Table des matières

Introduction

But et fonctionnalités : L'application de chat fournit une plateforme de communication en temps réel permettant aux utilisateurs de s'enregistrer, de se connecter, de rejoindre divers salons de chat, d'envoyer des messages privés et de participer à des discussions de groupe. Avec une interface conviviale, les utilisateurs peuvent facilement naviguer à travers différents salons et interagir avec d'autres utilisateurs.

Documentation Client

Introduction

Le client de mon application de chat est conçu pour fournir une expérience utilisateur fluide et interactive. Il permet aux clients/utilisateur de se connecter au serveur, de rejoindre des salons de discussion, d'envoyer et de recevoir des messages de plusieurs clients dans un salon mais aussi en privé et demander à rejoindre un salon. Avec une interface graphique intuitive construite à l'aide de la bibliothèque PyQt5, les utilisateurs peuvent facilement naviguer à travers l'application, rendant la communication en ligne plus agréable.

1. Connexion et Communication

1.1Connexion au Serveur :

- Les utilisateurs débutent par se connecter au serveur en fournissant l'adresse et le port. Une fois connectés, ils peuvent s'authentifier ou s'inscrire pour accéder aux fonctionnalités du chat.

La classe ServerConnectionWindow gère la connexion initiale au serveur.

```
class ServerConnectionWindow(QWidget):
    """
    Fenêtre pour la connexion au serveur de l'application de chat.

    Permet à l'utilisateur d'entrer l'adresse et le port du serveur auquel se
connecter,
    puis tente d'établir une connexion. En cas de succès, passe à l'interface
d'authentification.
    """
```

1.2Envoi et Réception de Messages :

- Les utilisateurs peuvent envoyer des messages textuels dans différents salons de chat auxquels ils ont adhéré. Ils peuvent également recevoir des messages d'autres utilisateurs en temps réel via la classe ChatClient et un ClientThread dédié.

```
class ClientThread(QThread):
```

```
    """
    Un thread client pour gérer la réception des messages du serveur.

    Cette classe hérite de QThread pour gérer la réception des messages
    de manière asynchrone. Elle émet un signal chaque fois qu'un message est
reçu.
    """
class ChatClient(QWidget):
    """
    Interface client pour l'application de chat.

    Cette classe gère la fenêtre principale de l'application client de chat.
    Elle initialise l'interface utilisateur, gère la connexion au serveur et
    l'affichage des messages reçus.
    """
```

2.Fonctionnalités Clés

2.1Interface Utilisateur :

- La classe ChatClient représente l'interface principale du client. Elle offre une zone de texte pour afficher les messages des salons, une zone pour saisir les nouveaux messages, et des boutons et menus pour différentes actions comme envoyer des messages, rejoindre et quitter des salons.

2.2Gestion des Salons :

- Les utilisateurs peuvent rejoindre différents salons disponibles ou en quitter, via la méthode def join_salon et leave_salon dans la classe ChatClient.

2.3Messages Privés :

- En plus des salons publics, les utilisateurs ont la possibilité d'envoyer des messages privés à d'autres utilisateurs, permettant des conversations plus personnelles et directes avec la méthode send_private_message dans la classe ChatClient.

2.4Notifications et Alertes :

- La méthode display_message dans ChatClient s'occupe de l'affichage des notifications et des alertes.

```
    def display_message(self, message):
        """
        Affiche les messages reçus du serveur.

        Cette méthode gère l'affichage des messages normaux, des
avertissements,
        et des notifications administratives. Elle gère aussi la fermeture
        de la session en cas de kick ou ban.
        """
```

3.Architecture et Technologies

ClientThread : Ce thread s'occupe de recevoir les messages du serveur de manière asynchrone pour éviter de bloquer l'interface utilisateur.

PyQt5 : Utilisé pour construire l'interface graphique, permettant une expérience utilisateur interactive et agréable.

Conclusion

La partie client de l'application de chat est essentielle pour une expérience utilisateur réussie. Elle offre une interface intuitive, supporte des communications fluides et sécurisées, et permet une interaction facile avec le serveur et les autres utilisateurs. En résumé, le client rend l'utilisation de l'application de chat accessible, engageante, et sécurisée pour tous.

Documentation Serveur

Introduction

Au début, ce projet de serveur de chat était contrôlé uniquement par des commandes textuelles dans le terminal. C'était fonctionnel, mais pas toujours pratique. Alors, j'ai ajouté une interface graphique pour rendre les choses plus simples et visuelles, tout en gardant la possibilité d'utiliser le terminal. Maintenant, vous avez le choix entre utiliser les commandes dans le terminal ou cliquer sur des boutons dans l'interface graphique (notification : dans l'interface graphique seules les notifications lors de l'utilisation des commandes administratives et gestions des salons apparait ; tandis que dans le terminal toute la notification apparait en plus des commandes administrative ex : client {alias} à rejoint/quitter le salon, Authentification de {alias} réussi…)

1.Classe et méthode

1.1 Connexion à la Base de Données

Le serveur utilise une base de données MySQL pour enregistrer des infos importantes comme les utilisateurs, leurs messages, l'authentification des utilisateurs, et la gestion des sanctions administrative. Pour se connecter et interagir avec cette base de données, il y a une méthode spécifique :

Méthode de Connexion :

 # Utilisation de la méthode get_db_connection de l'objet Server pour obtenir une connexion à la base de données dans les méthodes qui ont besoins d'être connecter (par ex : voir la méthode login)

```
def get_db_connection(self):
            """Établit une connexion à la base de données et la retourne."""
            try:
                connection = pymysql.connect(
                    host=self.db_host,
                    user=self.db_user,
                    password=self.db_password,
                    db=self.db_name,
                    charset='utf8mb4',
                    cursorclass=pymysql.cursors.DictCursor
                )
                return connection
            except Exception as e:
                print(f"Erreur lors de la connexion à la base de données : {e}")
                return None
```

1.2Pour démarrer le serveur, méthode start dans la classe Server :

```
def start(self):
        """
        Démarre le serveur pour écouter les connexions entrantes.

        Lie le serveur à l'adresse et au port spécifiés et commence à écouter
les connexions.
        Pour chaque nouvelle connexion, un thread client est démarré pour
gérer la communication.
        """
```

1.3La classe StartServerWindow pour l'interface de démarrage du serveur :

```
class StartServerWindow(QMainWindow):
    """
    Fenêtre pour démarrer le serveur.

    Cette interface permet à l'utilisateur de démarrer le serveur avant de
passer à l'interface d'authentification.
    """
```

La méthode login qui se trouve dans la classe AuthentificationWindow pour l'authentification du serveur.

```
    def login(self):
        """
        Gère la tentative de connexion du serveur.
        """
```

2.Fonctionnalités Clés

2.1Gestion des connexions :

Accepte les connexions des clients et crée un nouveau thread pour gérer chaque client.

```
while True:
                client_socket, addr = self.server_socket.accept()
                client_thread = threading.Thread(target=self.handle_client,
args=(client_socket, addr))
                client_thread.start()
```

2.2Gestion des salons :

Permet aux clients de rejoindre différents salons de chat (Général, Blabla, etc.) et de recevoir ou envoyer des messages au sein de ces salons.

2.3Authentification et gestion des utilisateurs :

Méthode login qui se trouve dans la classe Server. Authentifie les utilisateurs en utilisant les informations stockées dans une base de données MySQL.

```
def login(self, alias, password):
        """
        Vérifie les informations de connexion d'un utilisateur et
l'authentifie si elles sont correctes.
        Utilisez get_db_connection de l'objet Server pour obtenir une
connexion à la base de données
        """
        connection = self.get_db_connection()
```

2.4 Commandes Administratives

Au début, le serveur permettait la gestion via des commandes textuelles dans le terminal. Les administrateurs pouvaient utiliser des commandes telles que kick, ban, et kill pour modérer l'activité sur le serveur.

Administration du serveur :

Fournit des commandes administratives pour gérer les utilisateurs et le serveur.

```
def handle_admin_commands(self):
        """
        Gère les commandes administratives du serveur.

        Permet à l'administrateur d'exécuter des commandes dans le terminal
pour gérer le serveur et les utilisateurs,
        telle que l'acceptation ou le refus de demandes de participation à des
salons,
        le bannissement d'utilisateurs, l’exclusion et aussi l’arrêt du
serveur.
        """
```

2.5Exemple de Commandes dans le Terminal :

Tout d’abord, spécifier le nom d’utilisateur, ensuite le mot de passe pour ensuite effectuer des commandes, cette fonctionnalité est incluse pour l’interface et/ou s’il on veut utiliser, dans le terminal.

Kick {alias} avec la raison : Exclut temporairement un utilisateur du chat.

Ban {alias} avec la raison : Bannit un utilisateur de façon permanente.

Kill : Arrête le serveur.

accepter 1/ refuser 1 : commande qui accepte/refuse un utilisateur de rejoindre un salon.

Explication détailler des commandes d’administration dans la section « guide pour utiliser des commandes administratives »

Avec l'ajout de l'interface graphique, ces commandes peuvent également être exécutées à travers des boutons et des formulaires, rendant la gestion plus visuelle et moins dépendante de la connaissance des commandes textuelles.

Transition vers une Interface Graphique

L'ajout d'une interface graphique pour le serveur marque une évolution importante du projet. Cette interface, est réalisée avec PyQt5 et non pas tkinter, offre une visualisation en temps réel de l'état du serveur, des connexions clients, et permet l'exécution des commandes administratives de manière intuitive.

3.Panneau de Contrôle du Serveur :

La classe ControlPanelWindow est une fenêtre de l'interface graphique du serveur qui permet de visualiser et d'interagir avec le serveur de manière graphique. Elle intègre des fonctionnalités telles que l'affichage des logs, la gestion des utilisateurs connectés, et l'exécution de commandes telles que kick et ban, kill, warning.

```
class ControlPanelWindow(QMainWindow):
    """
    Fenêtre du panneau de contrôle du serveur.

    Cette interface graphique permet à un administrateur de gérer le serveur,
     compris le bannissement ou l'exclusion d'utilisateurs, l'arrêt du
serveur, etc.
    """
```

RequestManagementWindow est une fenêtre de l'interface graphique qui permet aux administrateurs de gérer les demandes d'adhésion aux salons

```
class RequestManagementWindow(QMainWindow):
    """
    Fenêtre pour gérer les demandes d'adhésion au salon.

    Permet aux administrateurs de voir et de répondre aux demandes de
participation des utilisateurs.
    """
```

Conclusion

En résumé, le serveur a commencé simple avec juste des commandes dans le terminal, mais maintenant il a aussi une interface graphique. Vous pouvez toujours utiliser le terminal si vous voulez, mais l'interface graphique rend les choses plus faciles, surtout pour des tâches administratives comme gérer les utilisateurs ou voir les messages. Tout cela aide à garder le chat sécurisé et agréable pour tout le monde.

Documentation pour les développeurs

Dans cette partie, je vais vous guider à comprendre comment le code est architecturé, les fichiers, les classes

1.Structure du Code

1.1Architecture Générale :

- Serveur (server.py) : C'est le cœur de notre application. Il gère toutes les connexions, les salons, les messages et les interactions avec la base de données MySQL.
- Client (client.py) : C'est l'interface utilisateur qui permet aux utilisateurs de se connecter au serveur, de participer à des salons, d'envoyer des messages, etc.

1.2Modules et Classes Importantes :

- Server Class : Gère les connexions des clients, les salons, la diffusion des messages, etc.
- ClientThread : Un thread côté serveur pour gérer la communication avec chaque client connecté.
- ChatClient : La classe principale côté client pour l'interface utilisateur et la gestion des interactions avec le serveur.

2.Base de Données

Pour bien comprendre comment l'application fonctionne, c'est crucial de connaître la structure de la base de données. Voici les tables que j'ai utilisées et leur rôle :

Utilisateurs :

- Contient des infos sur les utilisateurs, comme leur alias, nom, prénom et mot de passe.
- alias est la clé primaire ici, donc chaque utilisateur doit avoir un alias unique.

Salons :

- Liste des salons disponibles pour le chat.
- J'ai déjà inclus quelques salons de base comme 'Général', 'Blabla', 'Comptabilité', 'Informatique' et 'Marketing'

Messages :

- Stocke tous les messages échangés dans les salons.
- Chaque message est lié à un utilisateur (via user_alias) et un salon (via salon_name).

PrivateMessages :

- Pour les messages privés entre utilisateurs. (Permet de stocker tous les messages privés)
- Chaque message a un expéditeur (sender_alias) et un destinataire (recipient_alias).

BannedUsers :

- Utilisateurs bannis de l'application avec la raison et la date du bannissement.

Avertissements :

- Enregistrements des avertissements donnés aux utilisateurs, avec la raison et la date.

• Sécurité des Mots de Passe : Les mots de passe stockés dans la table Utilisateurs devraient toujours être hashés. C'est une bonne pratique de sécurité pour protéger les informations des utilisateurs.

Document de Réponse

Ce document sert à résumer les réalisations, discuter des bénéfices et des limitations de l'outil de chat développé.

1.Réalisation du Projet

1.1Fonctionnalités Implémentées (serveur) :

• Authentification sur le serveur pour réaliser des commandes
• Le serveur enregistre les identifiants dans une base de données pour l'authentification
• Le serveur enregistre les conversations émises par chaque client.
• Le serveur est fait sous forme texte et graphique.
• Le serveur se connecte sur une base de données permettant d'enregistrer notamment les identifiants, les messages émis par le client dans tous les salons mais aussi les messages privés.
•Commandes d'administration : Le serveur permet de kick ou ban des utilisateurs, d'arrêter le serveur et d'avertir des utilisateurs.

1.2Fonctionnalités Implémentées (client) :

• Client sous la forme d'une interface graphique
• Gestion des utilisateurs : Les utilisateurs peuvent s'inscrire, se connecter, et sont gérés dans une base de données.
• Graphique de forme originale
• Une page d'inscription avec l'identité (nom, prénom), le mot de passe et l'alias de la personne (unique).
• Une page de connexion en spécifiant l'alias et le mot de passe.
• Les clients peuvent demander à intégrer les salons suivants :
    - Général c'est le salon d'arrivée
    - Blabla : sur demande (il n'y a pas besoin de validation)
    - Comptabilité : sur demande et validation
    - Informatique : sur demande et validation
    - Marketing : sur demande et validation
• Messages privés : Possibilité d'envoyer des messages privés à d'autres utilisateurs.

1.3Fonctionnalités Implémentées (en plus) :

• Côté serveur :
-commande « Warning User » qui permet d'avertir un utilisateur avec l'alias et la raison, le client reçois un message d'avertissement et ceci s'implémente dans la base de données dans la table Avertissements.
-pour gérer les demandes d'adhésion de plusieurs clients, j'ai amélioré l'interface pour qu'il soit plus clair.
• Côté client : Ajout des historiques des messages envoyés dans les salons. Par exemple : un utilisateur qui envoie des messages dans le salon Général lorsqu'il quitte et revient dans le même salon les messages envoyés seront visible pour lui et pour les autres utilisateurs.

• Base de données : lorsqu'un utilisateur est créé avec le nom, prénom, alias et mot de passe, le mot de passe est haché dans la base de données, pour renforcer la sécurité des utilisateurs. Donc mêmes

si une personne malveillante à accès à la base de données il pourra voir les identités en clair mais pas le mot de passe car il est chiffré.

Utilisation de chatgpt pour le css et une aussi pour l'acceptation d'un client depuis le terminal !

1.4Ce qui n'a pas été fait :

Je n'ai pas eu le temps de garder en mémoire les salons auxquels chaque utilisateurs à adhérer.

Comment faire : Il faudra créer dans la base de données une table qui stocke chaque utilisateur qui ont rejoint un salon avec validation (et ajuster le code, si un client demande de rejoindre un salon →voir dans la base de données s'il a été déjà valider →si oui, annuler la validation et accepter à rejoindre le salon), pour qu'ensuite si l'utilisateur veut rejoindre un salon qui a été déjà adhérent n'aura plus besoin de validation.

1.5Ce qui peut être fait en plus :

Transfert de fichier entre chaque utilisateur mais aussi dans les salons en ajoutant dans le code des méthodes comme send_file / receive_file.

Ajouter des modérateurs qui peuvent aussi exécuter des commandes administratives.

Canal de Support : Mettre en place un moyen pour les utilisateurs de demander de l'aide, que ce soit via un forum, un email ou un système de ticket.

Mot de passe oublié : Ajout du fonctionnalité « mot de passe oublié » pour les utilisateurs qui ont oublié leurs mots de passe.

La commande ban à améliorer : dans notre cas lorsqu'un utilisateur est banni, il s'enlève de la table des utilisateurs dans la base de données pour qu'il ne puisse pas se connecter à nouveau avec le même identité et mot de passe. Mais il peut créer un nouveau compte avec le même identité et mot de passe. Remarque : dans la base de données, lorsqu'un utilisateur est banni, il n'apparait pas dans la table BannedUsers, car lorsque que l'utilisateur est banni, il s'enlevé de la base de donnée, donc comme on a utiliser un clé étrangère cela impacte la table BannedUsers.

Solution : Ajouter dans la table Utilisateurs, un nouveau champs « situation » qui indique si l'utilisateur est banni ou non, s'il est banni, il ne pourra plus créer de compte avec ses identité et mot de passe.

2.Discussion sur la Solution

2.1Bénéfices :

- Facilité d'utilisation : Les utilisateurs peuvent facilement s'inscrire, se connecter et commencer à échanger des messages. Il ne nécessite pas de formation complexe ni de compétences techniques avancées pour être utilisé efficacement. Cette simplicité d'utilisation signifie que les membres de l'équipe peuvent se mettre rapidement en ligne et commencer à communiquer sans tracas.

- Efficacité de la communication en temps réel : L'un des avantages majeurs de cet outil de chat est sa capacité à permettre une communication en temps réel. Les messages sont transmis instantanément, ce qui signifie que les membres de l'équipe peuvent discuter, partager des informations et prendre des décisions rapidement. Cela élimine les retards associés aux e-mails ou aux réunions en personne, ce qui est essentiel pour les projets et les tâches nécessitant une réponse rapide.

2.2Risques :

• Les messages privés sont visibles dans la base de données avec l'utilisateur qui à envoyer est le destinataire et le contenu du message. Donc une personne malveillante à accès à la base de données pourra voir les messages privés envoyé.

2.3Recommandations :

- Implémentez une protection contre les attaques par force brute en limitant le nombre de tentatives de connexion infructueuses et en imposant des verrouillages temporaires des comptes en cas de tentatives infructueuses répétées.

- Enregistrez toutes les activités importantes, y compris les tentatives de connexion infructueuses, les commandes du serveur et les activités de gestion. Les journaux doivent être stockés de manière sécurisée et surveillés régulièrement

---

Documentation installation :

Pour commencer, il faut s'assurer d'avoir Python (version récent recommandée), c'est essentiel pour faire tourner notre application. Il est aussi impératif de posséder MySQL Workbench (et/ou MySQL Command Line Client) pour importer/exporter des bases de données et pour visualiser les données qui ont été importer dans la base de données.

1. Installation de Python et des dépendances :

Voici ce qu'il est primordial de faire avant d'exécuter les programmes. :

1.Télécharger et installer Python depuis si vous ne possédait pas site officiel de Python.

Installer ces deux bibliothèques dont on a besoin (Il faudra ouvrir le terminal ou invite de commande pour tapez ces commandes):

2. Installer PyQt5 : pip install PyQt5 (Pour l'interface client et serveur)

3. Installer Bcrypt : pip install bcrypt (Qui sert à hacher les mots de passe)

Récupérer le fichier GitHub depuis ce lien : https://github.com/agozel5/SAE3.02.git. Donc lorsque dans le salon vous avez à disposition 3 dossier. Le premier dossier, où se trouve tous les documentation (vous pouvez les visualisez en les téléchargent). Le deuxième dossier sont les exercice faites en classe. Et le troisième fichier et celui de notre application, il se nomme « SAE », une fois cliquer dessus, il y'aura 4 fichiers qui apparaitrons dont 2 clients identiques, un serveur et le fichier sql (qu'il faudra télécharger et importer dans workbench) vous pouvez aussi les télécharger en sélectionnant le fichier puis appuyer sur les trois petit point pour appuyer sur le bouton download afin de télécharger le fichier.

2. Configuration de la Base de Données :

Je vais vous expliquez étape par étape sur comment créer et initialiser la base de données en utilisant le script SQL fournis.

- Mais Jammy à quoi sert une base de données ?

-Eh bien Fred une base de données est un système électronique qui **permet d'accéder facilement à un ensemble organisé de données, de les manipuler et de les mettre à jour**.

Côté client, pour qu'un utilisateur puisse s'authentifier […], il est primordial de démarrer MySQL. Pour cela deux façons s'offre à nous, soit en démarrant MySQL Command Line Client, puis mettre don mot de passe pour démarrer ou soit en démarrant MySQL Workbench et sélectionner le bon nom de connexion et un fois sélectionner il faudra mettre le mot de passe pour démarrer le service.

Premier cas, pour l'utilisation MySQL Command Line Client :

- Créez une base de données spécifiquement pour notre chat. Je l'ai appelée "application_sae" mais sentez-vous libre de choisir le nom que vous voulez.
- N'oubliez pas de noter vos paramètres de connexion, on va en avoir besoin pour configurer le serveur.

Voici les tables qu'il faudra implémenter :

```
CREATE TABLE Utilisateurs (
   alias VARCHAR(255) PRIMARY KEY,
   nom VARCHAR(255),
   prenom VARCHAR(255),
   mot_de_passe VARCHAR(255),
   date_creation TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE Salons (
   salon_name VARCHAR(255) PRIMARY KEY
);

INSERT INTO Salons (salon_name) VALUES
('Blabla'),
('Comptabilité'),
('Général'),
('Informatique'),
('Marketing');


CREATE TABLE Messages (
   message_id INT AUTO_INCREMENT PRIMARY KEY,
   user_alias VARCHAR(255),
   salon_name VARCHAR(255),
   message_content TEXT,
   timestamp DATETIME DEFAULT CURRENT_TIMESTAMP,
   FOREIGN KEY (user_alias) REFERENCES Utilisateurs(alias) ON DELETE CASCADE,
   FOREIGN KEY (salon_name) REFERENCES Salons(salon_name)
);

CREATE TABLE PrivateMessages (
```

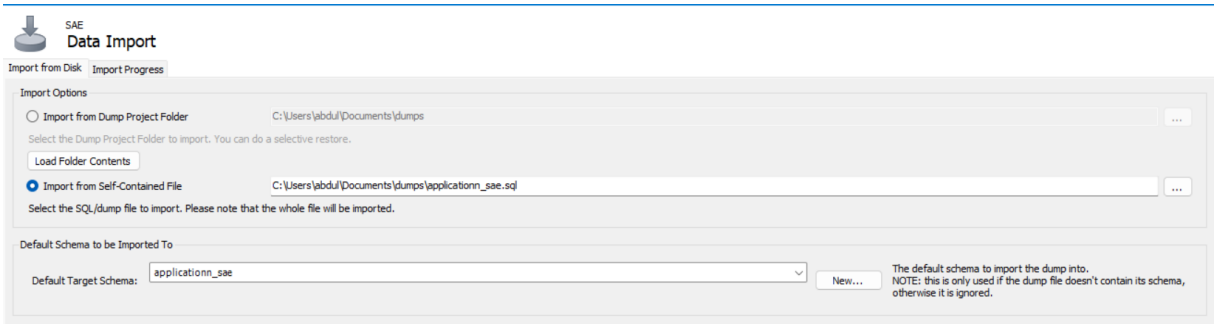
```
    id INT AUTO_INCREMENT PRIMARY KEY,
    sender_alias VARCHAR(255),
    recipient_alias VARCHAR(255),
    message_content TEXT,
    timestamp DATETIME DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE BannedUsers (
    alias VARCHAR(255),
    date_banned TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    ban_reason TEXT,
    PRIMARY KEY (alias),
    FOREIGN KEY (alias) REFERENCES Utilisateurs(alias) ON DELETE CASCADE
);


CREATE TABLE Avertissements (
    id INT AUTO_INCREMENT PRIMARY KEY,
    utilisateur_alias VARCHAR(255) NOT NULL,
    raison TEXT NOT NULL,
    date_avertissement TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (utilisateur_alias) REFERENCES Utilisateurs(alias) ON DELETE CASCADE
);
```

Remarque : Vous pouvez visualiser les données insérer dans la base de donnée directement depuis l'application Workbench ou dans le terminal Mysql en utilisant ce préfixe suivit du nom de la tables a visualiser : Select * from {nom de la Tables} ;

Dans notre projet, il suffit de sélectionner le fichier sql que je vous ai fourni puis l'importer dans l'application MySQL Workbench. Une fois que vous avez installez les bibliothèques et importer vous pouvez passer à l'étape du démarrage du serveur ainsi que du client.



! Attention à bien importer le fichier application_sae et ne pas oublier de choisir dans la séction Default Target Schema : application_sae

3. Mise en place du serveur

- Dirigez-vous vers le dossier contenant mon script serveur. Ça sera nommé server.py.

- Ouvrez-le et vérifiez bien que les infos de connexion à la base de données correspondent à celles que vous avez configurées dans MySQL.

- Maintenant, lancez-le ! Utilisez python server.py dans votre terminal. Si tout va bien, l'interface graphique de démarrage du serveur devra apparaitre, il suffira de cliquer su Start Serveur pour que le server démarre et soit attente des connexions sur le port configuré.

4. Mise en place du client

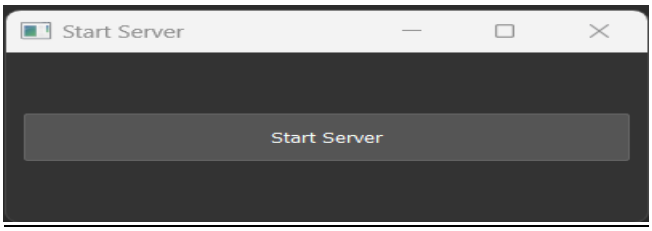
- Pareil que pour le serveur, allez au dossier contenant le script client, nommé client.py.

- Exécutez-le en utilisant python client.py. Vous devriez voir l'interface graphique du client s'ouvrir, vous permettant de vous connecter ou de vous inscrire au serveur en spécifiant l'adresse et le port auquel est connecter le serveur, dans notre cas l'adresse du serveur est : 127.0.0.1 et le port est : 12345, si vous voulez que votre serveur possède une adresse et port différent, il faudra aller tout en bas du code pour changer l'adresse et le port du code.

5.Guide pour utiliser les commandes administratives :

5.1Commandes du Serveur :

Pour faire des commandes il faudra tout d'abord démarrer le serveur puis ensuite une interface graphique sera lancé pour Start le serveur :

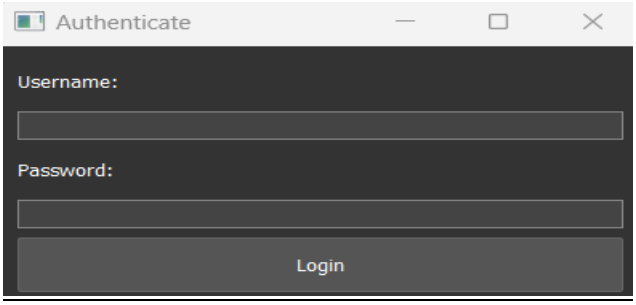


Vous avez le choix pour l'utilisation du serveur. Vous pouvez si vous souhaitez seulement utiliser l'interface graphique ou interface texte mais il faudra Start Server en appuyant sur le bouton de l'interface graphique.

5.1.1Authentification terminale :

```
Veuillez vous authentifier pour effectuer des commandes.
Nom d'utilisateur : Serveur en écoute sur 127.0.0.1:12345
Nouvelle connexion de ('127.0.0.1', 51104)
Client ('127.0.0.1', 51104) a rejoint le salon Général et a reçu l'historique des messages.
Authentification demandée par aq
admin
Mot de passe : password
Authentification réussie. Vous pouvez maintenant exécuter des commandes.
```

5.1.2Authentification graphique :



1. <u>Kick (Exclure) :</u>

- Parfois, il faudra exclure un utilisateur pour diverses raisons. Il faudra taper sur entrée une première fois puis utiliser la commande kick suivie de l'alias de l'utilisateur et suivie de la raison pour le déconnecter temporairement. (pour l'interface il suffit de cliquer sur « kick », puis spécifier l'alias à kick)

2. Ban (Bannir) :

- Pour les cas plus graves, bannir un utilisateur est nécessaire. Cela supprime l'utilisateur de la base de données et empêche toute nouvelle connexion. Utilisez la commande ban avec l'alias de l'utilisateur et suivie de la raison du bannissement. (pour l'interface il suffit de cliquer sur « ban », puis spécifier l'alias à kick)

3. Gérer les Demandes de Salon :

- Les demandes des utilisateurs pour rejoindre les salons privés et validez ou refusez-les ce fait en utilisant ces commandes : accepter 1 ou refuser 1. (Côté graphique : appuyer sur manage participation requests, une nouvelle interface s'ouvre, puis cliquer sur la demande que vous souhaiter pour ensuite accepter ou refuser)

4. Kill (Arrêter le Serveur) :

- En cas d'arrêt du serveur qui va impacter la déconnexion des clients, vous pouvez arrêter le serveur en toute sécurité avec la commande kill. (Côté interface : sélectionner le bouton kill)

5. Warning (Avertissement) :

- Lorsque que vous souhaitez avertir un utilisateur vous devez appuyer sur le bouton warning , spécifier l'alias de l'utilisateur et la raison (cette fonctionnalité n'est pas présent comme commandes terminal)