ChatHack
Client et Serveur

MANUEL UTILISATEUR

Mise en route

Le serveur doit se connecter à un serveur de base de données sur le port **7777** pour pouvoir authentifier les clients.
Pour cela il faut avant tout lancer le *ServerMDP.jar* (non inclus dans le projet) sur le port **7777** en lui indiquant un fichier contenant les identifiants et mots de passe.

```
java -jar ServerMDP 7777 password.txt
```

Ensuite le serveur ChatHack peut être lancé avec la commande :

```
java -jar chatHack_server-1.0.jar port
```

Où '*port*' correspond à un numéro de port disponible.

Une fois le serveur démarré, un ou plusieurs clients peuvent s'y connecter en exécutant le fichier correspondant :

```
java -jar chatHack_client-1.0.jar adresse port repertoire login [mot de passe]
```

Où '*adresse*' est l'adresse du serveur, '*port*' son numéro de port, '*repertoire*' le chemin vers un dossier dans lequel seront téléchargés et uploadés les fichiers, '*login*' le pseudonyme du client et '*mot de passe*' un éventuel mot de passe.
Si le mot de passe est renseigné, le client sera authentifié si son login/mot de passe correspond à une entrée dans la base de données.
S'il ne le renseigne pas, il sera authentifié si le login n'existe pas déjà dans la base et qu'aucun autre client n'utilise le même login.

Utilisation

Client

Messages publics et messages privés

Une fois connecté et authentifié au serveur, l'utilisateur peut envoyer un message public au serveur qui sera diffusé à tous les autres clients authentifiés, en ne spécifiant ni '@' ni '/' au début du message.

Pour envoyer un message privé :

```
@login message
```

Où '*login*' est le login du client destinataire, authentifié au serveur, et '*message*' le contenu du message à envoyer.

Pour envoyer un fichier :
`/login nom_fichier`

Où '*login*' est le login du client destinataire, authentifié au serveur, et '*nom_fichier*' le nom du fichier (situé dans le répertoire indiqué lors du lancement) à envoyer.

Établissement d'une communication privée

Si un client A sollicite un autre client B pour la première fois, le client B devra d'abord accepter d'établir une communication privée avec A.
B recevra un message de demande de communication privée.
S'il refuse la communication, il devra saisir :
`@login no`
ou
`/login no`

Où '*login*' est le login du client demandeur.
Si B décide d'accepter, il devra répondre à A (avec '@' ou '/') avec un autre « message » que '*no*'.

Tant que B n'accepte pas la demande de A, les messages et fichiers de A sont mis en attente avant d'être envoyé à B. (Ou détruits si B refuse la demande).

Envoi de fichiers

Si la communication privée est établie entre deux clients, ils peuvent s'échanger des fichiers. Plusieurs en même temps si souhaitée, sans bloquer le reste.

L'envoi de fichier est limité à 2Go par fichier *(car en fait le champ qui code la taille du fichier dans la trame est un entier signé sur 4 octets, ce qui borne le maximum à $2^{32}/2$)*.

Si une erreur a lieu pendant l'envoi du fichier, ou pendant la réception (parce que le fichier en train d'uploader/de télécharger est supprimé par exemple), l'upload/téléchargement s'interrompt, sans provoquer un arrêt du client.

Le suivi de toutes les informations du client sont stockées dans un fichier *client_log.log* généré à l'emplacement où le client a été exécuté.

Serveur

L'utilisateur peut interagir avec le serveur en saisissant 3 commandes :

Info : Informe sur le nombre de clients authentifiés sur le serveur.
Shutdown : Empêche des nouveaux clients de se connecter au serveur.
Shutdownnow : Stop le serveur et déconnecte tous les clients.

Interrompre le serveur ne rompt pas la connexion entre les clients qui se sont connectés entre eux.

Le suivi de toutes les informations du serveur sont stockées dans un fichier *server_log.log* généré à l'emplacement où le serveur a été exécuté.

Pas de bug n'est à signaler.