INSA de TOULOUSE
Département Génie Électrique & Informatique

INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Etudiant(s) :	MARTIN Mattew | CORELLA Gabriel | ROCHE Stéphane	**Date :**	27/01/2023
Matière :	Projet chat system	**Numéro :**	
Sujet :	Partie UML		
Remarques :			
Note :			

SYSTÈME DE CLAVARDAGE DISTRIBUE INTERACTIF MULTI-UTILISATEUR TEMPS RÉEL (UML)

Matière :	Projet chat system
Date :	27/01/2023
Etudiant(s) :	MARTIN Mattew | CORELLA Gabriel | ROCHE Stéphane
Sujet :	Partie UML

1 Acteurs

1.1 Acteurs primaires

Les agents (vus comme des utilisateurs d'un point de vue interne) : Ils sont ceux qui interagissent avec le système en faisant des demandes telles que se connecter, envoyer des messages, etc…
L'administrateur : Il est celui qui déploie et maintient le système, il a donc un lien très fort avec celui-ci.

1.2 Acteurs secondaires

Les autres agents vus par un même utilisateur et qui sont la cible de différentes intéractions comme la notification d'un changement de pseudo ou bien l'envoi/réception d'un message.

2 Diagramme de Cas d'Utilisation

2.1 Point de vue Utilisateur

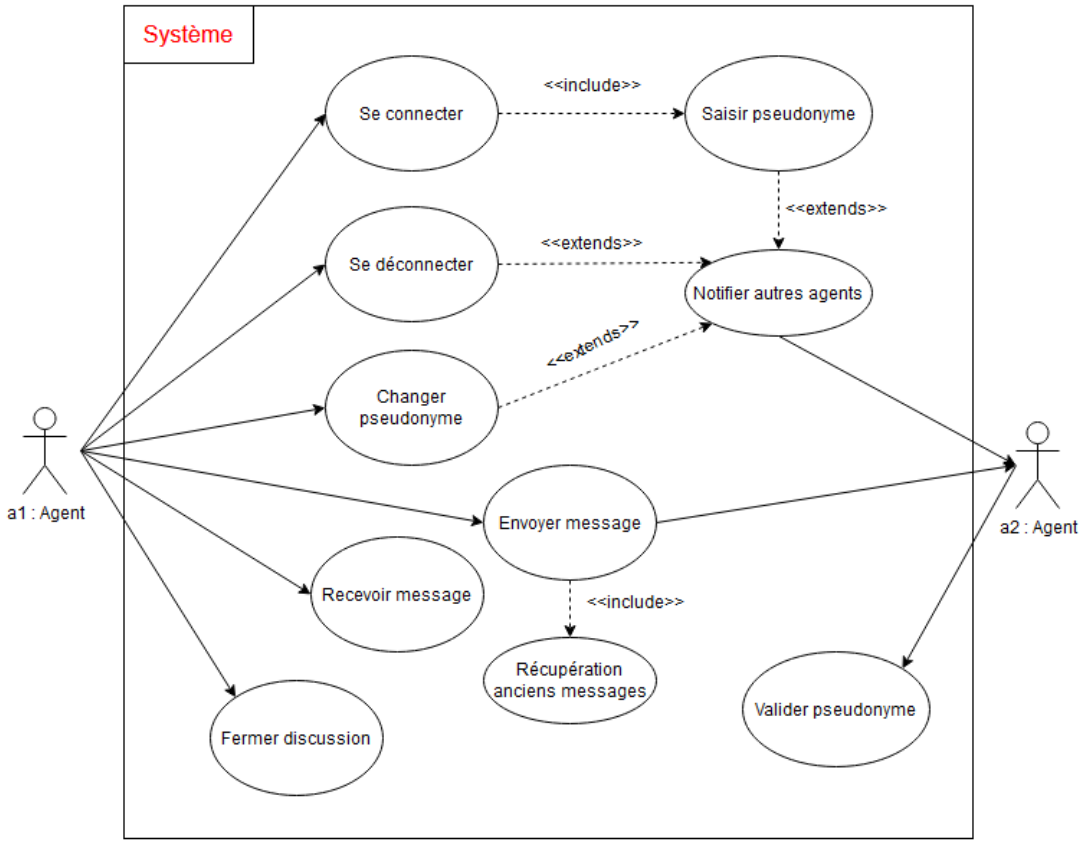


FIGURE 1 – Use Case Diagram User POV

Matière :	Projet chat system
Date :	27/01/2023
Etudiant(s) :	MARTIN Mattew | CORELLA Gabriel | ROCHE Stéphane
Sujet :	Partie UML

2.2 Point de vue Administrateur

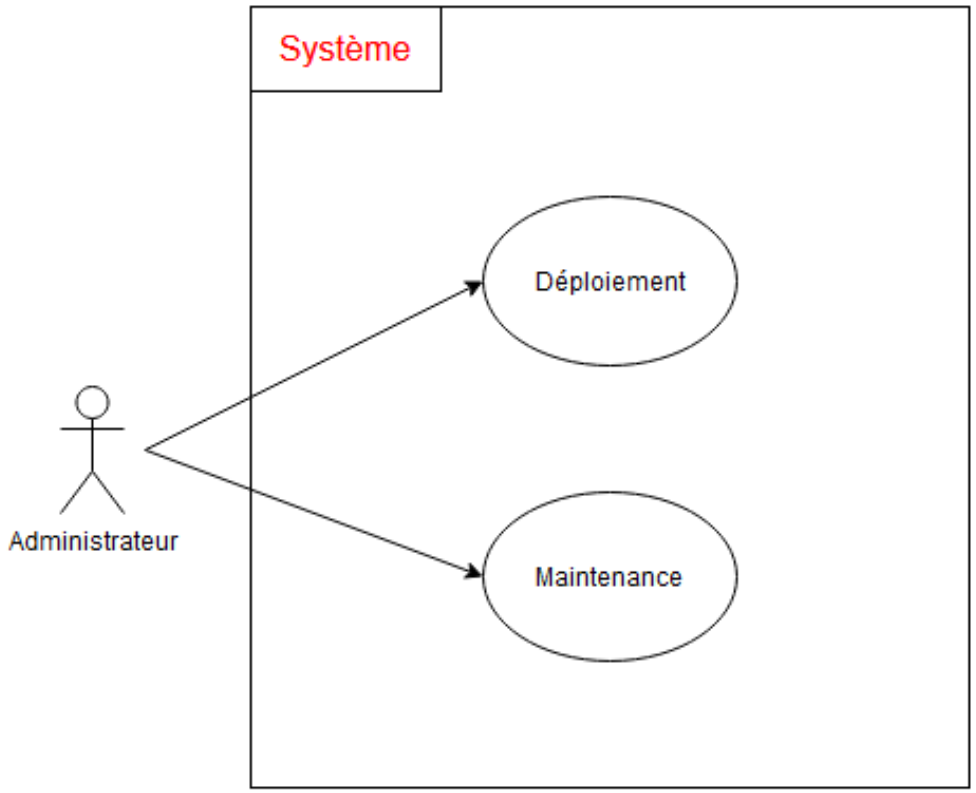


FIGURE 2 – Use Case Diagram Admin POV

3 Diagramme de Classes

Voir le fichier nommé “ClassDiagram.pdf” dans le dossier UML du repository

Matière :	Projet chat system
Date :	27/01/2023
Etudiant(s) :	MARTIN Mattew | CORELLA Gabriel | ROCHE Stéphane
Sujet :	Partie UML

4 Diagrammes de Séquence

4.1 Boîte noire

4.1.1 Connexion

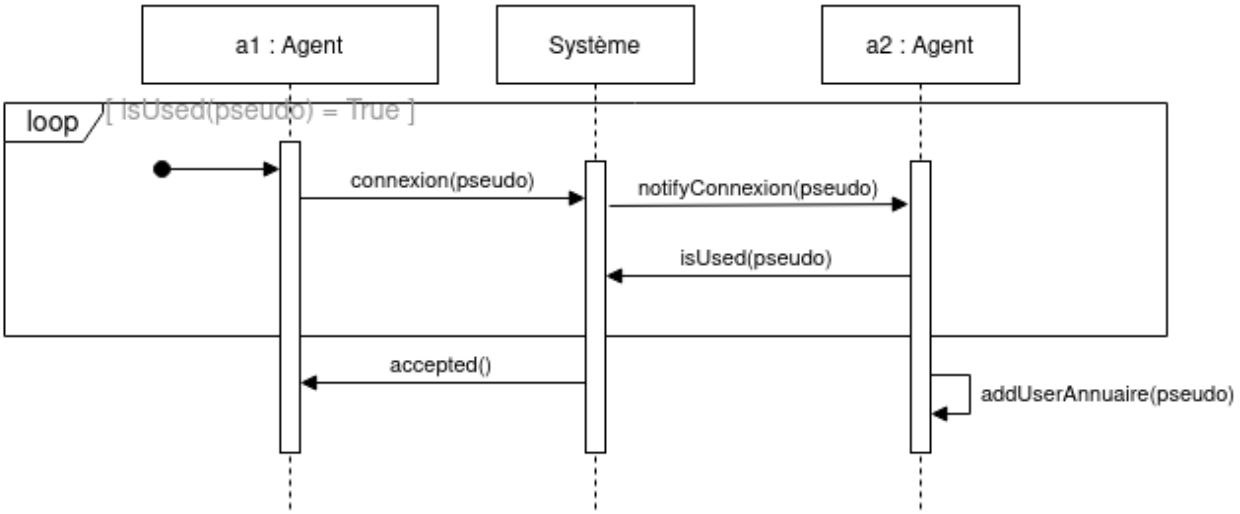


FIGURE 3 – Sequence Diagram Black Box (connection)

4.1.2 Déconnexion

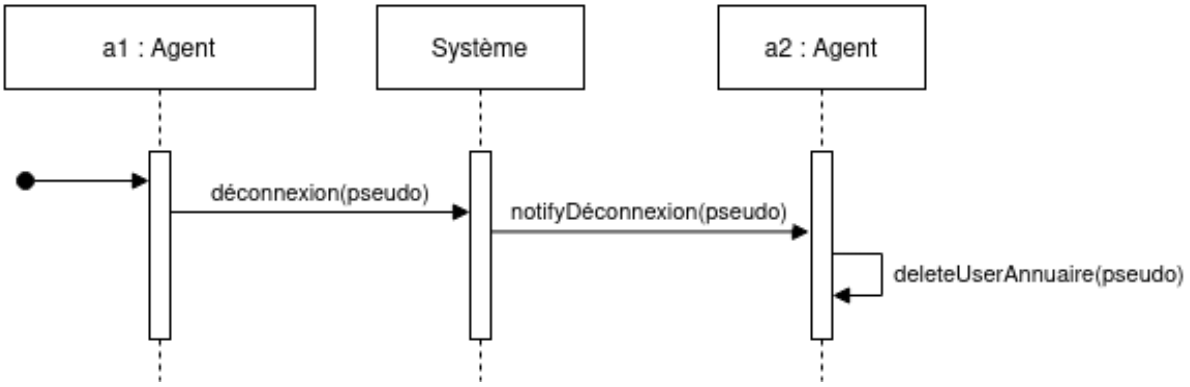


FIGURE 4 – Sequence Diagram Black Box (deconnection)

Matière :	Projet chat system
Date :	27/01/2023
Etudiant(s) :	MARTIN Mattew | CORELLA Gabriel | ROCHE Stéphane
Sujet :	Partie UML

4.1.3 Changement de pseudo

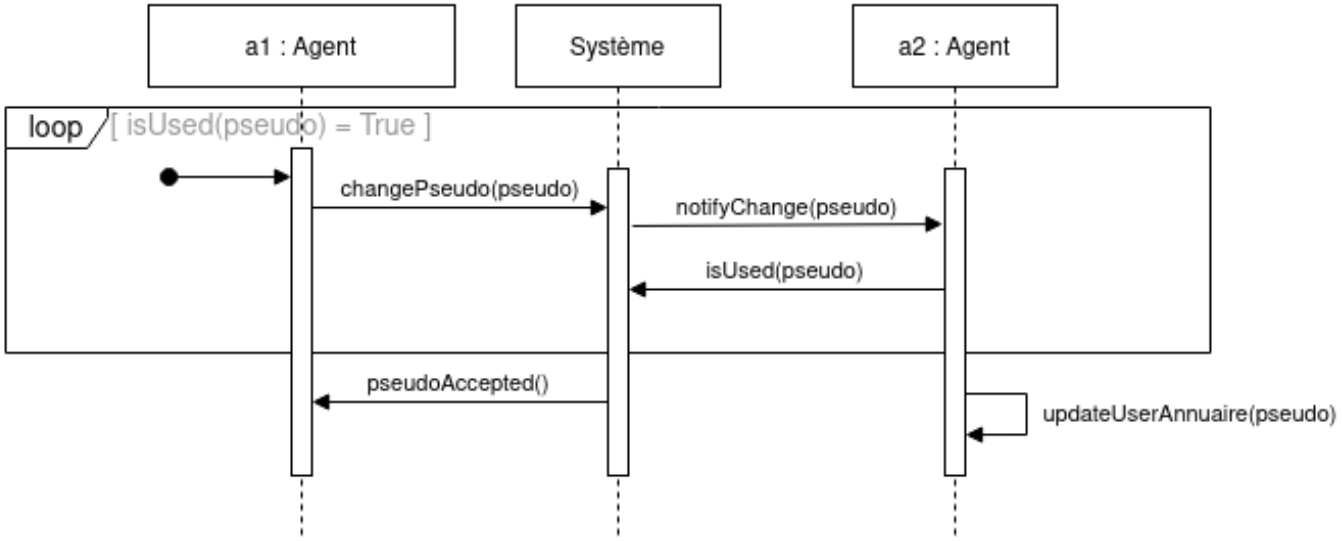


FIGURE 5 – Sequence Diagram Black Box (change pseudo)

4.1.4 Envoi d'un message

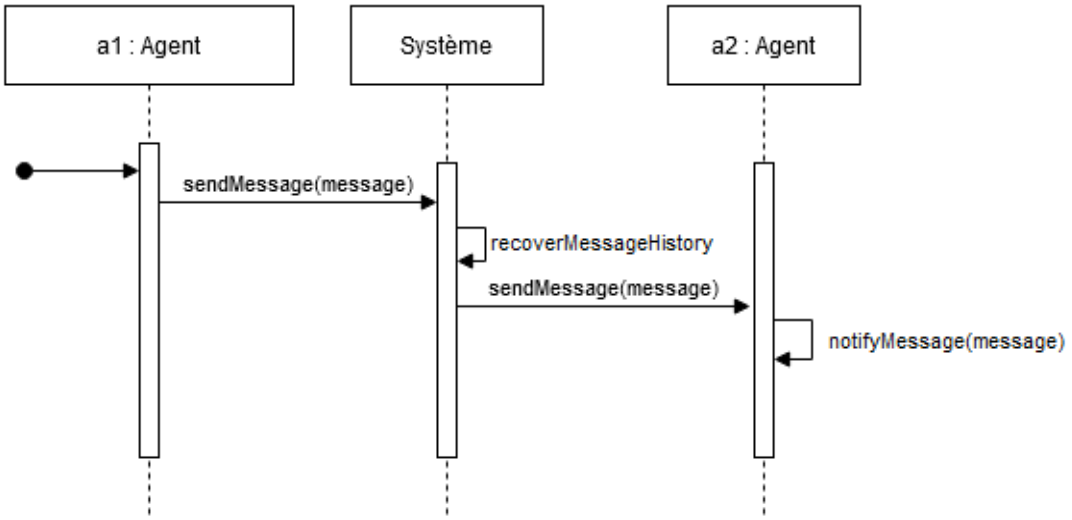


FIGURE 6 – Sequence Diagram Black Box (send message)

4.1.5 Réception d'un message

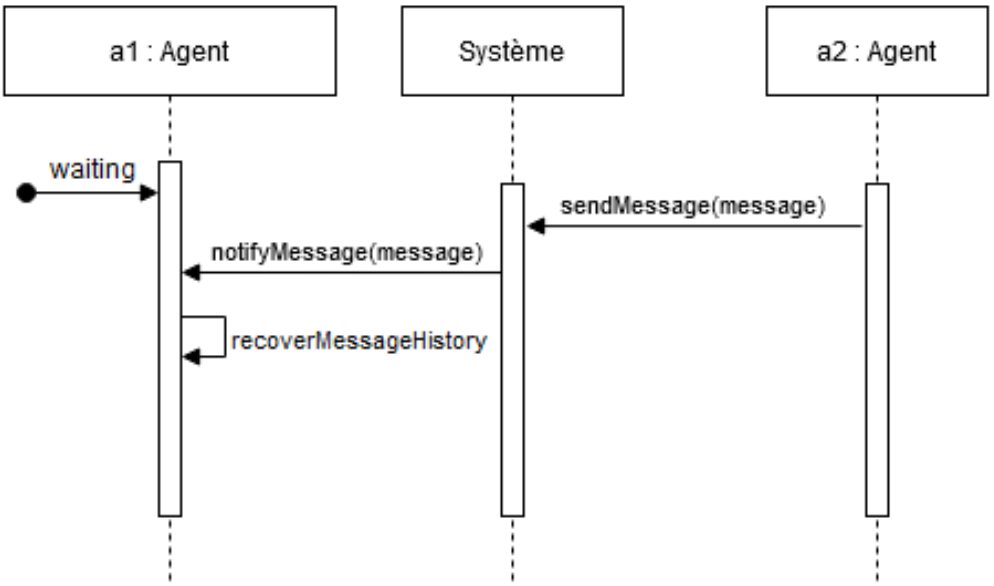


FIGURE 7 – Sequence Diagram Black Box (receive message)

4.1.6 Fermeture d'une discussion

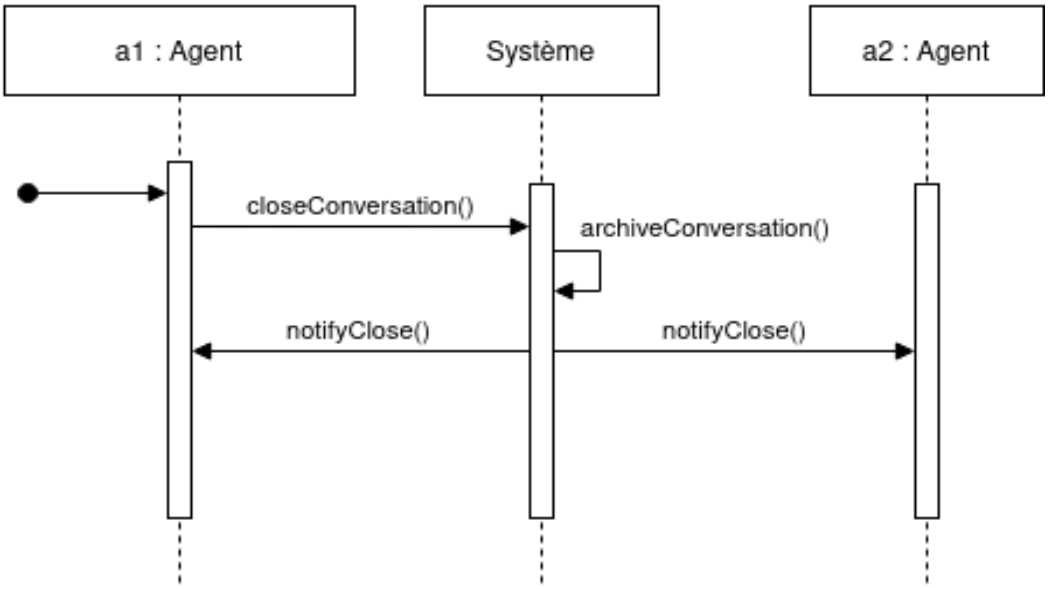


FIGURE 8 – Sequence Diagram Black Box (close conversation)

Matière :	Projet chat system
Date :	27/01/2023
Etudiant(s) :	MARTIN Mattew | CORELLA Gabriel | ROCHE Stéphane
Sujet :	Partie UML

4.2 Boîte blanche

Il est important de noter que pour cette partie, nous avons considéré le code en version CLI et non interface graphique. Dans le cas de l'interface graphique il aurait fallu remplacer la classe App par LaunchGUI.

4.2.1 Connexion

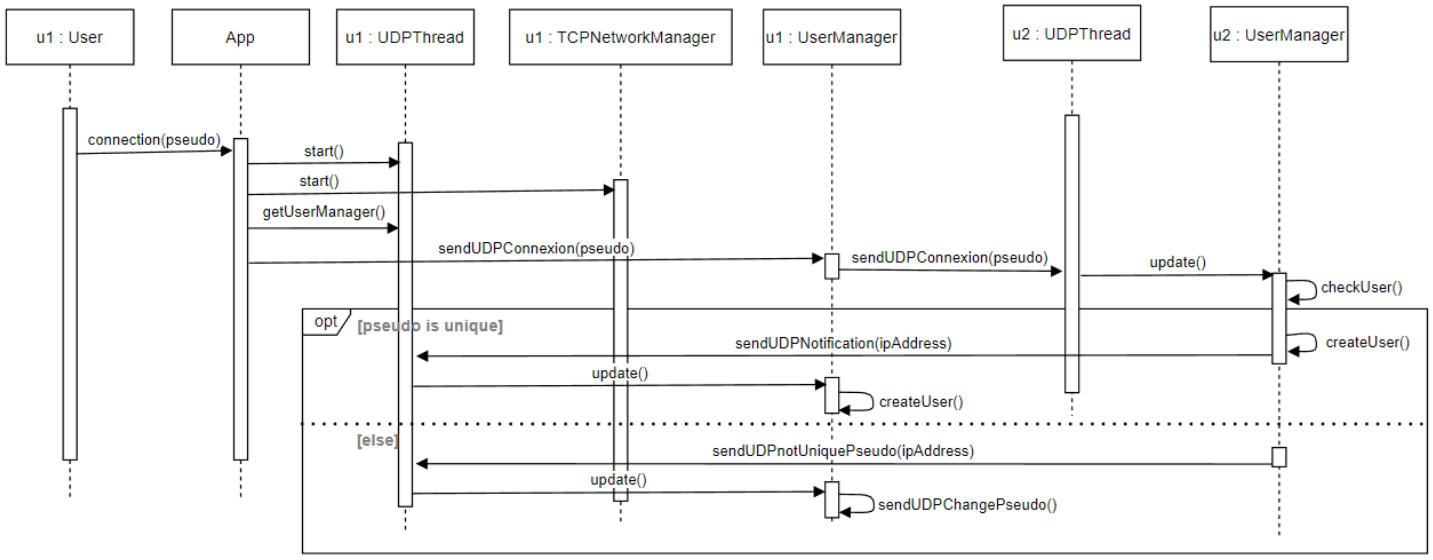


FIGURE 9 – Sequence Diagram White Box (connection)

Les méthodes **start()** impliquent qu'un objet de type UDPThread et TCPNetworkManager a été créé pour ensuite l'utiliser afin de lancer les deux threads.

La méthode **getUserManager()** permet d'utiliser un objet de cette classe pour appeler depuis App des méthodes de la classe UserManager.

En ce qui concerne la méthode **sendUDPNotification(pseudo)**, cette méthode notifie l'utilisateur 1 de l'existence de l'utilisateur 2 afin qu'il l'ajoute lui aussi à sa liste. Mais ce n'est pas tout car elle sert par la même occasion à confirmer à l'utilisateur 1 que son pseudo est bien unique (sinon il n'aurait pas reçu une notification de ce type là)

Il est important de noter que dans le cas où le pseudo n'est pas unique comme on peut le voir en bas du diagramme, l'utilisateur va de nouveau choisir un pseudo et la trame sera la même qu'à partir de sendUDPConnexion(pseudo).

4.2.2 Déconnexion

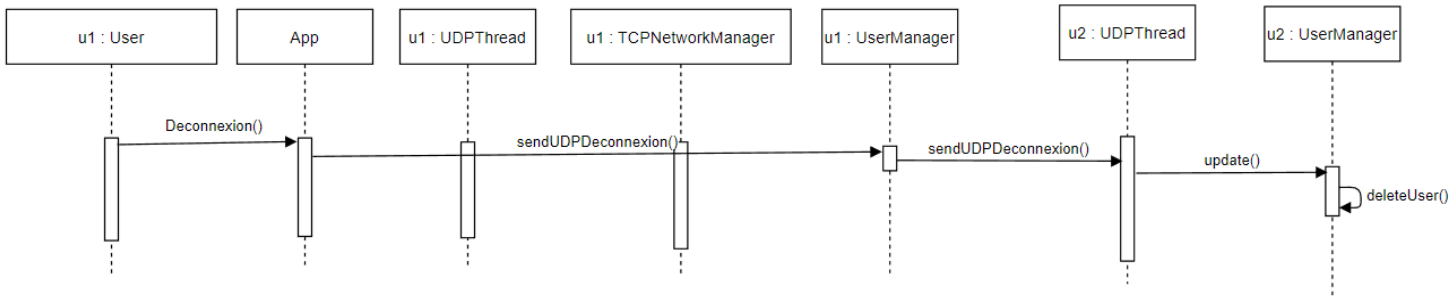
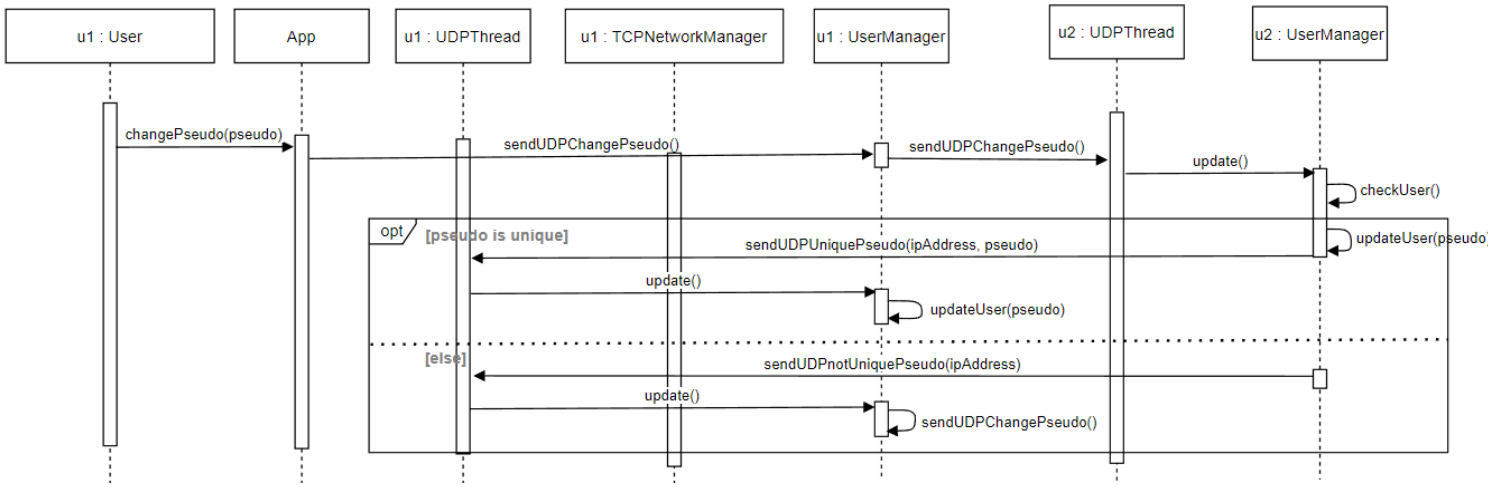


FIGURE 10 – Sequence Diagram White Box (deconnection)

Ici nous nous considérons dans une situation où l'utilisateur est déjà connecté depuis un moment d'où le fait que le serveur TCP, UDP Thread soient déjà lancés.

4.2.3 Changement de pseudo



Comme nous pouvons le voir ici, le fonctionnement ressemble trait pour trait à la connexion d'un utilisateur au chat system.
Il envoie son choix de pseudo aux autres users, ils traitent le paquet et vérifient que le pseudo est bien unique. Si oui, ils confirment l'unicité en envoyant un paquet sendUDPUniquePseudo(ipAddress, pseudo) avec pour ipAddress celle de l'utilisateur souhaitant changer de pseudo et comme pseudo le nouveau pseudo choisi. Ces données lui permettront de changer son propre pseudo depuis sa liste des utilisateurs.
Dans le cas où l'utilisateur n'a pas choisi un pseudo unique, les autres utilisateurs le notifient afin que l'utilisateur puisse choisir un nouveau pseudo (grâce à la méthode sendUDPChangePseudo())

4.2.4 Envoi/Réception d'un message

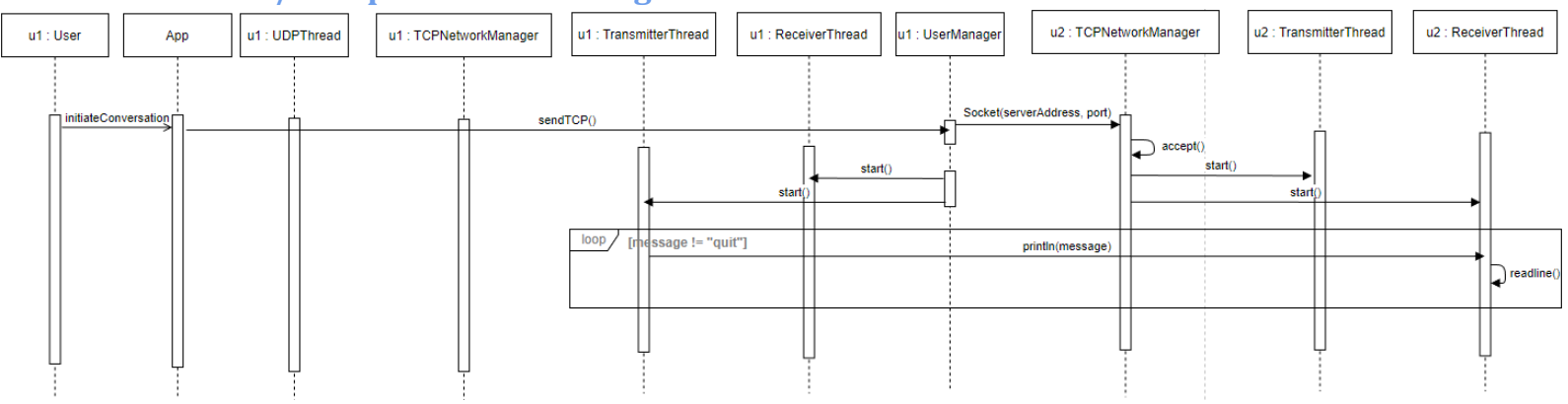


FIGURE 11 – Sequence Diagram White Box (send/receive message)

Dans le cas de l'envoi de message, nous passons d'abord par l'initiation d'une conversation via TCP cette fois-ci afin de créer un canal de communication dans les deux sens entre les deux utilisateurs. Nous pouvons ici voir le cas de l'envoi de message si nous nous plaçons du point de vue de l'user 1 mais aussi le cas de la réception de message si nous nous plaçons du point de vue de l'user 2.
Lorsqu'une connexion est initiée, le serveur de l'utilisateur 2 est sollicité via la méthode Socket(serverAddress, port), il accepte alors la connexion et lance les threads Transmitter et Receiver. En parallèle, une fois que la connexion est acceptée par le serveur, UserManager de l'utilisateur 1 se charge de lancer lui aussi les threads Transmitter et Receiver. Une fois les threads lancés des deux côtés (càd les canaux étant prêts), la conversation peut démarrer.

Matière :	Projet chat system
Date :	27/01/2023
Etudiant(s) :	MARTIN Mattew | CORELLA Gabriel | ROCHE Stéphane
Sujet :	Partie UML

5 Diagramme de Structure Composite

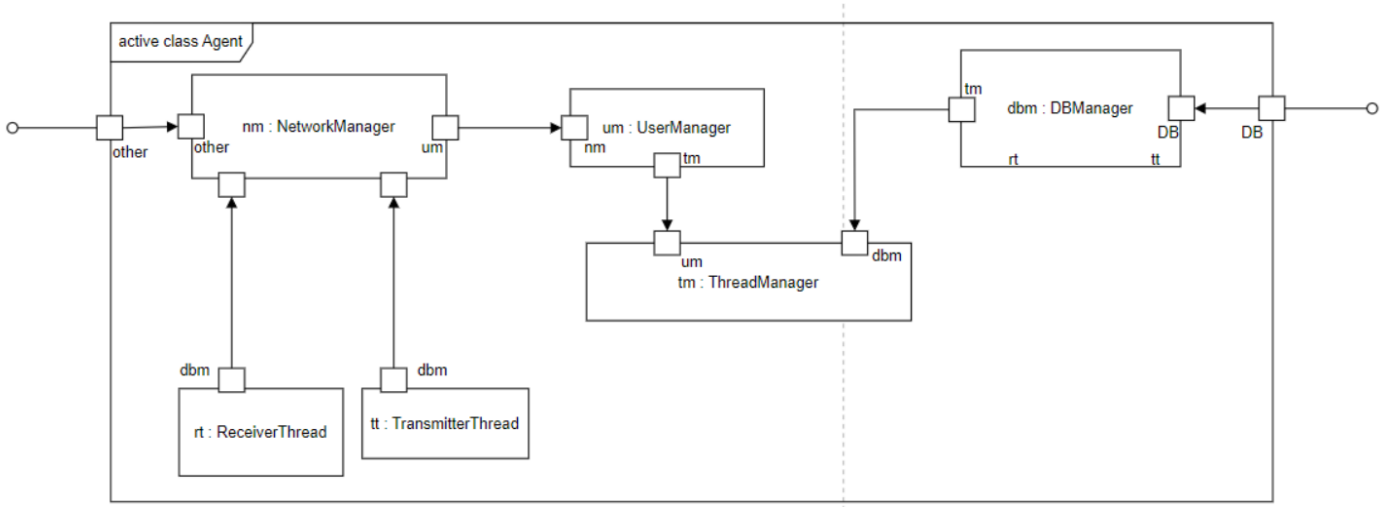


FIGURE 12 – Composite structure diagram

6 Diagramme de déploiement

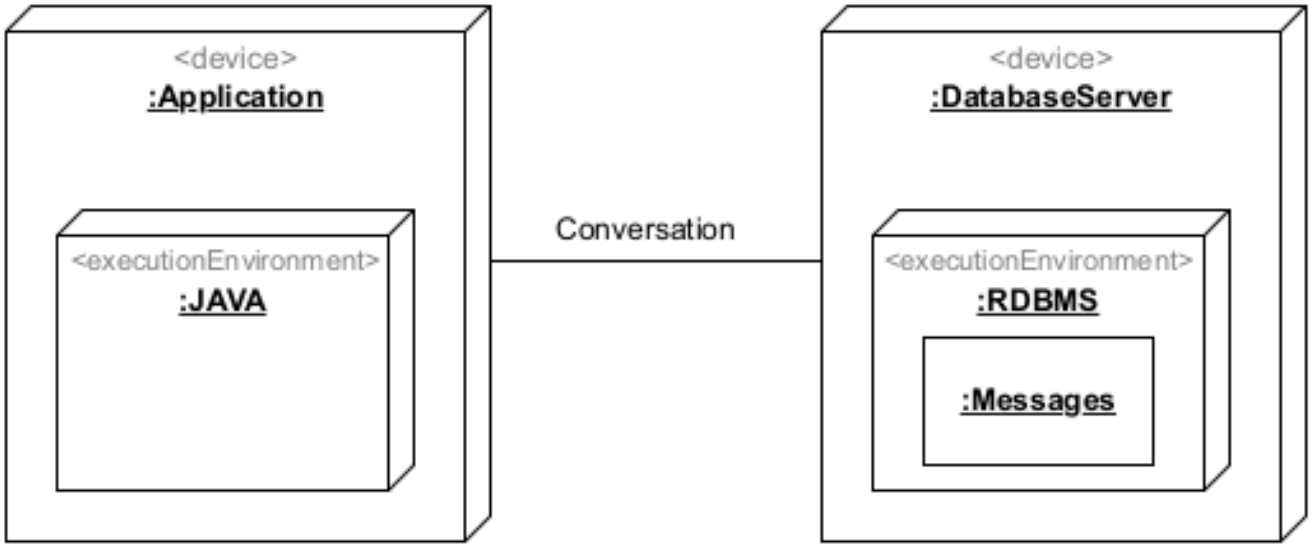


FIGURE 13 – Deployment diagram

Ici nous sommes dans le cas d'un déploiement très simple puisqu'il s'agit juste d'une application communicant avec sa BDD décentralisée.

Matière :	Projet chat system
Date :	27/01/2023
Etudiant(s) :	MARTIN Mattew | CORELLA Gabriel | ROCHE Stéphane
Sujet :	Partie UML

7 Schéma de la BDD

¨Pour identifier une conversation précise, on a besoin :

- @IP source
- @IP destination

On peut identifier par la suite chaque message découlant de la conversation par sa date. Ainsi grâce à ces informations, on peut retrouver le contenu du message.

De plus, on considère qu'en entreprise, les machines seront adressées de manière statique. C'est-à-dire que chaque machine aura sa propre adresse IP, et de plus on considère que chaque salarié sera assigné à une même machine. On décide donc d'une clé primaire composée des adresses IP source et destination et de la date des messages afin de discerner avec précision les messages échangés dans chaque conversation.

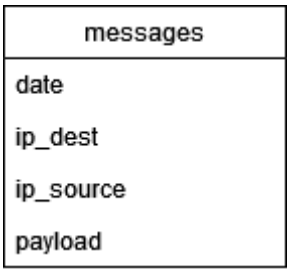


FIGURE 14 – Database diagram

La table messages est contenu dans le fichier messages.db associé à l'application. Cette table est donc propre à l'appareil.

Matière :	Projet chat system
Date :	27/01/2023
Etudiant(s) :	MARTIN Mattew | CORELLA Gabriel | ROCHE Stéphane
Sujet :	Partie UML

8 Architecture du système et choix technologiques

En ce qui concerne la BDD, nous nous sommes orientés vers une base de données décentralisée en utilisant SQLite. SQLite est globalement plus performant que HSQLDB et c'est également le plus simple en termes de mise en place. Par ailleurs, ce qui nous a confortés dans ce choix, c'est qu'on considère que dans une entreprise, le réseau de l'entreprise serait soit du BYOD (Bring Your Own Device) soit des postes assignés à chaque salarié. Ainsi, dans notre scénario, les personnes utilisant notre programme ne changent pas même si elles peuvent changer de pseudo. On considère donc que les adresses IP des machines sont assignées de façon statique ce qui est en accord avec la structure de notre base de données. Tous les échanges sont donc enregistrés en local dans un fichier Messages.db.

En ce qui concerne GUI, nous avons choisi d'utiliser JavaFX car :

- Il est **basé sur Java**, ce qui signifie qu'il est compatible avec d'autres bibliothèques Java et peut être intégré facilement dans des projets existants écrits en Java.
- Il offre des **fonctionnalités de rendu avancées**, telles que des effets visuels, des animations et des graphismes vectoriels, qui peuvent améliorer l'expérience utilisateur.
- Il offre une **large gamme de composants préconstruits** pour les interfaces utilisateur courantes, comme les boutons, les menus, les tableaux, etc.
- C'est une plate-forme **multi-plateforme**, ce qui signifie qu'une application développée avec JavaFX peut fonctionner sur **Windows**, **Mac** et **Linux** sans modification.

Matière :	Projet chat system
Date :	27/01/2023
Etudiant(s) :	MARTIN Mattew | CORELLA Gabriel | ROCHE Stéphane
Sujet :	Partie UML

9 Maquette GUI

9.1 Connexion

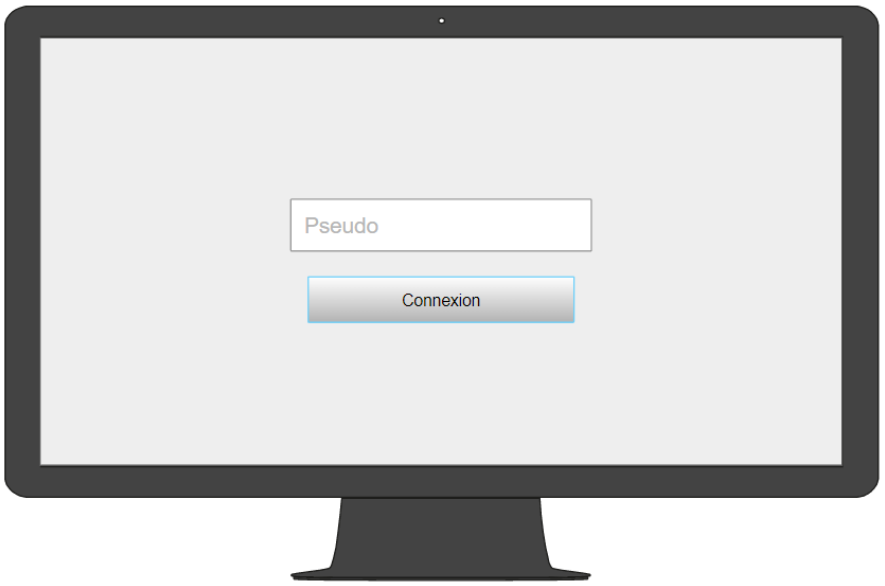


FIGURE 15 – Login GUI

Ceci est la première page affichée après exécution du programme, c'est-à-dire la **page d'accueil**.

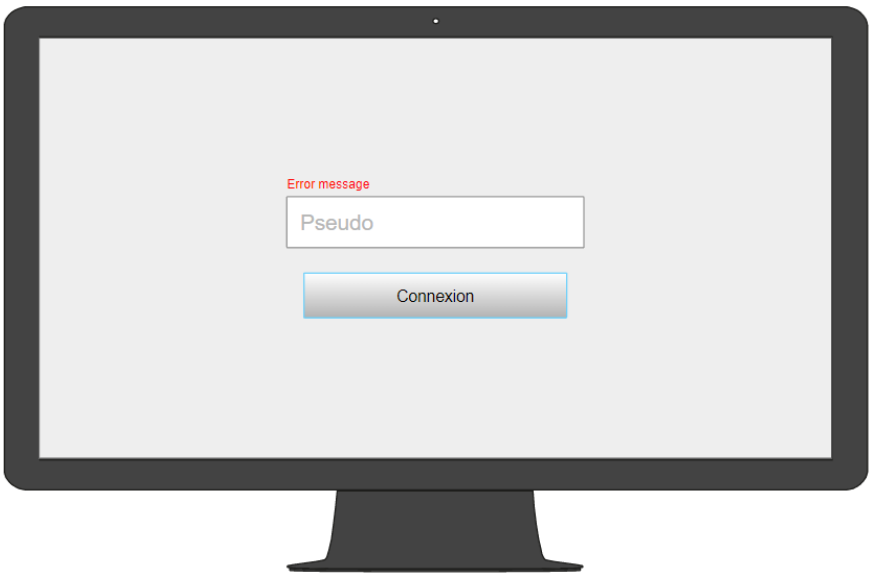


FIGURE 16 – Login GUI (On mismatch entry)

Si le pseudo choisi ne respecte pas les **contraintes d'unicité** et de **taille**, un **message d'erreur** sera alors affiché à l'écran en expliquant la cause de l'erreur.

Matière :	Projet chat system
Date :	27/01/2023
Etudiant(s) :	MARTIN Mattew | CORELLA Gabriel | ROCHE Stéphane
Sujet :	Partie UML

9.2 Interface principale

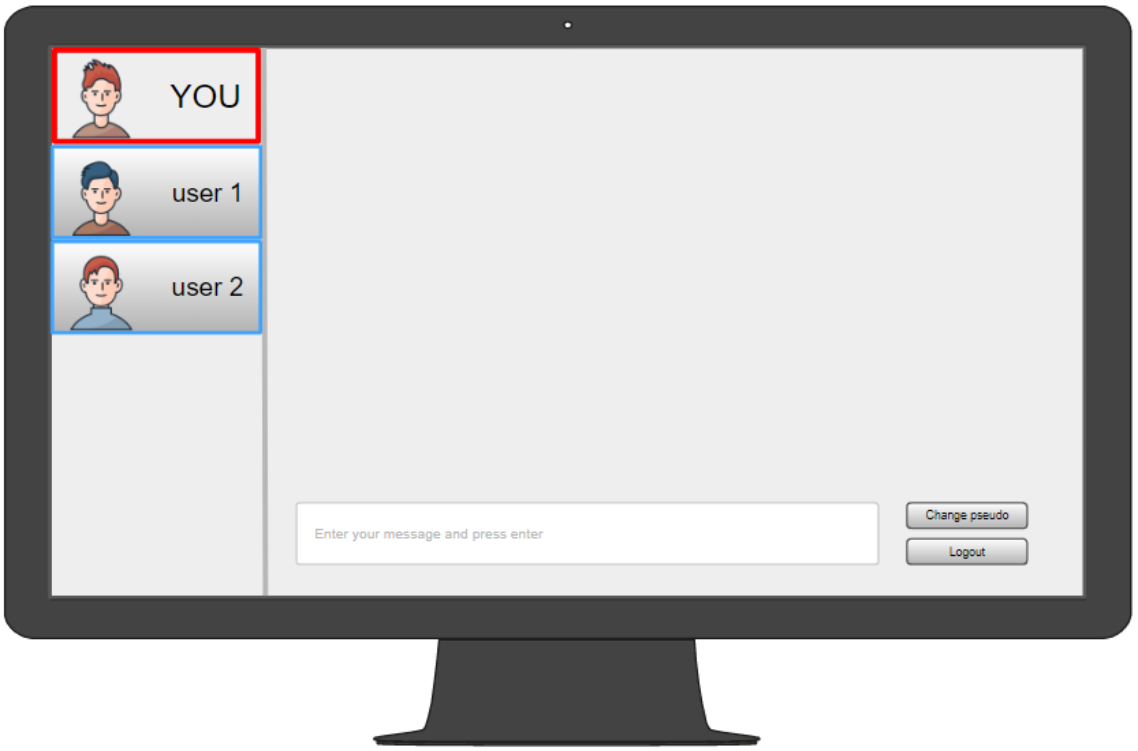


FIGURE 17 – Main scene GUI

Une fois connecté, un avatar nous est attribué et nous apparaissons de notre point de vue tout en haut à gauche de la liste des utilisateurs actuellement connectés au chat. Les autres utilisateurs sont affichés au-dessous dans l'ordre des connexions. Il est possible à partir d'ici de lancer des conversations avec les autres utilisateurs ainsi que de changer de pseudo et se déconnecter.

Matière :	Projet chat system
Date :	27/01/2023
Etudiant(s) :	MARTIN Mattew | CORELLA Gabriel | ROCHE Stéphane
Sujet :	Partie UML

9.3 Changement de pseudo

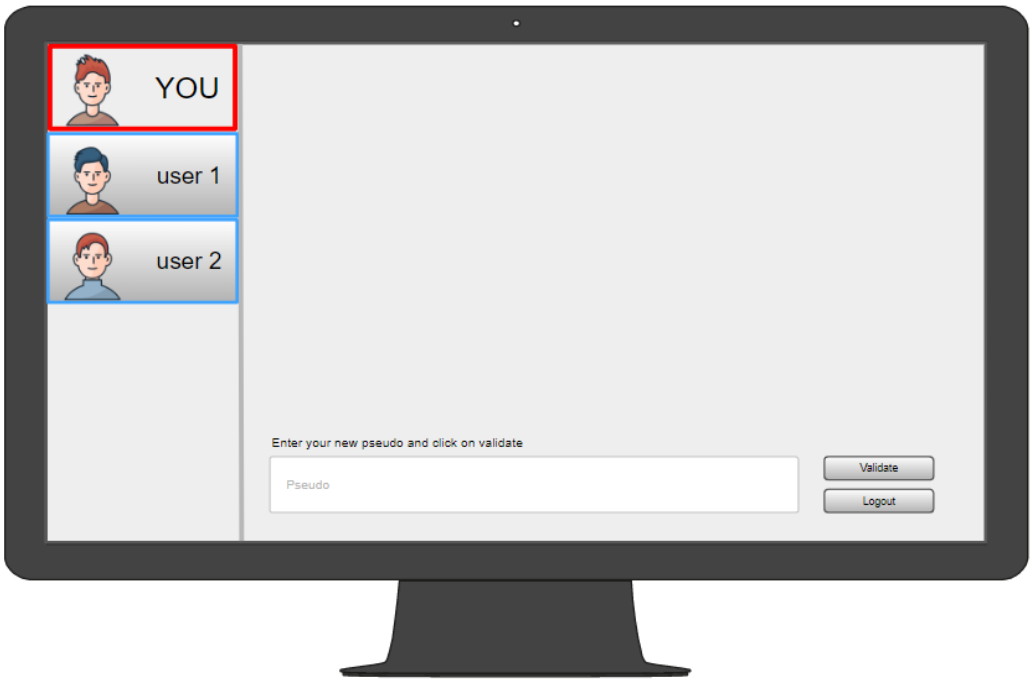


FIGURE 18 – Main scene GUI (On pseudo change)

Pour ce faire, il suffit de cliquer sur le bouton nommé "**Change pseudo**". Une fois cela fait, ce bouton sera remplacé par "**Validate**" et il ne restera plus qu'à entrer le nouveau pseudo et cliquer sur ce bouton.

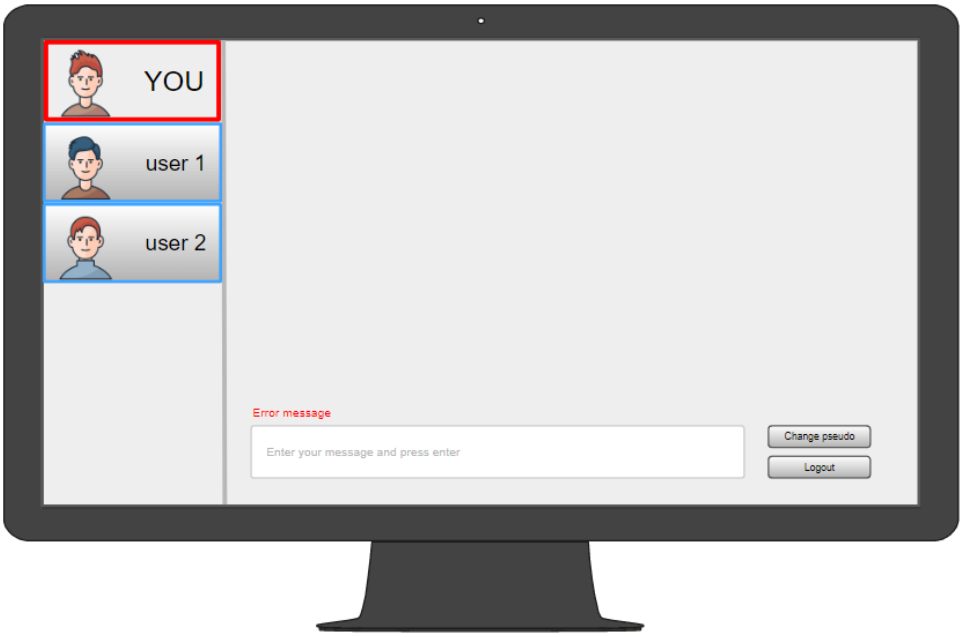


FIGURE 19 – Main scene GUI (On pseudo change mismatch)

Les **contraintes d'unicité** et de **taille** sont également prises en compte ici. Donc, en cas de non-respect de l'une de ces contraintes, un message d'erreur sera affiché avec l'origine de celle-ci.

Matière :	Projet chat system
Date :	27/01/2023
Etudiant(s) :	MARTIN Mattew | CORELLA Gabriel | ROCHE Stéphane
Sujet :	Partie UML

9.4 Conversation

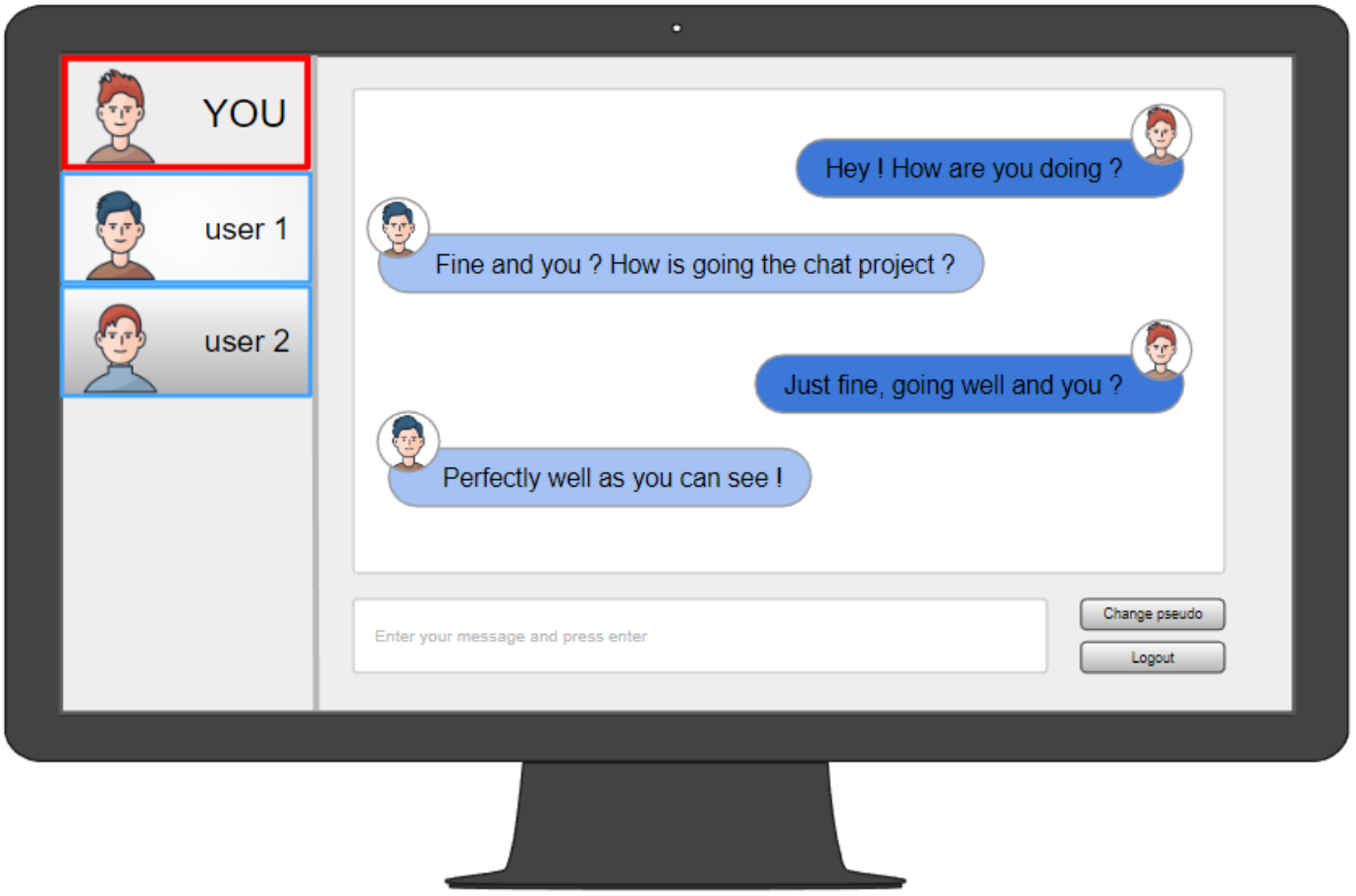


FIGURE 20 – Main scene GUI (On chatting)

Pour **lancer une conversation**, il suffit de cliquer sur le profil d'un des utilisateurs à gauche. Une fois cela fait, il vous sera possible de clavarder avec celui-ci. Les couleurs des bulles de messages ainsi que les avatars affichés à côté des messages permettent une meilleure lisibilité et compréhension de quels sont les messages entrants et sortants.

Matière :	Projet chat system
Date :	27/01/2023
Etudiant(s) :	MARTIN Mattew | CORELLA Gabriel | ROCHE Stéphane
Sujet :	Partie UML

10 Procédures d'évaluation et de tests

Il existe différents types de tests permettant d'évaluer la qualité et le bon fonctionnement de notre code. Dans le cadre de ce projet nous avons décidé d'en retenir certains que nous allons vous présenter :

- **Tests unitaires :** Il s'agit de tests qui vérifient le bon fonctionnement d'une unité de code spécifique, comme une méthode ou une classe. Il existe plusieurs frameworks de test et nous avons décidé d'utiliser JUnit4
- **Tests d'intégration :** Il s'agit de tests qui vérifient le bon fonctionnement des différentes parties d'un système lorsqu'elles travaillent ensemble. Par exemple, vérifier que la base de données et l'interface utilisateur communiquent correctement.
- **Tests utilisateur :** Il s'agit de tests réalisés par des utilisateurs finaux pour vérifier que l'interface utilisateur est intuitive et facile à utiliser.
- **Test d'acceptation :** Il s'agit de tests qui vérifient que le système répond aux besoins spécifiés c'est-à-dire au cahier des charges dans notre cas.

11 Procédures d'installation et de déploiement

Afin d'installer et déployer correctement l'application de clavardage il vous faudra un ordinateur fonctionnant sur un OS suffisamment récent et ce qu'importe la distribution (Windows, Linux , Mac).
Il vous faudra **Java** ainsi que le **Java Development Kit (JDK)** qui désigne un ensemble de bibliothèques logicielles de base du langage de programmation Java. De plus, **aven** sera nécessaire au bon déploiement puisqu'il vous servira pour la commande de déploiement.

Pour ce qui est du déploiement, une simple commande suffit à compiler et exécuter le programme :

```
mvn compile && mvn -X exec:java -Dexec.mainClass="userInterface.LaunchGUI"
```

Si jamais celle-ci ne fonctionne pas, il vous faudra préciser le chemin menant au fichier "mvn", il faudra alors pour chaque commande mvn ci-dessus se placer au bon endroit comme dans cet exemple :

```
../maven/bin/mvn compile && ../maven/bin/mvn -X exec:java -Dexec.mainClass="userInterface.LaunchGUI"
```

Une fois cela fait, vous vous retrouverez face à l'écran d'accueil de notre application vous demandant de choisir un pseudo puis de vous connecter.

Matière :	Projet chat system
Date :	27/01/2023
Etudiant(s) :	MARTIN Mattew | CORELLA Gabriel | ROCHE Stéphane
Sujet :	Partie UML

12 Manuel d'installation et d'utilisation

Installation :

1. Assurez-vous d'avoir une version récente de Java installée sur votre ordinateur.
2. Téléchargez le fichier de l'application à partir d'un référentiel de code source tel que GitHub.
3. Ouvrez une invite de commande dans le répertoire où se trouve le fichier téléchargé.
4. Pour construire le projet et pour lancer l'application, exécutez la commande :
   ```
   mvn compile && mvn -X exec:java -Dexec.mainClass="userInterface.LaunchGUI"
   ```

Utilisation :

1. Une fois l'application lancée, entrez votre pseudo dans le champ prévu à cet effet.
2. Cliquez sur le bouton "Connexion" pour vous connecter au chat.
3. Pour discuter avec d'autres utilisateurs, cliquez sur leur profil et envoyez-leur un message.
4. Pour changer de pseudo, cliquez sur le bouton "ChangePseudo" et entrez votre nouveau pseudo.
5. Pour vous déconnecter, cliquez sur le bouton “Deconnexion” prévu à cet effet