INSA
INSTITUT NATIONAL
DES SCIENCES
APPLIQUÉES
TOULOUSE

RAPPORT
APPLICATION
SYSTÈME DE CLAVARDAGE

CHABLOZ Ellias
AHAMDY Jdihadi

INTRODUCTION :

Le but de ce projet est la mise en place d'un système de communication, dans le cadre d'une entreprise afin d'améliorer l'efficacité et la communication au sein de l'entreprise. Le système doit permettre aux utilisateurs l'échange de messages textuels entre eux en temps réel.

Pour la mise en place de ce logiciel plusieurs étapes seront détaillées à la suite de ce documents, l'analyse du cahier des charges , la phase de conception orienté objet puis la mise en place d'une feuille de route pour travailler en méthode Agile (Jira System). Cette méthode permet lors de la phase de production une meilleure prise en main du projet, avec un suivi continu sur les objectifs définis en début de projet.

CAHIER DES CHARGES:

Définition des exigences :

Le système peut se décomposer en deux parties entre les fonctionnalités utilisateur et administrateur.

Les fonctionnalités administrateur sont :

→ Un déploiement facile sur les différents systèmes d'exploitations (Windows,Linux, OS X etc ..)

→ Un installation et accès simplifié à l'application

→ Une exigence de poids, les fichiers du système ne doivent pas excéder 50 Méga-Octets

Les fonctionnalités utilisateur sont :

→ Permettre l'utilisation d'un pseudonyme unique à l'ouverture du système.

→ La capacité d'identifier tous les utilisateurs actif sur le réseaux.

→ Permettre la mise en communication entre deux utilisateurs actif du système.

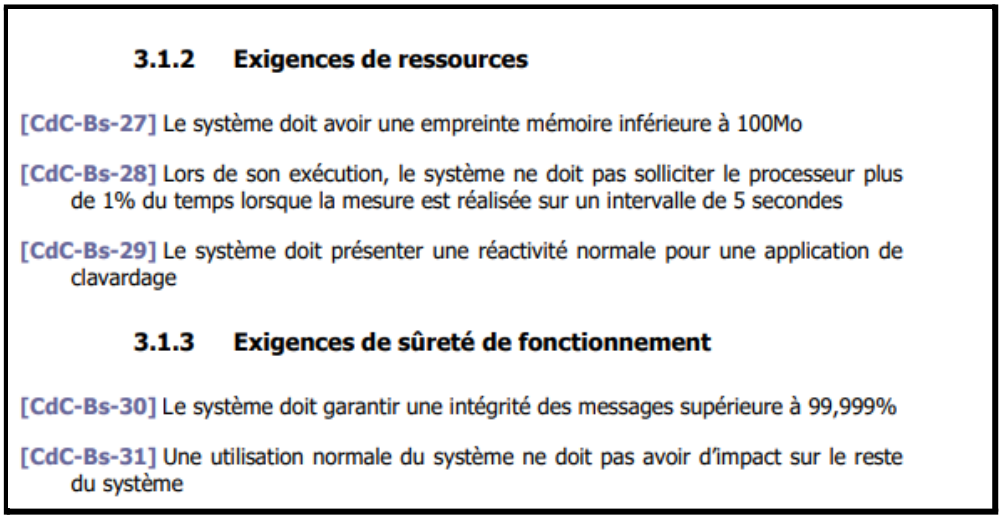
→ L'horodatage des messages échangés entre utilisateur.

→ Accès à l'historique des messages à l'ouverture d'une session de discussion.

→ La capacité de changer son pseudonyme à tout moment et garantir l'unicité de ce dernier.

→ Être notifié à chaque connexion et déconnexion d'utilisateur.

Puis il y a aussi les exigences de fonctionnalités opérationnelles non négligeable:



3.1.2 Exigences de ressources

[CdC-Bs-27] Le système doit avoir une empreinte mémoire inférieure à 100Mo

[CdC-Bs-28] Lors de son exécution, le système ne doit pas solliciter le processeur plus de 1% du temps lorsque la mesure est réalisée sur un intervalle de 5 secondes

[CdC-Bs-29] Le système doit présenter une réactivité normale pour une application de clavardage

3.1.3 Exigences de sûreté de fonctionnement

[CdC-Bs-30] Le système doit garantir une intégrité des messages supérieure à 99,999%

[CdC-Bs-31] Une utilisation normale du système ne doit pas avoir d'impact sur le reste du système

Extrait du cahier de charge cf moodle Conception orientés objet

Réalisation des exigences :

Parmi les fonctions et les exigences cités auparavant voici un bilan des buts fixés et atteints et une brève explication des résultats obtenus. Pour l'organisation de ce projet on peut distinguer trois majeures parties: le back-end, le front-end et l'intégration. Ce qui implique donc une vérification en trois phases (ou deux pour certaines fonctionnalités).

L'ensemble des fonctionnalités est réalisable en back-end, les vérifications sont faites par des données d'entrées fictives et les sorties se font via le terminal. Cependant ayant rencontrés des soucis lors de l'intégration certaines fonctionnalités restes sans visuels explicites, on a pu distinguer parmis ces cas des conditions particulières pour obtenir les fonctionnalités attendues.

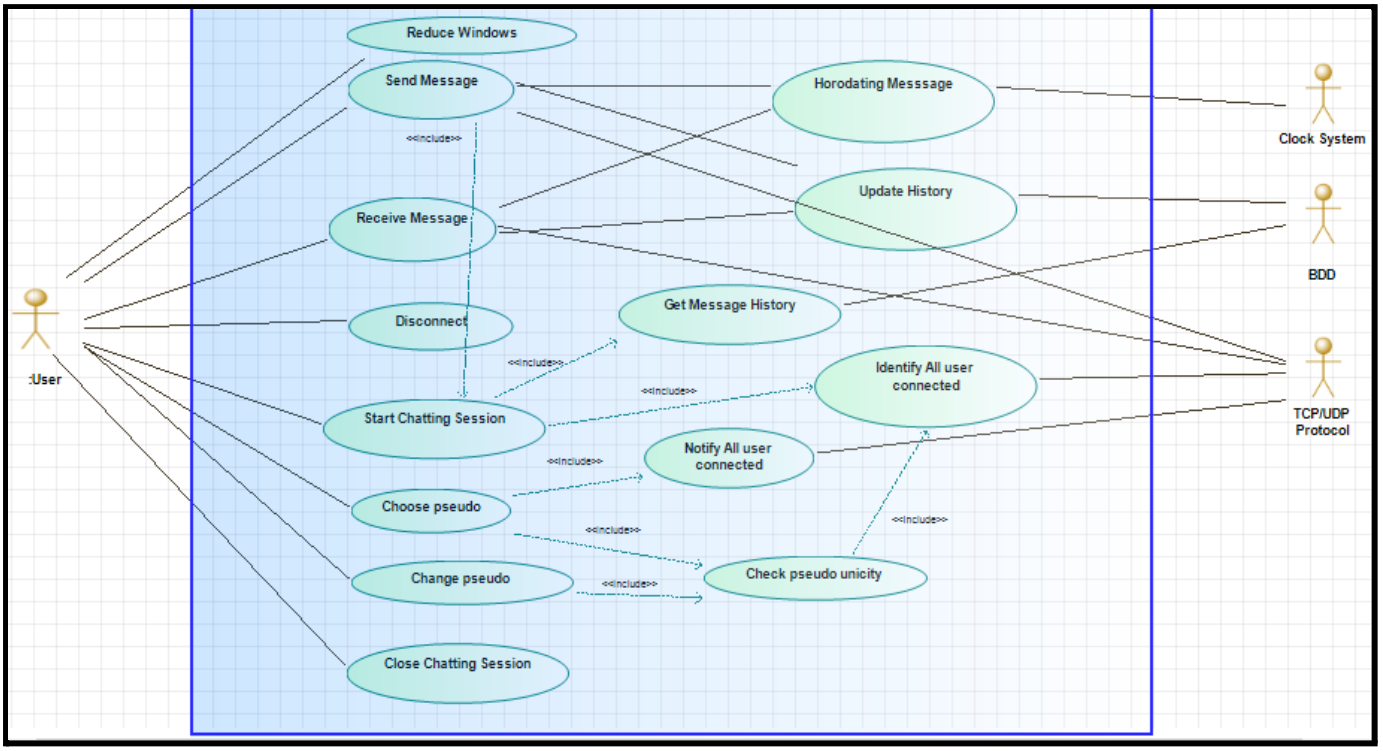
Fonctions principales :

- Reduce Window : OK
- Send/Receive Message : l'envoi de message effectué via TCP fonctionne entre deux utilisateurs si la session est ouverte, cependant on a du mal à afficher le message reçu au niveau de l'interface.
- Disconnect : Ok,
- Start Chatting Session : Crée bien un socket pour communiquer entre deux utilisateurs.
- Choose pseudo : Ok
- Change Pseudo : Ok. Le changement de pseudo s'affiche bien dans le terminale.
- Close Chatting Session : Ferme le socket ouvert précédemment.
- Horodatage : Ok
- Update History : Ok.
- Get History : Ok
- Identify Connected Users : On n'arrive pas à actualiser l'interface pour afficher la nouvelle liste malgré le fait que la nouvelle liste ait été mise à jour.
- Notify Connected Users : Ok
- Check pseudo unicity : Ok

COO :

Après la prise de connaissance du cahier des charges vient la phase de conception, ou en fonctions des exigences on dresse les diagrams UML, nous permettant ainsi d'avoir une meilleure approche du projet.

Pour la première analyse on établit un premier diagramme des cas d'utilisation pour formaliser et classer nos utilisateur et nos principales fonctionnalités.



Diagrammes des cas d'utilisation du système

Puis dans un second temps on établira des diagrammes de séquences en boîte blanche et boîte noire pour obtenir des idées des interfaces et classes à implémenter durant le projet.

Au vues des diagrammes obtenues et des choix faits pour la réalisation des fonctionnalités, on sait basé sur un design pattern pour la conception de ce système, on a choisi le model (Model, View et Controller)MVC. Aux vues des différentes fonctionnalités demandées, ce modèle correspond le mieux à nos besoins. En effet ayant besoin des fonctionnalités réseaux tel que TCP, UDP et d'un rendue graphique ce modèle nous paraît donc le plus pertinents dans les design patterns vues en cours. Les captures d'écrans n'étant pas optimaux pour une bonne visualisation vous pouvez retrouvez l'ensemble des diagrammes de notre parties conception via ce lien : https://a-djihadi.github.io/chatting_project/chatting_project/pdla_chatting_project/doc/UML_project/0.html

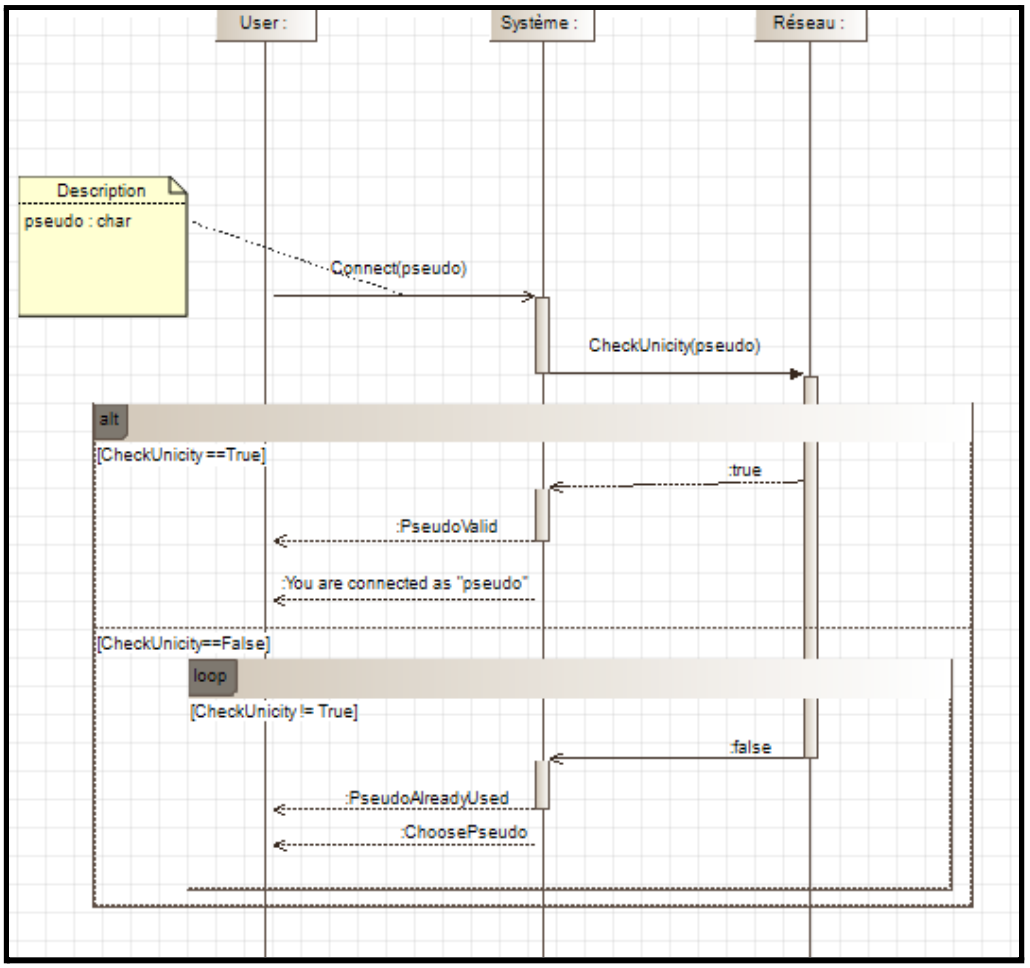


Diagramme de séquence boîte blanche

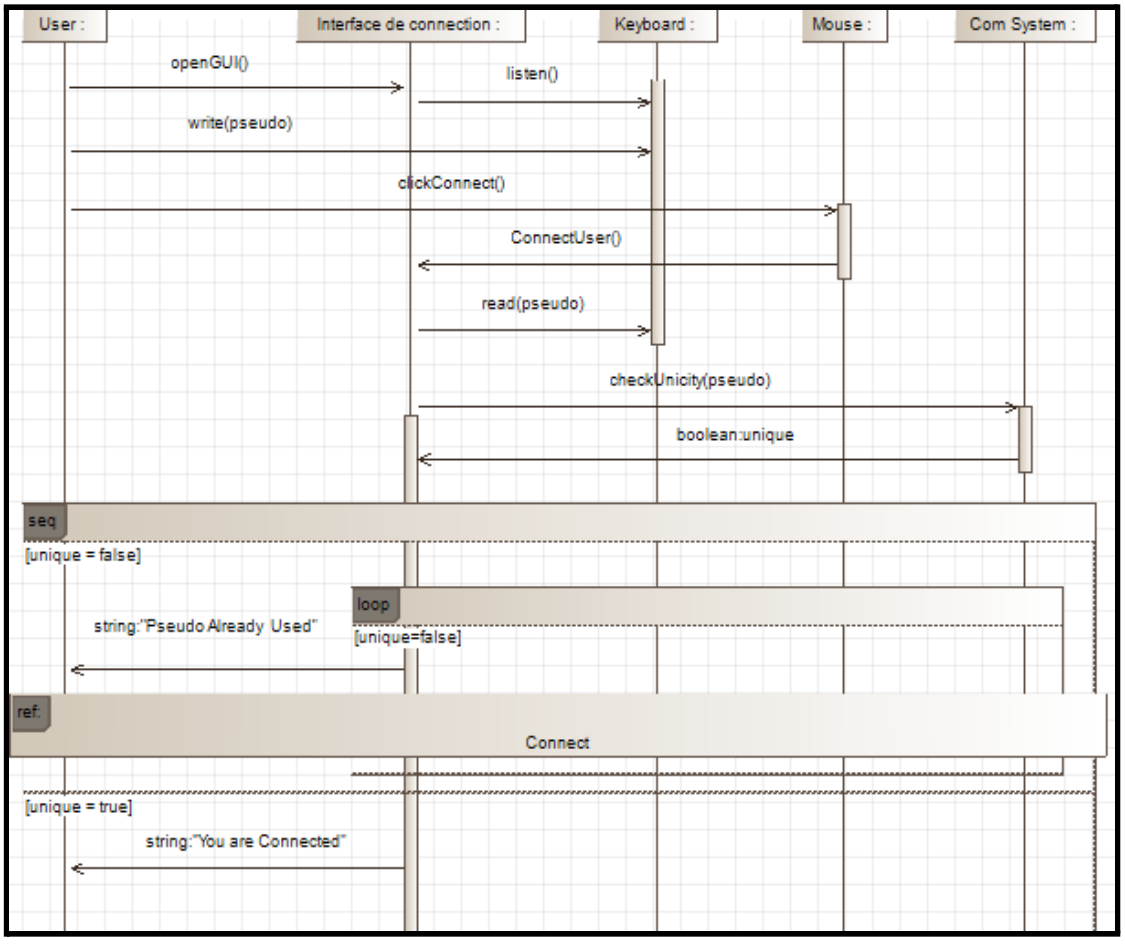
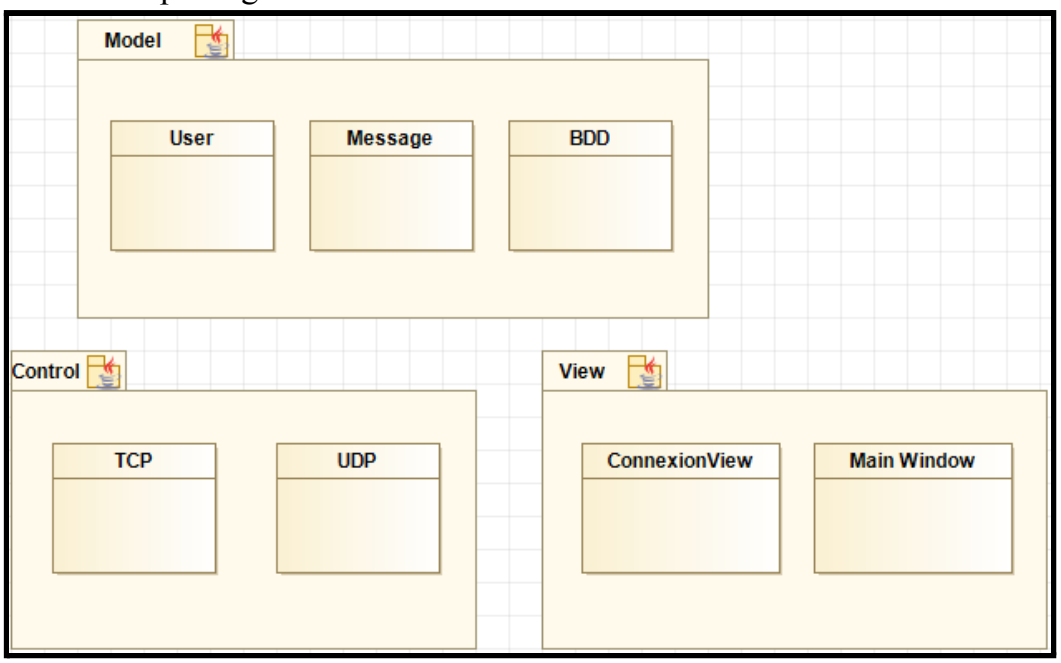


Diagramme boite noire:

On obtient donc ce package :



Package des classes obtenues selon le modèle MVC

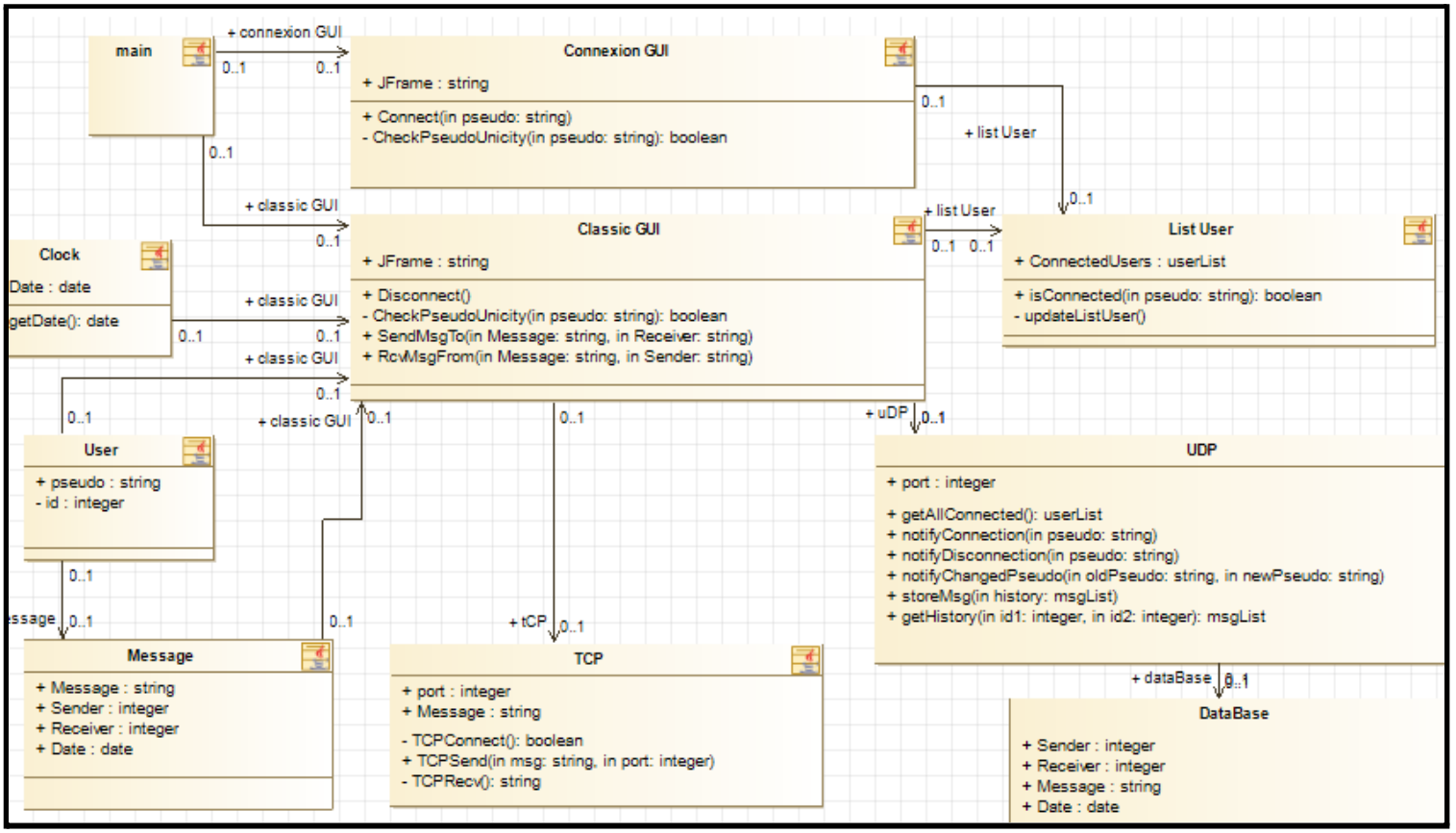


Diagramme des classes

POO:

Architecture du système :

BDD :

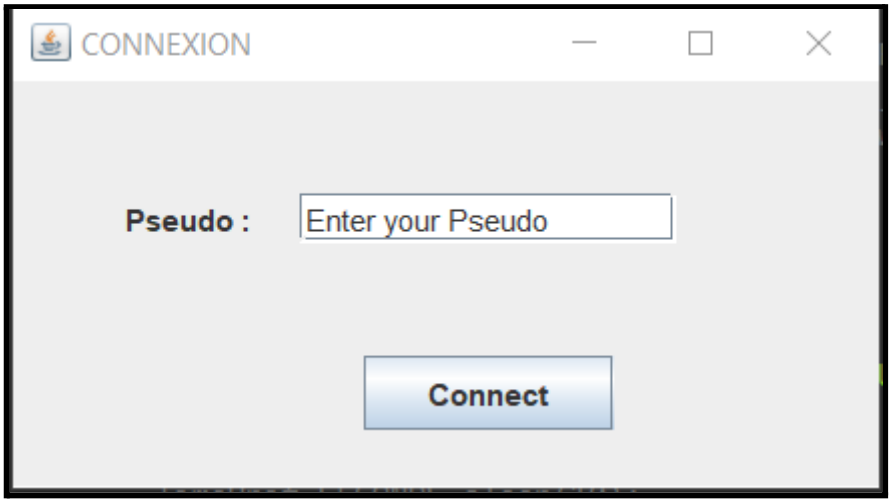
→Nous avons choisi la BDD centralisée proposée par l'INSA. Nous utilisons une table MESSAGES organisée de la forme suivante : (insert capture of mysql)

→ Lib utilisée : mysql connector Java

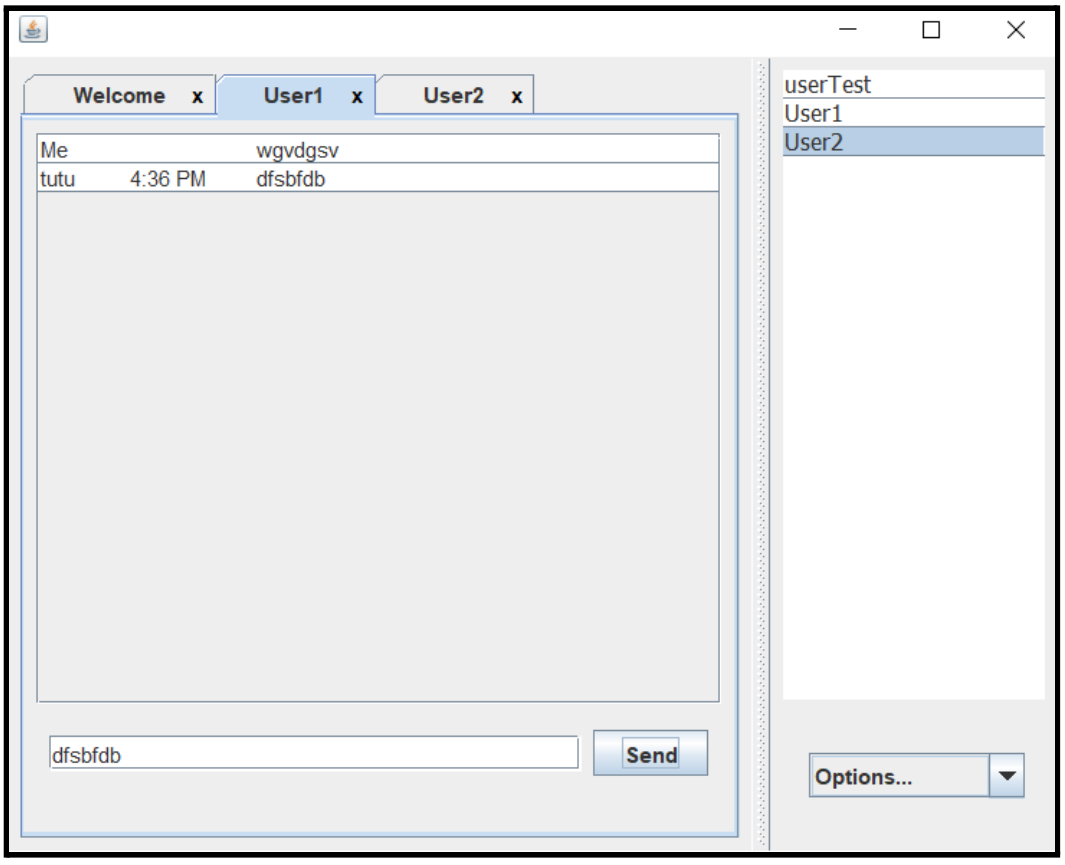
GUI :

Nous avons deux frames : une pour l'authentification (ConnexionGUI) et une principale (MainWindow). Nous avons également une "popup" pour le changement de pseudo.

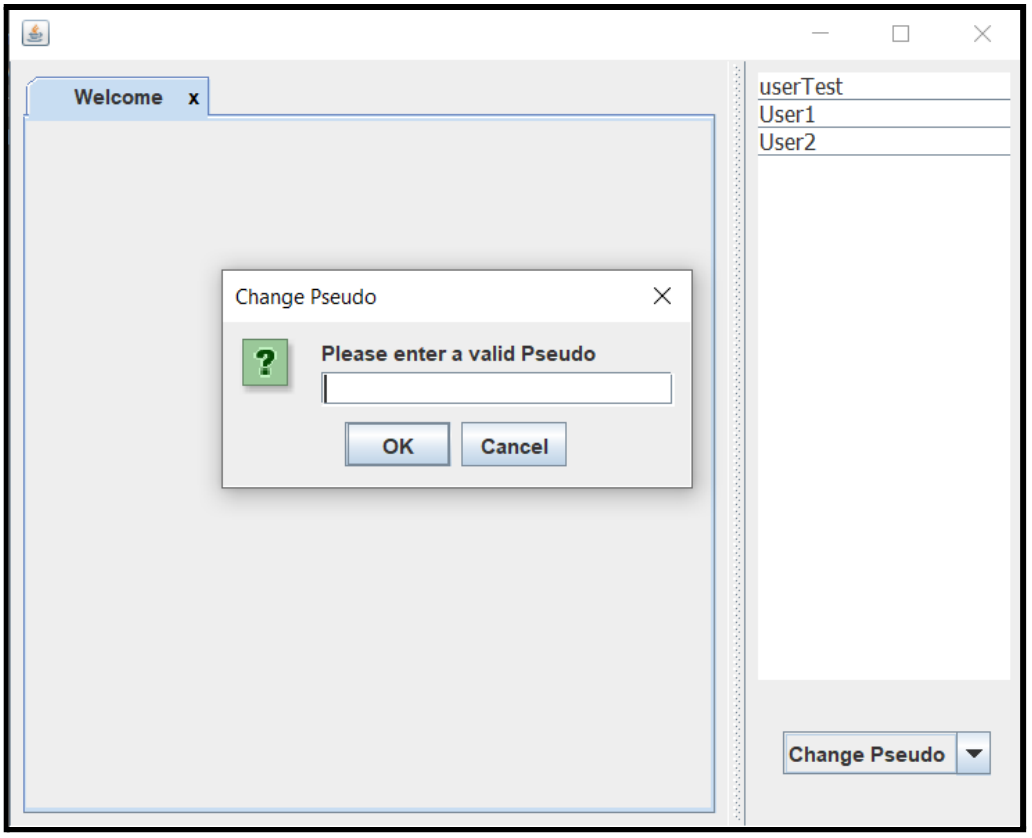
Afin de coder ces frames nous avons utilisé la librairie MigLayout.



ConnexionGUI



MainWindow



PopUp to Change Pseudo

Procédure de tests :

Un seul test JUnit à été codé : il permet de tester que les query dans la bdd est valide.

Déploiement :

Lancer dans le répertoire la commande : mvn package
Exécuter la commande : java -jar projet_chat.jar

Manuel d'utilisation simplifié :

Connexion :

Pour se connecter, entrez un pseudo dans la fenêtre de connexion. Si votre pseudo est unique, vous êtes authentifié, sinon il faut entrer un autre pseudo car il est déjà utilisé.

Communication :

Pour communiquer, cliquez sur un utilisateur connecté dans la liste à droite. Un onglet refermable s'ouvre vous pouvez entrer votre message puis l'envoyer.

Changement de pseudo :

Pour changer de pseudo, cliquez sur le bouton *Options* puis choisissez dans le menu déroulant l'option *Change pseudo*. Enfin, entrez votre nouveau pseudo.

Déconnexion :

Pour vous déconnecter, cliquez sur le bouton *Options* puis choisissez dans le menu déroulant l'option *Deconnection*.

Organisation du projet :

JIRA :

Nous avons réalisé 5 sprints au total :

- Tableau Sprint 1 : 02/12/21 - 09/12/21 -> début de codage sur UDP + Interface de connexion
- Tableau Sprint 2 : 15/12/21 - 21/12/21 -> Exploration de TCP, poursuite sur UDP
- Tableau Sprint 3 : 30/12/21 - 06/01/22 -> UDP fonctionnel, avec notifications(broadcast) + Liste d'utilisateurs connectés
- Tableau Sprint 4 : 08/01/22 - 14/01/22 -> TCP fonctionnel en sortie système, communication entre deux machines réussie + Interface de la MainWindow
- Tableau Sprint 5 : 14/01/22 - 28/01/22 ->Ajout de la BDD + Réunion entre les fonctions et les interfaces

Le début du projet a mis du temps à s'engager car nous avions commencé sur nos machines personnelles et pour des raisons inconnues les broadcasts ne fonctionnaient pas dans un sens de lancement et ce avec un code exactement similaire. Par la suite, nous avons vu qu'il valait mieux utiliser directement les machines de l'INSA pour les tests.
Autre problème rencontré, lors de l'implémentation nous avons perdu du temps sur une erreur qui après s'avérait être due à la nécessité du VPN sur nos machines personnelles.

BILAN:

Ce projet nous à beaucoup appris, et permis de mettre en application nos connaissances en réseau et en POO.
L'outil de gestion de projet est très utile même si notre utilisation est encore fragile et détachée, c'est à la fin du projet qu'on se rend compte de comment il aurait été mieux de faire au niveau des sprints.La partie codage est très enrichissante mais nous avons constaté des difficultés car nous nous sommes un peu détachés de notre travail de COO et avons par endroit beaucoup changé notre implémentation par rapport à ce que nous avions pensé au début. Ces changements entraînent donc des implémentations plus longues, des remaniements et au final notre projet n'est pas intégralement opérationnel. Nous avons également eu beaucoup de difficultés à réunir nos méthodes principales avec notre interface graphique, et ce alors que toutes les méthodes d'échange fonctionnent bien lors de tests via la sortie système, cela nous a aussi apporté beaucoup de recul sur la conception de projet en indépendant, car sans une conception bien défini, l'intégration du back-end au front-end est une opération périlleuse ou on est amené à refaire des choix pour respecter les fonctionnalités attendues.

INSA Toulouse
135, avenue de Rangueil
31077 Toulouse Cedex 4 - France
www.insa-toulouse.fr

Université Fédérale Toulouse Midi-Pyrénées
Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE
MINISTÈRE
DE L'ÉDUCATION NATIONALE,
DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE