Documentación de aplicación de reclamos

Descripción de la aplicación:

Esta es una aplicación de tickets construida en NestJS, con un modelo de API Rest y también provee un endpoint en Graphql para consultas.

Originalmente fue construida utilizando una base de datos postgres en docker pero también se le añadió la opción de utilizar una base de datos postgres en una RDS en AWS en la nube actualmente funcionando, en el ReadMe están todos los pasos para levantar la app tanto utilizando docker como AWS,

Aparte de la RDS para el tema de las imágenes las guarda en un bucket S3 en AWS, este bucket se utiliza tanto utilizando docker como el rds ya que las imágenes se guardan en el bucket con un id único como uuid y luego ese uuid se guarda al crear un ticket para enlazar las imágenes a los tickets.

En la Raiz del proyecto esta la coleccion de Postman con todos los servicios y objetos de prueba para cada servicio

También especifica cómo crear el archivo .env pero las credenciales para mayor seguridad no están en el repositorio, estas mismas las dejare a continuacion, cuidado de no difundir estas credenciales:

```
AWS_REGION = us-east-1
AWS_ACCESS_KEY = AKIAWFIWGE5XSHIRRG6R
AWS_SECRET_KEY = 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
AWS_BUCKET = bucketpruebafedecenco
```

Autentificación:

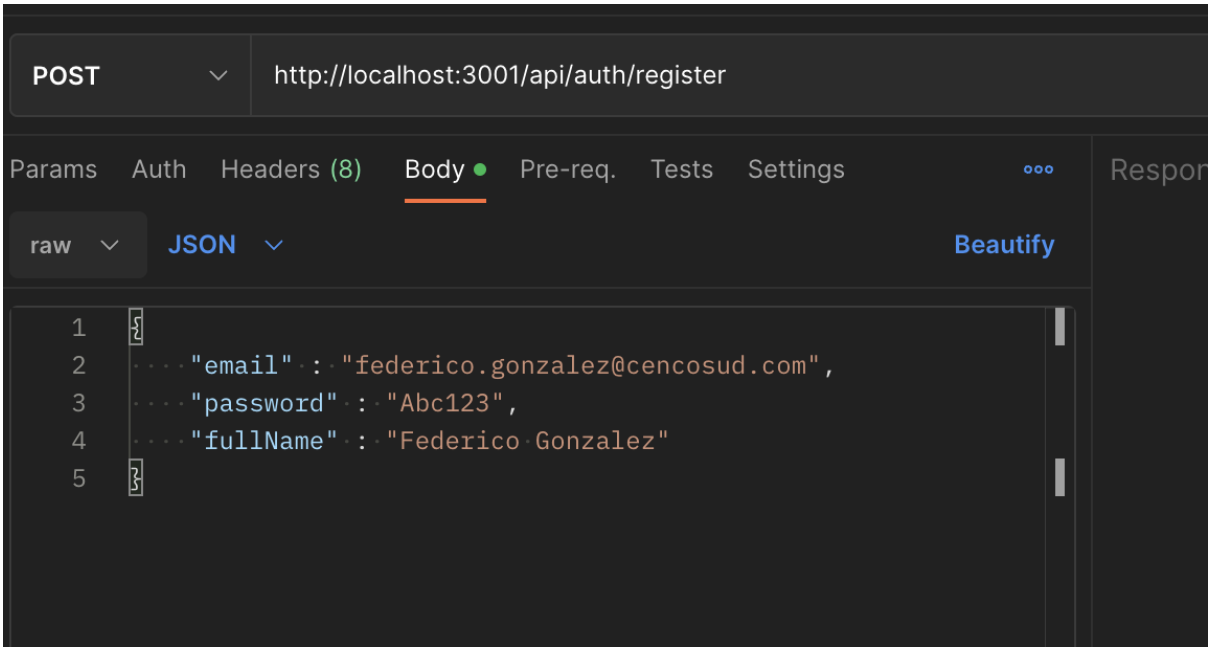
Servicios:

- register
- login
- makeAdmin

Todos los servicios menos los de este módulo piden autenticación mediante Jwt y utiliza un sistema de roles para que si no tenes el rol específico no puedas acceder a ciertos

servicios, por ejemplo con el rol usuario tienes acceso a ver las listas, los gets, pero no tienes acceso a modificar ni eliminar registros, solo los usuarios admin pueden.
Para esto tenes primero el servicio de register para crear un usuario

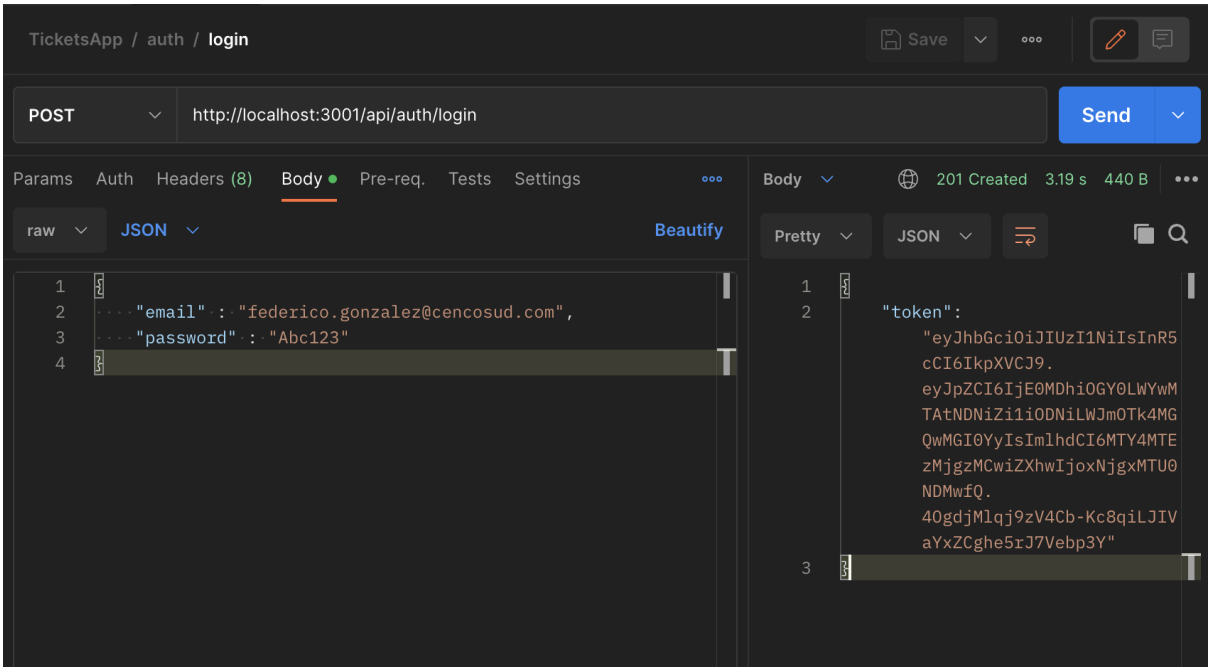
Servicio: register



Crea un usuario, importante en este servicio la contraseña se guarda encriptada en la base de datos utilizando la libreria bcrypt y por defecto el usuario se crea con el rol “user”

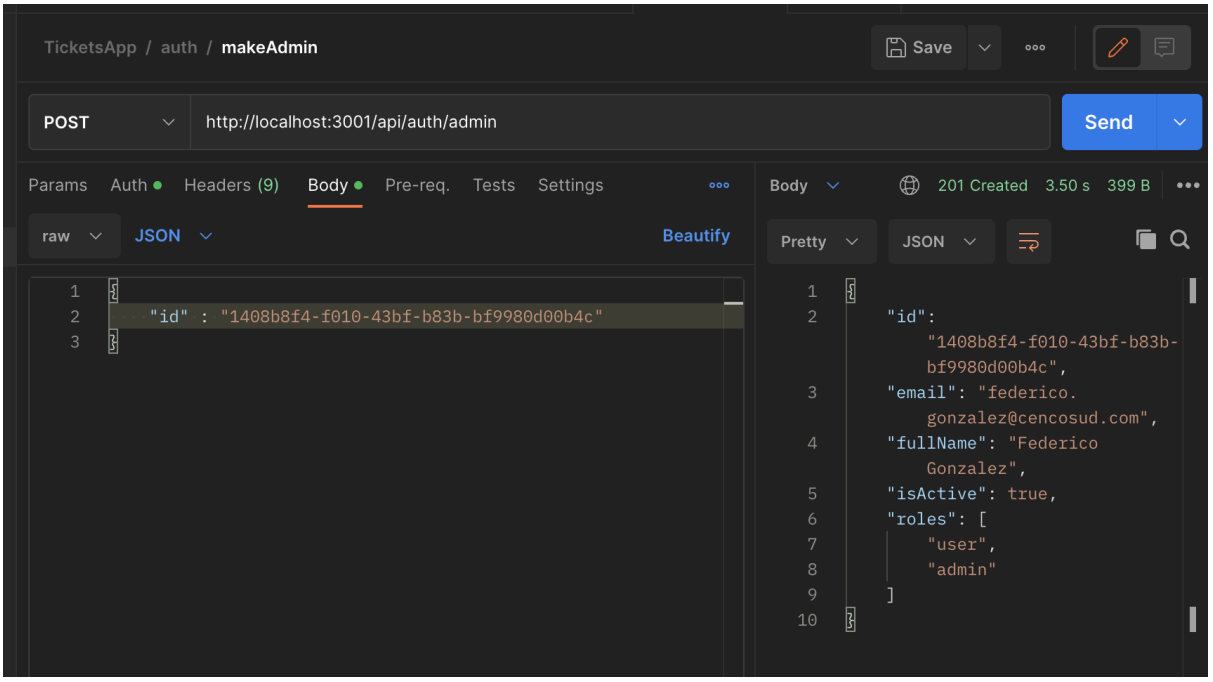
una vez registrados, nos podemos logear en la aplicacion para obtener el jwt

Servicio: login



Este servicio chequea si las credenciales son correctas en la bd y te devuelve el token para utilizar en la aplicacion.

Servicios:MakeAdmin



Con este servicio podemos asignarle el rol admin a un usuario para poder tener acceso total a la aplicacion.

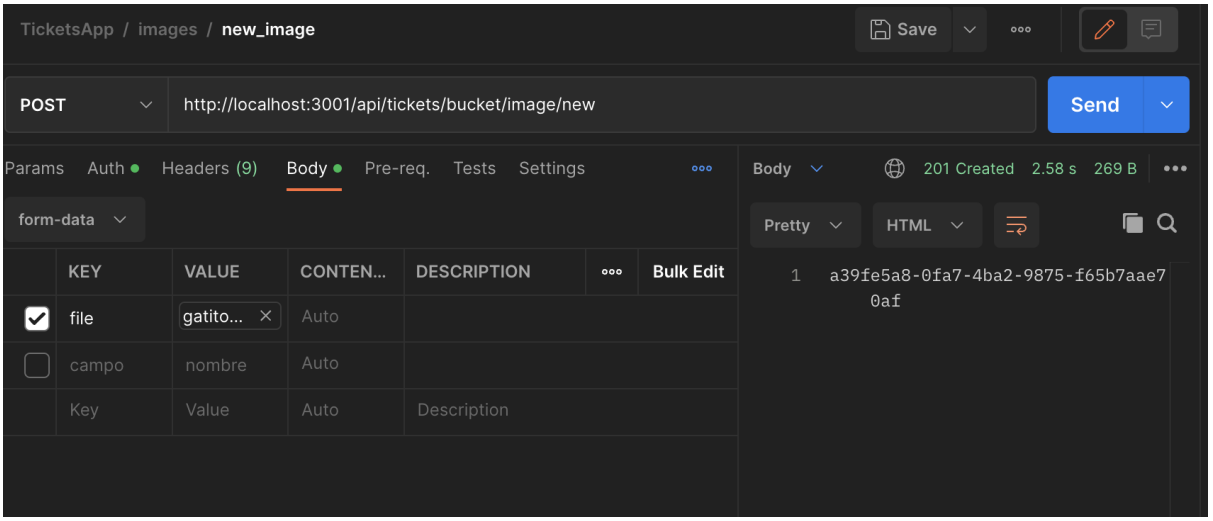
Tickets:

Creación de tickets:

Servicios:

- new_image
- new_ticket

Primero subimos la imagen: new_image



El servicio sube la imagen al bucket S3, la guarda y te devuelve un id único como uuid, para que posteriormente en el servicio se guarde.

Listar tickets:

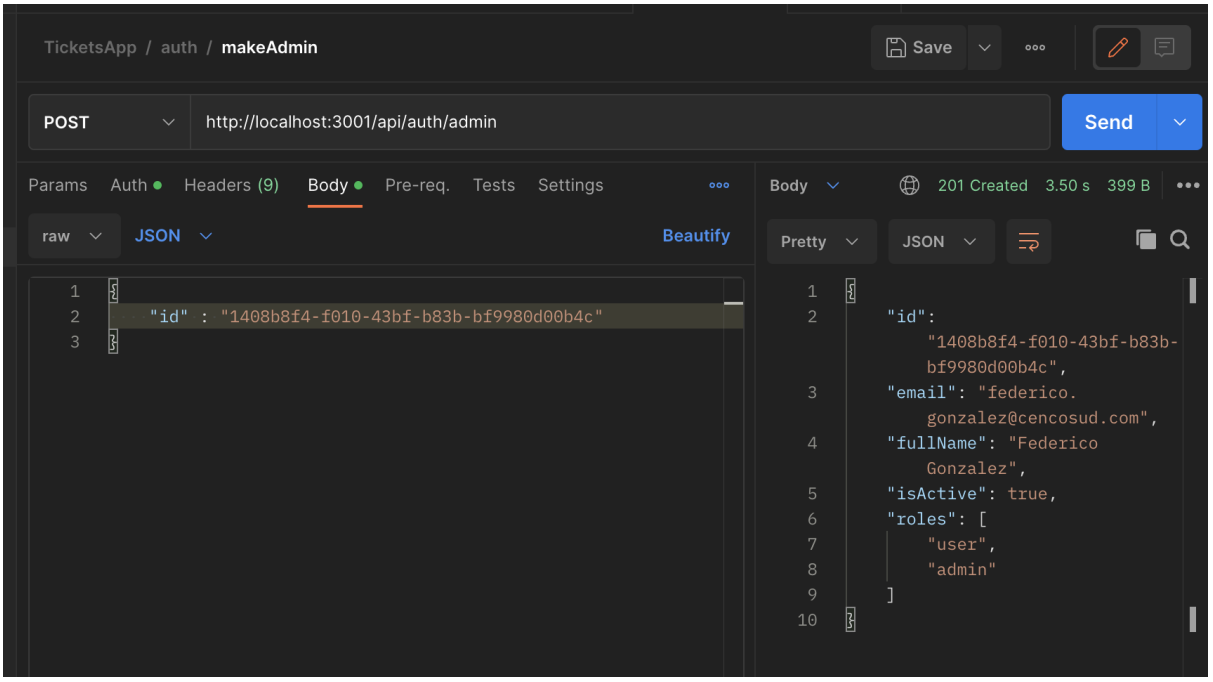
Para listar los tickets hay varios servicios para adaptarse a las necesidades del usuario existe el servicio:

- **getAll:** Traer todos los registros en la base de datos.
- **getCategory**: Trae registros por categoría: “Falla” “Reclamo” “Consulta”.
- **getOne**: Trae un registro por el id del mismo.
- **getLike**: Trae uno o varios registros filtrando por palabras específicas tanto en la descripción como en el titulo.

Para mas ejemplos en cada caso revisar la coleccion de Postman en la raiz del proyecto que contiene ejemplos de uso de cada caso y objetos de prueba.

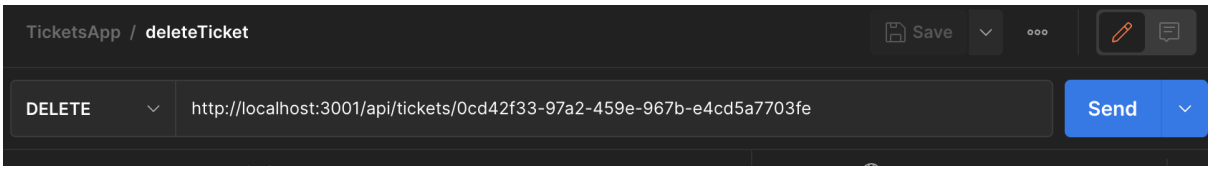
Actualizar Ticket:

Servicio: updateTicket

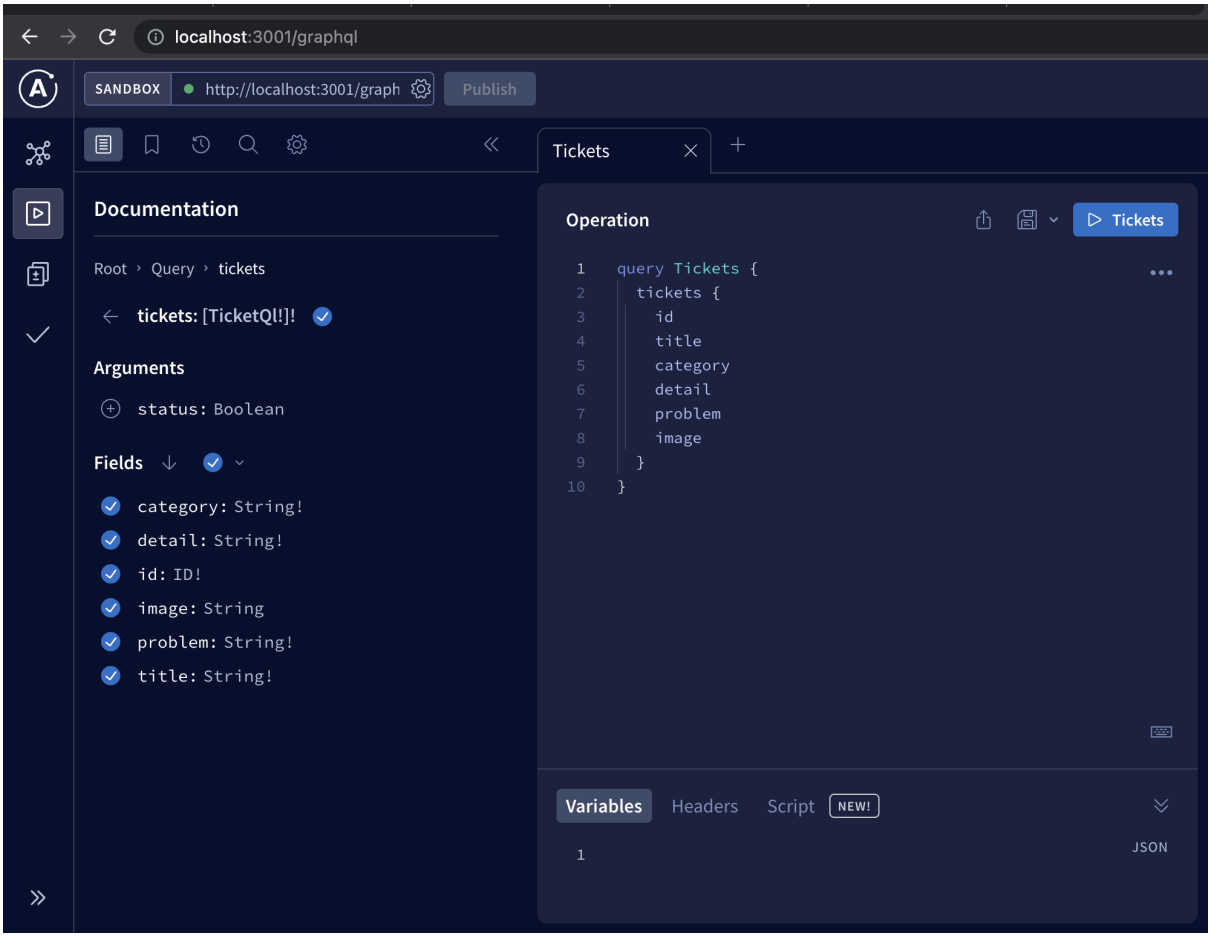


Eliminar Ticket:

Servicio: deleteTicket

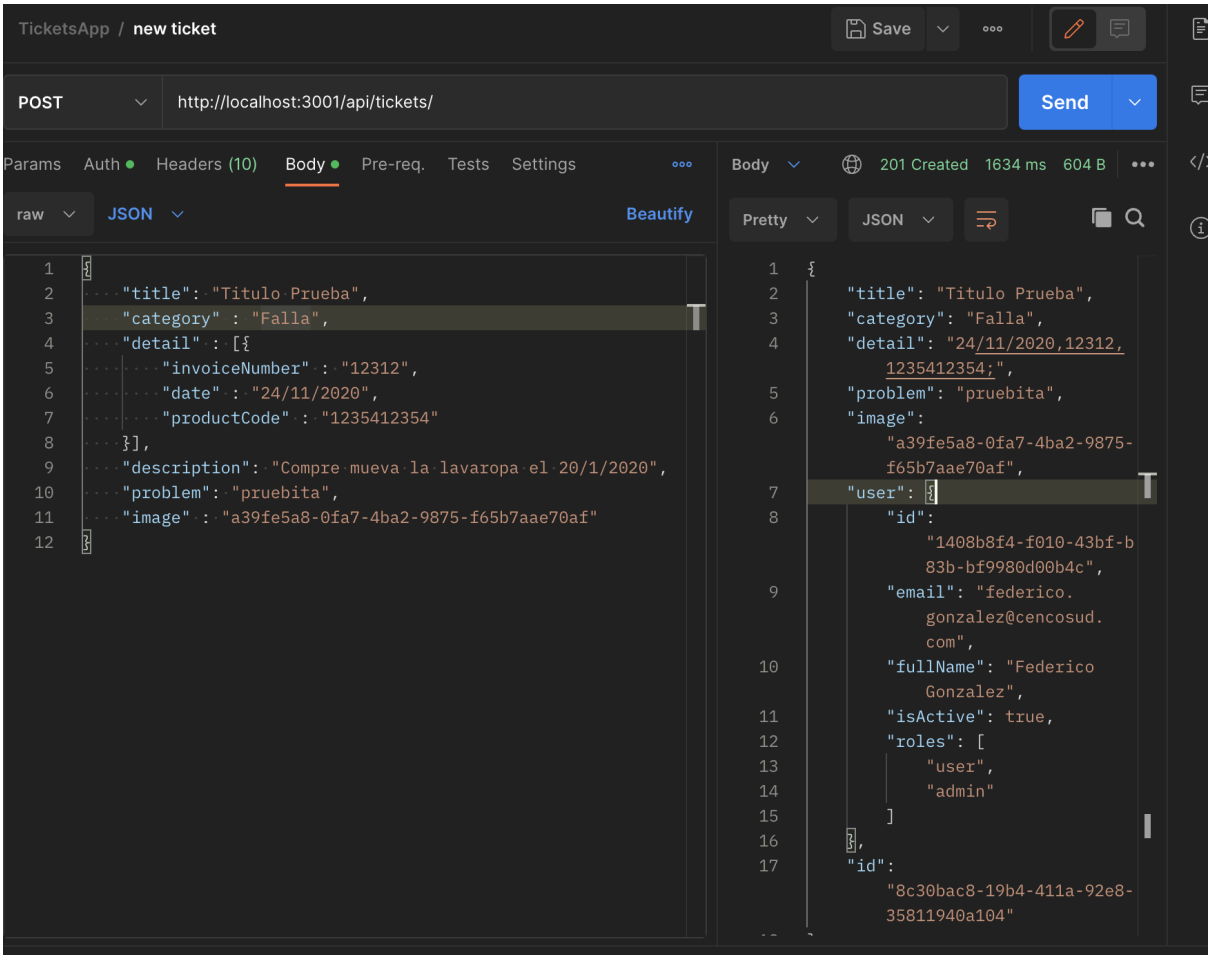


Graphql:



Este servicio provee un endpoint donde podes traer uno o varios registros utilizando graphql

Segundo creamos el Ticker: new_ticket



El servicio guarda el ticket y toma el usuario del token para tener un control de que usuario creo cada ticket