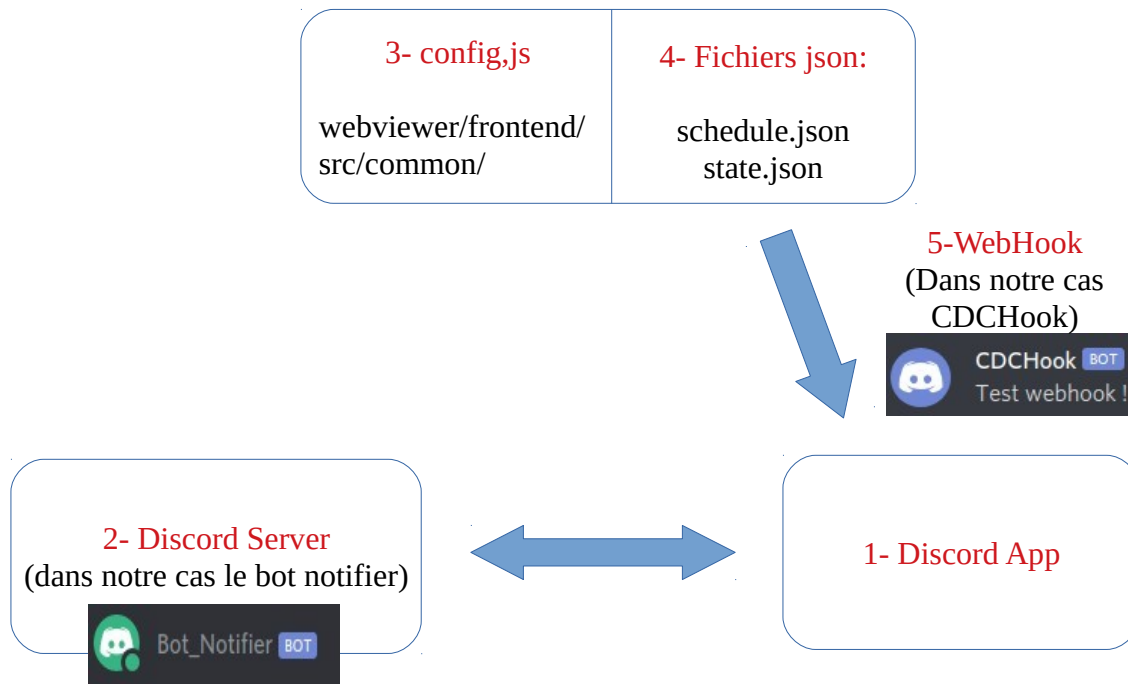
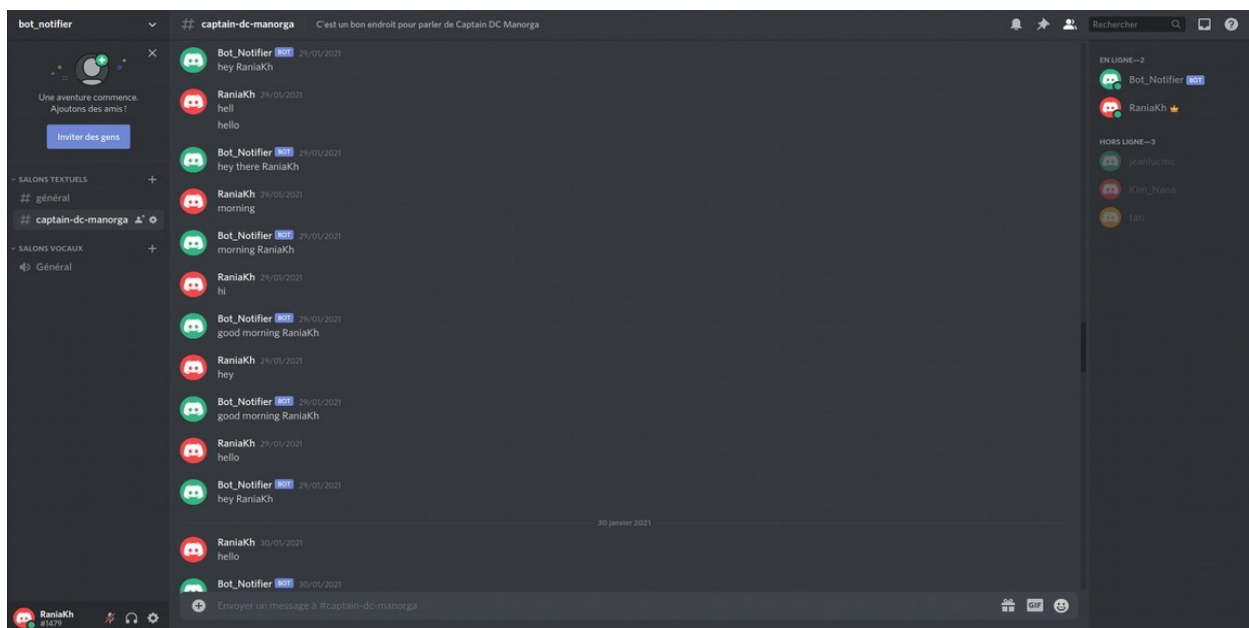


I. Architecture :



1- Discord Application : c'est l'interface où on discute



2- Discord Server :

ce serveur est écrit avec l'éditeur **repl.it** qui est un IDE en ligne.

Avantage de repl.it : permet d'avoir **une adresse ip public** (serveur hébergé gratuitement) .

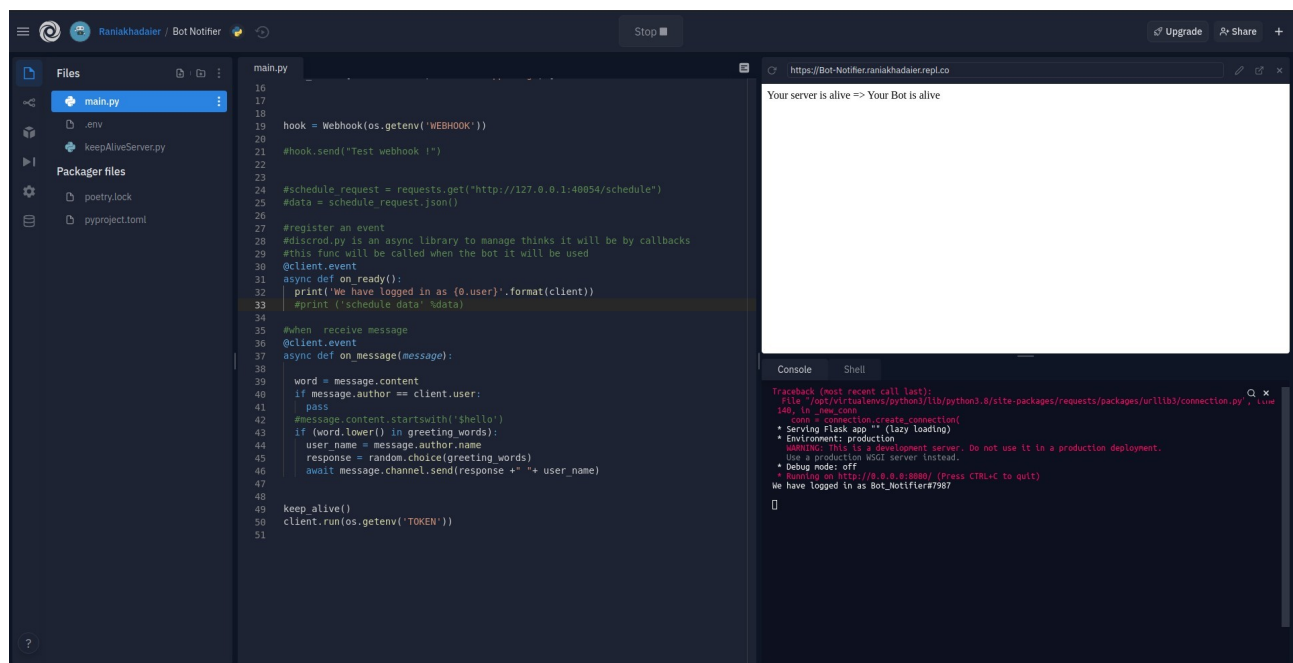
Cette adresse est utilisée dans **uptimerobot.com**

uptimerobot.com va nous servir à maintenir notre bot disponible 24/24 (avec un refresh chaque 5 minutes). En utilisant ces deux outils mentionnés on garantit la disponibilité du Bot Notifier.

Espace de repl.it : <https://repl.it/@Raniakhadaier/Bot-Notifier#main.py>

UptimeRobot.com : <https://uptimerobot.com/>

Espace repl.it :

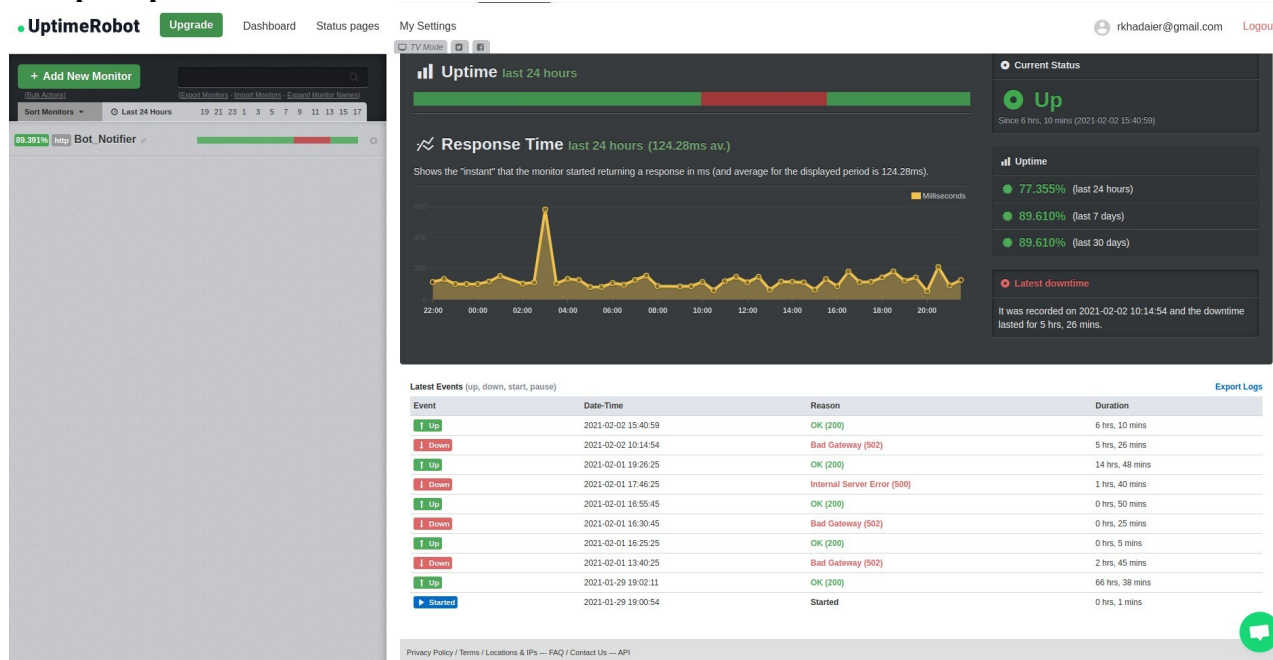


The screenshot displays the repl.it web interface. On the left, a file explorer shows a project named 'Bot Notifier' with files like 'main.py', '.env', 'keepAliveServer.py', 'poetry.lock', and 'pyproject.toml'. The main editor area shows the content of 'main.py', a Python script for a Discord bot using discord.py and requests. The script includes a webhook for uptime monitoring and a message handler that responds with a random greeting. On the right, a preview window shows the URL 'https://Bot-Notifier.raniakhadaier.repl.co' and a message 'Your server is alive -> Your Bot is alive'. Below the preview, a console window shows the output of the script, including a traceback for a connection error and a confirmation that the bot is running on http://0.0.0.0:8080/.

```
16
17
18
19 hook = Webhook(os.getenv('WEBHOOK'))
20
21 #hook.send("Test webhook !")
22
23
24 #schedule_request = requests.get("http://127.0.0.1:40854/schedule")
25 #data = schedule_request.json()
26
27 #register an event
28 #discord.py is an async library to manage things it will be by callbacks
29 #this func will be called when the bot it will be used
30 @client.event
31 async def on_ready():
32     print('We have logged in as {0.user}'.format(client))
33     #print('Schedule data: %data')
34
35 #when receive message
36 @client.event
37 async def on_message(message):
38
39     word = message.content
40     if message.author == client.user:
41         pass
42     #message.content.startswith('hello')
43     if (word.lower() in greeting_words):
44         user_name = message.author.name
45         response = random.choice(greeting_words)
46         await message.channel.send(response + " " + user_name)
47
48
49 keep_alive()
50 client.run(os.getenv('TOKEN'))
51
```

Traceback (most recent call last):
File "/opt/virtualenvs/python3.8/site-packages/requests/packages/urllib3/connection.py", line 140, in _new_conn
conn = connection.create_connection(
* Serving Flask app " (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
We have logged in as Bot_Notifier#7967

Compte UptimeRobot :



3- Fichier config.js :

Ce fichier sert à récupérer les seuils de températures , humidité et disque.

Ce fichier se trouve sous le répertoire : « webviewer/frontend/src/common/config.js »

4- Fichiers JSON :

Le Rest Server stocke les résultats de ses requêtes dans des fichiers JSON. Dans notre cas , on va utiliser deux fichiers.

schedule.json : afin de récupérer les dates de début/fin de l'événement.

state.json : afin de récupérer : le status du robot les valeurs de température , humidité , batterie et valeur d'espace vide de disque.

NB : on va tester sur CDC Manorga pour vérifier le choix sinon on passe à utiliser les topics ROS au lieu des fichiers JSON

5- Webhook :

qui se produit lorsque quelque chose se produit côté serveur ; une simple notification d'événement via HTTP POST.

6- Structure des notifications :

*****Notification de début d'événement*****

03/02/2021 09H00 => **Date et heure de début**

Failed: Next start AUTOPILOT NOW END 09H30 => **récupération de l'état du robot si Autopilot OK sinon Failed**

Failed: ROBOT STATUS AUTOPILOT

WARNING : Batterie status 0% minimum 10%

WARNING : Temperature 10.1°C bellow minimum 15 °C

WARNING : Hydrometry 51% above max value 40%

=> **récupération des valeurs de batterie/température/humidité avec leurs seuils : si les valeurs respectent les seuils donnés OK sinon WARNING**

*****Notification de fin de l'événement avec Status DOCKING*****

26/01/2021 17H30 => **Date et heure de fin**

OK: Next start AUTOPILOT NOW => **récupération de l'état du robot avant DOCKING si Autopilot OK sinon Failed**

OK: ROBOT STATUS changed to DOCKING => **récupération de l'état du robot si DOCKING OK sinon Failed**

WARNING : Batterie status 0% below minimum 10%

WARNING : Temperature 8.1°C bellow minimum 15 °C

WARNING : Hydrometrie 47% above max value 40%

=> **récupération des valeurs de batterie/température/humidité avec leurs seuils : si les valeurs respectent les seuils donnés OK sinon WARNING**

*****Notification avec le status DOCKED*****

26/01/2021 17H30 => **Date et heure où le robot est bien positionné sur la base de recharge**

Next start AUTOPILOT 27/01/2021 09H30 => **indiquer le prochain événement**

OK: ROBOT STATUS changed to DOCKED => **récupération de l'état du robot si DOCKED OK sinon Failed**

OK : Batterie status 100%

WARNING : Temperature 8.2°C bellow minimum 15 °C

WARNING : Hydrometrie 47% above max value 40%

=> **récupération des valeurs de batterie/température/humidité avec leurs seuils : si les valeurs respectent les seuils donnés OK sinon WARNING**

II. Exigences :

1- Installation de **Python 3.5** comme version minimale pour le fonctionnement de Webhook discord:

<https://www.python.org/downloads/release/python-3510/>

<https://tecadmin.net/install-python-3-5-on-ubuntu/>

2- Installation de **dhooks** pour interagir avec les webhooks Discord à l'aide de python :

<https://pypi.org/project/dhooks/>

3-Installation de **Threading** pour faire l'appel à la fonction d'envoi des notifications d'une façon répétitive avec un timer pour ne pas rater aucun événement de schedule :

<https://pypi.org/project/threaded/#description>

III. Sprint 26 Avril 2021 – 10 Mai 2021 :

User Stories :

1-En tant que User je veux recevoir la notification de début d'un event : **TO VERIFY=> Test sur CDC Manorga**

2-En tant que User je veux recevoir la notification de fin de l'event avec state DOCKING : **TO VERIFY=> Test sur CDC Manorga**

3-En tant que User je veux recevoir la notification de fin de l'event avec state DOCKED

4-En tant que User je veux savoir dans chaque notification les seuils de Batterie /Température /Humidité /Disque

IV. Next Step :

Intégration de tensorflow/dialogflow avec Bot de Discord afin d'avoir un chatbot intelligent qui peut interagir avec les membres de la chaîne Discord (A négociier)