SAE - 3.03

RÉSEAU SOCIAL

2023 - 2024

Demande

L'objectif de cette SAE était de mettre en œuvre une application représentant un réseau de communication, tel un réseau social. Cette mise en œuvre s'effectue par l'utilisation des notions de programmation parallèles, de sockets, de datagrammes, sans oublier le complément graphique par l'utilisation du framework Java, JavaFX, ces notions ayant été vues en cours.

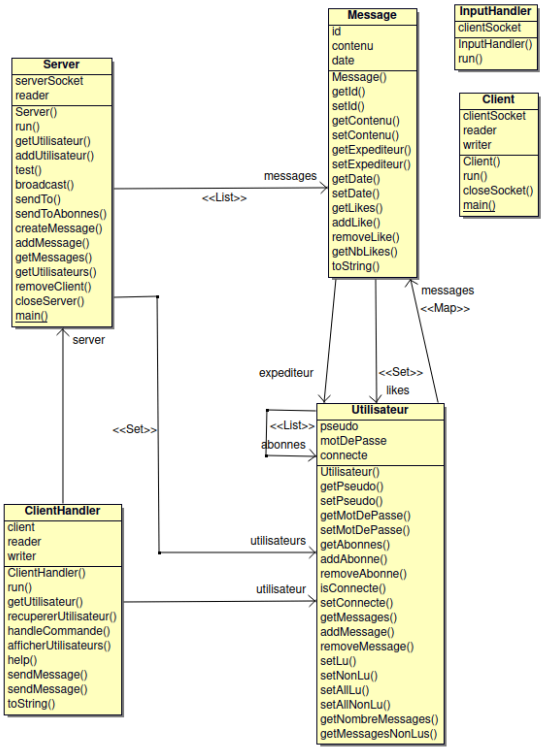
Il faut donc concevoir une application client-serveur, où les clients peuvent lancer une interface graphique et se connecter au serveur. Celui-ci sera responsable de gérer les requêtes des clients connectés, pouvant écrire des messages à leurs abonnés, au salon entier, ou des messages privés à d'autres clients connectés au serveur.

Partie 1 - Premier rendu

À l'aide des cours et des ressources à notre disposition, nous avons créé une application de client-serveur basique, permettant de lancer un serveur sur un port et d'accepter les sockets des clients se connectant à ce serveur.

Les fonctionnalités disponibles étaient l'instanciation d'un nouvel utilisateur qui devra renseigner un pseudo et son mot de passe à l'entrée sur le serveur, et l'envoi de messages en broadcast sur l'application.

Diagramme de classe



Partie 2 - Second rendu

Commandes

Pour la seconde partie, les messages sont envoyés en json et récupérés via la librairie gson. Cette implémentation nous permet de mettre à jour les états d'un message ou d'un utilisateur, notamment via ces commandes :

- /broadcast <content> : envoyer un message à tout le salon

- /follow <user>: permet de suivre un utilisateur
- /unfollow <user> : permet de ne plus suivre un utilisateur
- /msg <user> <content> : afin d'envoyer un message privé à un autre utilisateur
- /abonnes : consulter ses abonnés
- /suivi : consulter ses abonnements
- /list : permet de lister tous les utilisateurs présents dans le salon
- /like <id> : permet de liker un message ou une publication par son id
- /unlike <id> : permet de ne plus liker un message ou une publication
- /quit : quitter le salon

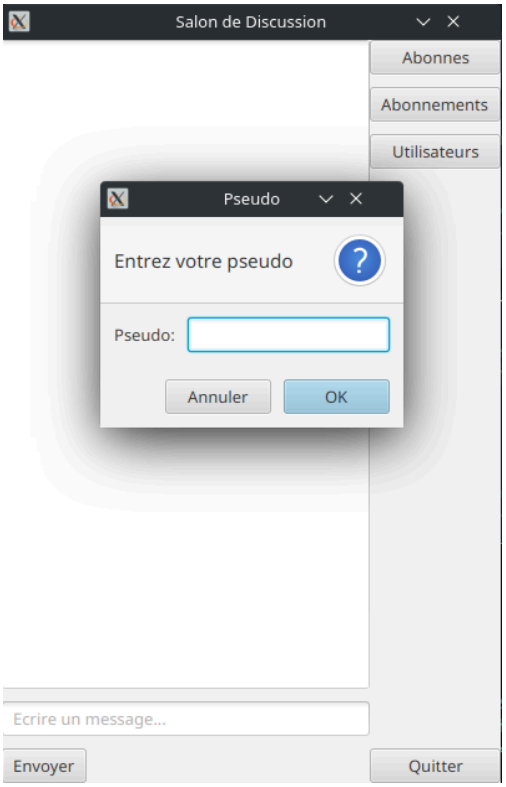
Formalités

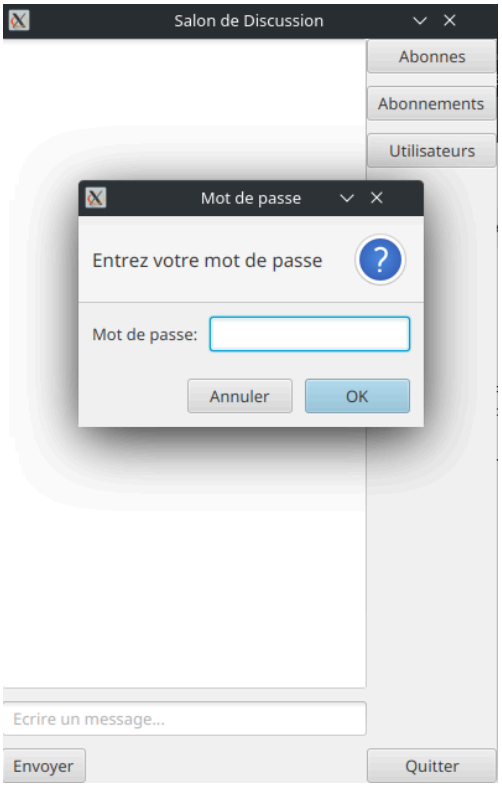
Nous avons décidé de déclarer 4 types distinct d'envoi de messages :

0. les messages d'utilisateur suivi (par-défaut)
1. les messages publics (/broadcast)
2. les messages privés (/msg)
3. les autres messages (messages du serveur comme "Utilisateur créé")
4. les retours (quand le serveur renvoie le message de l'utilisateur (pour avoir l'id,...))

Interface

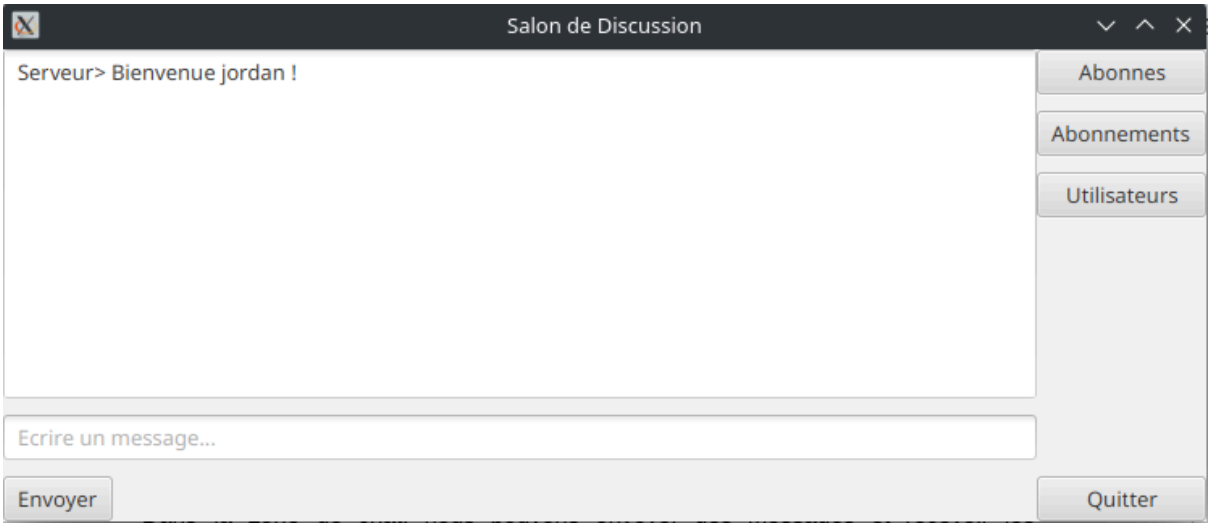
Pour l'extension de l'interface, nous avons implémenté une application JavaFX basique avec une page de connexion initiale afin de renseigner son nom d'utilisateur et son mot de passe.





Nous renseignons un mot de passe car nous aimerions stocker toutes les données échangées dans une structure de données harmonisée, et ainsi conserver les identifiants des utilisateurs connectés.

Cette application JavaFX, implémentée en suivant le modèle de conception MVC (modèle vue contrôleur), possède ensuite la zone de chat après la connexion / enregistrement dans le salon, où nous pouvons écrire dans le salon afin que le serveur traite nos messages et nos entrées (cf. commandes).

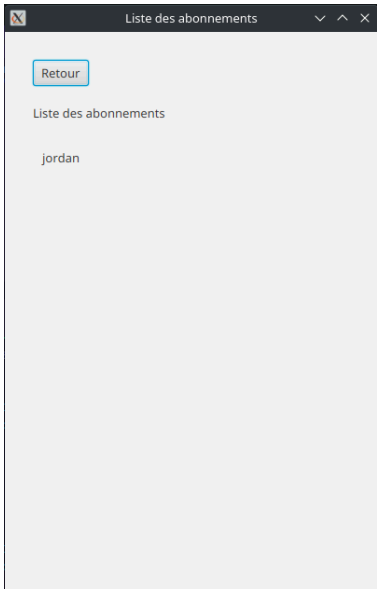


Dans la zone de chat, nous pouvons envoyer des messages et recevoir les messages des autres utilisateurs, affichés selon les conditions de réception.

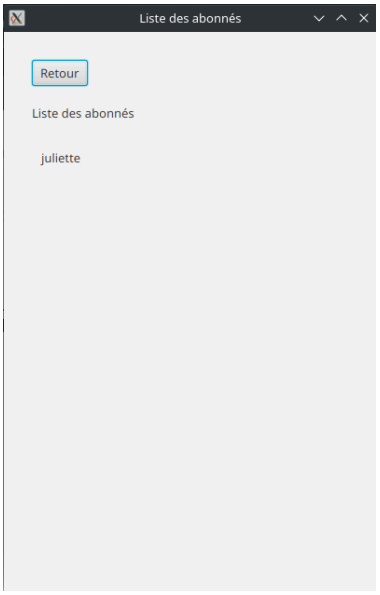
- Nous ne recevons par-défaut que les messages des utilisateurs suivis
- Nous recevons les messages envoyés en broadcast par les autres utilisateurs
- Nous recevons les messages privées qui nous sont envoyés

À droite de la zone de chat se trouve un menu permettant d'effectuer des commandes basiques sans les taper dans le champ de saisie, comme la consultation des abonnées, des abonnements ou des utilisateurs connectés au salon. Nous pouvons également quitter le salon de discussion, en déconnectant l'utilisateur du serveur.

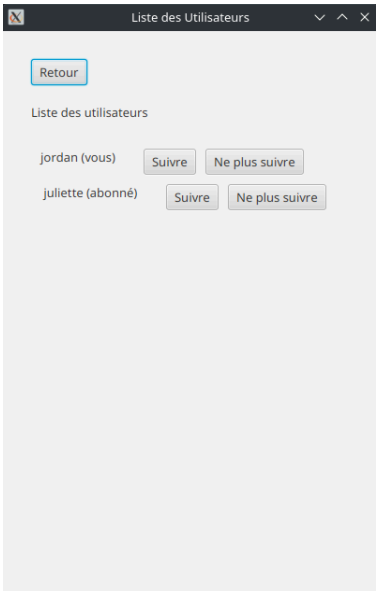
Consultation des abonnements



Consultation des abonnés



Consultation des utilisateurs connectés



Les boutons “Suivre” et “Ne plus suivre” simulent les commandes respectives /follow et /unfollow, permettant de suivre ou de ne plus suivre un utilisateur du salon. La gestion des exceptions tels qu’un utilisateur n’existant pas ou l’ajout d’un utilisateur qui est déjà suivi à nos abonnements est fonctionnel.

Persistance

En démarrant, le serveur tente de charger les fichiers “utilisateurs.txt” et “messages.txt” du dossier “sauvegardes” pour se remplir. S’il échoue, il créera simplement l’utilisateur Serveur. Les utilisateurs sont stockés un par ligne sous le format :
id;pseudo;mdp;abonne1,abonne2,...(-1 si aucun);abonnement1,abonnement2,...(-1 si aucun);message1:lu(true/false),message2:lu,...(-1 si aucun)
et les messages :
id;contenu;idExpediteur;date;idUtiLike1,idUtiLike2,...;supprime(true/false);type
On peut sauvegarder le serveur avec /save si on est l’utilisateur Serveur et il se sauvegarde en cas d’erreur avec les clients.

Diagramme de classe final