Documentation Technique :

Introduction :

Ce document vise à offrir une compréhension approfondie de la structure de mon code et de ma logique de développement. Cette explication reflète un état parfait du code, c'est-à-dire si tout fonctionne correctement. Cependant, étant donné que tout ne sera pas nécessairement opérationnel, je mettrai en évidence les fonctionnalités entamées mais non achevées en raison de contraintes de temps ou de compétences.

Serveur :

Def main :

Lorsque le serveur démarre, il initialise la connexion avec les futurs clients et entre en attente pour établir des connexions. Ensuite, nous entrons dans une boucle qui permet de lancer simultanément deux threads : l'un pour les commandes serveur et l'autre pour la réception des messages, ainsi que pour gérer toutes les exceptions (le corps principal). Cette boucle permet également d'arrêter le serveur : lorsqu'une condition est remplie, cela interrompt la boucle et arrête le serveur (stop-server).

Def server-commandes :

Cette section sert à envoyer des commandes aux clients, mais aussi à la base de données. Trois commandes sont implémentées ici : kill, ban et kick.

Kill : Pour le kill, il suffit d'envoyer "Kill" au client et de son côté, il s'arrêtera, comme expliqué dans la partie client. Du côté serveur, j'ai choisi que celui-ci envoie simplement "Kill" aux clients, et à ce moment-là, les clients enverront un message aux utilisateurs et se termineront.

Ban : La procédure pour le ban est un peu plus complexe. J'ai choisi de supprimer le nom ou l'adresse IP choisis. Si "cursor. rowcount" ne renvoie rien, alors l'utilisateur ou l'adresse IP n'existe pas. Mais s'il renvoie quelque chose, alors on envoie à tous les clients le message suivant : "!" + nom. Ainsi, les clients peuvent vérifier si le ban correspond à leur nom. Si oui, ils se feront bannir, sinon rien ne se passe.

Kick : Le kick est le plus complexe. On commence par vérifier si le nom existe dans la base de données, même procédure pour l'IP. Si oui, on regarde dans la table des kicks si cette personne n'est pas déjà kickée. Si oui, on met à jour, sinon on ajoute le nouveau kick. Ensuite, on envoie au client qu'il a été kické ainsi : "~nom : temps".

Def Server :

Le fonctionnement de cette définition est assez simple : chaque premier message sera soit "!#sign" ou "!#log" car le client se connecte ou s'inscrit. Avec cela, nous confirmons les identifiants et mots de passe pour la connexion et nous créons l'utilisateur lors de l'inscription.

Bien sûr, en cas d'échec, nous envoyons un message au client, de même si cela réussit. Lorsque le client se connecte, avant d'envoyer la confirmation, nous vérifions dans la table des kicks si l'utilisateur est kické. Si oui, nous lui envoyons le temps qu'il lui reste, sinon nous acceptons la connexion.

Ensuite, nous envoyons aux clients déjà inscrits les droits qu'ils ont concernant les canaux. Nous entamons ensuite une boucle de réception : elle permet de recevoir les messages des clients et de les renvoyer à tous les autres clients, tandis que les messages reçus sont enregistrés dans la table "messages" de la base de données avec le message, l'expéditeur et le destinataire.

Cette définition permet également de recevoir les droits de canaux pour la personne venant de s'inscrire, lesquels sont insérés dans la table "users" des droits de salon. Cette partie gère également la réception de tous les messages des clients et les transmet à tous les autres clients, excepté celui qui l'a envoyé en premier lieu.

Def send_user_list :

Cette section a pour but d'envoyer à un client qui souhaite discuter avec une seule personne la liste des utilisateurs inscrits.

Def validation :

Cette partie permet de valider les demandes de salon des clients.

Client :

Pour le client, l'expérience se divise en trois grandes parties, commençant par

Le Login/Sign-in :

Le processus de connexion suit cette logique : en entrant le nom d'utilisateur et le mot de passe, vous choisissez entre « login » et « sign-in ». Opter pour « login » enverra !#log au serveur, tandis que « sign-in » déclenchera l'envoi de !#sign au serveur. À partir de là, le client passe en mode d'attente.

Choix des canaux :

Une fois que le client obtient le feu vert du serveur pour continuer, dans le volet inscription, il est nécessaire de sélectionner les salons auxquels vous souhaitez accéder. Ces choix sont enregistrés dans une liste puis envoyés au serveur pour être intégrés à sa table des utilisateurs.

Page principale :

Ensuite, la page principale de mon discord s'ouvre, offrant plusieurs options telles que se déconnecter, envoyer des messages dans un groupe ou choisir une personne avec qui discuter

- Envoi de messages :
    - Pour que les groupes fonctionnent, lorsqu'on clique sur un groupe, une variable est mise en attente. Lorsque le message est envoyé, le destinataire est inséré dans le message de la forme « destinataire/message ». Ainsi, le serveur sait à qui l'envoyer.
    - Pour l'envoi de messages de personne à personne, à la réception d'un message, on vérifie à qui le message est destiné. Si c'est bien pour le destinataire, le message est affiché dans la discussion.

- Réception de messages :
    - Si le message commence par !, on vérifie le nom derrière. Si c'est le nôtre, le système se ferme.
    - Si le message commence par ~, on vérifie si le nom après correspond au nôtre. Si oui, le système se ferme.
    - Si on reçoit « Kill », le système se ferme et affiche une alerte indiquant que le serveur a été arrêté.
    - ...
    - Sinon, le message reçu est simplement affiché.

Donc, la réception de messages gère toutes les exceptions mais également l'affichage des messages reçus.

- Lorsque l'on accède à la page principale, une demande est immédiatement envoyée au serveur pour obtenir la liste des salons accessibles.

- Lorsqu'on appuie sur le bouton « Message privé », une liste des utilisateurs présents dans le discord apparaît, car nous envoyons « get_users_list » au serveur pour obtenir cette liste, puis nous les ajoutons en tant que boutons. Une fois pressé, un chat s'ouvre pour démarrer la conversation.

La réalité des choses :

Parmi toutes les choses écrites je n’ai pas pu faire plusieurs choses :

- La vérification des canaux du serveur pour donner son accord ou son désaccord. J'avais l'intention de l'intégrer dans mon interface serveur graphique, mais cela s'est écarté de mes priorités lorsque j'ai réalisé l'ampleur des tâches à accomplir. Malheureusement, faute de temps, cette fonctionnalité n'a pas pu être développée.
- La fonctionnalité de conversation de personne à personne n'a pas été finalisée.