**Fiche Technique de server.py**

**Nom du Fichier :**

server.py

**Description Générale :**

Ce fichier implémente un serveur de chat permettant à plusieurs clients de se connecter, d'envoyer des messages, et de recevoir des messages diffusés par les autres utilisateurs connectés. Il utilise les sockets pour la communication réseau et des threads pour gérer plusieurs connexions simultanément.

**Structure du Code :**

1. **Fonctions Utilitaires**
   * get\_ip()
   * keygen(chaine: str) -> str
   * keygenRev(key: str) -> str
2. **Classe Principale**
   * ChatServer
     + Gestion des connexions des clients.
     + Diffusion des messages.
     + Hébergement d'un serveur TCP.

**Fonctions :**

1. **get\_ip()**
   * Récupère l'adresse IP locale de la machine.
   * Retourne l'adresse IPv4 si possible, ou "127.0.0.1" par défaut.
2. **keygen(chaine: str) -> str**
   * Encode une chaîne en Base64.
3. **keygenRev(key: str) -> str**
   * Décodage inverse de keygen.
   * Retourne les données d'origine d'une chaîne encodée en Base64 par keygen.

**Classe : ChatServer**

* **Rôle** : Fournit les fonctionnalités de base pour un serveur de chat multi-clients.

**Attributs :**

* host : Adresse IP du serveur.
* port : Port d'écoute du serveur.
* server : Socket utilisé pour gérer les connexions.
* clients : Liste des clients connectés.
* pseudo : Liste des pseudonymes associés aux clients.

**Méthodes Principales :**

1. **\_\_init\_\_(host, port)**
   * Initialise le serveur avec une adresse IP et un port.
2. **start()**
   * Démarre le serveur.
   * Accepte les connexions entrantes.
   * Lance un thread pour chaque client connecté.
3. **handle\_client(client\_socket, address)**
   * Gère la communication avec un client particulier (client\_socket).
   * Traite les messages reçus et les diffuse.
4. **broadcast(message, sender\_socket)**
   * Diffuse un message à tous les clients sauf l'expéditeur.
5. **send(message, destination)**
   * Envoie un message spécifique à un client donné (destination).
6. **getClients()**
   * Retourne la liste des sockets des clients connectés.
7. **getPseudo()**
   * Retourne la liste des pseudonymes des clients connectés.

**Caractéristiques Techniques :**

* **Protocole** : TCP/IP via la bibliothèque socket.
* **Encodage des Messages** : UTF-8 pour la transmission et Base64 pour l'encodage optionnel des données.
* **Gestion de Concurrence** : Multithreading via la bibliothèque threading.
* **Portabilité** : Compatible avec toutes les plateformes supportant Python et les sockets (Windows, Linux, macOS).

**Dépendances :**

* **Standard Libraries :**
  + socket : Pour la gestion des communications réseau.
  + threading : Pour gérer plusieurs connexions simultanément.
  + base64 : Pour encoder et décoder les chaînes.