IUT du Littoral Côte d'Opale

UE216 Compléments Informatiques

Discord MemeBot

Enseignant :
Antoine Gamelin
Département Informatique

Groupe :
Isaac Ruchalski
Guillaume Le Louarn
Théo Helary

Table des matières

Table des figures

1 Introduction

L'étude qui va suivre analysera le produit final de notre projet, un bot Discord. Il permet à l'utilisateur de récupérer des Memes (macro d'image accompagné d'un slogan humoristique), et de les afficher dans une conversation Discord avec un texte personnalisé.

2 Présentation des outils utilisés

2.1 Discord

Discord est un logiciel gratuit de voix sur IP (VoIP) et de messagerie instantanée. Comptant plus de 250 millions d'utilisateurs en 2019, Discord est devenu une plateforme collaborative assez fournie où les utilisateurs peuvent rejoindre des groupes d'utilisateurs partageant les mêmes points communs.

Figure 1 – Logo de Discord

2.2 Discord Developer Portal

Discord Developer Portal est un portail Discord accessible depuis leur site Internet.

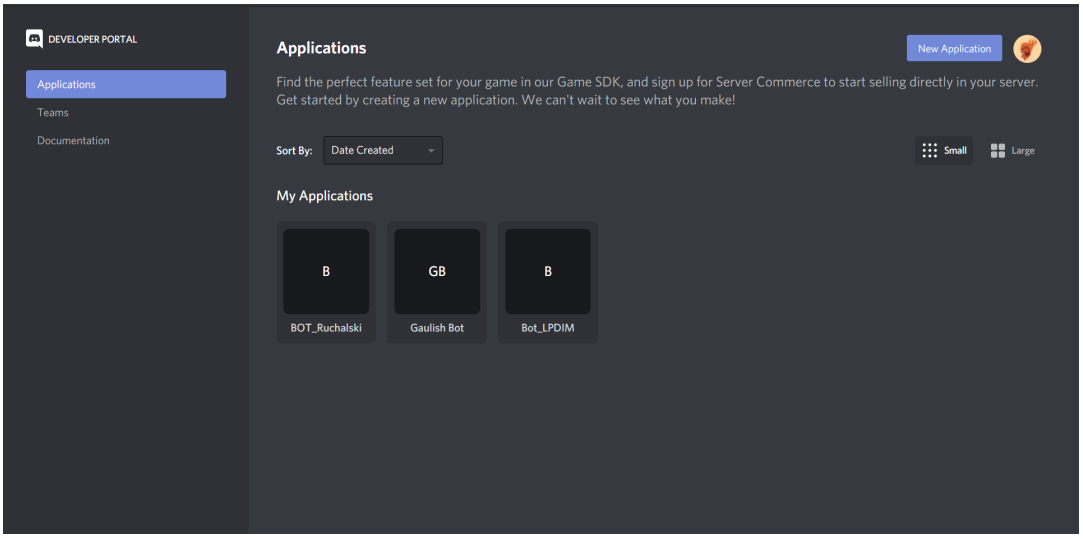

Figure 2 – Accueil de Discord Developer Portal

Discord Developer Portal permet de créer des applications en tout genre utilisant Discord (API, bots, webhooks, etc).

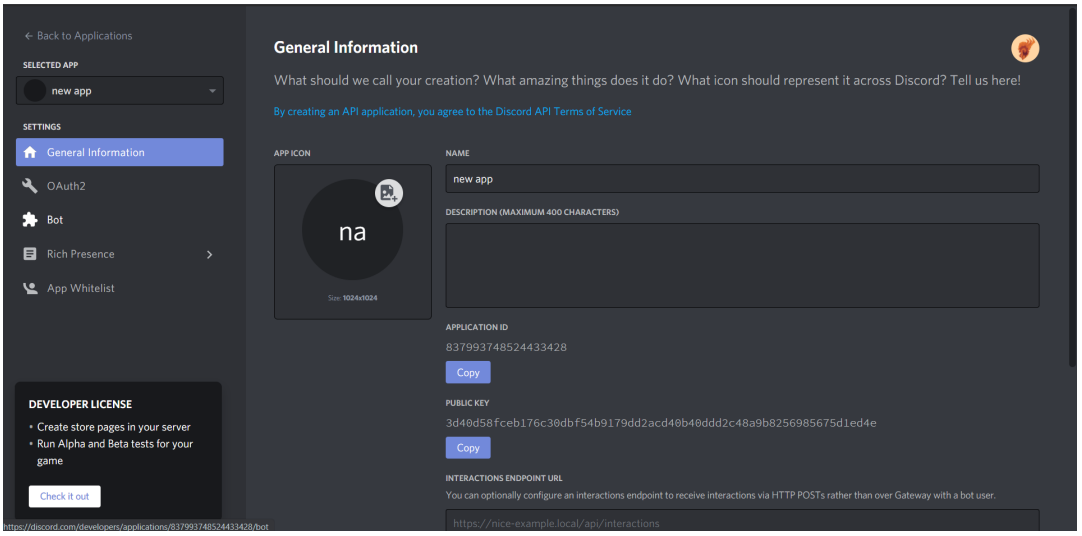

Figure 3 – Création d'une nouvelle application sur le portail

2.3 node.js

node.js est un environnement d'exécution Javascript orienté serveur. Celui-ci nous permet donc nativement d'exécuter du code Javascript hors navigateur, et donc de le run dans un serveur (ou node).

Figure 4 – Logo de node.js

2.4 discord.js

discord.js est un module node.js qui permet une communication avec l'API discord (Discord Developer Portal).

2.5 Imgflip API

Imgflip API est une API REST génératrice de memes, basée sur le site Imgflip.

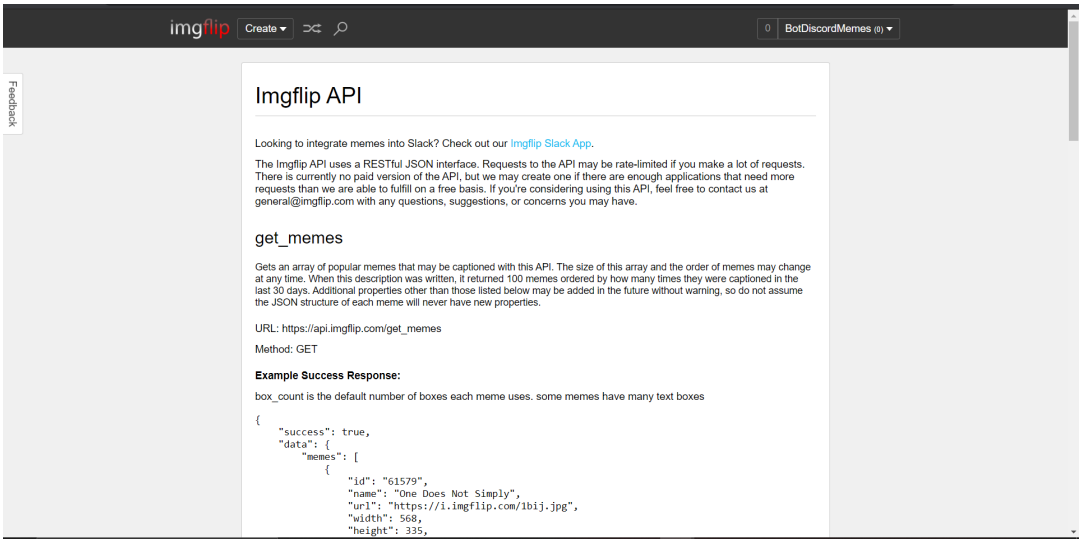

Figure 5 – Page d'Imgflip API

2.6 node-rest-client

node-rest-client est un package node.js qui permet une connexion à n'importe quelle API REST et une valeur de retour en objet JS (JSON).

3 Travail réalisé

3.1 Création d'un espace appli/bot

Comme dit précédemment, la création d'une application sur le DDP (Discord Developer Portal) permet la création d'un **bot**.
Ma première mission a donc consisté en la **création d'une application sur le DDP**

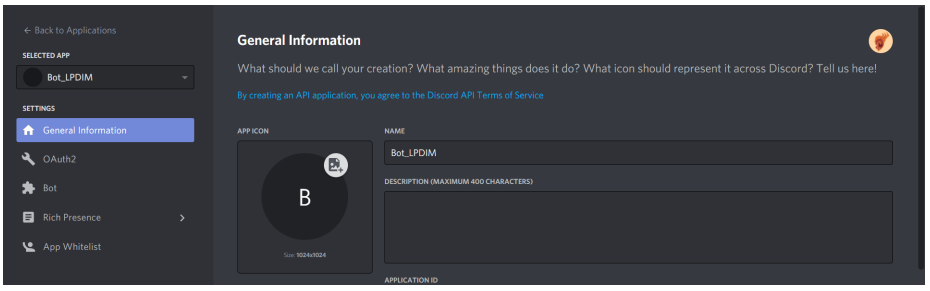

Figure 6 – Création d'une application pour le bot

Puis, une fois cette application créée, en la **création du bot**. Ces informations lui permettent d'être nommé, avoir une image de profil, exactement comme un utilisateur discord standard.

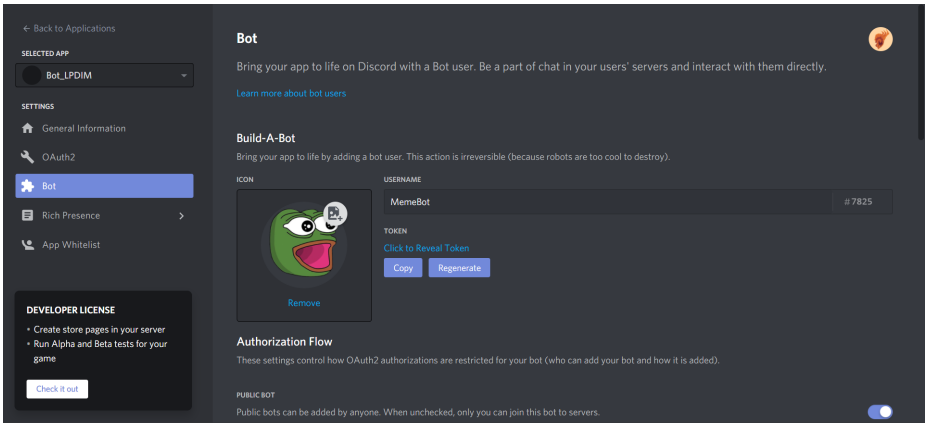

Figure 7 – Création du bot

l'API Discord utilise OAuth2 pour permettre à ses applications d'être authentifiées. Il faut donc fournir des paramètres d'authentification à l'application. Dans mon cas, je souhaite que, lorsque appelée, l'application n'ait que les droits "bot", c'est-à-dire que quiconque aura le lien vers mon application ne pourra que l'ajouter en tant que bot sur un serveur Discord de son choix.

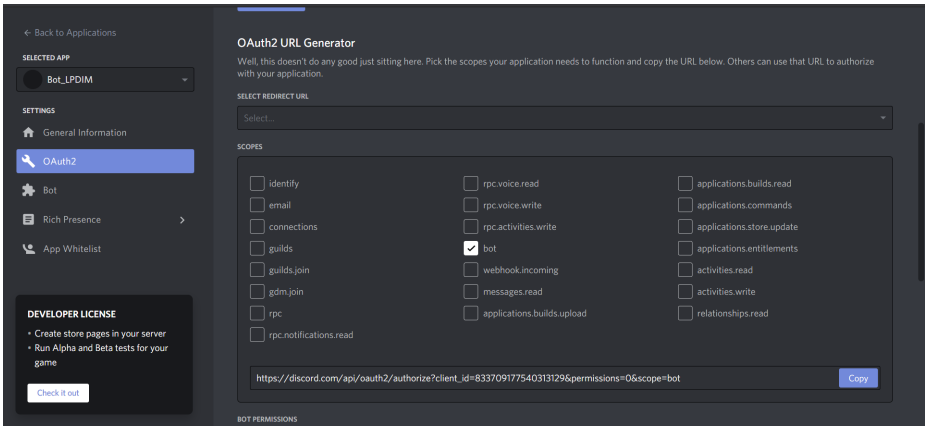

Figure 8 – Création du lien OAuth2

En utilisant le lien généré, n'importe qui peut donc ajouter le bot sur le serveur discord qu'il souhaite. Des informations complémentaires sur l'utilisation du bot sont également disponibles.

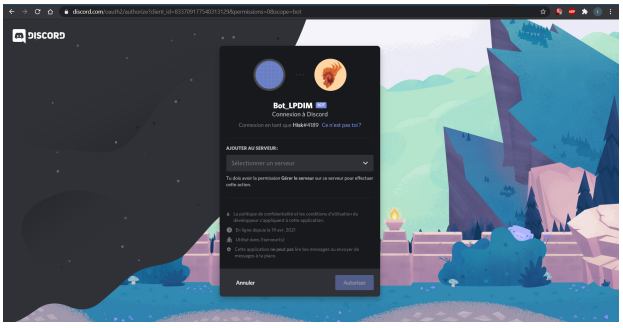

Figure 9 – Déploiement grâce au lien OAuth2

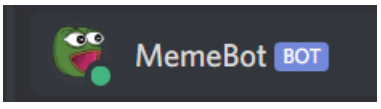

Figure 10 – Aspect du bot sur un serveur Discord

3.2 Création du projet node.js

Pour créer le projet node.js, j'ai utilisé la méthode standard pour la création d'un projet node.js avec une variation.

avant d'initialiser le projet, j'ai **créé un fichier** config.json, dans le quel j'ai **spécifié le token de mon bot**, trouvé sur le portail :

```
{ "BOT_TOKEN": "MON TOKEN" }
```

ensuite, il faut initialiser le projet node.js

```
npm init
```

et ensuite **installer discord.js**

```
npm install discord.js
```

Maintenant que discord.js est installé, il est possible d'utiliser ses éléments.

3.3 Détection des messages

Pour que le bot détecte des messages, la fonction client.on() permet de récupérer un message envoyé par un utilisateur.Cependant, cela pose des problèmes de logique : à chaque message envoyé,le bot va devoir effectuer un test pour voir si le message lui était destiné.

Il faut donc utiliser un signe qui ferait office d'ajout sémantique. Dans notre cas, nous utilisons un préfixe, le point d'exclamation " !". à chaque message, si le contenu du message commence par " !", alors cela fait référence au bot. Si le contenu ne commence pas par ce préfixe, alors les interactions avec le bot n'iront pas plus loin. Le préfixe est facilement modifiable dans le code par un autre signe en cas d'incompatibilité avec d'autres bots.

```
const prefix = "!"; //préfixe choisi : !

client.on("message", function (message) {
  // Initialisation après réception d'un message
  var erreur = true;
  if (message.author.bot) return;
  if (!message.content.startsWith(prefix)){
```

Figure 11 – Gestion du préfixe

Il a été décidé ensuite que notre bot aurait 3 fonctionnalités :

— récupérer des memes depuis l'API imgFlip
— créer un meme et l'envoyer dans le salon discord
— afficher un panel d'aide

Mon travail sur le bot a principalement porté sur la conception de la fonctionnalité de **récupération de memes**.

3.4 Récupération des memes

La fonction de récupération de memes s'appelle getMemes.

l'API Imgflip propose une URL, https://api.imgflip.com/get_memes.
Quand, en mode CRUD, la méthode GET est exécutée sur cette adresse URL, nous obtenons les 100 templates de memes les plus populaires sur imgflip.com des 30 derniers jours. Cela permet donc de varier les memes en fonction de la popularité de ceux-ci et donc d'alimenter constamment le bot avec de nouveaux memes.


```
{
   "success":true,
   "data":{
      "memes":[
         {
            "id":"181913649",
            "name":"Drake Hotline Bling",
            "url":"https:\/\/i.imgflip.com\/30b1gx.jpg",
            "width":1200,
            "height":1200,
            "box_count":2
         },
         {
            "id":"112126428",
            "name":"Distracted Boyfriend",
            "url":"https:\/\/i.imgflip.com\/1ur9b0.jpg",
            "width":1200,
            "height":800,
            "box_count":3
         },
```


Figure 12 – Retour de l'API en méthode GET

Une fois la méthode GET effectuée, mon bot a donc récupéré les memes. Pour les afficher, je dois donc détecter que le message de l'utilisateur correspond au préfixe suivi de la commande "getMemes".

```
//Fonction pour obtenir tous les memes
if (command === "getMemes") {
  var Client = require("node-rest-client").Client; //Génération d'un client note-rest-client
  var cli = new Client();

  //on get les memes
  cli.get("https://api.imgflip.com/get_memes", function (data, response) {
    var memes;
    var memesEmbed = [];

    for (var index in data) {
      memes = data[index].memes;
    }

    for (var key in memes) {
      var meme = memes[key];

      /*Je push les memes dans un objet JSON , le format utilisable par un Embed discord :

      "name": un nom,
      "value": une valeur

      */
      memesEmbed.push({
        name: meme.name,
        value: meme.url + "\n" + meme.id,
      });
    }

    //Envoi d'un embed
    message.channel.send(makeEmbedMemes("List of memes", memesEmbed));
  });

  //Pas d'erreur
  erreur = false;
}
```

Figure 13 – code de !getMemes

Les memes sont ensuite formattés en suivant le format des Embed discord.

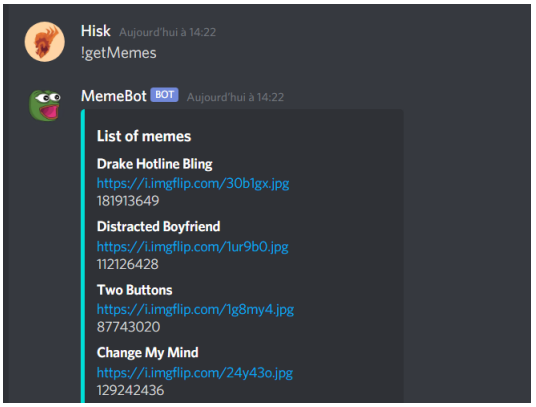

Figure 14 – Réponse du bot

4 Problèmes rencontrés

L'un des problèmes majeurs que j'ai rencontrés, a été le fait que les Embed discord, ce que le bot renvoie lors de l'exécution de !getMemes, ne peuvent contenir que 6000 caractères maximum.

avec Imgflip, un template meme est composé d'une multitude d'éléments, mais nous n'en avons gardé que 3 :

- un id
- un nom
- une URL

Mon idée au début était de n'afficher dans l'Embed que le nom, et l'URL du meme (l'URL permettant de visionner le template). Mais cela pose un problème majeur : tout le monde ne connaît pas forcément le nom complet d'un meme. De plus, si le bot utilise le nom d'un meme au lieu de son id, cela peut apporter des erreurs si jamais le nom d'un meme comporte un symbole similaire au préfixe ou au passage d'arguments de notre fonction !makeMeme &id &args.

Il faut donc afficher l'id également lors de l'affichage des templates de memes. Cela apporte un problème de débordement de texte sur l'embed.

même si nous récupérons donc 100 templates de memes avec l'API Imgflip, seulement 25 apparaitront dans le message du bot, les 25 les plus populaires. J'ai essayé de faire un système de "pagination", ou envoyer :

!getMemes 2

renverrait les memes 25 à 50 , !getMemes 3 les memes 50 à 75 , ...

Mais ce n'est pas concluant, car les Embed n'aiment pas avoir un contenu dynamiquement généré. Malheureusement, le bot n'affichera donc que les 25 premiers templates de memes. Il est toujours possible pour un utilisateur avancé d'utiliser cette page pour récupérer les id des 100 memes :

https://api.imgflip.com/get_memes

5 Conclusion

Ayant déjà développé des bots discord par le passé, ce complément d'information node.js/discord.js était une révision. Apprenant des langues par mes propres moyens, j'ai déjà conçu un bot permettant de m'entraîner avec un système de question réponses et d'aide à la demande, avec pour projet de mettre en place une traduction directe en français et/ou anglais.

Néanmoins, j'ai tout de même appris sur les embeds, ces composants discord.js, leurs limites, et leurs utilisations.

J'ai cependant énormément appris sur l'utilisation du LaTex et ses possibilités.