Documentation de l'Application de Communication en Python

Installation et Configuration du Client

Prérequis

- Python 3.12 installé sur votre machine.
- PyQt5 et les autres dépendances nécessaires installées. Vous pouvez les installer via `pip` en exécutant :
  Copy code
  ```
  pip install PyQt5 mysql-connector-python bcrypt
  ```

Installation

1. **Télécharger le code source** : Téléchargez les fichiers source de l'application depuis leur emplacement de stockage (par exemple, un dépôt GitHub).
2. **Configuration de la base de données** : Assurez-vous que MySQL est installé sur votre machine et créez une base de données pour l'application.
3. **Configurer les Paramètres de Connexion** : Modifiez les paramètres de connexion dans la classe `DatabaseManager` pour correspondre à votre configuration de base de données MySQL.

Lancement de l'Application

- Ouvrez un terminal ou une invite de commande.
- Naviguez jusqu'au dossier contenant les fichiers de l'application.
- Exécutez le script principal `python client_main.py`.

1. Classe `Client`

Gère la communication réseau côté client pour une application de chat.

Attributs :

- `message_received (pyqtSignal)`: Signal pour la réception d'un message.
- `connection_failed (pyqtSignal)`: Signal pour une erreur de connexion.
- `connection_success (pyqtSignal)`: Signal pour une connexion réussie.
- `formatted_message_received (pyqtSignal)`: Signal pour les messages formatés reçus.
- `connection_closed (pyqtSignal)`: Signal pour la fermeture de la connexion.

Méthodes :

- `__init__(self, username, host, port)`: Initialise le client avec un nom d'utilisateur, une adresse hôte et un port.
- `connect_to_server(self)`: Connecte le client au serveur et lance la réception des messages.
- `receive_messages(self)`: Reçoit les messages du serveur en continu.
- `send_messages(self, message)`: Envoie un message au serveur.
- `close_connection(self)`: Ferme la connexion avec le serveur.

2. Classe `ClientUI` (Hérite de `QMainWindow`)

Interface utilisateur pour le client de chat.

Attributs:

- `client_logic (Client)`: Instance de la logique client.
- `textAreas (dict)`: Zones de texte pour chaque canal de chat.

Méthodes :

- `__init__(self, username)`: Initialise l'UI avec la logique client spécifiée.
- `initUI(self)`: Initialise l'UI avec des onglets pour différents canaux de chat.
- `createChannelTab(self, channel_name)`: Crée un onglet pour un canal de chat.
- `eventFilter(self, obj, event)`: Filtre les événements clavier pour l'envoi de messages.
- `sendMessage(self, channel_name, message, textArea, inputField)`: Envoie un message au serveur et met à jour l'UI.
- `logMessage(self, message)`: Affiche un message reçu du serveur.
- `connect_client_signals(self)`: Connecte les signaux du client aux slots appropriés.
- `close_client(self)`: Ferme l'UI du client.
- `setupClient(self, client)`: Configure le client avec la logique existante.
- `onConnectionClosed(self)`: Gère la fermeture de la connexion avec le serveur.

3. Classe `Login` (Hérite de `QDialog`)

Interface utilisateur pour la fenêtre de connexion.

Attributs :

- `stacked_widget (QStackedWidget)`: Widget empilé pour la navigation.
- `client (Client)`: Instance client pour la connexion au serveur.

Méthodes :

- `__init__(self, stacked_widget)`: Initialise l'interface de connexion.
- `loginfunction(self)`: Gère la tentative de connexion.
- `on_connection_failed(self, error_message)`: Gère l'échec de la connexion.
- `on_connection_success(self)`: Gère une connexion réussie au serveur.
- `isValidLogin(self)`: Vérifie si les identifiants sont valides.
- `gotocreate(self)`: Dirige vers la fenêtre de création de compte.

4. Classe `CreateAcc` (Hérite de `QDialog`)

Interface utilisateur pour la fenêtre de création de compte.

Attributs :

- `stacked_widget (QStackedWidget)`: Widget empilé pour la navigation.

Méthodes :

- `__init__(self)`: Initialise l'interface de création de compte.
- `createaccfunction(self)`: Gère la création du compte utilisateur.
- `go_to_login(self)`: Ramène à l'écran de connexion.
- `set_stacked_widget(self, stacked_widget)`: Configure le widget empilé.
- `validate_credentials(self, username, password, confirm_password)`: Valide les informations d'inscription.
- `username_exists(self, username)`: Vérifie si le nom d'utilisateur existe déjà.