The following document provides a step-by-step guide detailing the process I undertook to download, install, and configure the RocketChat instance. It includes screenshots and explanations of each action taken.

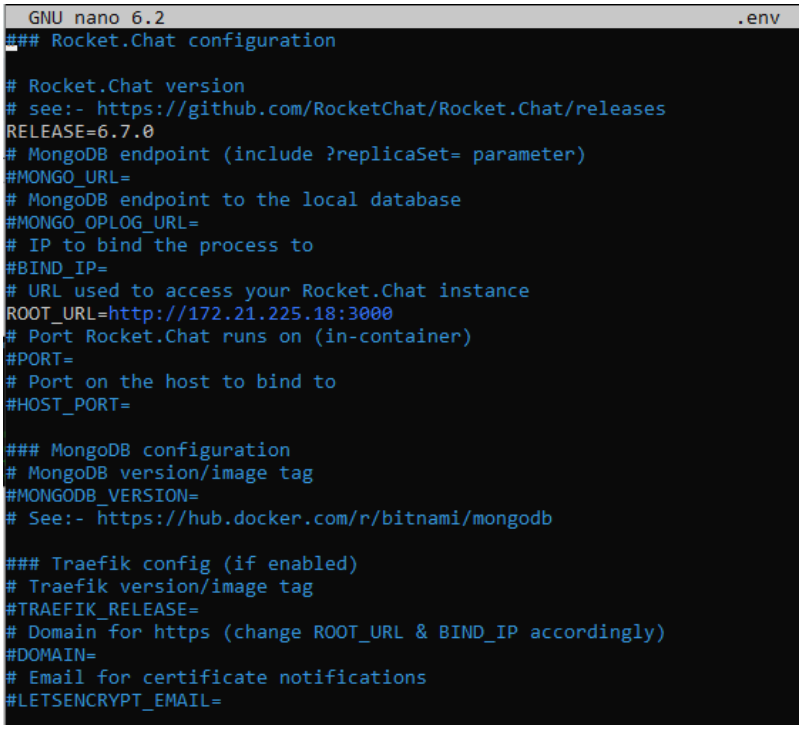
- I initiated the download of the file 'compose.yml' on Ubuntu using WSL 2, executing the command provided in the documentation:

```
10  curl -L https://raw.githubusercontent.com/RocketChat/Docker.Official.Image/master/compose.yml -O
11  ls
```

- I used the following command to download the latest version of RocketChat image via Docker:

```
gregorytrovao@DESKTOP-VKODANU:~$ docker pull registry.rocket.chat/rocketchat/rocket.chat:latest
latest: Pulling from rocketchat/rocket.chat
26c5c85e47da: Pull complete
96da4c1974ec: Pull complete
286584c9c618: Pull complete
ec51043fad6b: Pull complete
10845595c672: Pull complete
23b3c9ae79f3: Pull complete
e8711a648170: Pull complete
ac96427e2f3a: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:2b45c666aafcf09fe6c83861303eaeab5fcd3850a39a30af4b4481c9f07febdc
Status: Downloaded newer image for registry.rocket.chat/rocketchat/rocket.chat:latest
registry.rocket.chat/rocketchat/rocket.chat:latest
```

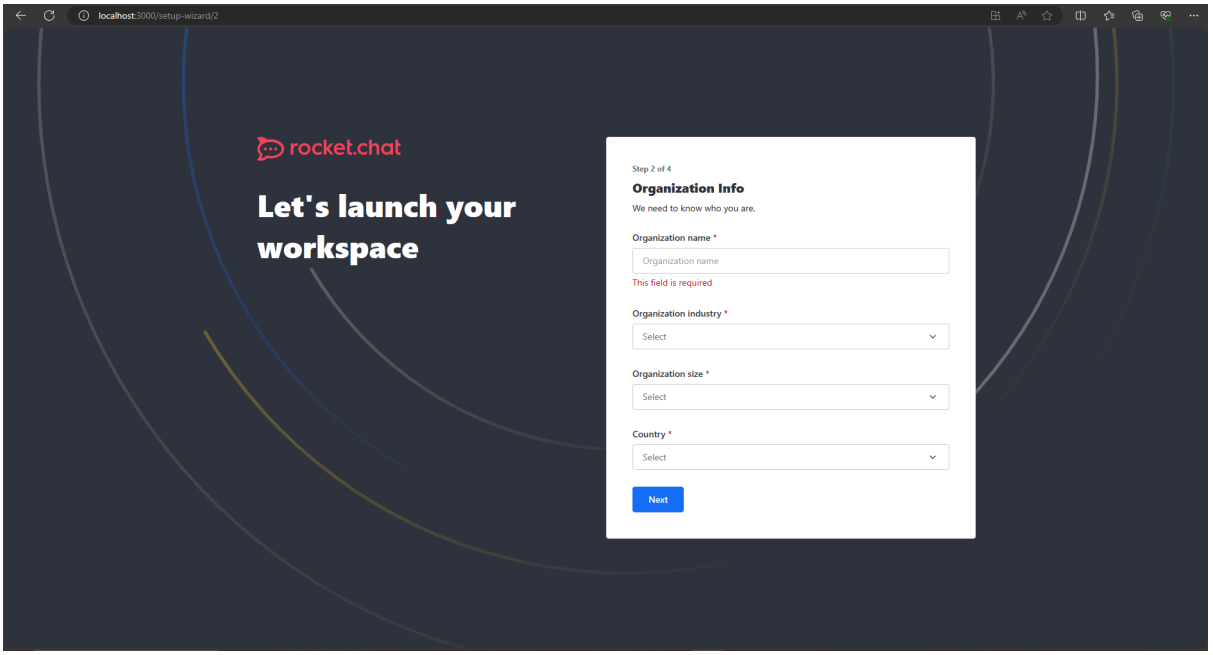
- Following the documentation I used the “nano” command to edit the file, implementing the required changes.



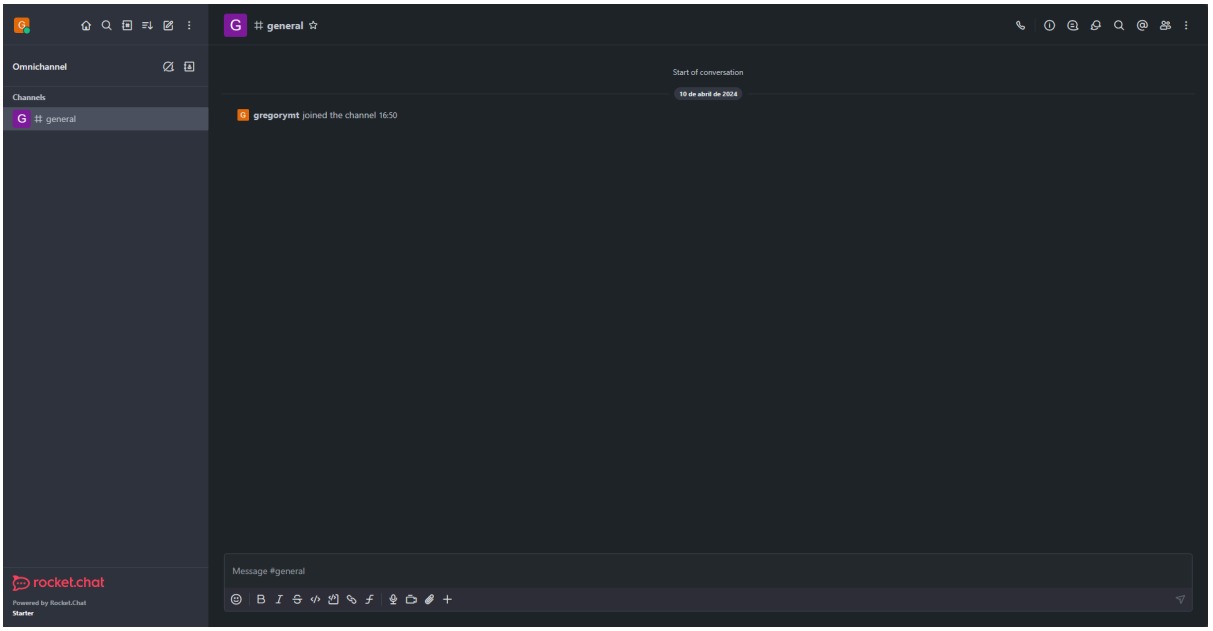
- I initiated the container using Docker Compose:

```
gregorytrovao@DESKTOP-VKODANU:~$ docker compose up -d
[+] Running 3/3
 ⠿ rocketchat Pulled                                                    3.2s
 ⠿ mongodb Pulled                                                      41.6s
   ⠿ 6754bb5cae91 Pull complete                                        23.5s
[+] Running 4/4
 ⠿ Network gregorytrovao_default          Created                       0.1s
 ⠿ Volume "gregorytrovao_mongodb_data"    Created                       0.0s
 ⠿ Container gregorytrovao-mongodb-1      Started                       3.1s
 ⠿ Container gregorytrovao-rocketchat-1   Started                       0.0s
```

- I accessed the link specified in the compose.yml to conduct a test, confirming its proper functionality:

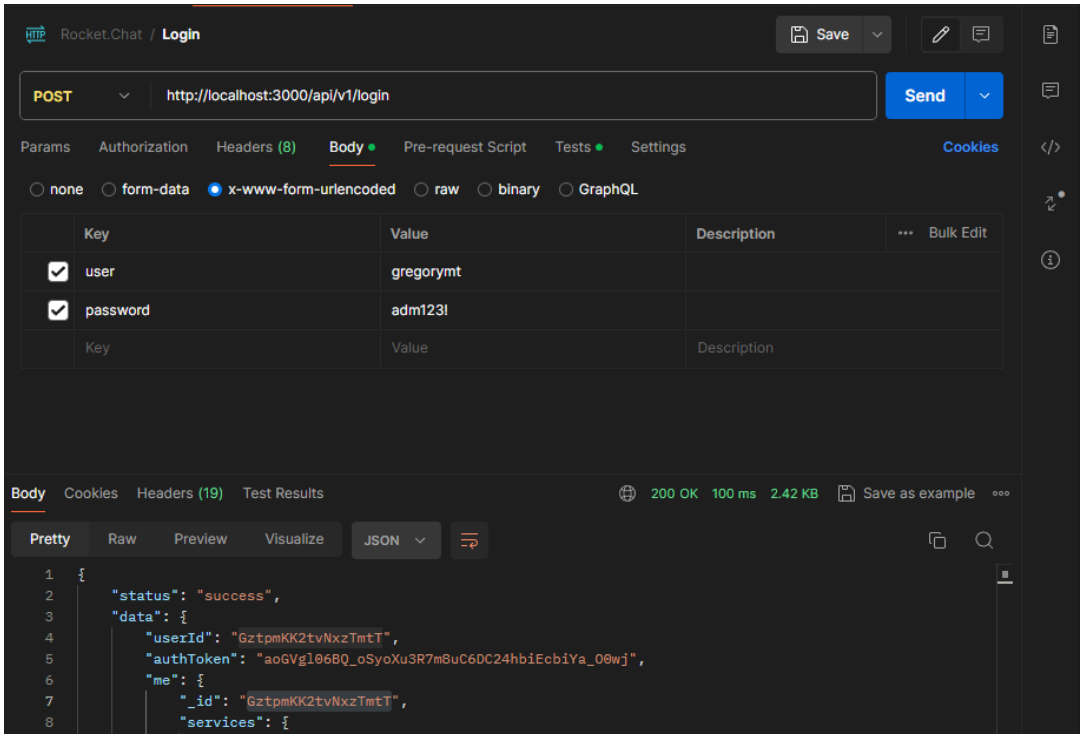


- I proceeded to register and subsequently logged in as an admin:



Subsequently, I commenced the execution of the three tasks outlined in the challenge, utilizing Postman.

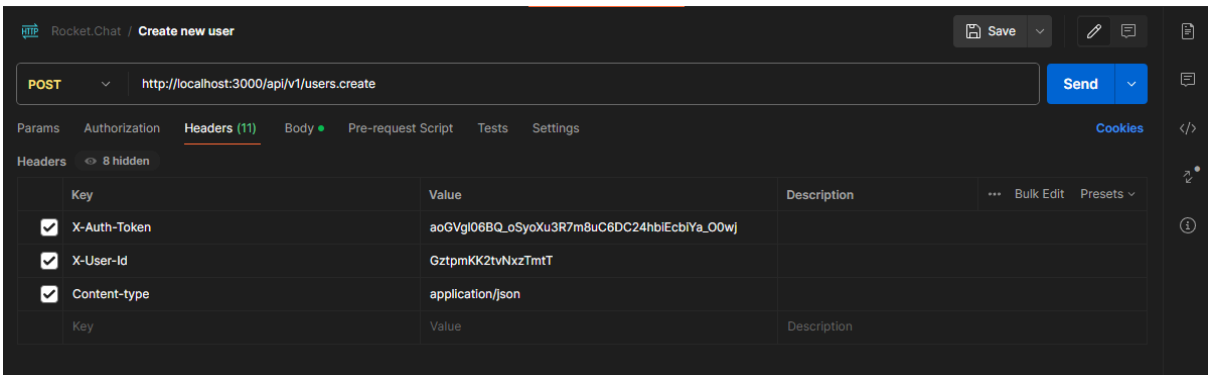
- **Create a new user via an API endpoint:**

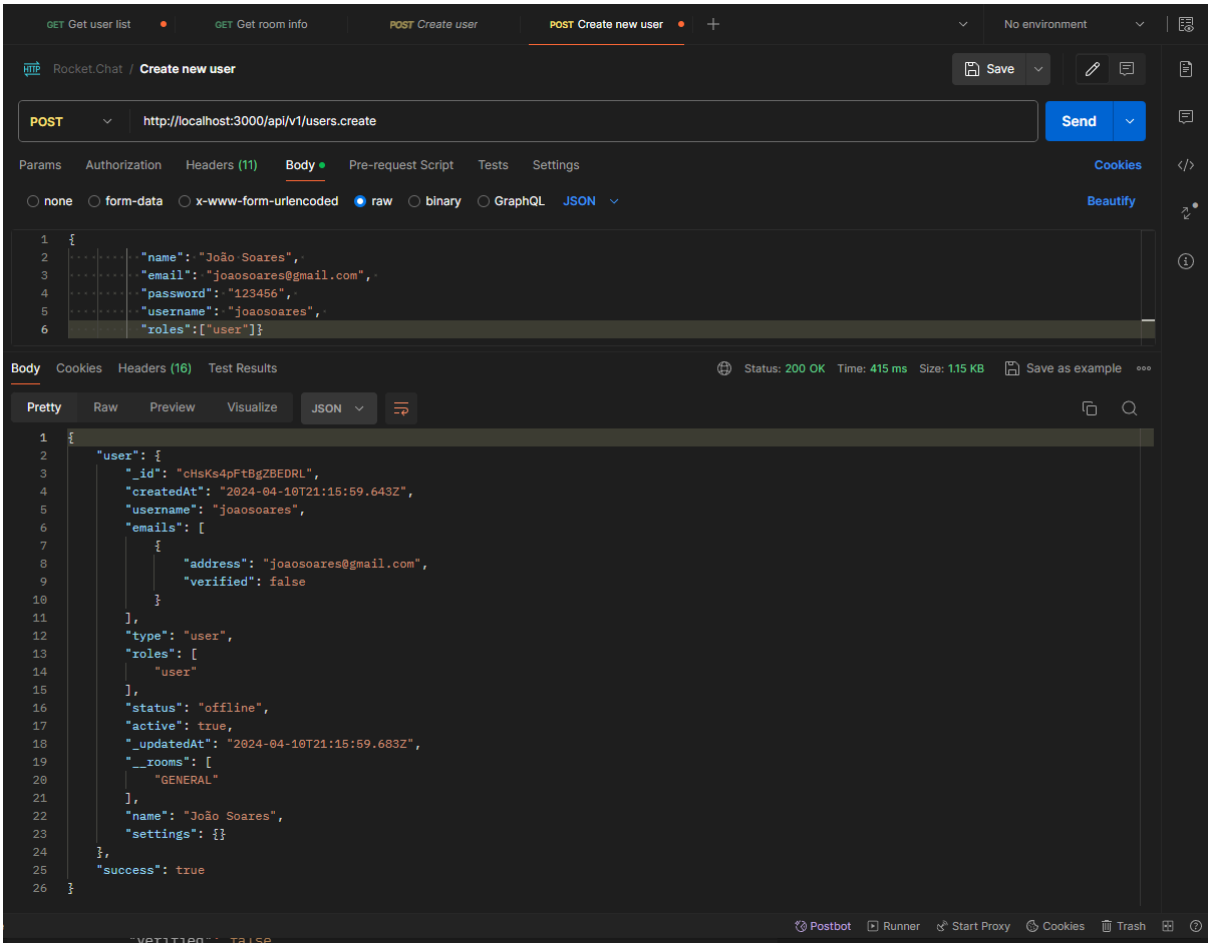


- I observed that the documentation specified authentication requirements for user creation. Therefore, I used the 'Login with Username and Password' API endpoint to obtain my userID and authToken, then utilizing them to create a new user.

- Login Function used:

```
curl http://localhost:3000/api/v1/login \
    -d "user=gregorymt&password=adm123!"
```





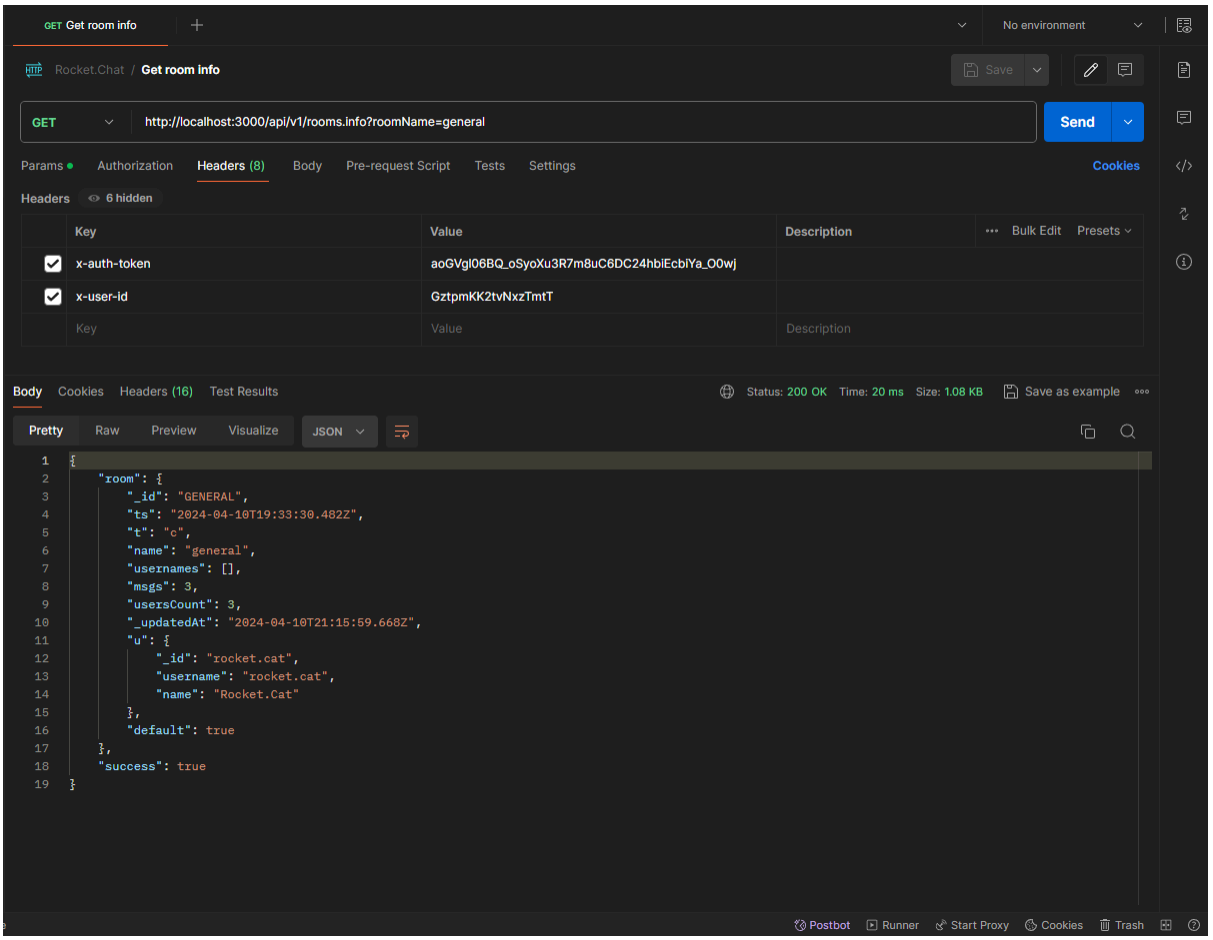
- After obtaining my token and ID, I utilized the 'create user' command as provided in the documentation, modifying the user information:

```
curl -H "X-Auth-Token: 9HqLlyZOugoStsXCUfD_0YdwnNnunAJF8V47U3QHXSq" \
    -H "X-User-Id: aobEdbYhXfu5hkeqG" \
    -H "Content-type:application/json" \
    http://localhost:3000/api/v1/users.create \
    -d '{
        "name": "name",
        "email": "email@user.tld",
        "password": "anypassyouwant",
        "username": "uniqueusername",
        "roles":["bot","user"]}'
```

- **Get the room information via an API endpoint:**

- I followed the command in the documentation but chose to utilize the roomName instead of the roomID, referencing the name of the room created during the deployment of the local instance.:

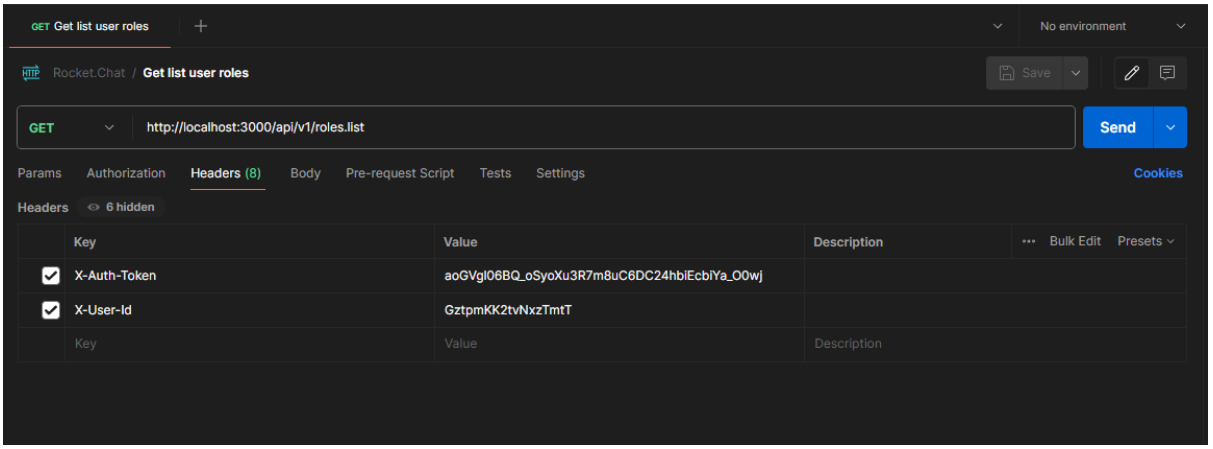
```
curl -H "X-Auth-Token: 9HqLlyZOugoStsXCUfD_0YdwnNnunAJF8V47U3QHXSq" \
    -H "X-User-Id: aobEdbYhXfu5hkeqG" \
    http://localhost:3000/api/v1/rooms.info?roomName=general
```

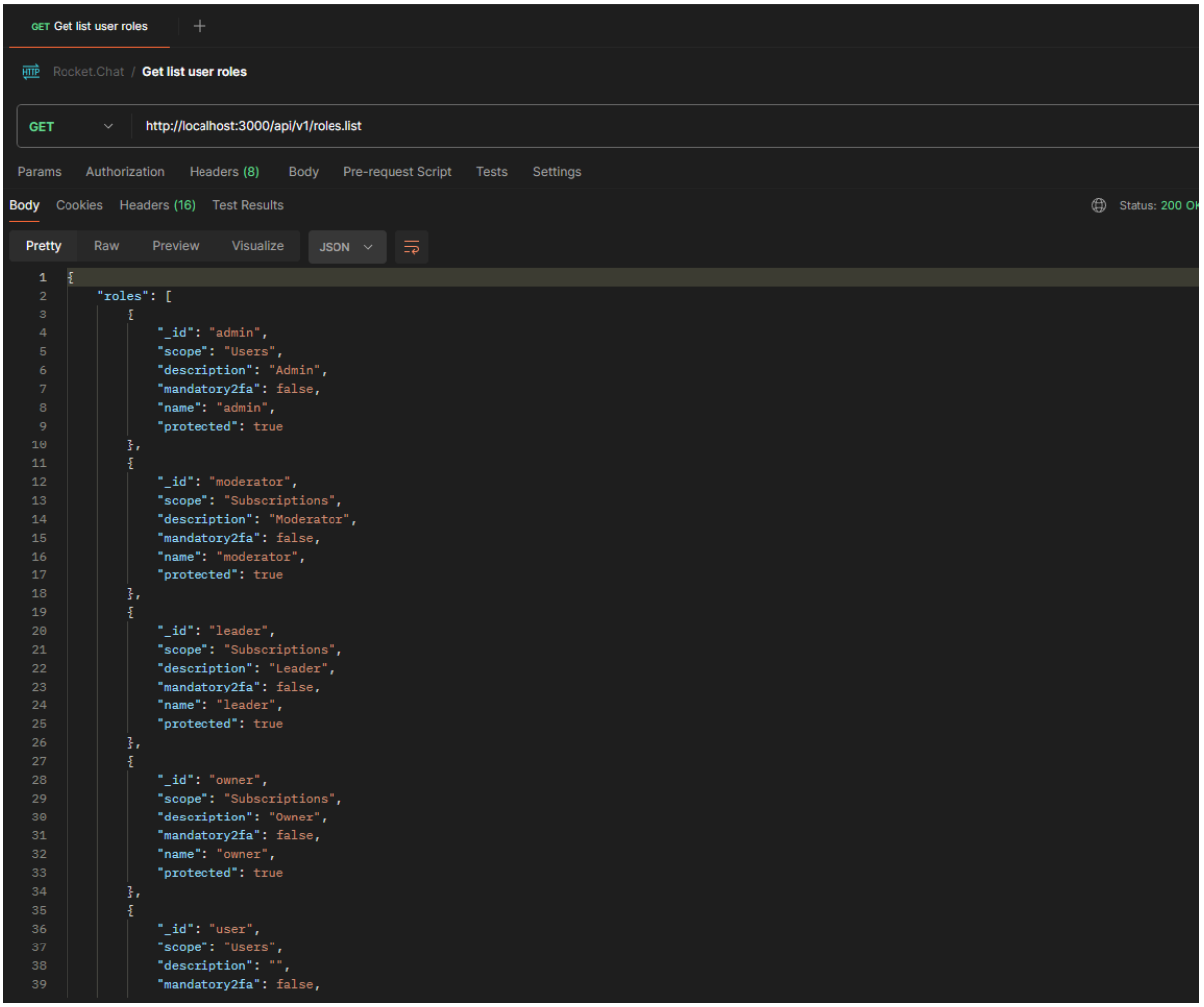


- **Get a list of all user roles in the system via an API endpoint:**

- I used the List Roles Endpoint available on the documentation, just changing to my authToken and userID:

```
curl -H "X-Auth-Token: 9HqLlyZOugoStsXCUfD_0YdwnNnunAJF8V47U3QHXSq" \
    -H "X-User-Id: aobEdbYhXfu5hkeqG" \
    http://localhost:3000/api/v1/roles.list
```



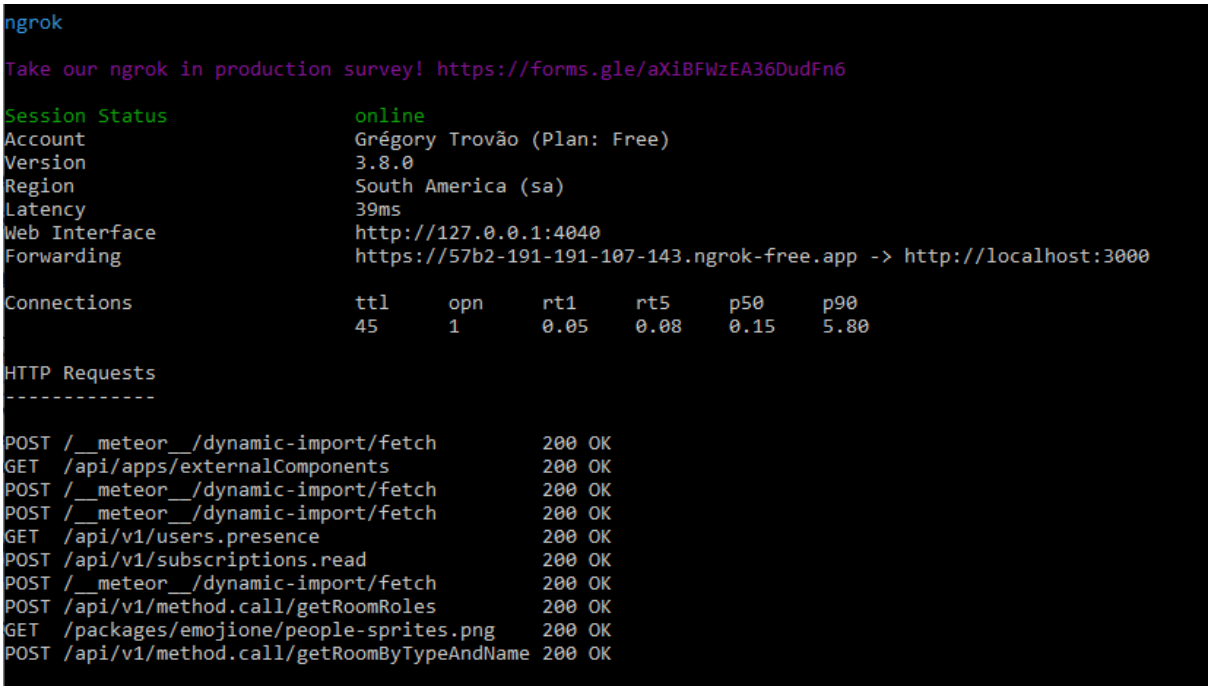


- **Optional: Feel free to go a little crazy and integrate your Rocket.Chat server whatever tool, platform, or service you see fit, ok? Please, amaze us :)**

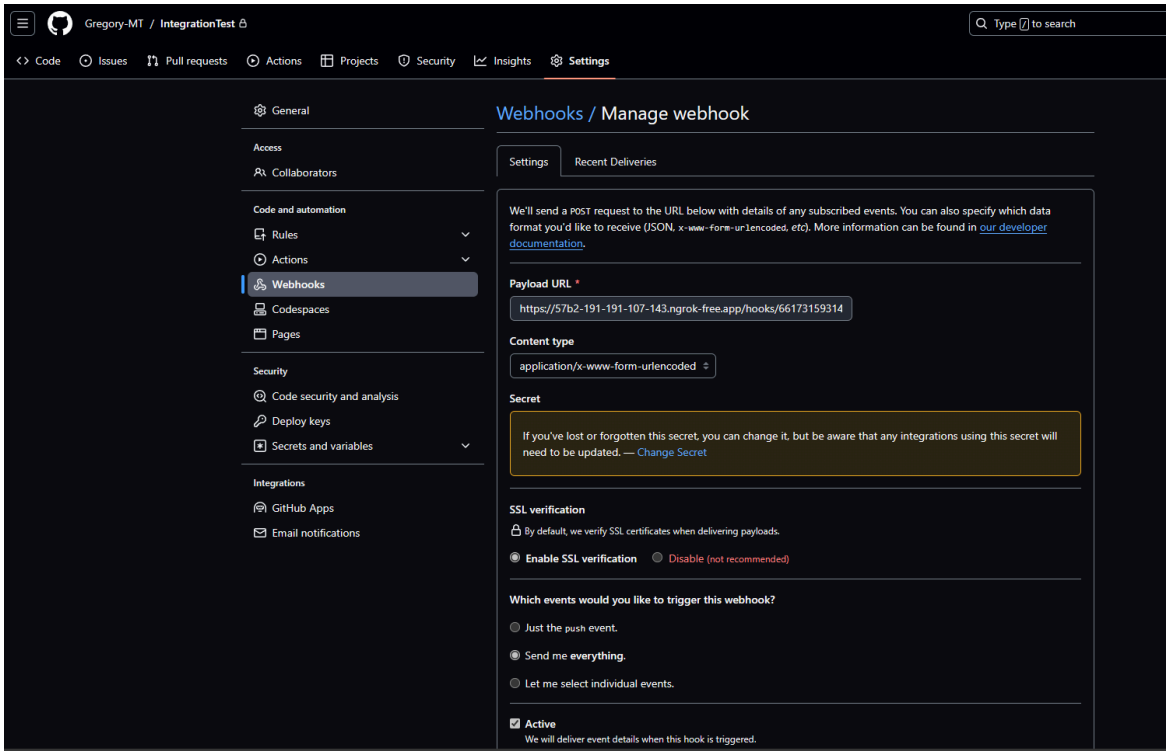
- I created a new channel through Postman to test the integration, changing the “channelname” to “Testes”:

```
curl -H "X-Auth-Token: 9HqLlyZOugoStsXCUfD_0YdwnNnunAJF8V47U3QHXSq" \
    -H "X-User-Id: aobEdbYhXfu5hkeqG" \
    -H "Content-type: application/json" \
    https://localhost:3000/api/v1/channels.create \
    -d '{
        "name": "channelname" }'
```

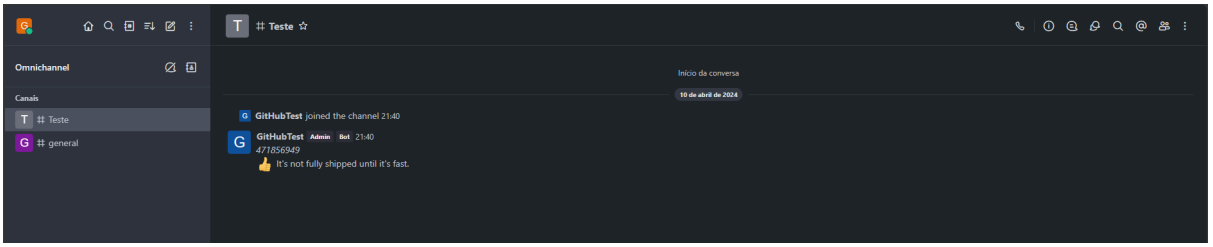
- Using Ngrok I exposed my localhost to external access, then I used the documentation to integrate a webhook from GitHub:



- Following the documentation I created a new incoming integration with the first example script, generating the URL and the Token, then I used them in the GitHub WebHook from the repository I created for this Challenge Test:



- After saving it, I received a message on the “Teste” channel, validating the integration



- I exported the Postman collection to a folder as well as this document and used Git to make the first commit to the GitHub repository