

Rapport

Projet Bot Discord

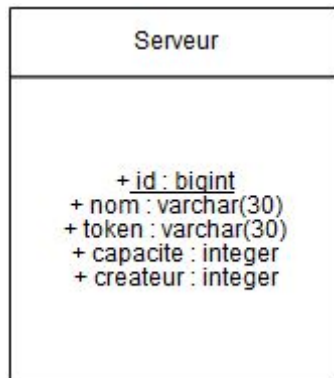
Introduction

Dans le cadre du module Web nous avons pu découvrir différentes technologies permettant la gestion client mais aussi serveur. Discord en tant que logiciel permettant de gérer des serveurs vocaux, des channels de discussion et bien d'autres éléments, son étude est tombé sous le sens. L'objectif a donc été de découvrir le fonctionnement de cet outil en développant un bot (robot) de modération de serveur à l'aide de la plateforme NodeJS. Ce projet s'est déroulé en différentes étapes allant de la conception de la base de données à la mise en place d'une interface web de gestion en passant par le développement du bot à proprement parler.

Sommaire

Introduction	2
Sommaire	3
1. Adaptation de la BDD	4
2. Le Bot	4
2.1. Commande kick	4
2.2. Commande ban	5
2.3. Commande mute	5
2.4. Commande deaf	6
2.5. Anti-spam	6
3. Interface de gestion	7
3.1. Page de connexion	7
3.2. Page de gestion	8
3.2.1. Création de rôles	9
3.2.2. Activation / Désactivation	9
3.2.3. Création d'une commande	10
3.2.4. Association d'une commande à un rôle	10
Conclusion	11

1. Adaptation de la BDD



La base de données de l'application n'a été que très peu modifiée. En effet, il a juste fallu adapter certains éléments comme les ID. Les ID fournis sur Discord ne passent pas sur un type Integer car ils sont trop gros. Il a donc fallu ajuster la BDD pour que tous les ID soient en BigInt au lieu de Integer.

Il a cependant fallu rajouter un nouvel élément à la table Serveur pour permettre au Owner du serveur d'accéder à l'interface de gestion. Le Token est donc une chaîne de caractères générés aléatoirement, liée au serveur.

2. Le Bot

2.1. Commande kick

Cette commande permet à tout membre ayant les droits nécessaires sur un serveur, de pouvoir expulser un membre d'un serveur Discord.

Dans un premier temps, on récupère le membre à expulser (grâce une une expression régulière) ainsi que le nom du serveur sur lequel la commande est exécutée. Ensuite, on teste la bonne présence de ce membre sur le serveur. Si il existe, alors on l'expulse et on modifie notre base de données. La table Sanction va être modifiée afin d'ajouter la nouvelle sanction avec sa raison (optionnelle). La table EstMembre, elle, sera mise à jour pour faire correspondre la sanction à l'utilisateur expulsé.

2.2. Commande ban

Cette commande permet à tout membre ayant les droits nécessaires sur un serveur, de pouvoir bannir un membre d'un serveur Discord.

Dans un premier temps, on récupère le membre à bannir (grâce à une expression régulière) ainsi que le nom du serveur sur lequel la commande est exécutée. Ensuite, on teste la bonne présence de ce membre sur le serveur. S'il existe, alors on le bannit et on modifie notre base de données. La table Sanction va être modifiée afin d'ajouter la nouvelle sanction avec sa raison (optionnelle). La table EstMembre, elle, sera mise à jour pour faire correspondre la sanction à l'utilisateur banni.

2.3. Commande mute

Cette commande permet à tout membre ayant les droits nécessaires sur un serveur, de pouvoir rendre muet un membre d'un serveur Discord.

Dans un premier temps, on récupère le membre à mute (grâce à une expression régulière) ainsi que le nom du serveur sur lequel la commande est exécutée. Ensuite on teste la bonne présence de ce membre sur le serveur. S'il existe, alors, deux options se présentent.

Si le rôle "muted" n'existe pas sur le serveur, on le crée lors de l'exécution de la commande et on l'applique à l'ensemble des salons (vocaux et textuels) avec des droits restreints :

- impossible d'envoyer des messages dans un salon textuel,
- impossible de réagir à des messages dans un salon textuel,
- impossible de parler dans un salon vocal.

Remarque, les permissions d'un salon sont prioritaires sur celles d'un utilisateur. Nous avons donc choisi de ne pas mettre de permissions directement au niveau de l'utilisateur mais plutôt d'appliquer le rôle "muted" sur chaque salon.

Ensuite, il ne reste plus qu'à appliquer ce rôle au membre à rendre muet et à modifier la base de données (mêmes modifications de la base de données que pour la partie kick et ban).

Dans le cas où le rôle existe déjà sur un serveur, nous avons uniquement besoin d'appliquer le rôle au membre à rendre muet et à modifier la base de données.

2.4. Commande deaf

Cette commande permet à tout membre ayant les droits nécessaires sur un serveur, de pouvoir rendre sourd un membre d'un serveur Discord.

Dans un premier temps, on récupère le membre à deaf (grâce à une expression régulière) ainsi que le nom du serveur sur lequel la commande est exécutée. Ensuite, on teste la bonne présence de ce membre sur le serveur. S'il existe, alors, deux options se présentent.

Si le rôle "deafed" n'existe pas sur le serveur, on le crée lors de l'exécution de la commande et on l'applique à l'ensemble des salons (vocaux et textuels) avec le droit restreint suivant : impossible de se connecter à un serveur vocal.

Remarque, les permissions d'un salon sont prioritaires sur celles d'un utilisateur. Nous avons donc choisi de ne pas mettre de permissions directement au niveau de l'utilisateur mais plutôt d'appliquer le rôle "deafed" sur chaque salon.

Ensuite, il ne reste plus qu'à appliquer ce rôle au membre à rendre sourd et à modifier la base de données (même modifications de la base de données que pour la partie kick et ban).

Dans le cas où le rôle existe déjà sur un serveur, nous avons uniquement besoin d'appliquer le rôle au membre à rendre sourd et à modifier la base de données.

2.5. Anti-spam

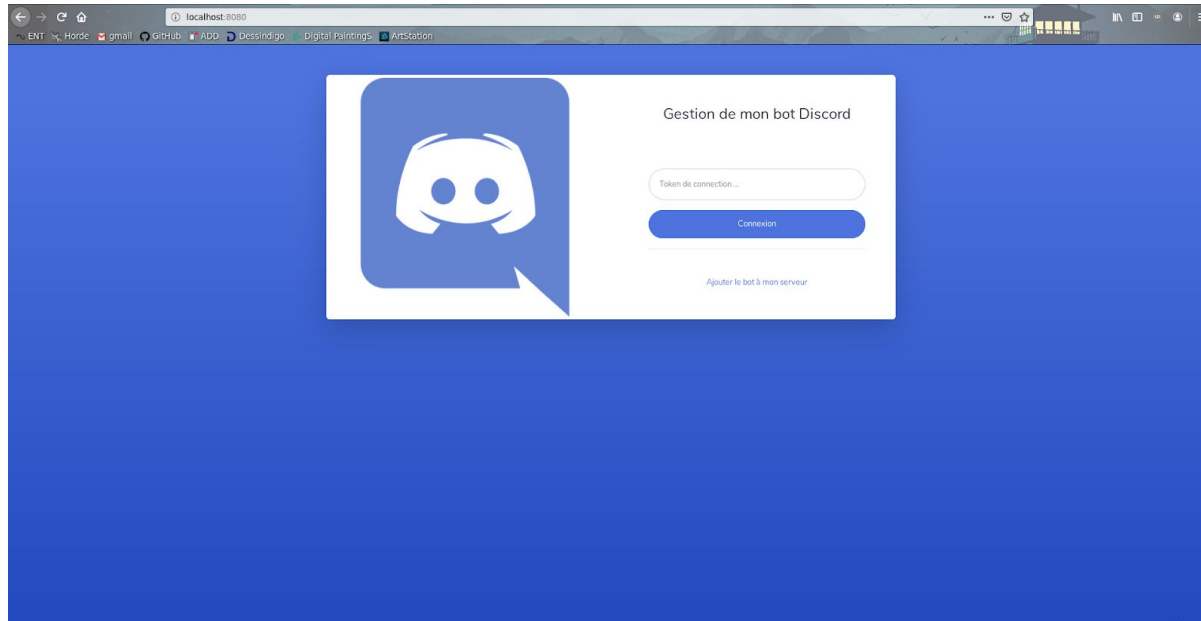
Par manque de temps, nous avons choisi d'utiliser le module discord anti-spam. Ce module permet de détecter un spam :

- en comparant le delta de temps entre deux messages d'un même membre,
- en comparant deux messages identiques d'un même membre.

Si une de ces deux détections est vérifiée, alors le bot va incrémenter un compteur. Dans le cas où un spam a été détecté plus de 4 fois, le bot va alerter le spammeur avec un message. S'il décide de continuer malgré l'avertissement, il sera banni du serveur au bout de 7 messages considérés comme spam.

3. Interface de gestion

3.1. Page de connexion



Le Owner d'un serveur Discord peut donc effectuer différentes actions grâce à l'interface web disponible. Lorsqu'il se lance dessus, une page de connexion apparaît pour lui permettre soit de se connecter au panel d'administration d'un serveur dont il est le Owner, soit il peut inviter le bot sur l'un de ses serveurs.

Pour se connecter, l'administrateur doit entrer un token lié au serveur. Ce token stocké en base de données est généré lorsque le bot arrive sur le serveur et envoyé au Owned par message privé pour qu'il puisse l'utiliser directement.

L'utilisation d'un système de token a été utilisé pour faciliter la connexion à l'interface en évitant de devoir se connecter à Discord pour récupérer les informations de l'utilisateur. De plus en termes d'efficacité cela revient au même car seul le Owner peut se connecter car seulement lui reçoit le token par message privé.

A côté de la génération de ce token, lorsque l'on invite le bot sur un serveur, on va alors récupérer toutes les informations nécessaires à la base de données et les enregistrer pour pouvoir en avoir une à jour avec toutes les informations nécessaires :

- Information sur le serveur
- Liste des utilisateurs et donc des membres
- Liste des salons
- Liste des rôles

Ce peuplement de la base s'effectue bien entendu avec des tests pour éviter la redondance d'informations similaires en base de données.

3.2. Page de gestion

The screenshot displays a web application interface for managing a Discord bot. On the left is a blue sidebar with the text "GESTION DE MON SERVEUR" and a "Connection" link. The main content area is titled "TestBot" and contains three primary sections: "Activation des logs" with "Activer les logs" and "Désactiver les logs" buttons; "Créer un rôle" with input fields for "Nom du rôle" and "Position 0, 1, 2 ..." and a "Valider" button; and "Créer une commande" with input fields for "Nom de la commande", a dropdown menu showing "KICK, BAN, MUTE, DEAF", and a field for "Nom du salon où appliquer la commande". On the right, a "Gestion des rôles : ajout d'une commande" panel includes input fields for "Nom du rôle", "Nom de la commande", and "Nom de l'atom de la commande", along with an "Ajouter" button.

La page de gestion permet d'effectuer différentes actions sur un serveur Discord spécifique:

- Création de rôles
- Activation / désactivation des logs
- Création d'une commande
- Association d'une commande à un rôle

Par manque de temps d'autres éléments de gestions n'ont pas pu être ajoutés.

3.2.1. Création de rôles

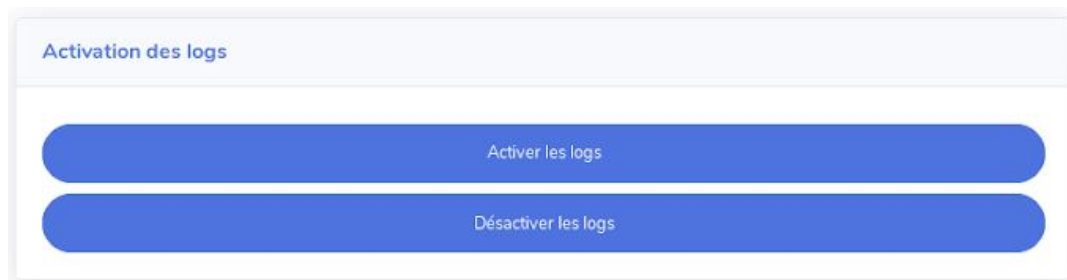


The image shows a web form titled "Créer un rôle" (Create a role). It contains two input fields: the first is labeled "Nom du rôle" (Role name) and the second is labeled "Position 0, 1, 2 ..." (Position 0, 1, 2 ...). Below these fields is a blue button labeled "Valider" (Validate).

Il est possible au travers d'un formulaire de créer un nouveau rôle en spécifiant son nom, sa couleur et sa position dans la hiérarchie des rôles.

Une fois le formulaire validé, ce nouveau rôle sera créé sur le serveur Discord puis enregistré en base de données.

3.2.2. Activation / Désactivation

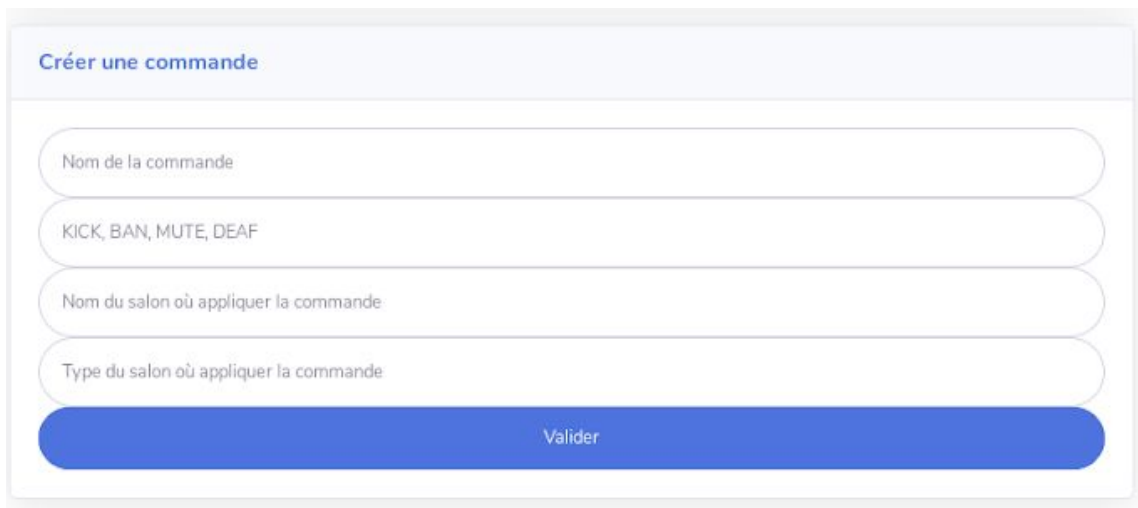


The image shows a web form titled "Activation des logs" (Log activation). It contains two blue buttons: the top one is labeled "Activer les logs" (Activate logs) and the bottom one is labeled "Désactiver les logs" (Deactivate logs).

Il est possible d'activer ou de désactiver les logs. Pour ce faire, l'activation va entraîner la création d'un salon textuel nommé bot-logs où le bot spécifie tous les logs s'il existe.

D'un autre côté, la désactivation entraîne la suppression de ce salon, le bot n'écrit donc plus de logs.

3.2.3. Création d'une commande



The screenshot shows a web form titled "Créer une commande" in a light blue header. Below the header, there are four input fields stacked vertically, each with a light blue border and rounded corners. The first field is labeled "Nom de la commande". The second field contains the text "KICK, BAN, MUTE, DEAF". The third field is labeled "Nom du salon où appliquer la commande". The fourth field is labeled "Type du salon où appliquer la commande". At the bottom of the form is a wide, solid blue button with the white text "Valider".

Le formulaire de création de commande permet d'entrer tous les champs nécessaires pour créer une commande et l'enregistrer en base de données. Il est donc nécessaire de remplir son nom, son atom, le nom du salon qui lui est associé, le type de salon.

3.2.4. Association d'une commande à un rôle



The screenshot shows a web form titled "Gestion des rôles : ajout d'une commande" in a light blue header. Below the header, there are three input fields stacked vertically, each with a light blue border and rounded corners. The first field is labeled "Nom du rôle". The second field is labeled "Nom de la commande". The third field is labeled "Nom de l'atom de la commande". At the bottom of the form is a wide, solid blue button with the white text "Ajouter".

Le formulaire d'association d'une commande à un rôle dispose de trois champs, le nom du rôle, le nom et l'atom de la commande. Une fois validé tout est enregistré en base de données.

Conclusion

Ce projet a donc été très intéressant et nous a permis de découvrir la technologie NodeJS au travers d'un outil utilisé aujourd'hui par des millions de personnes, Discord. Le développement d'un bot et d'une interface web nous a permis de comprendre et d'appréhender le fonctionnement d'une telle application tout en nous formant à l'utilisation d'une plateforme destinée à la gestion côté serveur. Le manque de temps ne nous a cependant pas permis de mener ce projet totalement à bien, ce qui aurait été réellement très enrichissant.