Rapport sur la conception du bot et de l'interface pour le projet 'Bot de modération Discord'

Dans le cadre de notre projet web, nous cherchons à mettre en place un bot sur l'application de chat en ligne **discord**, le but étant de mettre en place un bot pouvant réaliser les tâches suivantes :

> - Gérer la modération en permettant au bot de créer ou supprimer des modérateurs sur le serveur Discord
> - Gérer les sanctions en permettant au bot de punir des utilisateurs en recevant une commande d'un modérateur
> - Détecter les comportements illicites tel que le spamming et le punir en conséquence sans passer par le biais d'un modérateur. En prime, nous mettront en place un interface web permetttant de gérer le bot sur un site à part, ce site web disposera des fonctionnalitées suivantes :
> - Permettre la diffusion publique du bot
> - Chaque utilisateur ayant le bot installé sur son serveur doit avoir accès à un panel d’administration. Ce panel d’administration doit permettre d’activer/désactiver ou/et de configurer certaines fonctionnalités du bot. Notre mission doit donc nous amener à comprendre le fonctionnement d'un serveur node js, et établir un lien entre notre applicatino et un élément extérieur.

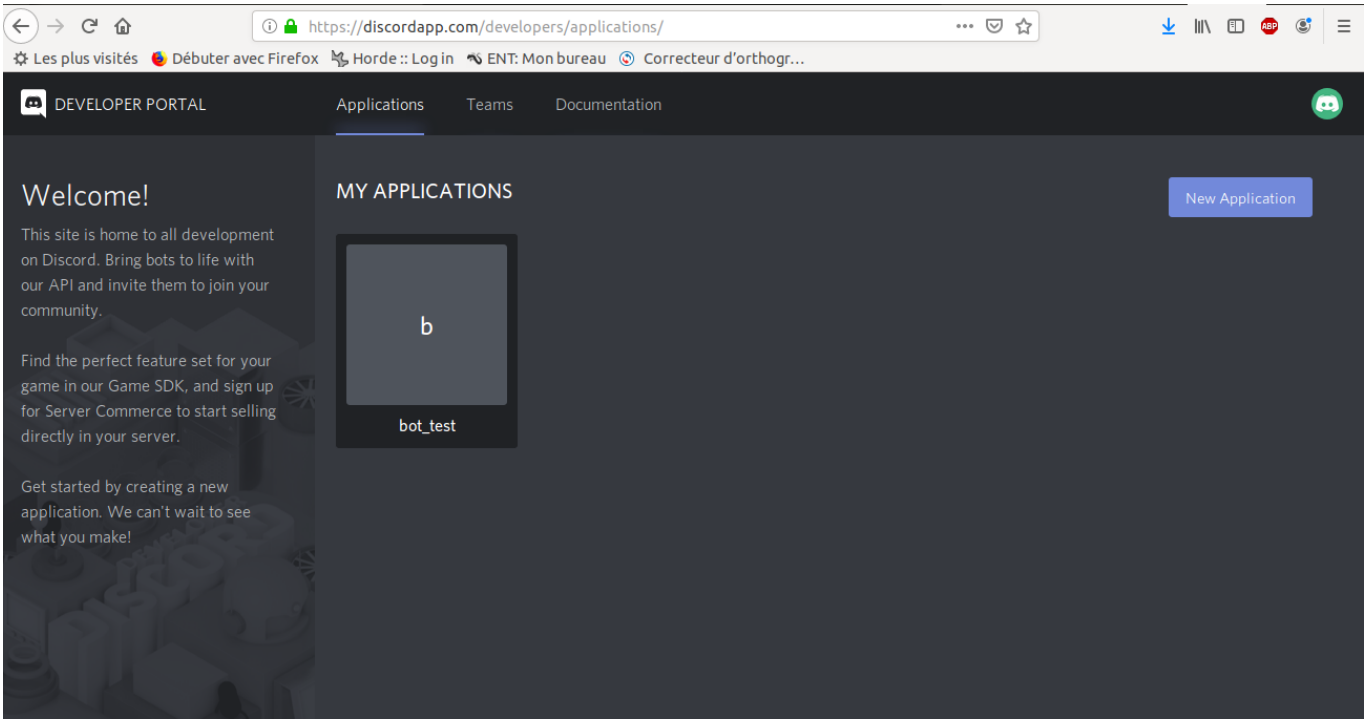
Au travers de ce rapport nous verrons la manière dont nous avons réalisé ces tâches, en commencant par l'élaboration d'un serveur node js avec un webSocket, puis la réalisation du bot avec Discord.js et enfin la réalisation de notre panel de gestion du bot.

Pour toutes questions en ce qui la base de données, vous pouvez vous référez à notre précédent rapport `rapport_conception_db.md`.

Création d'un bot Discord

La première partie de notre travail de création d'un bot Discord est évidemment de créer le bot. Pour ce faire nous nous renderons sur le portail Developers de discord à l'adresse :

'https://discordapp.com/developers/applications/'



On crée une application qui deviendra notre futur bot. Discord nous fournit ce portail afin de simplifier la création d'application sur leur plateforme.

On se rend ensuite dans l'onglet **"Bot"** afin de confirmer que notre application est un utilisateurs robot, une fois que l'on confirme la nature de notre application nous serons données par Discord un `Token` qui sera utilisé par notre application node js afin d'appeler le bot sur un serveur.

Mise en place du WebSocket

Afin de comprendre le principe derrière la discussion entre node js et Discord, nous avons commencé par établir un webSocket qui nous permettrait d'établir une discussion avec le serveurs discord, mette en ligne notre bot et mettre en place une fonctionnalité qui retournerait un message **'Pong'** en cas de réception d'un message **'Ping'**.

L'interêt de cette partie est double, comprendre la discussion entre node js et Discord par le biais de node js, et tester la fonctionnalité du bot vu précédemment.

Conception du bot avec Discord.js

> Afin de faciliter la réalisation de notre bot, nous avons choisi d'utiliser la bibliothèque **discord.js**.

Grâce à l'outil discord.js, nous pouvons simplifier la mise en place du bot, à l'aide des différentes fonction déjà présente dans l'extension. Cependant la simplification ouvre la prote à l'étendu du problème de la réalisation de l'application et afin de la simplifier, nous avons découpé l'application en plusieurs dossiers, chacun devant traités un aspect spécifique du bot :

index.js

Le fichiers en charge de traiter la connexion du bot au serveur, ainsi que la récupération les messages addressé au bot avant de les envoyer vers un index de traitement. Cela inclus notamment la détection de si le message est destiné au bot et vient d'un autre bot (dans ce cas il est ignoré) ou d'un utilisateur humain.

Dans ce cas le message est envoyé dans la commande `check_and_run()` de src/command.index.js afin de vérifier si le message vient d'un propriétaire légitime ou non.

Le fichier index.js est disponible dans /src/

command/index.js

Le fichier index.js dans ./src/command/ à pour but de traiter les commandes envoyés par un utilisateur. Il commence d'abord par vérifier si l'utilisateur est légitime dans sa demande en vérifiant s'il dispose des droits nécessaires.

S'il dispose des droits nécessaires la commande est passé au travers d'un moulinage de **Regex** afin de comprendre de quel commande il s'agit. Si la commande est trouvée, elle est immédiatement exécuter par l'appel d'une fonction `callfunc()` qui traitera la fonction reçu. Si le match() ne retourne rien, alors il ne s'agit pas d'une fonction et ce n'est pas traiter. Une liste des commandes globale est disponibles ci-contre :

Listes des commandes globale

> Il s'agit des commandes prédéfini par notre service, et qui sont globale pour tous les serveurs.

- Bannir un utilisateur :
  `!ban @<user> <reason:text> [-d <duration:time(sec)>, -c <channels:list>]`
- Exclure un utilisateur :
  `!kick @<user> <reason:text>`
- Rendre sourd un utilisateur :
  `!deaf @<user> <reason:text> [-d <duration:time(sec)>, -c <channels:list>]`
- Rendre muet un utilisateur :
  `!mute @<user> <reason:text> [-d <duration:time(sec)>, -c <channels:list>]`
- Avertir un utilisateur :
  `!warn @<user> <reason:text>`
- Créer une commande personnalisé sur un serveur :
  `!create (ban|kick|deaf|mute) -d <duration_restriction> -c (<channels_restriction>)`
- Annuler une sanction par son id :
  `!cancel <id_sanction>`
- Ajouter un role de moderation à un utilisateur :
  `!rankup @<user> <role_id>`
- Retirer un role de moderation à un utilisateur :
  `!derank @<user> <role_id>`
- Ajouter un rôle (le créer) :
  `!addrole <name> <priority>`
- Supprimer un rôle :
  `!delrole <id>`
- Ajouter une commande à un rôle :
  `!role add <role_id> <command_id>`
- Retirer une commande à un rôle :
  `!role del <role_id> <command_id>`

- Récupérer les sanctions appliquées à un utilisateur :
  `!getto @<user>`
- Récupérer les sanctions appliquées par un modérateur :
  `!getfrom @<modo>`
- Vérouiller un ou des channels :
  `!lock <channels:list>`
- Déverouiller un ou des channels :
  `!delock <channels:list>`
- Supprimer les messages d'un channels (message d'un joueur et/ou depuis x sec) :
  `!delmsg <channel> [-d <duration>, -u @<user>]`