

Mode d'emploi :
Déploiement et mise en place d'une
application serveur de discussion
interne

Table des matières

I.	Introduction	3
1.	Objectif de la Documentation	3
2.	Prérequis.....	3
II.	Préparation de l'Environnement	4
1.	Création de la Base de Données.....	4
III.	Configuration du Serveur	5
1.	Introduction	5
2.	Configuration des Paramètres Réseau	5
3.	Configuration des Paramètres de Base de Données.....	5
IV.	Configuration du client.....	6
1.	Ajustements des modules et lancement	6
V.	Tests et Validation.....	6
1.	Connexion au serveur	6
2.	Inscription des clients	6
3.	Connexion au client	7
4.	Test de l'Adhésion à un Salon.....	8
5.	Test de l'Envoi et de la Réception de Messages	10

I. Introduction

1. Objectif de la Documentation

Cette documentation a pour objectif de guider les techniciens à travers le processus complet d'installation de l'application de chat, y compris la mise en place de la base de données associée. Les instructions détaillées fournies ici visent à assurer une installation sans heurts et une configuration correcte de l'environnement nécessaire.

Objectifs Spécifiques :

- Fournir une vue d'ensemble claire de l'installation de l'application de chat, en mettant l'accent sur les étapes critiques.
- Détailler les prérequis matériels et logiciels pour garantir une compatibilité optimale.
- Guider les techniciens à travers la configuration de la base de données, y compris la création des tables nécessaires.
- Expliquer la configuration du client et du serveur, en mettant l'accent sur les paramètres réseau et la connexion à la base de données.
- Fournir des instructions détaillées pour les tests et la validation de l'installation.
- Offrir des conseils de maintenance et de dépannage pour résoudre les problèmes potentiels.
- Conclure en récapitulant les points clés et en fournissant des contacts pour un support technique supplémentaire.

Cette documentation est conçue pour être utilisée comme référence complète tout au long du processus d'installation et est destinée aux techniciens responsables de la mise en œuvre de l'application de chat dans un environnement donné.

2. Prérequis

Avant de commencer le processus de déploiement de l'application, assurez-vous que votre environnement satisfait à toutes les exigences nécessaires.

Prérequis Logiciels :

IDE : Il est préférable pour le confort et les fonctionnalités de développement d'utiliser un IDE tel que VsCode ou PyCharm.

Python : Le langage que l'on va utiliser pour développer notre application

MySQL Server : La base de données utilise MySQL.

Client MySQL : Pour faciliter la gestion de la base de données, installez un client MySQL tel que MySQL Workbench.

Connexion Internet : Assurez-vous d'avoir une connexion Internet stable pour télécharger les fichiers source de l'application et installer les dépendances nécessaires.

II. Préparation de l'Environnement

1. Création de la Base de Données

Après avoir installé toutes les dépendances de MySQL et de Python, nous allons procéder à la création de la base de données nécessaire au fonctionnement de l'application de chat. Assurez-vous que votre serveur MySQL est en cours d'exécution et que vous avez les privilèges nécessaires pour créer une nouvelle base de données.

Dans mon cas j'ai deux utilisateurs qui fonctionnent pour effectuer des actions sur cette base de données :

Username	Host	Password
root	localhost	lutC_MySQL !68#
test	localhost	test

On peut soit recréer la base de données à l'aide de ces commandes :

Commandes	Explication
<pre>CREATE TABLE clients (client_id INT AUTO_INCREMENT PRIMARY KEY, username VARCHAR(255) NOT NULL, alias VARCHAR(255) NOT NULL, password_hash VARCHAR(64) NOT NULL, status VARCHAR(20) DEFAULT 'disconnected'); INSERT INTO clients (username, alias, password_hash, status) VALUES ('admin', 'Admin User', 'hashed_admin_password'), ('server', 'Server User', 'hashed_server_password');</pre>	<p>Cette commande crée une table clients pour stocker les informations des utilisateurs.</p> <ul style="list-style-type: none"> client_id est une clé primaire auto-incrémentée. username est le nom d'utilisateur unique. alias est l'alias de l'utilisateur. password_hash stocke le hash du mot de passe. status indique le statut de connexion de l'utilisateur (par défaut à "disconnected"). <p>On ajoute de base deux utilisateurs server et admin pour la gestion des commandes sur le serveur. Le reste des utilisateurs pourra se créer après dans l'interface du client avec la fenêtre d'inscription. On les supprimera ultérieurement pour les réinscrire avec le client afin qu'il aient un vrai password hashé.</p>
<pre>CREATE TABLE salons (salon_id INT AUTO_INCREMENT PRIMARY KEY, nom_salon VARCHAR(255) NOT NULL); INSERT INTO salons (nom_salon) VALUES ('Général'), ('Blabla'), ('Comptabilité'), ('Informatique'), ('Marketing');</pre>	<p>Cette commande crée une table salons pour stocker les informations sur les salons.</p> <ul style="list-style-type: none"> salon_id est une clé primaire auto-incrémentée. nom_salon contient les noms des salons, et des salons prédéfinis sont insérés. <p>Pour les salons on ne peut pas les créer après donc on les instancie directement. Ils sont fixes donc pas besoin de les supprimer ni d'en rajouter.</p>
<pre>CREATE INDEX idx_username ON clients(username); CREATE TABLE salons_adheres (id INT AUTO_INCREMENT PRIMARY KEY, username VARCHAR(255) NOT NULL, salon_id INT NOT NULL, FOREIGN KEY (username) REFERENCES clients(username), FOREIGN KEY (salon_id) REFERENCES salons(salon_id));</pre>	<p>Cette commande crée un index sur la colonne username dans la table clients, ce qui peut améliorer les performances lors de la recherche par nom d'utilisateur. Cette commande crée une table salons_adheres pour enregistrer les adhésions des utilisateurs aux salons.</p> <ul style="list-style-type: none"> id est une clé primaire auto-incrémentée. username est une clé étrangère, ref à la table clients. salon_id est une clé étrangère, ref à la table salons.
<pre>CREATE TABLE messages (message_id INT AUTO_INCREMENT PRIMARY KEY, client_id INT NOT NULL, salon_id INT NOT NULL, contenu TEXT, timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,</pre>	<p>Cette commande crée une table messages pour stocker les messages envoyés par les utilisateurs.</p> <ul style="list-style-type: none"> message_id est une clé primaire auto-incrémentée. client_id est une clé étrangère faisant référence à la table clients. salon_id est une clé étrangère faisant référence à la table salons.

FOREIGN KEY (client_id) REFERENCES clients(client_id), FOREIGN KEY (salon_id) REFERENCES salons(salon_id));	<ul style="list-style-type: none"> • contenu stocke le texte du message. • timestamp enregistre le moment où le message a été créé (par défaut à la date et l'heure actuelles).
--	---

Soit on importe le fichier sql :

Sous Windows : `mysql -u[utilisateur] -p [nom_base_de_donnees] < fichier.sql`

Sous Linux : `mysql nom_base_de_donnees < fichier.sql`

III. Configuration du Serveur

1. Introduction

On peut maintenant commencer à toucher à l'application. Tout d'abord on va devoir télécharger le fichier `Server.py` dans le GitHub. Ensuite on l'ouvre ainsi que son dossier dans l'IDE.

2. Configuration des Paramètres Réseau

Dans cette section, nous allons nous assurer que les paramètres réseau du serveur. Étant donné que nous utilisons tout en localhost, cela signifie que le serveur sera exécuté sur la même machine que la base de données et le client.

```
self.server_address = ('localhost', 5000)
```

Il faut s'assurer que le port est disponible et qu'il n'y a pas d'erreurs lors du lancement du serveur.

3. Configuration des Paramètres de Base de Données

Avant de déployer l'application de chat, nous devons configurer les paramètres de connexion à la base de données du serveur. Ces étapes garantiront que l'application peut interagir correctement avec la base de données MySQL.

On retrouve cette configuration dans la fonction `__init__` de la classe `Server`.

```
self.db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': 'IutC_MySQL!68#',
    'database': 'sae'
}
```

Il faut bien s'assurer d'adapter ces quatre options à votre configuration MySQL, sinon cela ne fonctionnera pas !

Après ceci le server est désormais prêt à être lancé mais avant on va s'occuper des clients.

IV. Configuration du client

1. Ajustements des modules et lancement

Les codes clients sont dans le même dossier que le Server.py ouvert précédemment. On les retrouve sous le nom de Client1.py et Client2.py.

Il n'y a pas de configuration de code à vérifier mais au lancement certains modules de bibliothèque ne sont pas installés par défaut. Il suffit alors d'exécuter dans la console un pip install [nom du module] et celui-ci sera opérationnel.

V. Tests et Validation

1. Connexion au serveur

Dès le lancement du server on nous demande dans la console de nous authentifier. Cette authentification est nécessaire au lancement du server et permet de s'assurer que le server a été démarré par l'utilisateur server ou administrateur. Cela permet également de protéger la console car seuls ces utilisateurs doivent être capables d'utiliser des commandes permettant de kick un utilisateur (le déconnecter du serveur et empêcher sa reconnexion pendant un délai de 5min), de le ban (le déconnecter définitivement, son compte est supprimé), d'arrêter le serveur avec un kill ou d'accepter et rejeter une demande d'adhésion du client pour un salon défini.

Pour se connecter il suffit de se s'authentifier en tant que serveur (username : server ; password : server) ou en tant qu'administrateur (username : admin ; password : admin)

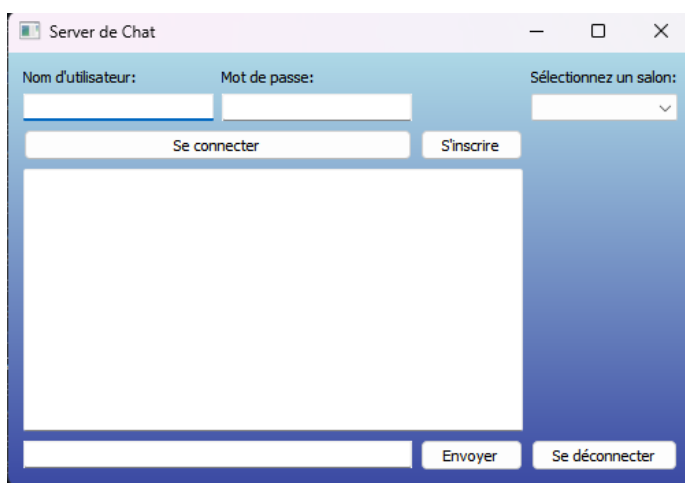
```
Nom d'utilisateur : server
Mot de passe : server
Authentification réussie pour l'utilisateur server.
Serveur en attente de connexions...
Tapez une commande (KICK/BAN/KILL/ACCEPT/REJECT):
```

A partir de cette étape, les clients peuvent se connecter

2. Inscription des clients

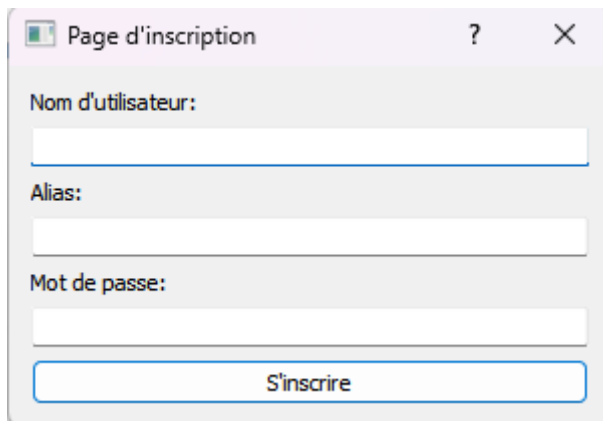
Donc on peut maintenant se rendre sur un client et le lancer.

On se retrouve alors avec cette interface :



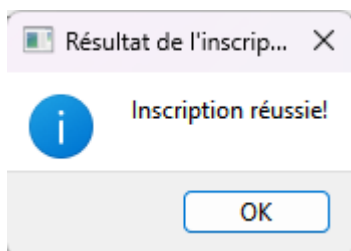
A partir de là on peut déjà se connecter avec les deux utilisateurs admin et server

On peut également créer son premier utilisateur en cliquant sur le bouton « s'inscrire ».



Celui nous ouvre cette fenêtre d'inscription nous permettant de renseigner le nom de l'utilisateur, son alias ainsi que son mot de passe. Il y a cependant quelques règles à respecter, sinon cela va générer des erreurs. On n'a pas le droit d'avoir deux utilisateurs avec le même nom ou avec le même alias.

Si l'inscription se déroule comme prévu, cette fenêtre de confirmation s'affichera



Dès que ce message apparaît on s'aperçoit que notre utilisateur a bien été intégré à la liste des clients dans la base de données

Si vous avez importé le fichier de base de données les utilisateurs admin et server sont déjà configurés. A l'inverse si vous avez récréé la base de données, il faudra penser à recréer les utilisateurs server et admin pour qu'ils aient un mot de passe hashé.

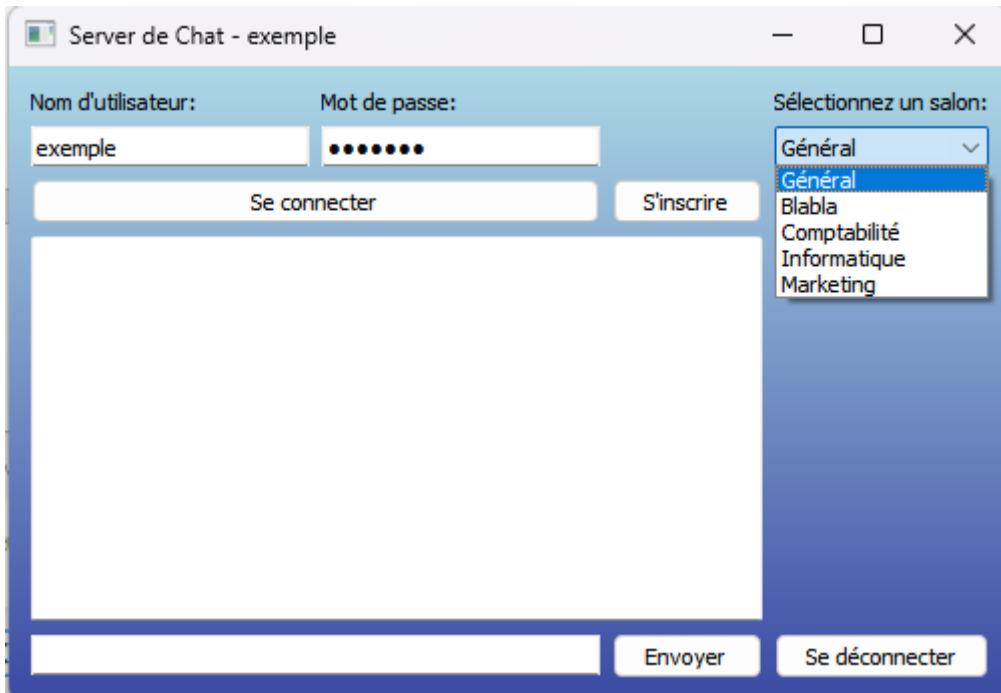
Et ainsi penser à supprimer les anciens pour ne pas créer des problèmes de sécurité.

```
DELETE FROM Clients WHERE client_id = 1 ;
```

```
DELETE FROM Clients WHERE client_id = 2 ;
```

3. Connexion au client

Une fois qu'on a fait ça on va pouvoir se connecter avec le nouvel utilisateur que l'on vient de créer dans les sections Nom d'utilisateur et Mot de passe.



On peut voir qu'après la connexion établie, on a la liste des salons qui s'est affichée en haut à gauche. Par défaut, chaque utilisateur dès son inscription a adhéré au salon général. De ce fait il peut désormais échanger des messages avec d'autres utilisateurs dans ce salon.

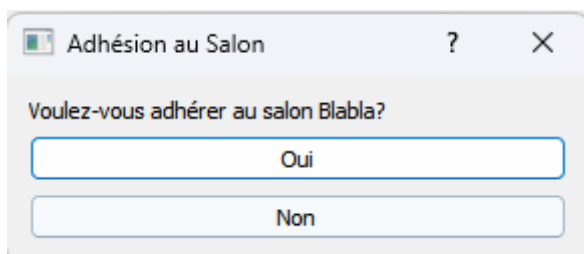
Mais ce n'est pas tout, dès la connexion réussie, l'utilisateur voit son statut changer en connected.

client_id	username	alias	password_hash	status
1	admin	admin	8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a918	disconnected
2	toto	toto	31f7a65e315586ac198bd798b6629ce4903d0899476d5741a9f32e2e521b6a66	disconnected
3	server	server	b3eacd33433b31b5252351032c9b3e7a2e7aa7738d5decdf0dd6c62680853c06	disconnected
5	titi	titi	cce66316b4c1c59df94a35afb80cecd82d1a8d91b554022557e115f5c275f515	disconnected
9	toto	test	9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08	disconnected
11	test	ax	9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08	disconnected
13	exemple	exemple	b77ad2edf199a4b4479347afaf387d435a12480d0dc5b8e042c4c8991587d080	connected

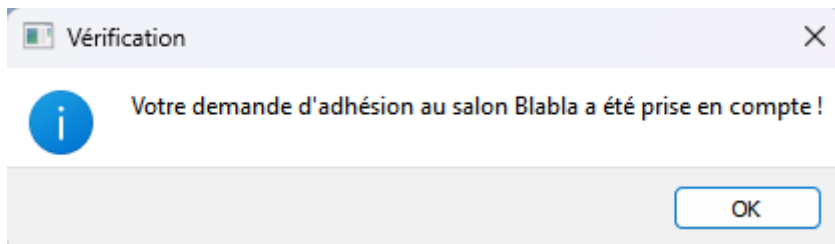
4. Test de l'Adhésion à un Salon

Maintenant ce qu'on aimerait c'est de pouvoir accéder à d'autres salons

Donc on va appuyer sur le salon Blabla



Pour chaque salon auquel l'utilisateur connecté n'a pas adhéré, cette fenêtre s'ouvre pour lui demander s'il souhaite l'intégrer. S'il répond non, le salon reste dans le même état, c'est-à-dire, l'utilisateur ne peut ni voir ni écrire de messages dans ce salon. A l'inverse s'il répond oui, il aura ce message :



Et dans le serveur, cette ligne sera affichée dans la console :

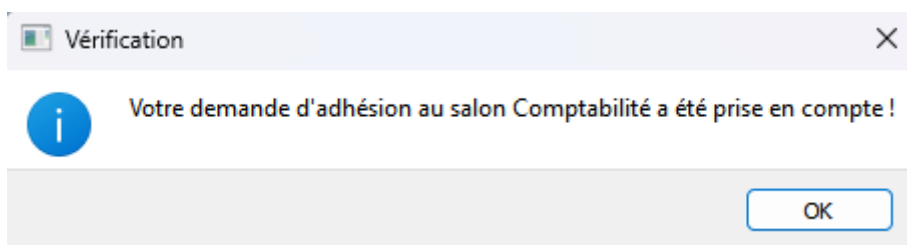
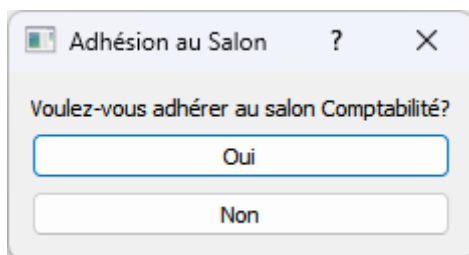
```
Utilisateur exemple ajouté au salon 2.
```

Et nous voici intégré dans le salon Blabla !

Attention ! : Il est parfois possible que la synchronisation ne se fasse pas toujours et qu'il faille se déconnecter puis se reconnecter pour accéder au salon.

Afin de rejoindre les trois autres salons restant c'est un peu différent mais c'est le même principe pour les trois.

On retrouve toujours cette fenêtre de demande d'adhésion :



Mais cette fois ci quand on clique sur « oui », on a une demande d'adhésion qui est envoyée au serveur, mais avant d'accéder au salon, soit l'utilisateur server ou admin, dépend de qui est connecté sur le server doit accepter ou refuser la demande du client à l'aide des commandes « ACCEPT:username:nom_salon » ou « REJECT:username:nom_salon »

Par exemple :

```
Join request from exemple for salon Comptabilité.  
  
Tapez une commande (KICK/BAN/KILL/ACCEPT/REJECT): ACCEPT:exemple:Comptabilité  
Utilisateur exemple ajouté au salon 3.  
Tapez une commande (KICK/BAN/KILL/ACCEPT/REJECT): L'utilisateur exemple est déjà adhérent au salon Blabla.  
Join request from exemple for salon Informatique.  
  
Tapez une commande (KICK/BAN/KILL/ACCEPT/REJECT): REJECT:exemple:Informatique  
Utilisateur exemple retiré du salon 4.
```

Et dans le client on a ça :

Nom d'utilisateur: exemple Mot de passe: Sélectionnez un salon: Informatique

Se connecter S'inscrire

exemple a rejoint le salon Comptabilité.
Votre demande d'adhésion au salon Informatique a été refusée. Accès restreint

Envoyer Se déconnecter

La base de données stocke les utilisateurs et les salons dans un table `salon_adheres` et permet qu'à chaque reconnexion du client, il soit directement intégré au salon et qu'il n'ait plus besoin de faire de demande d'adhésion.

```
Connexion établie avec exemple.  
Salons adhérents: [{"salon_id": 1, "salon_name": "Général"}, {"salon_id": 2, "salon_name": "Blabla"}, {"salon_id": 3, "salon_name": "Comptabilité"}]
```

5. Test de l'Envoi et de la Réception de Messages

On peut voir qu'à l'ajout d'un autre utilisateur, chacun de son côté reçoit les messages et peut en envoyer.

Server de Chat - exemple

Nom d'utilisateur: exemple Mot de passe: Sélectionnez un salon: Général

Se connecter S'inscrire

exemple dans le salon Général: Bonjour !
admin dans le salon Général: Salut !

Envoyer Se déconnecter

Server de Chat - admin

Nom d'utilisateur: admin Mot de passe: Sélectionnez un salon: Général

Se connecter S'inscrire

exemple dans le salon Général: Bonjour !
admin dans le salon Général: Salut !

Envoyer Se déconnecter

La base de données est également au courant de chaque message envoyé par tel et tel utilisateur et dans quel salon :

```
mysql> select * from messages;
```

message_id	client_id	salon_id	contenu	timestamp
1	13	1	Bonjour !	2023-12-31 17:46:59
2	1	1	Salut !	2023-12-31 17:47:05