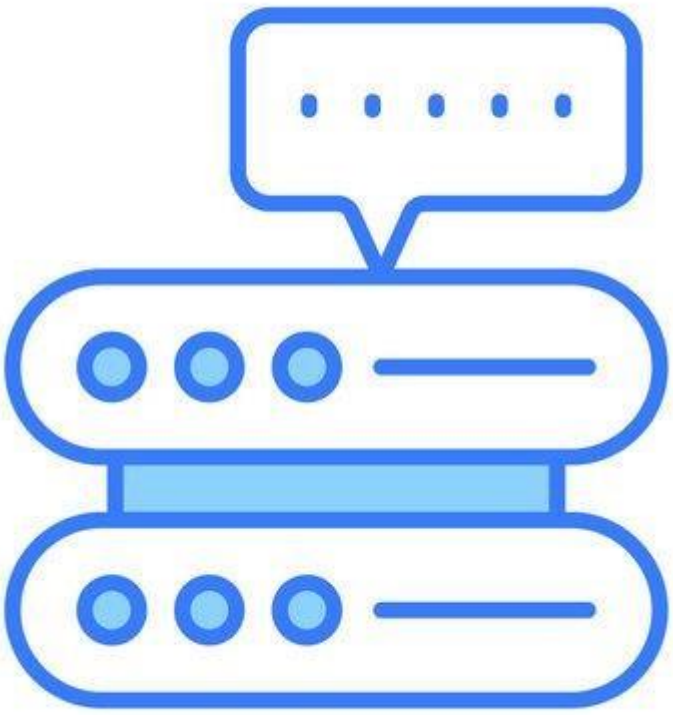
Louison FINK

SAE 302

Un serveur de discussion interne



Aperçu de la fonctionnalité du client :

Le client est conçu pour faciliter la communication dans un environnement de chat. Il utilise la bibliothèque PyQt5 pour l'interface utilisateur graphique (GUI) et la programmation de sockets pour la communication réseau. Le client permet aux utilisateurs de saisir un nom d'utilisateur, de se connecter au serveur, d'envoyer des messages et de consulter l'historique du chat.

Fonctionnalités détaillées du client :

1. **Initialisation :**
   - Le client initialise un socket en utilisant le module **`socket`**.
   - Il se connecte à l'adresse du serveur (**`("127.0.0.1", 1024)`**) pour la communication.

2. **Configuration de l'interface utilisateur :**
   - Le client invite l'utilisateur à saisir un nom d'utilisateur à l'aide d'une boîte de dialogue (**`QInputDialog`**).
   - La fenêtre principale de l'interface graphique est créée en utilisant **`QWidget`** et comprend un **`QTextEdit`** pour afficher les messages, un **`QLineEdit`** pour saisir les messages et un **`QPushButton`** pour envoyer les messages.

3. **Connexion au serveur :**
   - Le client se connecte au serveur en utilisant le socket initialisé.
   - Il envoie le nom d'utilisateur saisi au serveur pour l'identification.

4. **Envoi de messages :**
    - Lorsque l'utilisateur clique sur le bouton "Envoyer", le client envoie le message saisi au serveur à l'aide du socket.
    - Le champ de saisie utilisateur est effacé après l'envoi d'un message.

5. **Réception de messages :**
    - Le client reçoit continuellement des messages du serveur en utilisant un thread séparé (`ClientThread`).
    - Les messages reçus sont émis sous forme de signaux pour mettre à jour l'interface graphique (`fonction updateTextArea`).

6. **Affichage des messages :**
    - L'interface graphique affiche les messages entrants dans le widget `QTextEdit`, permettant à l'utilisateur de consulter la conversation.

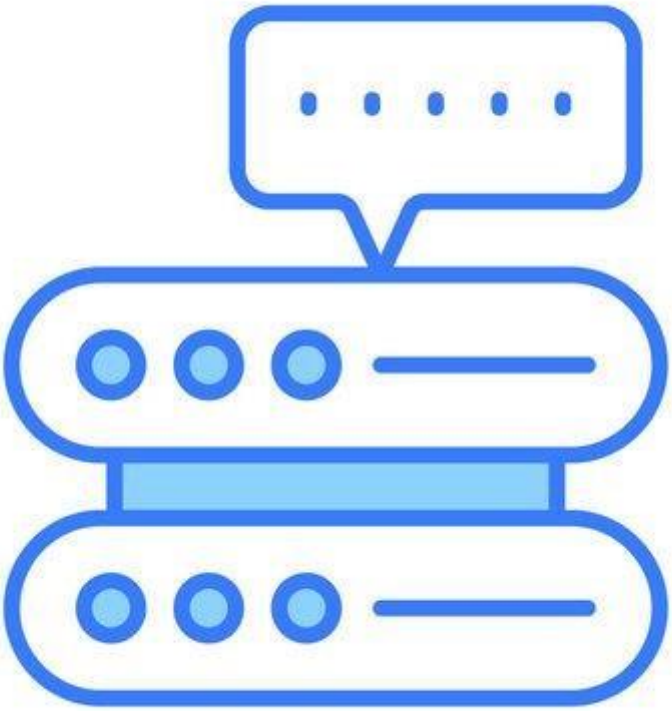
7. **Threads de l'interface utilisateur graphique :**
    - Le client utilise `QThread` pour gérer les threads de l'interface graphique et du réseau séparément.
    - Le `ClientThread` s'exécute en arrière-plan pour gérer la réception des messages.

8. **Gestion des exceptions :**
    - Le client inclut une gestion des exceptions pour gérer les erreurs lors de la réception des messages ou des mises à jour de l'interface utilisateur.

Conclusion :

Le client établit avec succès une connexion avec le serveur, permet aux utilisateurs d'envoyer et de recevoir des messages, et offre une interface graphique de chat simple et intuitive. L'utilisation de PyQt5 simplifie le développement de l'interface utilisateur graphique, tandis que la programmation de sockets assure une communication efficace avec le serveur.

An internal chat server



Overview of the client's functionality:

The client is designed to facilitate communication in a chat environment. It utilizes the PyQt5 library for the graphical user interface (GUI) and socket programming for network communication. The client allows users to enter a username, connect to the server, send messages, and view the chat history.

Detailed functionality of the client:

1. **Initialization:**
   - The client initializes a socket using the **socket** module.
   - It connects to the server's address (**("127.0.0.1", 1024)**) for communication.

2. **User Interface Setup:**
   - The client prompts the user to enter a username using a dialog box (**QInputDialog**).
   - The main GUI window is created using **QWidget** and includes a **QTextEdit** for displaying messages, a **QLineEdit** for entering messages, and a **QPushButton** for sending messages.

3. **Connection to the Server:**
   - The client connects to the server using the initialized socket.
   - It sends the entered username to the server for identification.

4. **Sending Messages:**
    - When the user clicks the "Send" button, the client sends the entered message to the server using the socket.
    - The user input field is cleared after sending a message.

5. **Receiving Messages:**
    - The client continuously receives messages from the server using a separate thread (`ClientThread`).
    - Received messages are emitted as signals to update the GUI (`updateTextArea` function).

6. **Displaying Messages:**
    - The GUI displays incoming messages in the `QTextEdit` widget, allowing the user to view the conversation.

7. **Graphical User Interface Threads:**
    - The client uses `QThread` to manage the GUI and networking threads separately.
    - The `ClientThread` runs in the background to handle message reception.

8. **Handling Exceptions:**
    - The client includes exception handling to manage errors during message reception or UI updates.

Conclusion:

The client successfully establishes a connection with the server, allows users to send and receive messages, and provides a simple and intuitive graphical interface for chat interaction. The use of PyQt5 simplifies GUI development, while socket programming ensures effective communication with the server.