateProject enviando la observación modificada.

**Response:**

Imagen que contiene Tabla

Descripción generada automáticamente

En el response se evidencia la modificación en la observación del avance en el proyecto.

**Prueba de funcionamiento desde GraphQL:**

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

En la imagen se evidencia que el líder podrá actualizar el campo de observaciones del avance seleccionado, Comprobando solución de la **HU\_018**.

* **Historia de usuario: HU\_019**

**Como** estudiante

**Dado** que ingresé al sistema de gestión de proyectos

**Cuando** requiera ver la lista de proyectos registrados en la plataforma

**Entonces** podré ver la lista de los proyectos registrados en la plataforma

**Request:**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Para que el estudiante pueda ver la lista de los proyectos se debe hacer uso del método Projects del api encargada de los proyectos <http://localhost:3000/Project>.

**Response:**

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Imagen que contiene Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Imagen que contiene Tabla

Descripción generada automáticamente

En el response se evidencian 10 proyectos creados, Se evidencian todos los proyectos creados en la BD.

**Prueba de funcionamiento desde GraphQL:**

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

En la imagen se evidencia que el estudiante podrá ver la lista de los proyectos registrados en la plataforma, Comprobando solución de la **HU\_019**.

* **Historia de usuario: HU\_020**

**Como** estudiante

**Dado** que estoy viendo la lista de los proyectos registrados en la plataforma

**Cuando** requiera inscribirme en un proyecto

**Entonces** podré generar una solicitud de inscripción al proyecto

Para que el estudiante pueda generar una solicitud de inscripción al proyecto usaremos el siguiente proyecto.

Imagen que contiene Tabla

Descripción generada automáticamente

**Request:**

Texto

Descripción generada automáticamente con confianza media

Para que un estudiante genere una solicitud de inscripción a un proyecto debe hacer uso del método updateProject, agregando un dato más con su Id y como rol estudiante.

**Response:**

Imagen que contiene Tabla

Descripción generada automáticamente

En el response se evidencia que se agrega un nuevo dato a él array inscritos que se evidencia en el response.

**Prueba de funcionamiento desde GraphQL:**

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

En la imagen se evidencia que el estudiante podrá generar una solicitud de inscripción a un proyecto, Comprobando solución de la **HU\_020**.

* **Historia de usuario: HU\_021**

**Como** estudiante

**Dado** que mi inscripción a un proyecto fue aceptada

**Cuando** requiera listar los avances de un proyecto en el que estoy inscrito

**Entonces** podré ver la lista de los avances del proyecto registrados

Para ver la lista de avances del proyecto registrados por parte del estudiante, usaremos el siguiente usuario de tipo estudiante.

Texto

Descripción generada automáticamente

**Request:**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Para ver la lista de los avances de los proyectos registrados por parte de los estudiantes se debe hacer uso del método projectByidUser, enviando su id como parámetro de entrada.

**Response:**

Imagen que contiene Tabla

Descripción generada automáticamente

En este response se observan los proyectos a los cuales esta inscrito el estudiante además de estar aprobado, se pueden ver los avances del proyecto.

**Prueba de funcionamiento desde GraphQL:**

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

En la imagen se evidencia que el estudiante puede listar los avances de un proyecto en el que el estudiante está inscrito, Comprobando solución de la **HU\_021**.

* **Historia de usuario: HU\_022**

**Como** estudiante

**Dado** que estoy viendo la lista de proyectos

**Cuando** requiera registrar avances a un proyecto en el que estoy inscrito (Y este se encuentra en estado activo)

**Entonces** podré ingresar la descripción de mi avance en el proyecto

Para realizar la modificación del proyecto ingresando un nuevo avance, se realizará uso del siguiente proyecto.

Imagen que contiene Tabla

Descripción generada automáticamente

**Request:**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Para que el estudiante pueda registrar avances en el proyecto debe hacer uso del método updateProject, agregando un nuevo avance al mismo.

**Response:**

Imagen que contiene Tabla

Descripción generada automáticamente

En el response se evidencia que el estudiante agrego una nueva especificación al proyecto al cual se encuentra inscrito.

**Prueba de funcionamiento desde GraphQL:**

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

En la imagen se evidencia que el estudiante puede registrar avances a un proyecto en el que está inscrito (Y este se encuentra en estado activo), Comprobando solución de la **HU\_022**.

* **Historia de usuario: HU\_023**

**Como** estudiante

**Dado** estoy viendo la lista de avances de un proyecto

**Cuando** requiera actualizar la información de un avance

**Entonces** podré modificar la descripción del avance

El proyecto que vamos a modificar en la descripción del avance por parte del estudiante es el siguiente.

Imagen que contiene Tabla

Descripción generada automáticamente

**Request:**

Texto

Descripción generada automáticamente

Para que el estudiante pueda modificar la descripción del avance debe hacer uso del método updateProject en su parte Avance.

**Response:**

Imagen que contiene Tabla

Descripción generada automáticamente

En el response se observa que el usuario modifico la descripción del avance del proyecto.

**Prueba de funcionamiento desde GraphQL:**

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

En la imagen se evidencia que el estudiante puede modificar la descripción de un avance al cual se encuentre escrito, Comprobando solución de la **HU\_023**.