Rapport de projet : Système de clavardage

Andréa CHENEL & Pierre LAUR

I - Présentation générale & installation/déploiement

Notre application est une application de clavardage permettant à deux utilisateurs connectés sur le même réseau local de pouvoir chatter entre eux. Il leur est possible de choisir un pseudonyme au démarrage de l'application.

Nous avons travaillé en *pair programming* une partie du temps, et utilisé git pour gérer les versions le reste du temps. Nous avons fait des pushs réguliers sur ce dépôt : https://github.com/andreachenel/Projet_Chat. Nous nous sommes appuyés sur Jira pour construire le développement du projet. 5 sprints étaient prévus à l'origine.

II - Manuel d’utilisation

1. Se connecter au système

Pour pouvoir se connecter au système, l'utilisateur va faire face à une interface de connexion lui demandant de renseigner son identifiant et son mot de passe. Si l'identifiant est connu de la base de données mais que le mot de passe ne correspond pas, l’utilisateur se voit refuser l’accès à l’application. Si l’identifiant n’est pas connu de la base de données, un nouveau compte est créé.

Dans le cas où mot de passe et identifiant concordent et celui dans lequel l’identifiant était inconnu, l’utilisateur a accès à l’interface suivante et donc à l’application de clavardage.

2. Voir les utilisateurs connectés

Une fois sur la fenêtre principale, pour voir les utilisateurs connectés, il faut cliquer sur le bouton “refresh” afin de voir apparaître tous les utilisateurs connectés dans la liste déroulante sur la gauche.

3. Commencer une session de clavardage

Pour démarrer une session de clavardage, il faut cliquer sur la liste déroulante, puis de choisir l’utilisateur avec lequel on souhaite chatter, et enfin de cliquer sur le bouton “Select”, l’historique des conversations s’affiche ainsi.

5. Envoyer et recevoir des messages

Pour envoyer un message, il faut d’abord sélectionner l’utilisateur avec lequel on souhaite chatter et ensuite taper le message dans la barre de texte en bas de la fenêtre. Enfin, il faut appuyer sur le bouton “Send”. Les messages reçus s’affichent automatiquement dans la conversation avec leur expéditeur.

6. Se déconnecter du système

Pour se déconnecter du système, il suffit à l’utilisateur de cliquer sur le bouton “log out” en haut à droite de l’interface.

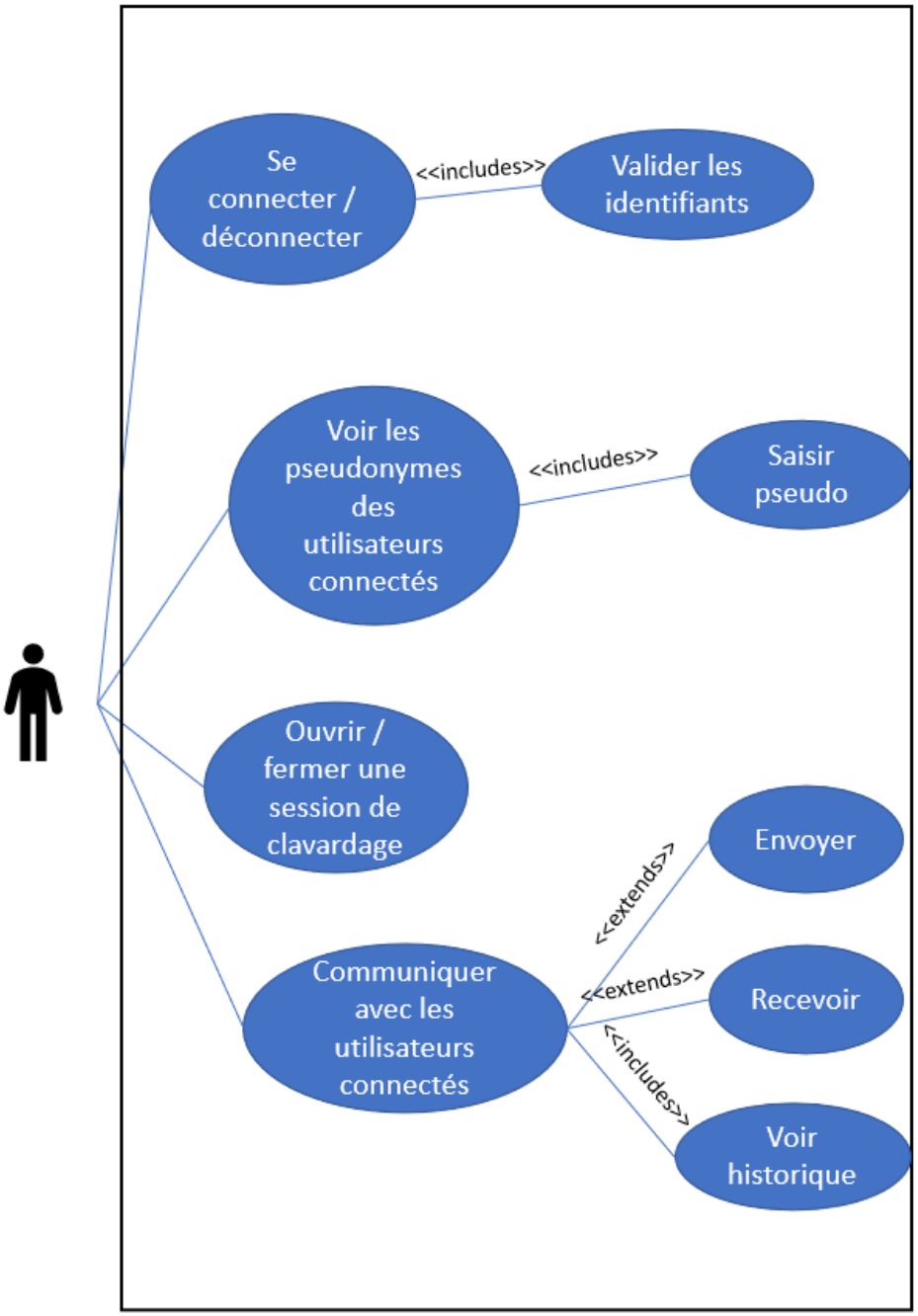
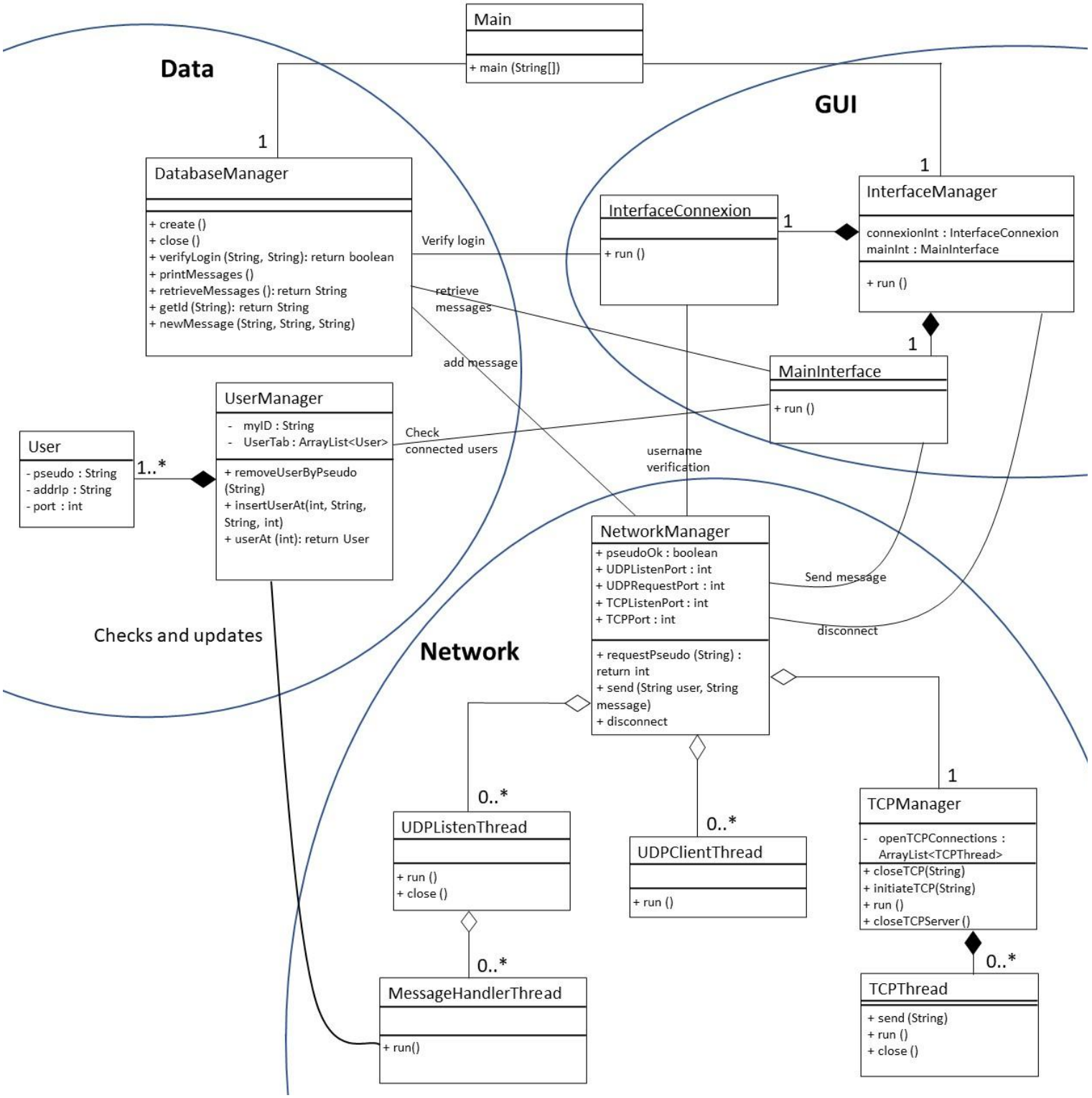


Diagramme de cas d'utilisation du système

III - Architecture du système



Le système est organisé en trois parties, représentées par les packages :

- **network**, responsable de l'établissement de la connexion au système, de l'envoi et de la réception des messages etc.
- **data**, où est stocké le tableau des utilisateurs connectés, et dont fait partie la classe responsable du lien avec la BDD
- et **GUI**, les interfaces.

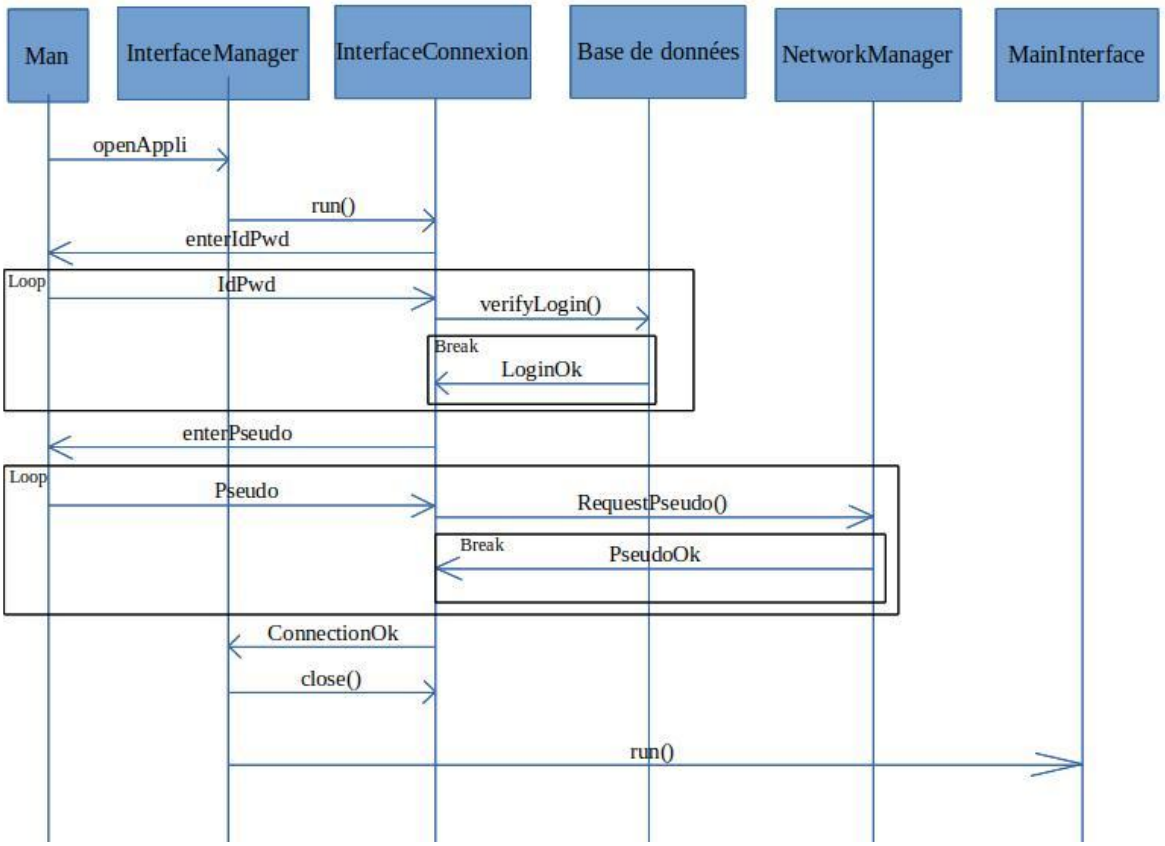
IV - Fonctionnement

1. Interface

Nous avons choisi de créer l'interface en utilisant Java Swing. Une première JFrame (InterfaceConnexion) permet de gérer la connexion, la deuxième (MainInterface) est l'interface principale qui permet d'envoyer et de recevoir les messages.
Le InterfaceManager créé à l'ouverture de l'application lance l'interface de connexion, puis l'interface principale lorsque la connexion est terminée.
Les actions s'exécutent grâce à des ActionListeners associés aux boutons des interfaces.

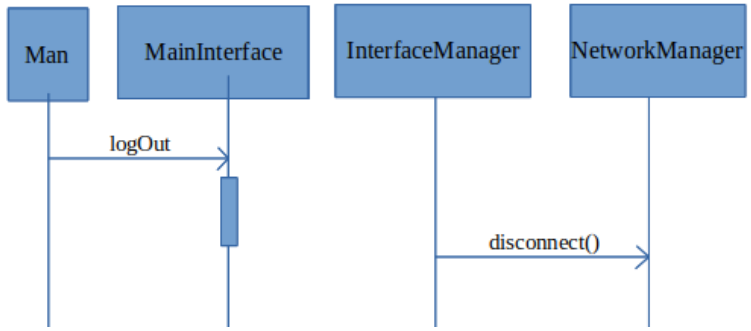
2. Connexion

L'utilisateur arrive sur la fenêtre de connexion, il lui est demandé de renseigner son identifiant et son mot de passe, on vérifie qu'ils correspondent avec une table de correspondance entre ID et mot de passe dans la base de données.
Nous avons choisi d'utiliser SQL pour créer une base de données centralisée, ce qui permet aux utilisateurs de s'identifier depuis n'importe quelle machine du réseau.
Si l'ID n'existe pas dans la base de données, on crée une nouvelle entrée (ID + mot de passe) dans la base de données et la connexion est validée.
Si l'ID existe mais ne correspond pas au mot de passe, ils doivent re-saisir leurs identifiants (un message d'erreur s'affiche). Dans le cas contraire, ces identifiants sont validés, ils doivent maintenant saisir un pseudo.
Pour garantir l'unicité du pseudo sur le réseau, un broadcast UDP est alors envoyé aux autres utilisateurs connectés. Chaque application dispose d'une table dans laquelle elle stocke les pseudos des utilisateurs connectés : elle vérifie que ce pseudo n'existe pas déjà dans la table et répond via UDP.
Si aucun d'eux n'a déjà le pseudo demandé, ce pseudo est alors déclaré valide et l'utilisateur peut entrer dans l'application. Un broadcast UDP est envoyé pour confirmer le choix du pseudo, et chaque application l'ajoute dans sa table.



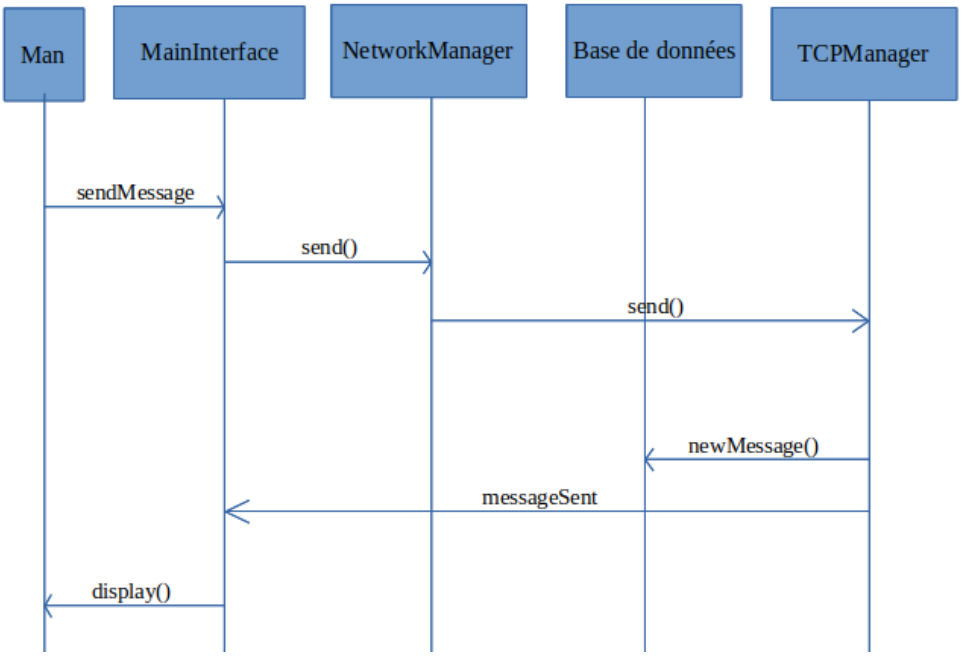
3. Déconnexion

Lorsque l'utilisateur clique sur "log out", la fenêtre de clavardage se ferme et un broadcast UDP est envoyé pour en notifier les autres utilisateurs. L'utilisateur déconnecté est supprimé de la table des utilisateurs connectés sur les autres applications.



4. Envoyer un message à un utilisateur connecté

Une fois l'utilisateur avec lequel on souhaite chatter sélectionné, le message tapé et le bouton "send" actionné, MainInterface envoie une requête au NetworkManager qui transmet au TCP Manager. Ce dernier envoie une requête de connexion au TCPManager de l'application distante, et un Thread TCP est créé des deux côtés pour gérer les messages reçus et envoyés. Lorsqu'un message est envoyé, le thread TCP concerné ajoute le message dans la base de données pour que celui-ci soit visible depuis toutes les machines.



5. Recevoir un message d'un utilisateur connecté

Une fois l'utilisateur connecté, NetworkManager crée un TCPManager qui crée des connexions TCP avec les applications distantes qui le demandent. L'interface principale envoie une requête à la base de données toutes les secondes pour actualiser les messages affichés.