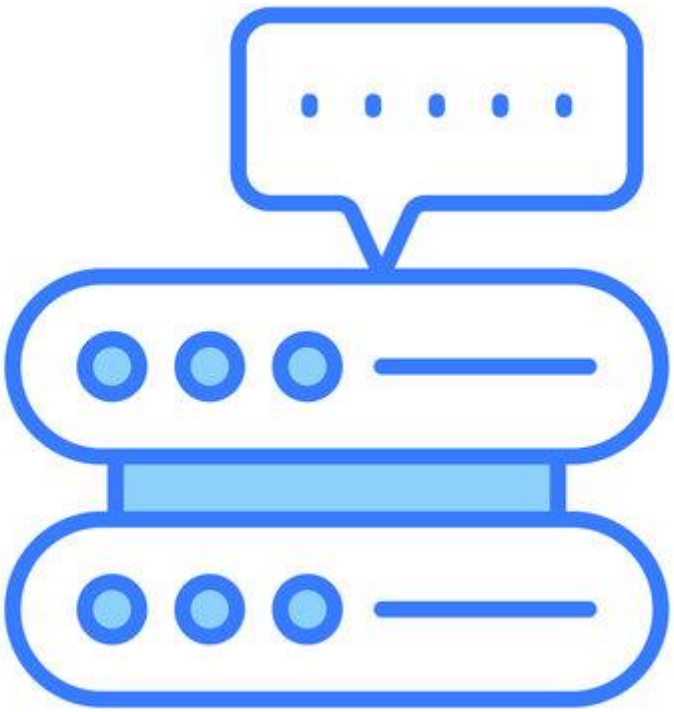
SAE 302

Un serveur de discussion interne



Aperçu du fonctionnement du serveur :

Le serveur est conçu pour faciliter la communication entre plusieurs clients dans un environnement de chat. Il utilise la programmation par sockets pour la communication réseau et intègre le multithreading pour gérer simultanément plusieurs connexions clientes. Le serveur garde une trace des clients connectés, diffuse des messages à tous les clients et enregistre l'historique du chat.

Fonctionnement détaillé du serveur :

1. **Initialisation :**
   - Le serveur initialise un socket en utilisant le module **socket**.
   - Il lie le socket à l'adresse **('0.0.0.0', 1024)** pour écouter sur toutes les interfaces disponibles et le port 1024.
   - Le serveur entre dans un état d'écoute, en attente des connexions clientes entrantes.
2. **Gestion des connexions clientes :**
   - Lorsqu'un client se connecte, un nouveau thread est créé pour gérer la communication avec ce client.
   - L'adresse IP du client est vérifiée par rapport à une liste noire pour déterminer si la connexion est autorisée.
3. **Nom d'utilisateur et message de connexion :**
   - Le serveur reçoit le nom d'utilisateur du client.
   - Si l'adresse IP du client est autorisée, le serveur diffuse un message de connexion à tous les clients connectés, indiquant l'arrivée d'un nouvel utilisateur.
4. **Envoi du journal de chat au nouveau client :**
   - Le serveur envoie le journal de chat (s'il est disponible) au nouveau client connecté.
   - Le journal de chat est stocké dans un fichier appelé "chat_log.txt".
5. **Gestion des messages :**
   - Le serveur reçoit continuellement des messages des clients.

- Si le message est "stop", le serveur se termine.
- Si le message est "bye", le serveur note la déconnexion du client.
- Sinon, le serveur enregistre le message et le diffuse à tous les clients connectés.

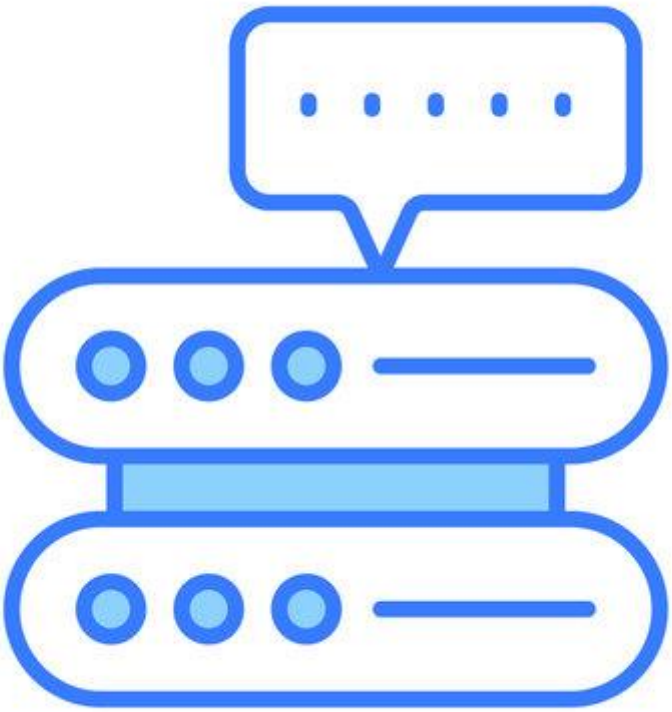
6. **Fichier de journal de chat :**
    - Chaque message reçu est ajouté au fichier "chat_log.txt".
    - Ce fichier sert d'enregistrement persistant de l'historique du chat

7. **Déconnexion du client :**
    - Lorsqu'un client se déconnecte, le serveur supprime les informations du client de la liste des clients connectés.
    - Le serveur ferme la connexion pour le client déconnecté.

Conclusion :

Le serveur gère avec succès les connexions clientes, relaie les messages entre les clients et conserve un journal de chat. L'utilisation du multithreading garantit que plusieurs clients peuvent interagir simultanément. En enregistrant les messages dans un fichier, le serveur préserve l'historique du chat entre les sessions. Globalement, le serveur fournit une base solide pour une application de chat multi-client.

An internal chat server



Server Operation Overview:

The server is designed to facilitate communication between multiple clients in a chat-like environment. It utilizes socket programming for network communication and incorporates threading to handle multiple client connections simultaneously. The server keeps track of connected clients, broadcasts messages to all clients, and logs the chat history.

Detailed Server Operation:

Initialization:

The server initializes a socket using the socket module.

It binds the socket to the address ('0.0.0.0', 1024) to listen on all available interfaces and port 1024.

The server enters a listening state, awaiting incoming client connections.

Client Connection Handling:

When a client connects, a new thread is spawned to handle the communication with that client.

The client's IP address is checked against a blacklist to determine if the connection is allowed.

Username and Join Message:

The server receives the username from the client.

If the client's IP is allowed, the server broadcasts a join message to all connected clients, indicating that a new user has joined.

Sending Chat Log to New Client:

The server sends the chat log (if available) to the newly connected client.

The chat log is stored in a file named "chat_log.txt."

Message Handling:

The server continuously receives messages from clients.

If the message is "stop," the server terminates.

If the message is "bye," the server notes the client's disconnection.

Otherwise, the server logs the message and broadcasts it to all connected clients.

Chat Log File:

Each received message is appended to the "chat_log.txt" file.

This file serves as a persistent record of the chat history.

Client Disconnection:

When a client disconnects, the server removes the client's information from the list of connected clients.

The server closes the connection for the disconnected client.

Conclusion:

The server successfully manages client connections, relays messages among clients, and maintains a chat log. The use of threading ensures that multiple clients can interact concurrently. By logging messages to a file, the server preserves the chat history between sessions. Overall, the server provides a robust foundation for a multi-client chat application.