



INSTITUT NATIONAL
DES SCIENCES
APPLIQUÉES
TOULOUSE

Rapport de projet de COO

Service de messagerie instantanée

Malik SEDIRA, Matthias RIFFARD
4^{ème} année IR, groupe C1

RAPPORT DE PROJET DE COO

-

**Conception et développement d'un
service de messagerie instantanée**

Sommaire :

I. CONCEPTION..... - 2 -

- A. Diagramme des cas d'utilisation - 2 -
- B. Diagrammes de séquence - 2 -
- C. Diagramme de classes..... - 4 -

II. CHOIX ET CONTRAINTES TECHNIQUES - 6 -

- A. Interface graphique - 6 -
- B. Base de données - 6 -
- C. Choix de programmation - 6 -

III. Gestion de projet..... - 7 -

- A. Gestion de version - 7 -
- B. Planification - 7 -
- C. Automatisation de développement..... - 7 -
- D. Intégration continue - 7 -

IV. Manuel d'utilisation - 8 -

- A. Installation - 8 -
- B. Connexion - 8 -
- C. Utilisation du chat..... - 9 -

Conclusion : - 11 -

INTRODUCTION

Dans le cadre du cours de Conception et Programmation avancée de 4ème année Informatique, nous avons l'opportunité de créer et développer un service de messagerie instantanée (chat).

Nous avons pu appliquer nos connaissances acquises depuis le début du semestre en Conception Orientée Objet, en Programmation Orientée Objet, ainsi qu'en Processus de Développement de Logiciel Automatisé et gestion de projet.

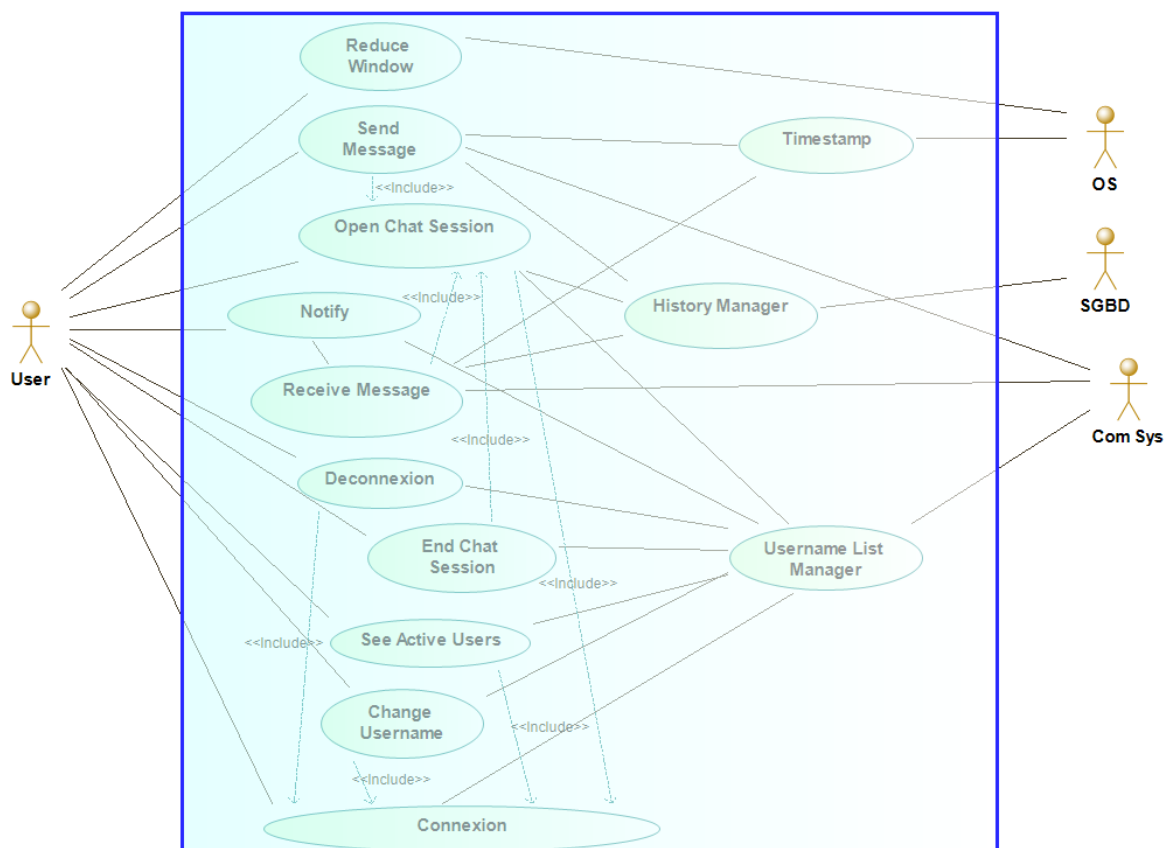
Nous nous appuyons sur le standard UML pour la conception, tandis que le développement se fera en langage Java.

Dans un premier temps, nous présenterons les différentes étapes de conception qui ont précédé le développement. Nous justifierons ensuite les choix technologiques que nous avons adopté. Nous présenterons dans une troisième partie les outils que nous avons utilisés pour nous aider dans la gestion de projet. Enfin, nous nous intéresserons aux méthodes d'automatisation de développement logiciel qui ont pu faciliter le développement et le déploiement dans une dernière partie.

I. CONCEPTION

A. Diagramme des cas d'utilisation

Voici tout d'abord le diagramme des cas d'utilisation sur lequel nous nous sommes basés pour concevoir notre application. Nous nous sommes basés sur le cahier des charges fourni pour le projet afin de le concevoir. Le diagramme suivant est le dernier sur lequel nous nous sommes arrêtés après de multiples révisions avec le « client » représenté par notre encadrant de TD.



Nous verrons dans la suite de ce projet que toutes les fonctionnalités de ce diagramme ont pu être réalisées à l'exception de la fonction notify a été laissée de côté pour privilégier le travail sur les fonctionnalités qui nous ont semblé plus essentielles.

B. Diagrammes de séquence

Nous avons ensuite réfléchi plus précisément au processus qui pourrait mener chacune des fonctions à être réalisées. Pour cela nous avons réalisé des diagrammes de séquences plus ou moins complexes selon les cas. Là aussi, nous avons changé plusieurs fois ces diagrammes après avoir mieux réalisé comment nous pourrions mettre en œuvre

les différents cas d'utilisation en parallèle. Voici quelques-uns de ces diagrammes :

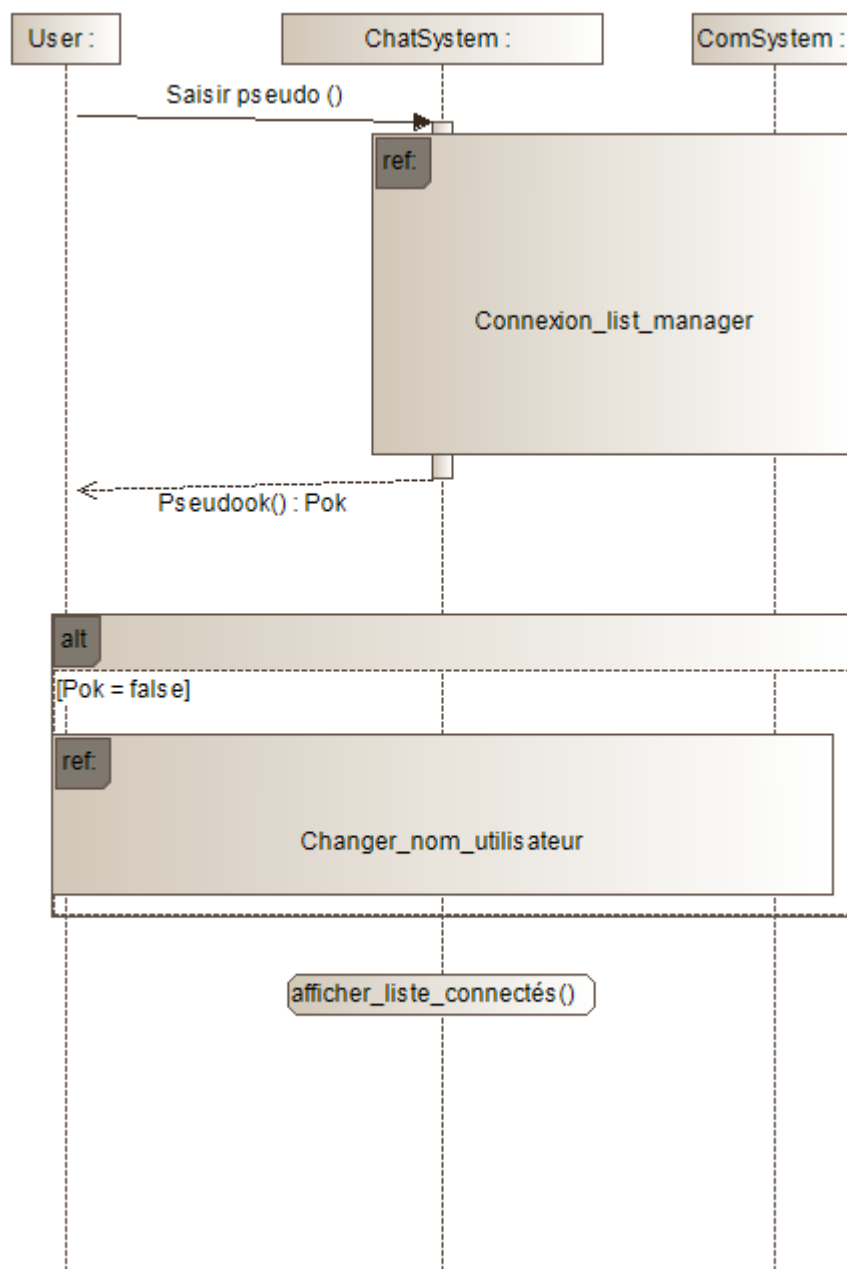


Diagramme de séquence – Connexion

Ce diagramme a été réalisé tôt durant la phase de conception. Le second ci-dessous est beaucoup plus abouti et montre l'évolution que nous avons eu dans la structure de programme durant toute cette première phase.

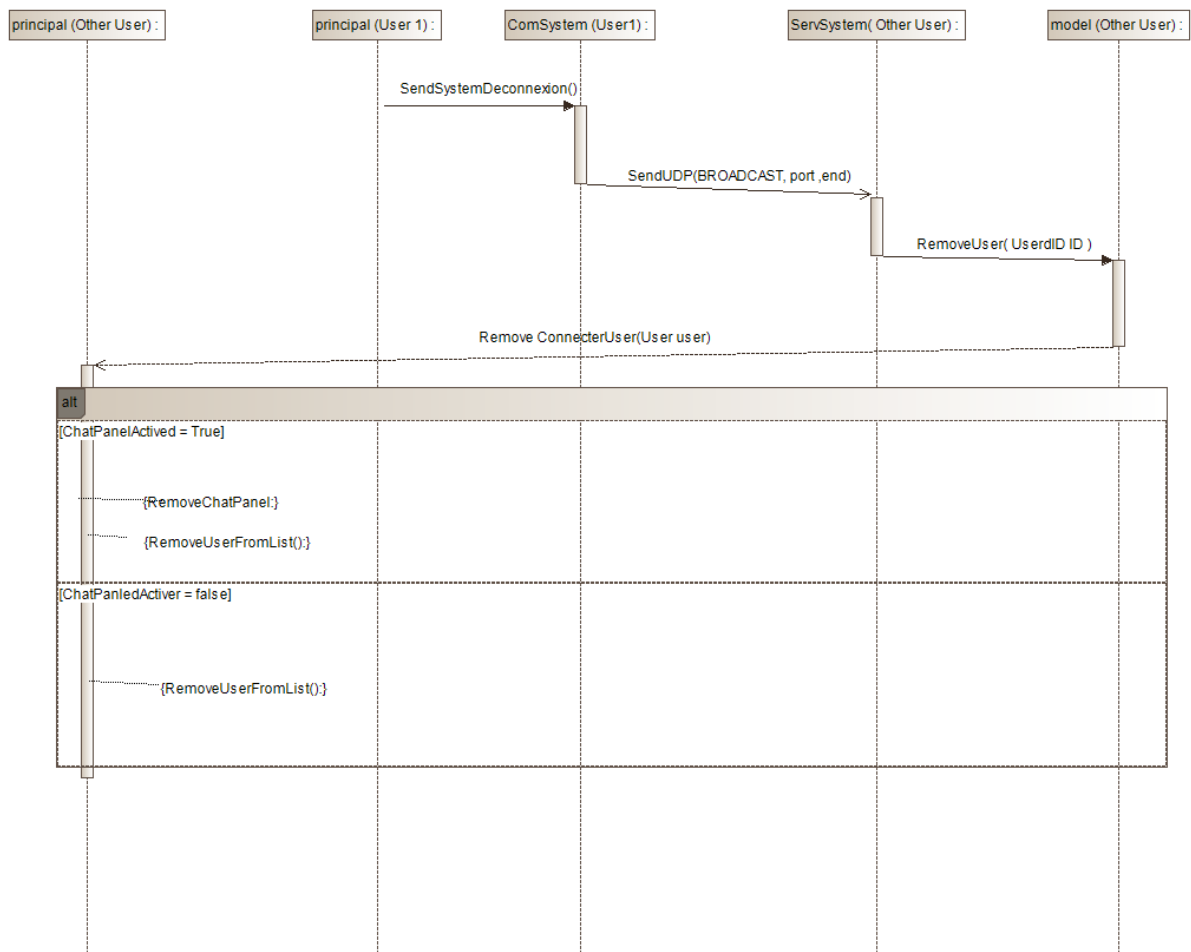


Diagramme de séquence – Déconnexion

C. Diagramme de classes

La dernière étape de notre processus de conception a été la réalisation d'un diagramme de classes, sur lequel nous nous sommes basés pour développer notre application. En particulier, nous avons utilisé Modelio pour générer une structure de code et gagner beaucoup de temps sur l'écriture de code en Java. Celle-ci nous a offert une précieuse base de développement sur laquelle nous avons pu nous appuyer pour travailler en parallèle sur des sections de code différentes.

Le diagramme qui suit est une révision du diagramme dont nous avons initialement généré le code, car nous avons dû opérer quelques changements après nous être confrontés à des difficultés de développement sur la partie réseau notamment.

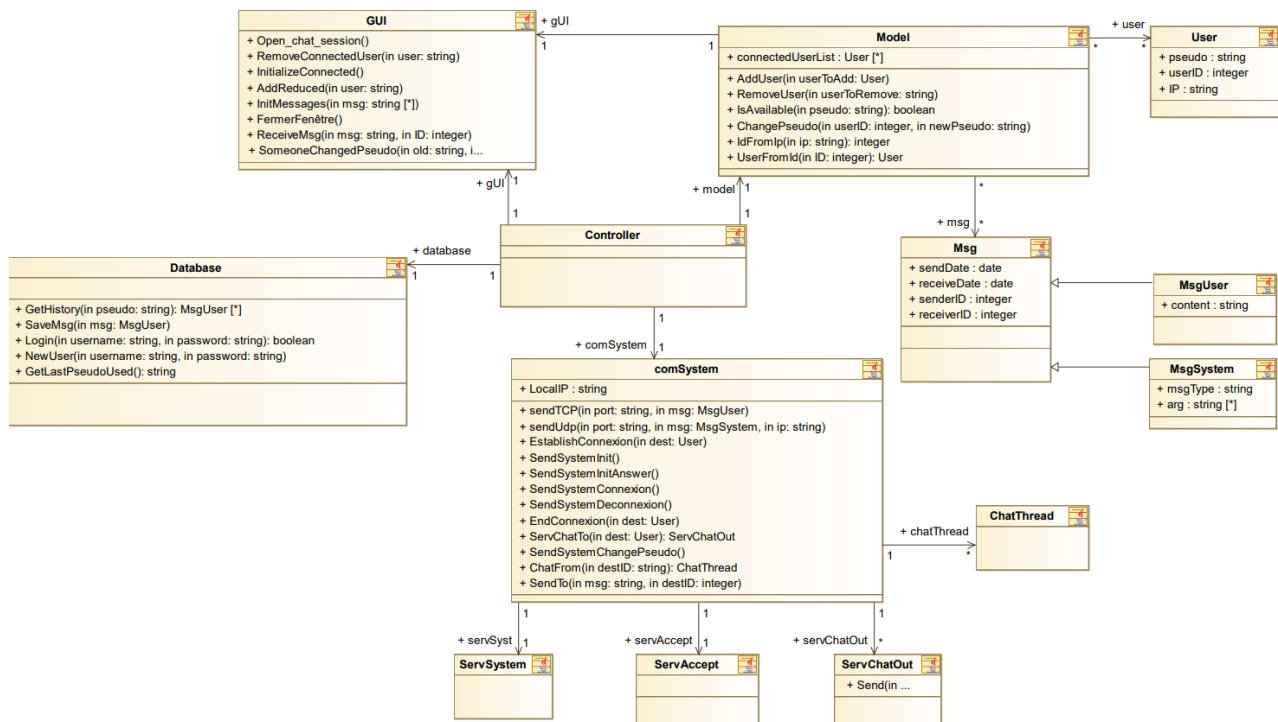


Diagramme de classes

Nous sommes partis d'une base MVC pour construire notre diagramme. Ces classes se sont plus révélées être des packages qui englobent les autres classes que de véritables objets la plupart du temps. Cependant cette base de modèle nous a permis de construire correctement notre logiciel. Même si toutes les classes n'ont pas servi, elles nous ont été très utiles lorsque nous devons faire des liens entre les différentes parties de l'application.

II. CHOIX ET CONTRAINTES TECHNIQUES

A. Interface graphique

Nous avons choisi de réaliser l'interface graphique de notre code en Java Swing, un langage que nous avons déjà utilisé en TD plus tôt dans l'année. Nous avons utilisé l'éditeur Window Eclipse Builder afin de faciliter la génération de code et sa visualisation. Nous avons maqueté notre interface dans un premier temps avant de la reproduire dans l'éditeur, afin de la faire valider auparavant.

B. Choix de programmation

Deux contraintes techniques nous avaient été imposées pour ce projet, la première étant de rechercher les autres utilisateurs connectés par un broadcast UDP, et la seconde de procéder à des échanges directs par TCP pour les messages. Nous avons donc mis en place sur chaque instance de l'application deux serveurs d'écoute différents aux numéros de port fixes : un premier pour les échanges UDP, et un second pour accepter les demandes de communication TCP.

Le premier (ServSystem dans le code) envoie et traite des messages bien définis. Un message « Init » est envoyé lorsqu'un utilisateur lance l'application. « InitAnswer » est la réponse à un tel message, et contient le nom de l'utilisateur de cette instance, ainsi l'émetteur de Init reçoit une liste des pseudonymes indisponibles. Une fois un pseudo libre choisi, « Connexion » est envoyé pour permettre à tous les utilisateurs du réseau d'avoir connaissance de la présence en ligne de l'utilisateur. Lorsqu'un utilisateur change de pseudo, un datagramme portant le message « ChangePseudo » est émis. Enfin, « Deconnexion » informe de la déconnexion de son utilisateur par un nouveau broadcast.

Le second serveur crée un nouveau Thread à chaque sollicitation. Ces threads sont mis en écoute sur un socket destiné à la réception des messages d'un unique utilisateur chacun.

L'envoi se fait lui sur des sockets dédiés, où un thread attend que des données à envoyer lui soient remises. Nous avons pour cela utilisé le préfixe synchronized afin de mettre le thread en attente avec wait().

C. Base de données

Nous avons utilisé une base de données pour stocker les comptes des utilisateurs (nom, mot de passe, historique de messages). Celle-ci nous a été fournie par l'INSA et est stockée sur un serveur de l'Ecole. Nous avons utilisé un package importé pour faciliter l'utilisation de la base de données : my SQL connector java.

Les logins sont stockés sous une table comprenant l'identifiant, le nom d'utilisateur, et le mot de passe. Une autre table contient l'historique de tous les messages, avec deux colonnes pour les identifiants (sender / receiver), une colonne pour la date du message, et une colonne contenant le texte.

III. GESTION DE PROJET

A. Gestion de version

Nous avons utilisé Git comme gestionnaire de versions. Notre approche lors de ce projet a été de limiter au maximum les problèmes de fusion, afin de gagner du temps lors des push vers notre dépôt. Pour cela nous avons travaillé en parallèle sur des parties différentes du code, plutôt base de données et interface pour l'un et mise en réseau des utilisateurs pour l'autre. Grâce à cette répartition, nous avons rarement modifié simultanément les mêmes lignes de code, et n'avons pas ressenti le besoin de créer plusieurs branches.

Nous avons parfois rencontré quelques difficultés avec nos fichiers *.gitignore*, créant des conflits qui n'avaient pas lieu d'être.

B. Planification

Nous avons planifié notre avancement grâce à l'outil Jira qui nous a été présenté en TD au début de l'année. Nous avons eu quelques difficultés à estimer le temps nécessaire pour réaliser certaines tâches, mais fixer des échéances nous a permis de travailler plus régulièrement et de faire des points sur notre avancement à chaque étape du projet.

Nous avons choisi de mettre en place des sprints hebdomadaires. Les user stories n'étaient pas toujours finies à temps et ont donc parfois été reportées sur le sprint suivant. Le fait d'avoir planifié assez tôt nous a permis de garder du temps après le dernier sprint pour finir les tâches en retard et d'inclure du temps non prévu pour des corrections de bugs et des améliorations.

C. Automatisation de développement

Pour faciliter le développement, nous avons créé un projet Maven qui nous permet d'intégrer automatiquement des dépendances au projet. Ici nous avons une seule dépendance, celle du package de connexion à la base de données. Maven nous a surtout permis de générer un unique fichier exécutable pour l'application, ce qui est très confortable pour le déploiement.

D. Intégration continue

Nous avons régulièrement testé notre code au fur et à mesure de son développement. Puisque l'application n'a été réellement utilisable que tard dans le développement, nous avons exécuté des tests manuels après chaque intégration de fonctionnalité sur les dernières séances, et les avons présentées pour nous assurer de leur conformité. Puisque nous avons passé une longue période à la conception de notre programme, les intégrations successives ont rarement posé un problème.

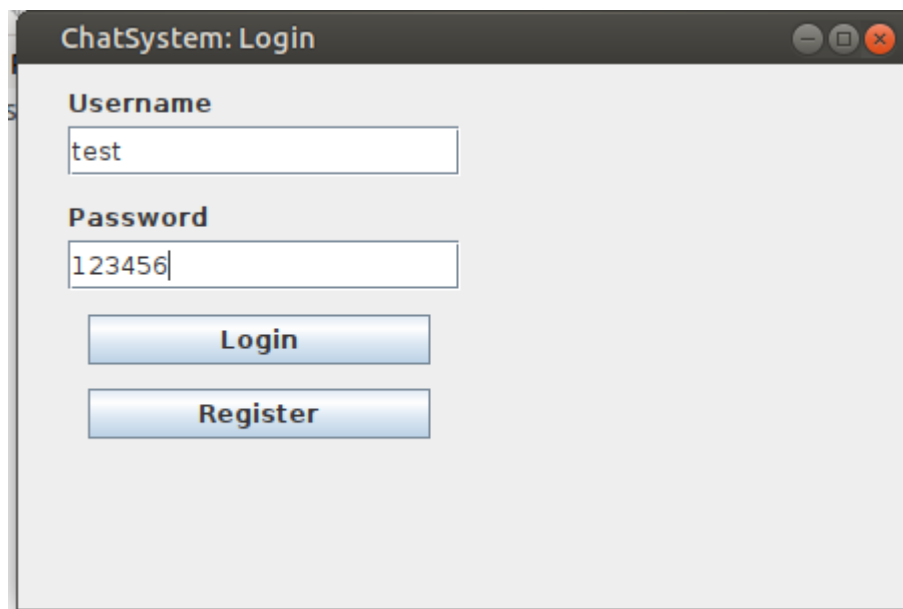
IV. MANUEL D'UTILISATION

A. Installation

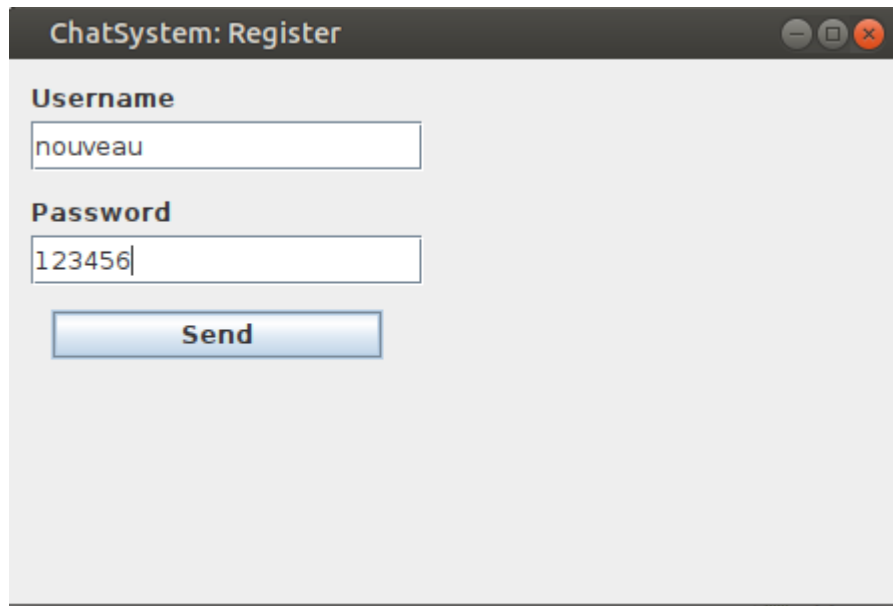
L'ensemble du code est disponible sur le git du projet : https://github.com/moutth/projet_coo/ . L'installation se fait facilement en téléchargeant le fichier ChatSystem.jar, puis en double cliquant sur le fichier (Windows) ou en utilisant la commande `java -jar`.

B. Connexion

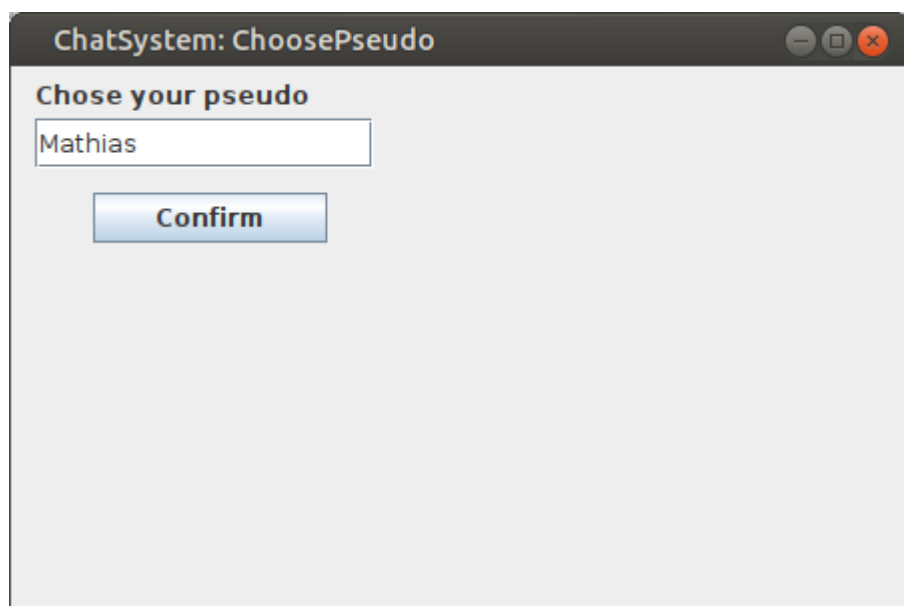
La connexion se fait depuis une fenêtre qui s'ouvre automatiquement au lancement. Si l'utilisateur possède déjà un compte avec lequel il souhaite se connecter, il peut entrer son pseudo et mot de passe puis cliquer sur le bouton Login.



Si l'utilisateur ne possède pas de compte, il peut s'enregistrer en cliquant sur Register. Il lui faut ensuite saisir son pseudo puis son mot de passe, puis cliquer sur Send. Il sera alors renvoyé vers la fenêtre précédente où il pourra se connecter.

A screenshot of a window titled "ChatSystem: Register". It has a dark grey title bar with standard window controls. The main area is light grey. It contains two text input fields: the first is labeled "Username" and contains the text "nouveau"; the second is labeled "Password" and contains the text "123456". Below these fields is a blue button with the text "Send".

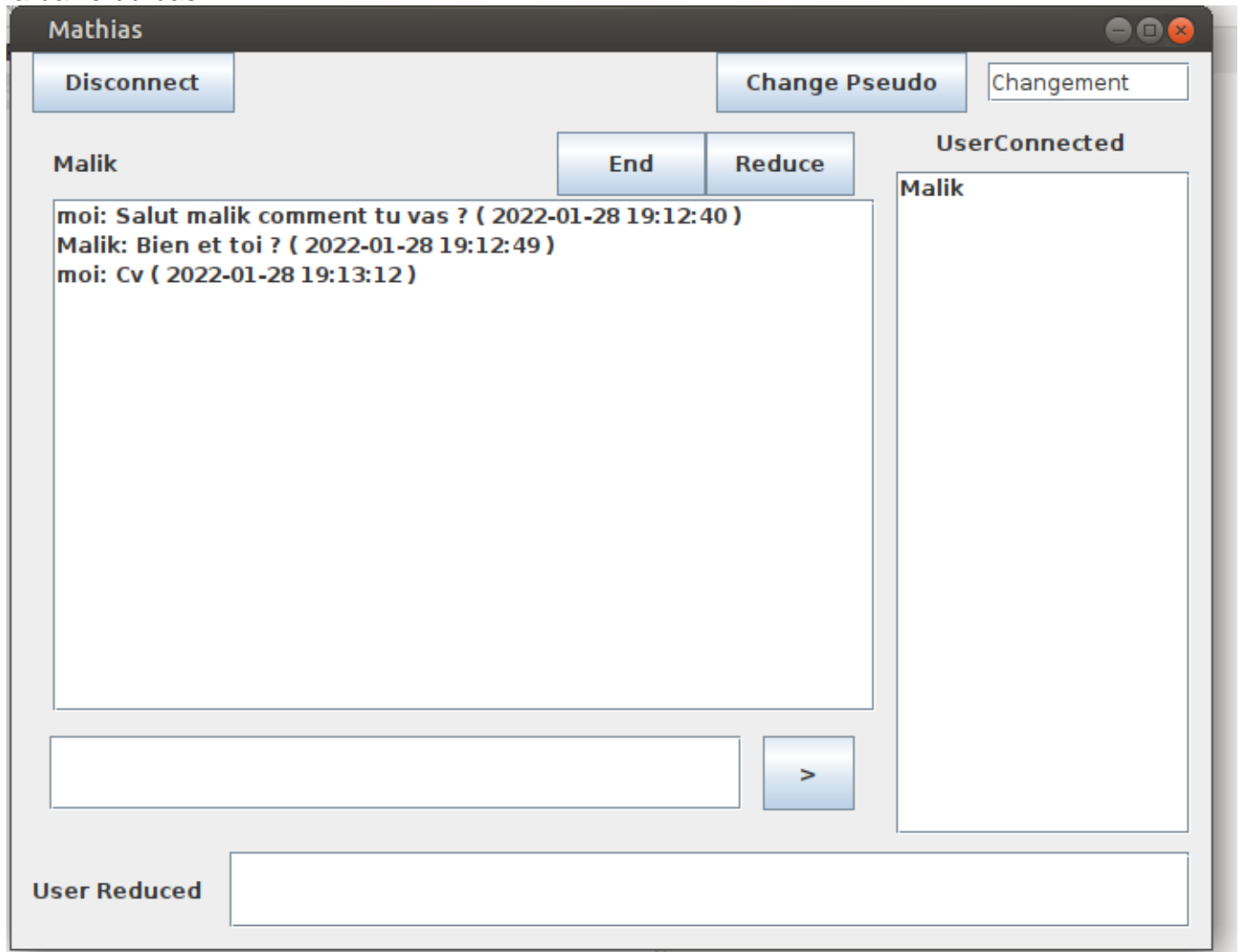
Il est ensuite demandé à l'utilisateur de saisir le nom sous lequel il souhaite apparaitre chez les autres utilisateurs. Si le pseudo souhaité n'est pas disponible, cela signifie qu'un autre utilisateur est déjà connecté sous ce même nom. Cela lui sera précisé et il faudra saisir un autre nom.

A screenshot of a window titled "ChatSystem: ChoosePseudo". It has a dark grey title bar with standard window controls. The main area is light grey. It contains a text input field with the text "Mathias". Below the field is a blue button with the text "Confirm".

C. Utilisation du chat

Une fois connecté avec succès, la fenêtre principale du logiciel s'ouvre. Un bouton Disconnect permet de se déconnecter et de revenir à la fenêtre de connexion. Attention : il est fortement recommandé de se déconnecter avant de fermer l'application. Il est possible de changer de pseudo à tout moment en saisissant un nouveau nom dans la zone de texte située en haut à droite de l'écran puis en cliquant sur Change Pseudo. Le pseudo courant est affiché dans la barre supérieure de l'application.

La liste des utilisateurs connectés apparaît à droite de l'écran. L'ouverture d'une session de chat se fait en cliquant sur le nom d'un des utilisateurs. L'historique des conversations avec cet utilisateur apparaît alors. L'envoi d'un message se fait en tapant dans la barre située sous la fenêtre des messages puis en cliquant sur la flèche. Il est possible de mettre fin à la session en cliquant sur End. Les messages de l'interlocuteur ne seront pas perdus et sauvegardés pour apparaître dès la prochaine connexion. Enfin, il est possible de réduire la fenêtre de dialogue en cliquant sur Reduce. Cette fenêtre sera alors mise en attente dans la barre du bas.



Conclusion :

Nous sommes satisfaits du produit réalisé. Bien qu'imparfait (quelques bugs subsistent, en cas de fermeture de l'application sans déconnexion par exemple), nous avons développé toutes les fonctionnalités que nous avons planifiées. Ce projet était pour nous le premier projet de programmation de cette envergure. Il nous a permis de réaliser l'importance de planifier un projet à son commencement, et les nombreuses séances de gestion de projet se sont révélées bénéfiques pour mener à leurs termes les différentes phases du projet. Nous avons retenu l'importance de tester régulièrement le produit, mais surtout de prendre le temps de concevoir l'application avec le client pour éviter de développer de fastidieuses fonctionnalités qui seront rejetées et pour poser des bases qui seront essentielles au développement.

Nous tenons à remercier nos enseignants de TD et TP qui ont pris le temps de nous faire des retours complets et de précieuses critiques constructives tout au long de ce semestre.

– Déconnexion