Walid Boulaghzalate
Yann Faussette

ISN

Informatique
et Sciences
du Numerique

Année 2017-2018

Sommaire

Introduction

Un ChatBot est un robot sous forme de software qui à pour tâche de dialoguer avec un individu via un service de conversations comme facebook, what'sapp etc...
L'univers des chatbots s'étant de jour en jour dans le monde. Dès IA qui peuvent guider des prospects sur une plateforme de vente en ligne, jusqu'au ChatBot Facebook et What'sapp qui, finalement, auront des fonctions similaires. D'après silicon.fr « À horizon 2020, 25 % des services clients et de support dans le monde utilisent des assistants virtuels ou agents conversationnels (chatbots), contre moins de 2 % en 2017.» C'est un pari sur l'avenir que le monde des robots et des IA se lance. Un horizon de fraîcheur qui pourrait bien

aider au développement des IA qui s'améliorent d'années en années. On peut parler notamment de Sophia, un Robot Humanoïde conçu pour apprendre en s'habituant au comportement des humains à se sociabiliser.
L'objectif de notre projet était de se familiariser avec cet univers.

I)Présentation des infrastructures lié à notre projet

1- Discord

Discord est un logiciel multiplateforme gratuit de VoIP (un protocole qui permet la transmission de flux audio, vidéo). C'est une plateforme plutôt jeune qui date de 2015. Elle a connu divers problèmes mais continue d'exister. Son but est d'éviter de disposer d'une armée de plateforme de discussion comme skype, teamspeak, mumble , slacks et bien d'autres.. Il regroupe les

avantages majeurs de chacun d'entre eux. Tel que les vidéos-conférence, la gestion de serveur de discussion, les partages d'écran et la fonctionnalité qui nous intéressent tout particulièrement, le chatbot. En effet, Discord, au même titre que ses compères dispose de moyen de développement simplifié de Bot. Nous verrons plus tard plus en détail comment tout cela fonctionne. Il faut néanmoins savoir qu'un Bot nécessite un développement dans un langage de programmation, il n'est pas (ou presque pas) clé en main. Dans notre cas, nous utiliserons NodeJS.

2- Node.JS

Node.JS est une plateforme logicielle libre, soit une plateforme qui supporte et accompagne le langage de programmation Javascript via un système de virtualisation. Node.JS utilise la Machine Virtuelle V8, un outil qui accélère grandement la vitesse de Javascript autrefois considéré comme extrêmement lente et gourmande, surtout sur l'aspect graphique que peut offrir ce langage.

Node.JS implémente en son sein également CommonJS, un projet de développement Javascript qui va lui permettre de s'utiliser autre part qu'à sa place initiale (Une page web). CommonJS va donc lui permettre de s'utiliser notamment en BackEnd (en arrière plan, la partie que l'utilisateur ne voit pas) comme avec NodeJS. Node.JS est considéré comme une plateforme scalable, c'est-à-dire qu'elle est prévue pour supporter des charges fortes sans problème. C'est pourquoi de grandes intreprise comme Paypal et Netflix migre vers NodeJS.

II) Explication de nos choix

1- La machine

Initialement, une première version à été développée sur Windows, une première version qui en son sein comportait de nombreuses sources externes. La deuxième version fut raccourcie, plus sobre car l'objectif du projet n'était pas tant de fournir un Bot surqualifié mais plutôt d'introduire ces notions en plein développement depuis peu. Pour cette renaissance nous avons choisi de développer notre ChatBot sous Linux, plus précisément sur Ubuntu 17.10 passé entre temps sous 18.04 LTS.

Pourquoi choisir un système Linux ?

Un chatbot est un logiciel léger, l'objectif est donc d'avoir une charge la plus faible possible pour par exemple l'héberger sur de petite plateforme tel un raspberry ou des machines virtuelles à faible performance. C'est également une des raisons de la sélection de NodeJS car son faible taux de librairies native l'allège. Ainsi, ce bot supporterait facilement une forte charge de requête sur une machine à faible performance comme un simple dualcore à moyenne fréquence avec moins d'1Go de RAM.

Néanmoins, l'utilisation d'un système Linux pose des problèmes au niveau de la gestion des émissions de flux audio que le chatbot est théoriquement capable de produire. En effet, l'émission de ce type de flux dépend de la dépendance FFMPEG et il nécessite une configuration plus poussée sur Linux sur laquelle nous ne nous sommes pas pencher.

Une fois le système installé, il nous faut installer NodeJS, ses dépendances. Une fois cela fait, nous pouvons créer notre projet et

nous installons notre dépendance à savoir discord.js. Une fois tout cela fait, c'est parti, on peut commencer la programmation.

2- Pourquoi choisir NodeJS et le Javascript?

Comme nous l'avons vu dans notre première partie, NodeJS est un programme plutôt intéressant quant à sa rapidité. Il peut supporter de grosses charges ce qui est plutôt pratique si le ChatBot est beaucoup sollicité. Nous voulions également apprendre un nouveau langage. L'avantage de celui-ci est également le fait qu'il est fait pour interagir avec des APIs (un ensemble de classes qui pourront être intégrées à une autre application).

3 – Github

Github est un logiciel de gestion de version, de projet. Nous l'avons utilisé pour centraliser notre code. Grâce à Github, notre code peut être partagé, distribué.

4 – La répartition des tâches

La répartition des tâches aura finalement été l'une de nos plus grosses épreuves lors de la réalisation de ce projet. Finalement, Walid à officier comme une sorte de Chef Projet, il s'est occupé de la réalisation de notre mémoire et de la mise en page pour notre projet en général. Nous avons initié une longue séance de Brainstorming visant à mieux avancer sur le projet. Après nos étapes de coordination, c'était Yann qui relevait la tâche de la programmation, comme technicien du projet.

III)Fonctionnement/code

```
// On dit que lorsque l'on utilise la constante Discord, on appel les fonctions de la bibliothèque discord.js issue des bibliothèques npm.
const Discord = require("discord.js");
// On dit que lorsque l'on utilise la constante bot, on appel la fonction Client.
const bot = new Discord.Client();
const erreurperm = "Erreur permission invalide"
const erreurarg = "Erreur Argument"
const PREFIX = "!";

bot.on("ready", function () {
// Le bot doit changé le "jeu" affecté au bot (voir documentation)
  bot.user.setGame("Aiiroz Bot | !help", "https://github.com/faroke/Aiirozbot")

// Etant donné que le bot à pour statue "ready", on indique dans la console qu'il est connecté.
  console.log("Aiiroz Bot - Connected");
// On demande le nombre d'utilisateurs qui ont accès au bot
  console.log("Membres : " + bot.users.size)
});
```

```
155   bot.login('NDE1ODcyMDYxNDA0MjE3MzU1.DfgNVg.CaosrIzxEiOtVBnSr7PPoqZiV0Y')
```

```
yann@Lenovo:~/AiirozBot$ node .
Aiiroz Bot - Connected
Membres : 13
(node:21771) DeprecationWarning: ClientUser#setGame: use ClientUser#setActivity
instead
```


```
7   const PREFIX = "!";
```

```
40    if (!message.content.startsWith(PREFIX)) return;
41
42    // On définit la variable args qui sera égal au message commençent par le prefix, chaque mot correspondra à une valeur de 0 à K
43    var args = message.content.substring(PREFIX.length).split (" ");
```

```
switch(expression) {
    case n:
        code block
        break;
    case n:
        code block
        break;
    default:
        code block
}
```

Le switch, c'est comme un sac à dos qui contient toutes nos commandes, nous

définissons ce que nous cherchons, à savoir les instructions commençant par args[0] soit le préfixe, soit “!”.
Les commandes que nous plaçons dans notre sac à dos des “livres” qui ont pour première de couverture “case” suivis du titre du “livre” (n) et pour quatrième de couverture “break;”. Le contenu de nos livres est donc dans “code block”.

Les avantages de ce type de forme d’écriture sont premièrement, la lisibilité pour les développeurs, la scalabilité* et donc d'alléger notre code.

Pourquoi cette forme l’allège?

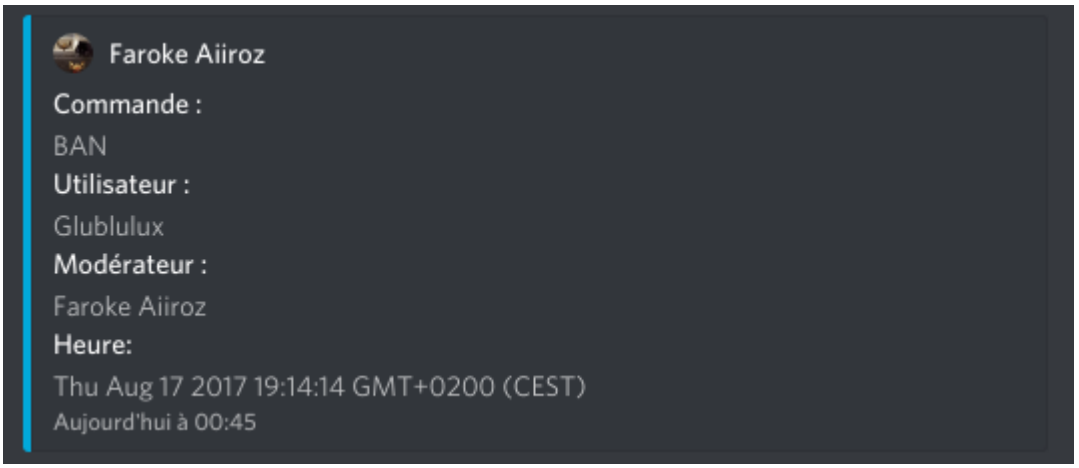
On pourrait écrire notre Chatbot à l’aide de la condition “if” en continue. Le problème étant qu’il faudrait pour chaque message que le chatbot vérifie qu'aucun des messages n’entre dans les conditions “if”.
Prenons un exemple:
Lors d’une conversation. Un interlocuteur A dit à un interlocuteur B qu’il a un ballon.
L’interlocuteur B ne peut utilisé que les frisbee.
Si l’interlocuteur B utilise les conditions “If”, il cherchera à chaque fois si le ballon ressemble à un de ses frisbee. Alors que si l’interlocuteur B utilisait le “switch” il ne réagirai que lorsqu'on lui parlait de Frisbee et ne chercherai donc rien lorsqu'on lui parle de ballon.

```
case "ban":
    if(!message.member.hasPermission("BAN_MEMBERS")) return message.channel.sendMessage("Erreur permission invalide");
    if(!modlog) return message.reply("Je ne trouve pas de channel mod-log.");
    if (message.mentions.users.size < 1) return message.reply("Erreur Argument")
    message.guild.ban(user, 2);

    var embed = new Discord.RichEmbed()
    .addField("Commande :", "BAN")
    .addField("Utilisateur :", user.username)
    .addField("Modérateur :", message.author.username)
     .addField("Heure:", message.channel.createdAt)
    .setColor("#01A9DB")
    .setAuthor(message.author.username, message.author.avatarURL)
    .setTimestamp()
    member.guild.channels.find("name", "mod-log").sendEmbed(embed);
    message.delete();
    break;
```

Pour décrire les fonctionnalités sur lesquelles nous avons travaillé, nous prenons pour exemple la commande “!ban”.

Nous y avons implanté un système de gestion des permissions fournit par Discord (https://discordapp.com/developers/docs/topics/permissions).
L'objectif de la commande est donc de bannir un utilisateur de notre serveur. Cependant, le bot s'occupe de créer un système de log dans un salon textuel.



IV)Test difficulté / Conclusion

Dans un premier temps, nous voulions initialement développer une infrastructure en Script Shell visant à montrer comment une surveillance de masse pourrait être mise à grande échelle dans le monde. Le problème étant que nous nous dispersions dans notre projet car le thème est trop vaste. C'est ce pourquoi nous avons décidé de choisir un projet qui, avec des objectifs dès plus simples initialement. C'est ce pourquoi notre ChatBot ne dispose finalement pas de grande fonctionnalité. L'objectif étant d'apprendre à travailler avec des APIs et d'apprendre l'optimisation algorithmique, c'est ce pourquoi nous avons choisi le NodeJS et cette forme d'écriture du code.
Les difficultés au niveau de l'organisation du projet que nous avons rencontrées sont principalement un manque d'assiduité collective vis-à-vis du maintien à jour du carnet de bord pour informer de l'avancé du projet.

Fin