Chat

Table of Contents

Introduction

Ce projet est un chat pour le projet de SAE dans le cadre du diplome de BUT informatique. Il s'agit d'un chat simple avec un serveur et un client.

Quoi installer

Pour lancer ce projet, vous devez installer java et javaFX.

Lancer

Lancer le serveur avec la commande suivante :

```
java -jar Serveur.jar
```

Sur un autre terminal, lancer le client avec la commande suivante :

```
java --module-path "{JAVAFX_DIR}\lib" --add-modules javafx.controls,javafx.fxml -
cp bin launch.ChatApplication
```

Une fois le client lancé, vous pouvez vous connecter avec un pseudo. Et utiliser la commande /help pour voir les commandes disponibles. Pour ce connecter sur un autre ordinateur sur le même réseau, lancer `/ip` pour avoir l'adresse ip du serveur.
Ensuite, lancer le client sur l'autre ordinateur et quand l'adresse ip est demandée, entrer l'adresse ip obtenue grâce au client local.

Vous pouvez maintenant découvrir le chat.

Fonctionnalitées

Ip inconnue

On peut renseigner l'ip du serveur si elle n'est pas connu.

Nom d'utilisateur

Le nom d'utilisateur est unique. Si un utilisateur tente de se connecter avec un nom déjà utilisé, il lui est demandé d'en choisir un autre.

Salon Général

Ce salon permet de discuter avec tous les utilisateurs connectés. Il est accessible par défaut.Et envoie le message à tous les autres salons.

Home

Quand clique sur le bouton home, on revient en mode chat salon.

new salon

Nous avons décidé que les salons sont créés par les utilisateurs. Quand appui sur bouton "nouveau salon", on saisie le nom du salon que l'on veut créer. Si il n'est pas déjà pris cela le créer et on est ajouté dedans et déconnecté de l'ancien salon.

Tous les salons

Quand appui sur bouton "tous les salons", on peut voir tous les salons existants. On peut cliquer sur un salon pour y accéder.

Users connectés

Quand on appui sur le bouton "Messages privés", on peut voir tous les utilisateurs connectés. On peut cliquer sur un utilisateur pour lui envoyer un message privé. Avec la commande `/nbuser` on peut voir le nombre d'utilisateurs connectés.

Message privé

Pour envoyer le premier message privé à une personne cliquer sur "Messages privés" puis sur le nom de l'utilisateur avec qui on veut discuter. La personne reçois un message le bouton message privé devient rouge. Et le nom de la perssonne qui à envoyer aussi. Pour répondre utiliser le textField du bas.

- To Do: Faire en sorte que les messages privéee soit visiible sans rechager le chat privé.

Uptime

Permet de voir le temps qu'un salon existe.

Quit

Quand on quitte le chat, par la croix ou par la commande `/quit` , on est déconnecté du serveur et on peut se reconnecter avec un autre pseudo (le pseudo actuel est libéré).

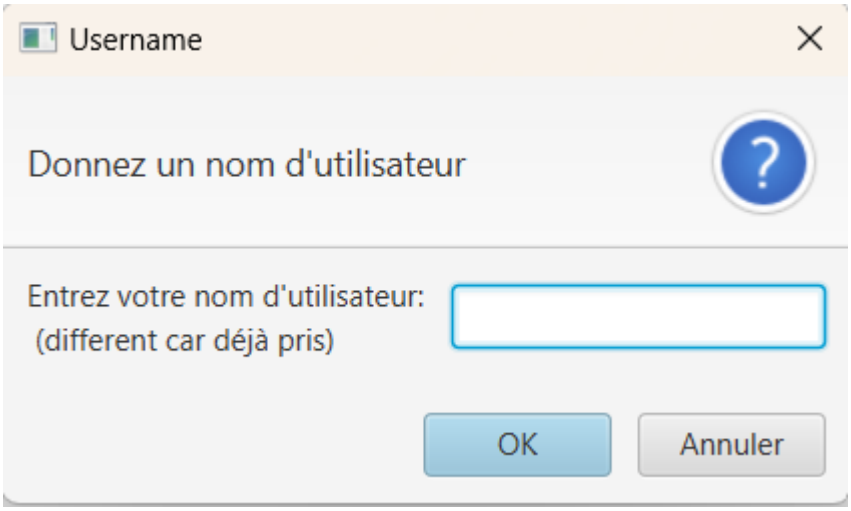
Help

`/help` permet d'avoir la liste des commandes disponibles.

Explication des fonctionnalitées

Username unique

Quand un user lance son client, il se trouve dans le salon `config`. Quand un user fait la commande `/username nomUtilisateur` (cela ce fait automatiquement), le serveur renvoie si le nom est déjà pris par une personne connecté. Si il est déjà pris, une nouvelle pop up s'affiche demandant un nouveau username.



Saisie adresse ip

Quand le client ce connect, il essaye de ce connecté à l'adresse ip localhost. Si sur cette adresse il n'y a pas le serveur, une pop demandant l'adresse ip du serveur est affiché et si on donne la bonne adresse la connexion au serveur ce fait.

Les boutons de l'ihm

Quand on clique sur les boutons de l'ihm, cela envoie la commande correspondant au serveur est attend le retour pour pouvoir afficher ce qu'il faut.

Message privé

Les messages privés avec une personne sont stocké en cache chez les clients.

Explication du code

Le serveur

Un serveur demande pour être instancié un port. Lorsque le serveur est lancé si le port 5001 est disponoble alors le message
```
PS C:\Users\benja\Desktop\Partage\Cour\S3\Sae_reseau> java -jar .\Serveur.jar
Serveur démarré
```
est affiché, le serveur ce lance sur l'addresse ip `localhost`. Sinon le message `Erreur lors de la création du serveur` est affiché.
Lors de son instantiation le serveur est initié avec deux salons: Geneneral et Config. Puis la fonction `launch()` permet d'attendre les connexions des clients. Quand un client ce connecte, une nouvelle session est instanciée et le message `Client connecté` est affiché. Si il y a une erreur le message `Erreur lors de la connexion d'un client` est affiché.
Quand cette session est instanciée, elle est ajoutée à la liste des sessions connectées.

Deco

Prend en paramètre un socket et un salon. Cette fonction lance un thread ServeurGestSalon qui permet de gérer les salons. Ce thread est instancié avec les paramètres `oldSolon: salon`, `socket: socket`, `serveur: le serveur actuel` et `action: "deco"` .

Les messages

Un message envoyé du client vers le serveur ressemble à ceci:
```
salon{nomSalon} nomUtilisateur : message
```
Le serveur va traiter l'en-tête du message pour connaitre le nom d'utilisateur, le salon et l'action à effectuer(message est une commande ou un message).

Session

Une session demande pour être instanciée un socket et un serveur.
Lors de son instantiation, la session est initiée avec un nom `Anonyme`, un nouveau thread `ServeurEcouter` est instancié et lancé.

ServeurEcouter

Le thread `ServeurEcouter` demande pour être instancié un socket et un serveur. Ce thread permet d'écouter les messages envoyés par le client. Son constructeur associe un `out` qui est un `PrintWriter` et un `in` qui est un `BufferedReader` au socket.
Le `run()` permet d'écouter les messages envoyés par le client. Si le message est `/quit` alors le client est déconnecté grâce à la fonction `deco` du serveur et le out est fermé. Tant que le message ne fini pas par "/quit" alors le message est écrit dans le terminal du serveur pour tous déboguage. Par la suite on regarde si le message correspond à une commande. Cette vérification est est faites avec la fonction `matches()` de la classe `String` , elle vérifie si un message correspond à celui rentré en paramètre, si l'on ne veut pas qu'une partie du message soit toujours le même on met un `.*` à la place de la partie variable.

ServeurGestSalon

Thread qui permet de gérer des actions à effectuer sur un salon (création, suppression, ajout d'un utilisateur, suppression d'un utilisateur, ...).

ServeurEnvoie

Thread qui permet d'envoyer un message. Ce thread est instancié à chaque fois qu'un message est envoyé.

L'IHM

Les cliques sur les boutons de l'IHM envoie des commandes et le retour de cette commande est géré.

Le client

Le client instancie 2 thread un pour écouter et un pour envoyé.

Javadoc

La javadoc est disponible dans le dossier `docs` du projet. Pour la lancer il faut ouvrir le fichier `index.html` dans un navigateur.

Membres

Paul JUPILLIAT Benjamin GUERRE