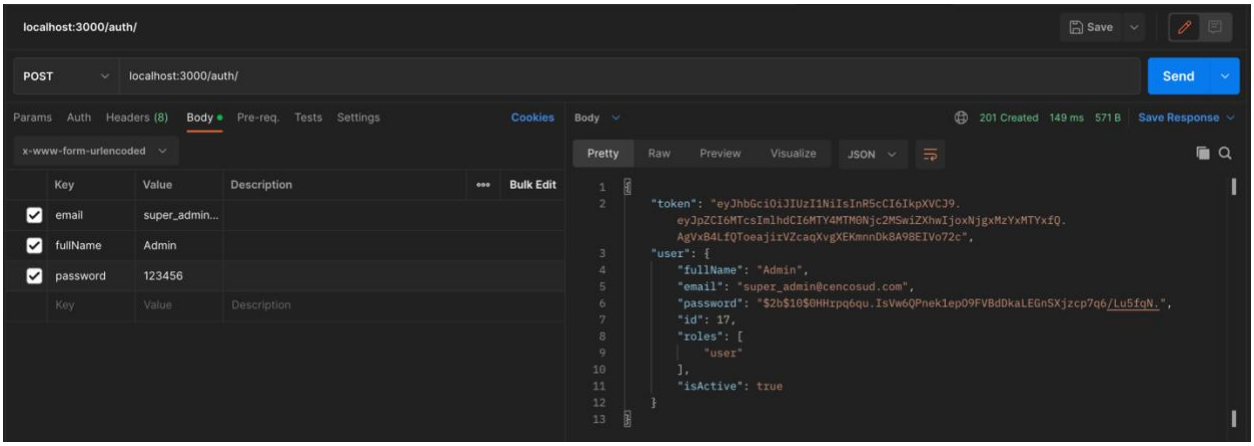
Trabajo integrador.

Por Nathaniel Boyd

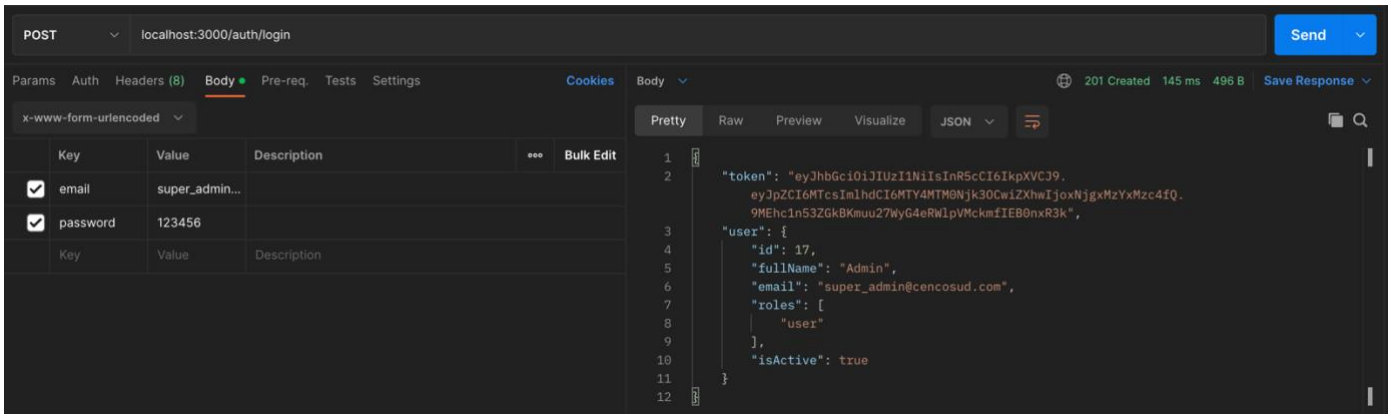
Instrucciones para usar la aplicación:

Con la aplicación de postman, cree un nuevo usuario usando el método POST y la ruta localhost/3000/auth ingresando las llaves ‘email’, ‘fullName’, y ‘password’

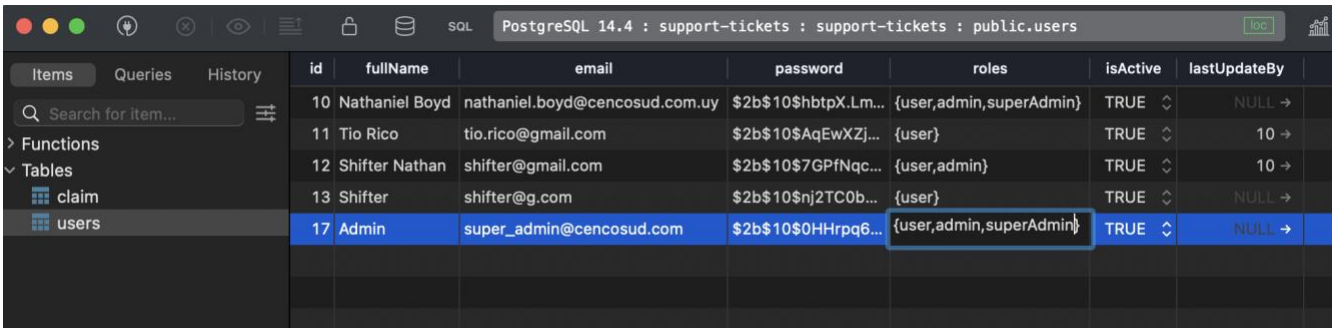


Copie el email que ingresó y usando el mismo método POST y la ruta existente, agregue una barra seguido de login para ingresar ingresar como usuario existente, la ruta debería quedar así: localhost:3000/auth/login .

En los valores de las llaves ‘email’ y ‘password’ pegue el email copiado e ingrese su contraseña, esto le devolverá un JWT token valido por 4 horas y establecerá el usuario ingresado como parámetro global hasta la expiración del JWT token. Para terminar con el ingreso asegúrese de copiar el token provisionado sin las comillas, el mismo será usado en adelante.



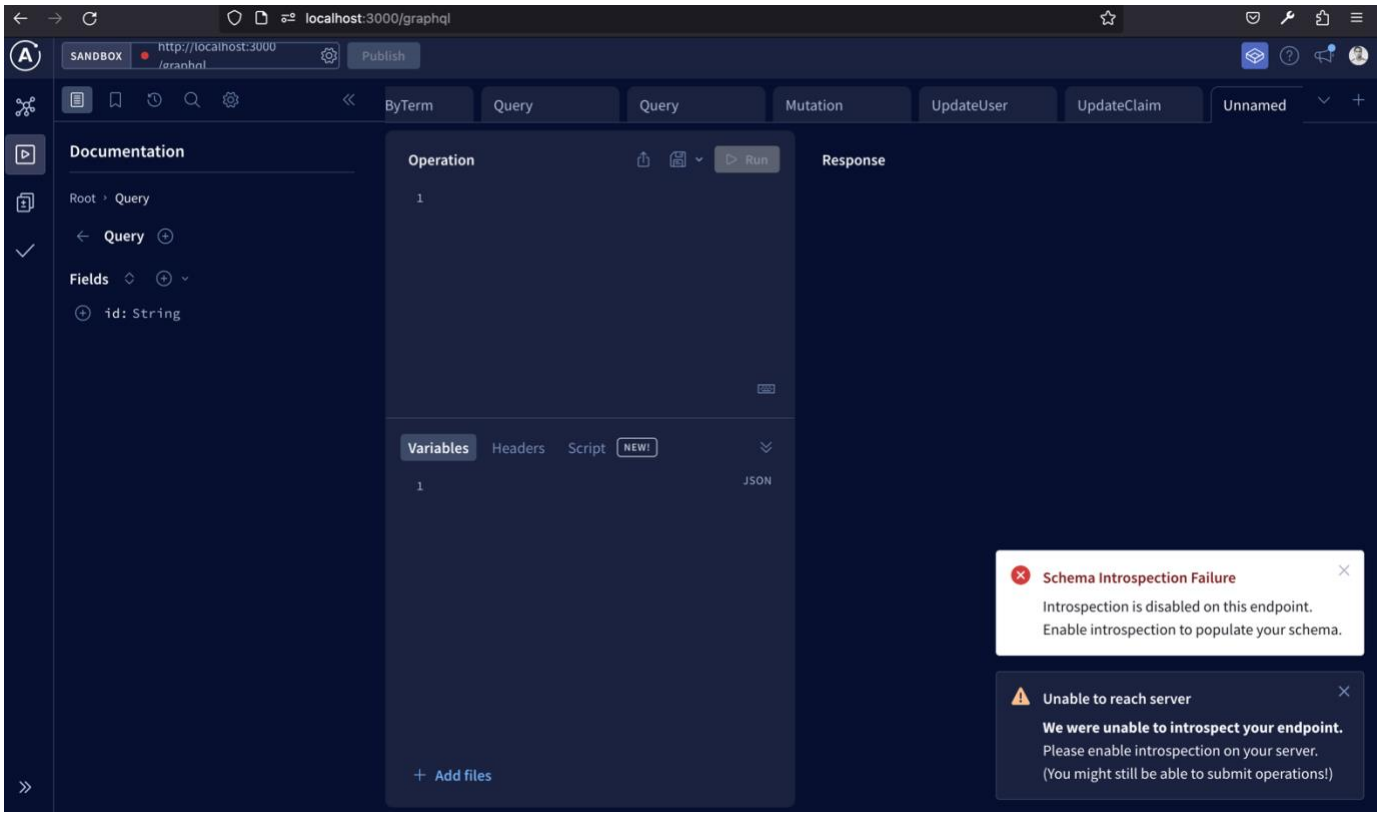
Diríjase a TablePlus y posiciónese en la base de datos conectada con el container de Docker, si la base de datos aun no está conectada, por favor siga las instrucciones proporcionadas en el readme.md de este repositorio. Una vez en TablePlus abra la tabla ‘users’ y modifique el campo roles, simplemente agregando una coma después de cada palabra e ingresando los valores de ‘admin’ y ‘superAdmin’, pulse ctrl+S o command+S para guardar los cambios. Esto está pensado para hacerse una única vez, ya que solo el rol de ‘superAdmin’ es el único poseedor de privilegios necesarios para editar a otros usuarios, incluyendo su rol o estado activo en caso de que haya sido bloqueado.



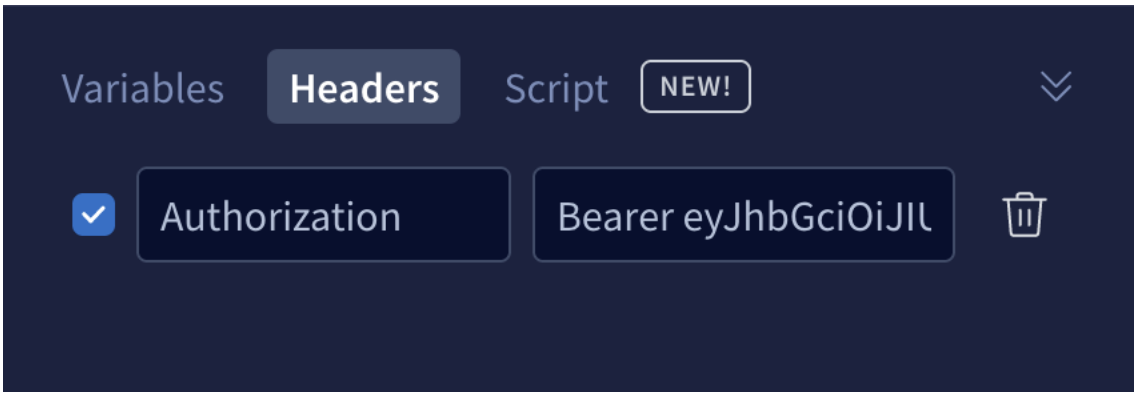
Luego, con la aplicación en ejecución y el usuario con el rol de superAdmin guardado en la base de datos, cree otro usuario de la misma manera para hacer pruebas futuras con el rol ‘user’. Abra su navegador de preferencia y diríjase a la ruta:

localhost:3000/graphql

Esto le llevara al Apollo Server Studio activado de manera global para la aplicación. Notará que no existe esquema alguno de graphql para hacer pruebas, incluso este le arrojará un error. Esto se debe a que la solicitud no está autenticada

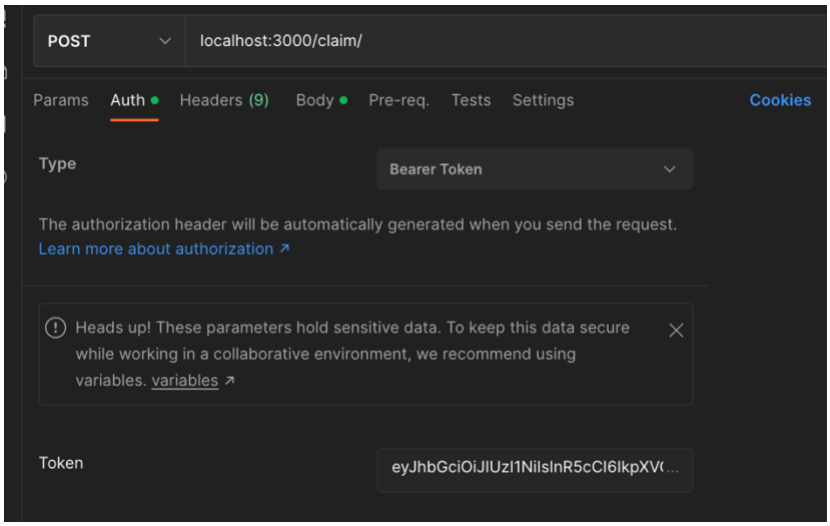


Para autenticar la solicitud, haga click en ‘Headers’ en la sección inferior de ‘Operation’ del Apollo Server y haga click en ‘New Header’, esto le activara 2 campos para ingresar texto, en el primero ‘header key’ ingrese la palabra ‘Authorization’ al literal, respetando mayúsculas y sin comillas, en el campo ‘value’ ingrese la palabra ‘Bearer’ seguido de un espacio y pegue el token que copió al momento de ingresar con el ‘superAdmin’.
Los campos llenos deberían lucir así:

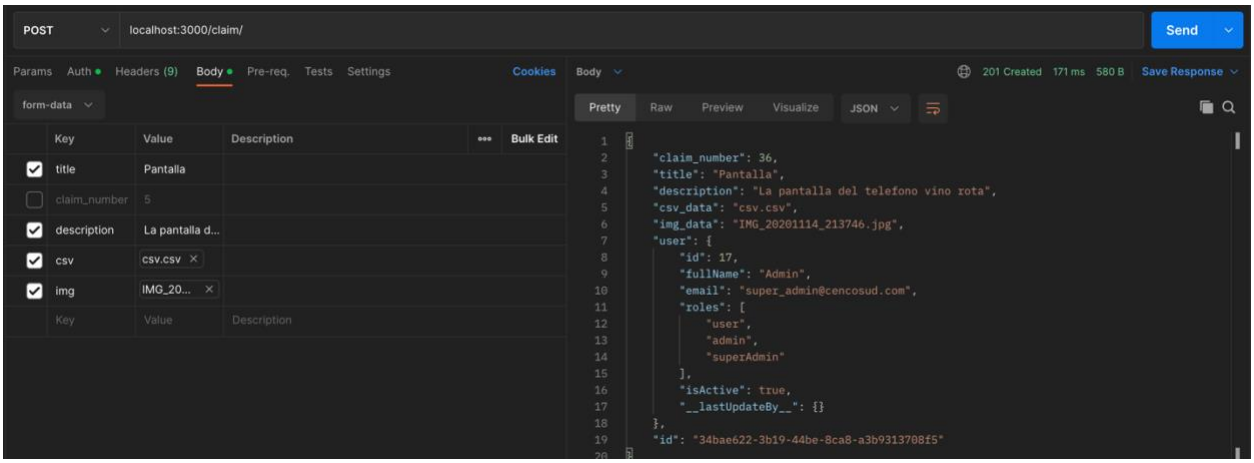


Asegurese de ingresar un token proporcionado a través del ingreso de un ‘superAdmin’ de lo contrario no podrá ejecutar muchas de las ‘Queries’ y ‘Mutations’ de GraphQL, ya que están protegidas por ‘Guards’ que en su mayoría, verificarán el rol del usuario ingresado antes de permitir ejecutar la petición.

Ahora cree uno o varios reclamos, para ello regrese a Postman y escriba la ruta **localhost/3000/claim** con el método POST. Para enviar el POST deberá ingresar el JWT token en la sección ‘Auth’ seleccionando en el ‘Type’ la opción ‘Bearer Token’, por último deberá pegar el token en el campo proporcionado



Con el JWT token ingresado en la ruta claim, ahora diríjase a la sección ‘Body’ y en el botón desplegable escoja la opción ‘form-data’, escriba las llaves ‘title’, ‘description’, ‘csv’ e ‘img’. Llene los dos primeros valores de llaves a su gusto, pero para el csv deberá subir un archivo con extensión .csv de manera obligatoria, de lo contrario la solicitud no podrá ser enviada, puede descargarse uno de internet o generar el suyo propio a través de un archivo Excel, esto está pensado para que cuando el usuario cree un reclamo relacionado a un producto, este producto ya tenga adjunto un archivo csv con toda la información necesaria a procesar, pero por motivos de pruebas, por favor suba su propio archivo ‘.csv’. Lo mismo sucederá con el campo ‘img’, las extensiones admitidas son ‘.jpg’, ‘.jpeg’ y ‘.png’ pero esto es opcional, puede dejarlo en nulo o simplemente desmarcar la llave.



Haga tantos reclamos como desee, estos se asociarán al id del usuario asociado al token ingresado. Para asociarlos a otros usuarios, haga el progreso de login y utilice ese token en la sección ‘Auth’. Una vez hecho esto, todos los demás procesos de consulta y modificación de datos se hacen en el Apollo Server Studio de GraphQL. Juegue todo lo que quiera con las consultas y pulse Ctrl+Espacio o Command+Espacio para sugerencias de escritura en las Queries/Mutations.

Proceso de desarrollo:

1. Creación de API REST para hacer CRUD de tickets de reclamos
3. Implementación de base de datos Postgres a través de una imagen de Docker.
3. Creación e integración de variables de entorno para la BD y app.
4. Conexión del CRUD de la API REST con la base de datos
5. Implementación de multer para subida de archivos
6. Implementación de csv-parser para transformar archivos .csv en strings y guardarlos en la BD
7. Implementación de graphql para consultar todos los registros y registros por id
8. Creación automática y en secuencia verificada de números de reclamo, logrando ocupar cada número en orden ascendente del 1 a n a pesar de que se elimine un registro de reclamo.
9. Implementación de método findOne() a través de graphql el cual admite 3 términos de búsqueda para un registro ‘uuid’, ‘title’ y ‘claim_number. El parámetro de title encuentra todos los registros similares a un texto ingresado, eliminando tildes, espacios y transformando el texto a minúsculas para mejorar la búsqueda.
10. Creación de usuarios utilizando bcrypt para hashear la contraseña
11. Implementación de jwt y guards para validar el ingreso de un usuario y proteger los endpoints de usuarios no validados / permitidos.

12. Bloqueo lógico de usuarios, a través de una mutation se cambia el campo isActive de true a false
13. Actualización de usuarios por mutation, el id es el único campo requerido
14. Desacoplación de graphql de los métodos singUp y login para creación y autenticación de usuarios, transferidos a una API rest, con el objetivo de bloquear globalmente el esquema de graphql si no se ha provisto un token valido, solo obtenible al hacer login y de permitir la creación e ingreso de usuarios sin ingresar un token.
16. Graphql Mutations relacionadas al módulo de usuario protegidas con custom Guard con el objetivo de permitirle el bloqueo o edición de usuarios solo a usuarios con rol 'superAdmin' y las consultas findAll y findOne a los usuarios con rol 'admin' y 'superAdmin'
15. Implementación de peticiones autenticadas, los usuarios de rol 'user' solo pueden obtener información relacionada a su 'user.id' mientras que los usuarios con rol 'admin' o 'superAdmin' pueden obtener información de cualquier usuario
16. Implementación de autenticación y carga opcional de imágenes para la creación y edición de reclamos disponible para cualquier usuario
17. Refactorización de varios métodos y documentación de la API

A través de REST:
POST /claim = crea un nuevo reclamo, requiere los campos 'title', 'description', 'csv_data' (el cual debe ser un archivo csv)

DEL /claim:uuid = elimina un registro existente a través del uuid, utiliza el findOne que admite 3 terminos pero este posee un pipe y validación para procesar el método únicamente a través del uuid.

POST /auth = crea un nuevo usuario, esta ruta debe ser ejecutada primero para generar el token de usuario
keys necesarias para enviar el post:
- email

- fullName
- password

POST /auth/login = Comprueba que el email y la clave des-hasheada almacenados en la base de datos sean correctos, si se cumple la condición genera un JwT token valido por hasta 4 horas, de lo contrario lanzara un error.
Keys necesarias para el post:
- email
- password
Una vez obtenido el token, se deberá copiar para usarlo en las consultas de graphql, en la sección '**Headers',** se debe crear un nuevo Header haciendo click en el botón inferior izquierdo, en el **'header key'** debe estar escrito literal '**Authorization'** y en el value debe ir la palabra '**Bearer**' seguido de un espacio y el token obtenido en el login de postman, la llave y el valor mencionados deben ir sin comillas