INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Conception Orientée Objet

Système de clavardage distribué interactif multi-utilisateurs temps réel

Shivaree PIMENTA, Nathan LEMMERS
4ème année Systèmes Informatiques - IR A2

Table des matières

1 Introduction **3**

2 Conception et diagrammes réalisés **3**
2.1 Diagramme des cas d'utilisation 3
2.2 Diagramme de classes 4
2.3 Diagramme de séquences 8

3 Architecture du sytème et choix technologiques **10**
3.1 Architecture de la base de données 10
3.2 Architecture des interfaces GUI 11

4 Procédures **12**
4.1 Procédures d'évaluation et de tests 12
4.2 Procédures d'installation et de déploiement 12

5 Manuel d'utilisation simplifié **13**

.

1 Introduction

Le système a pour mission d'être le support communicant des équipes de l'entreprise afin de leur permettre d'accroitre leur efficacité naturelle.

D'autre part, en tant que relais de communication, le système permet aux intervenants de se coordonner par l'échange de messages textuels entre eux mais également des illustrations sous la forme d'images, des schémas en plus de permettre l'échange de fichiers de travail.

Lien vers notre code : https ://github.com/nathanlemmers/ChatSystem

2 Conception et diagrammes réalisés

2.1 Diagramme des cas d'utilisation

Le diagramme des cas d'utilisation permet de capturer les principales caractéristiques et interactions de notre système de clavardage lorsqu'il est executé. Les acteurs primaires du chat system sont l'utilisateur qui est initiateur du message et l'administrateur qui va déployer le réseau sur les différents systèmes d'exploitation. En théorie, même si cela n'est pas sa fonction principale, l'administrateur peut se créer un compte utilisateur par exemple pour vérifier que le système est bien déployé. C'est pourquoi il existe un lien unilatéral entre ces deux acteurs. L'acteur secondaire est le deuxième utilisateur, celui qui va être à la réception du message, en sortie du système.

Un administrateur est donc principalement en contact avec le système pour le déployer. L'utilisateur, lui, utilise principalement le système pour communiquer avec les autres utilisateurs connectés. C'est à dire se connecter/se déconnecter, envoyer des messages (ou fichiers), et en recevoir. Pour cela, il doit pouvoir avoir accès à l'annuaire à jour des utilisateurs connectés. Lors de sa connexion, la correspondance pseudo/mot de passe est vérifiée. En parallèle, l'utilisateur peut aussi modifier son pseudo ainsi que son mot de passe.

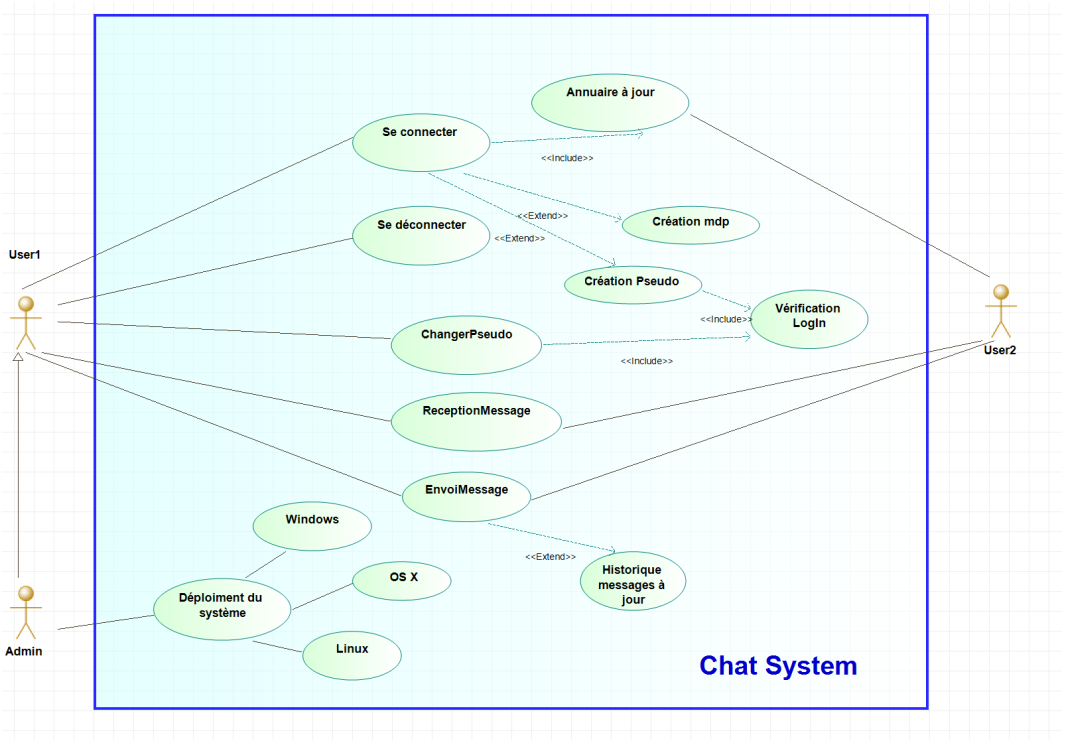

FIGURE 1 – Diagramme des cas d'utilisation réalisé sur Modelio.

2.2 Diagramme de classes

Le diagramme des classes permet de décrire de manière statique les différentes classes composant le chat system et les relations qui existent entre elles. Le système s'organise autour de **trois classes principales** : Controller, Model et View.

La **classe Controller** concerne la partie logique de notre projet : les algorithmes et les procédures. La **classe Modèle** concerne toute la gestion des données, i.e toutes les données stockées au niveau de l'utilisation qui sont volatiles. Ainsi, on sépare les données et leur traitement. Finalement, nous avons besoin d'une **classe View**. Une interface graphique sera dédiée à l'étape d'authentification, tandis que la seconde interface sera l'interface principale avec visualisation chat et de l'annuaire des personnes connectées.

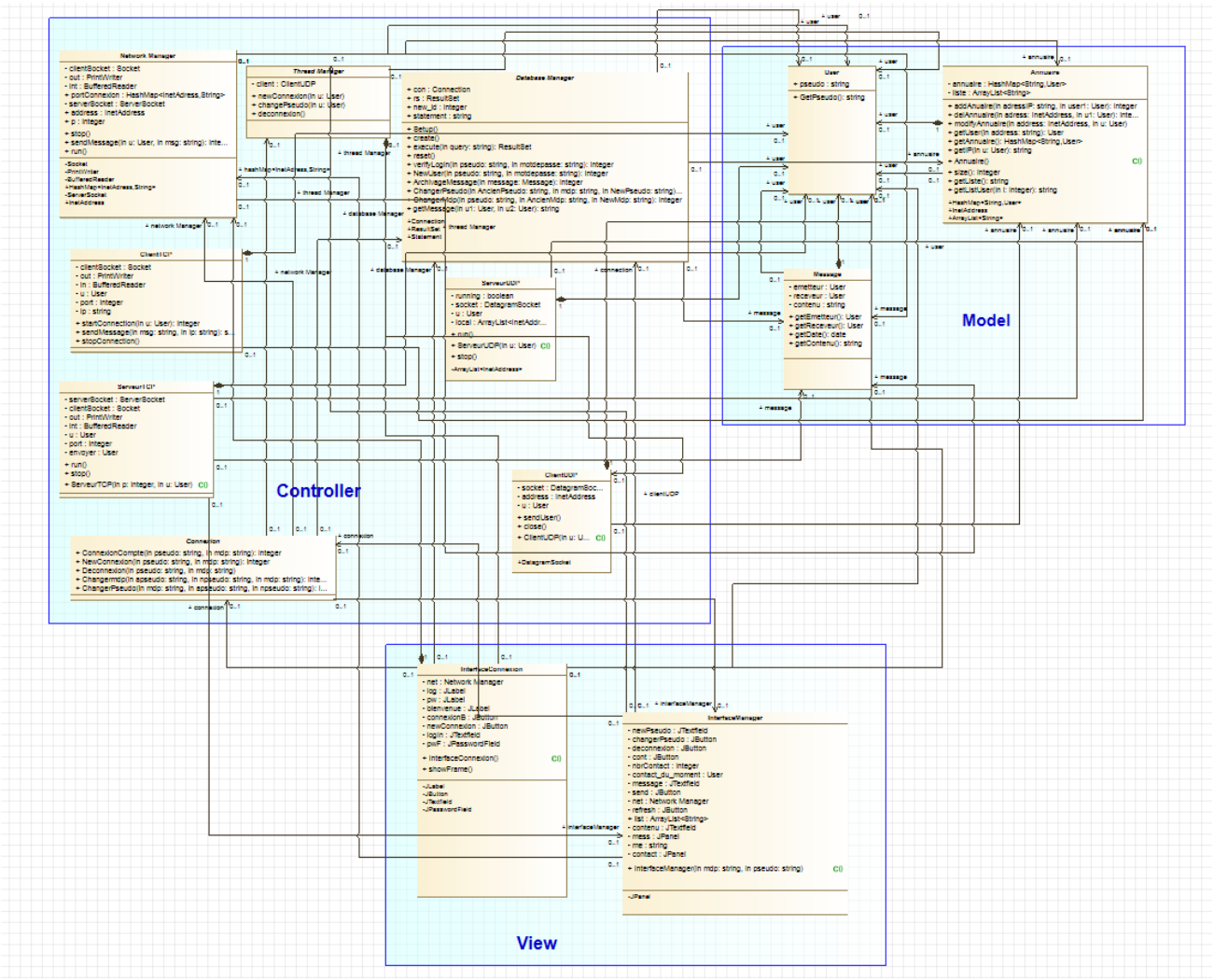

FIGURE 2 – Diagramme des classes réalisé sur Modelio.

La classe **Controller** : Elle est notamment composée d'un NetworkManager qui joue le rôle de pilote entre l'application et le réseau. Il va aussi notifier les autres managers. Le serveur du NetworkManager est un protocole TCP, il est en attente d'un message. Il crée un nouveau thread pour chaque utilisateur qui commence une nouvelle conversation. Le ThreadManager va donc initier l'envoi des messages depuis la classe interfaceManager.

Le client TCP est pour l'envoi des messages entre utilisateurs et le serveur TCP pour leur réception. Le client ainsi que le serveur UDP nous permettent l'envoi de l'adresse IP d'un nouvel utilisateur à tous les autres utilisateurs connectés et la récupération de leurs adresses IP par la création d'un serveur de stockage. Chaque message reçu est stocké par les serveurs.

Le Database Manager est d'une importance primodiale : elle gère la création, mise en route de la Database mais assure aussi la création des nouveaux utilisateurs, vérifie la correspondance d'authentification et effectue aussi les opération de changement de pseudo et de mot de passe lorsqu'ils sont initités par l'utilisateur via la classe Connexion.

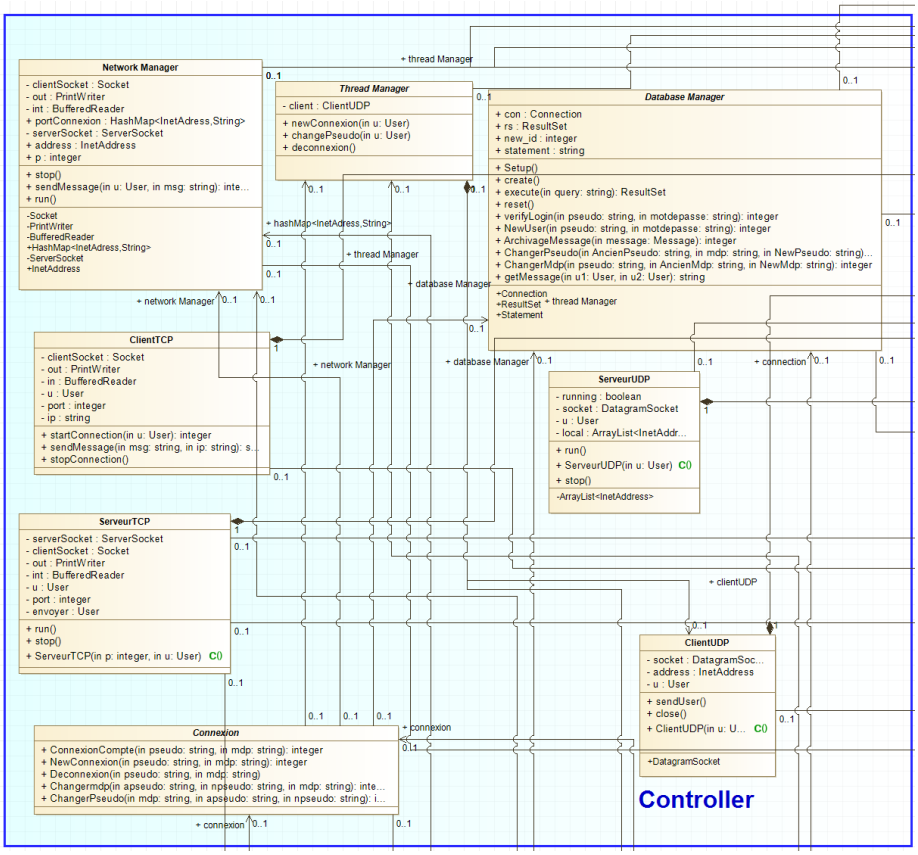

FIGURE 3 – Classe Controller réalisée sur Modelio.

La classe **Model** : Elle permet de caractériser notre utilisateur par son pseudo mais aussi de tenir un annuaire à jour des utilisateurs connectés, qui peut être modifié, par ajout ou suppresion d'un utilisateur grâce à son adresse IP. L'annuaire est de type Hashmap<String,User> de façon à faire correspondre un string, donc un pseudo et un utilisateur. Un liste des pseudos est aussi conservée. Enfin, un message est caractérisé par un utilisateur émetteur, un utilisateur receveur et un contenu.

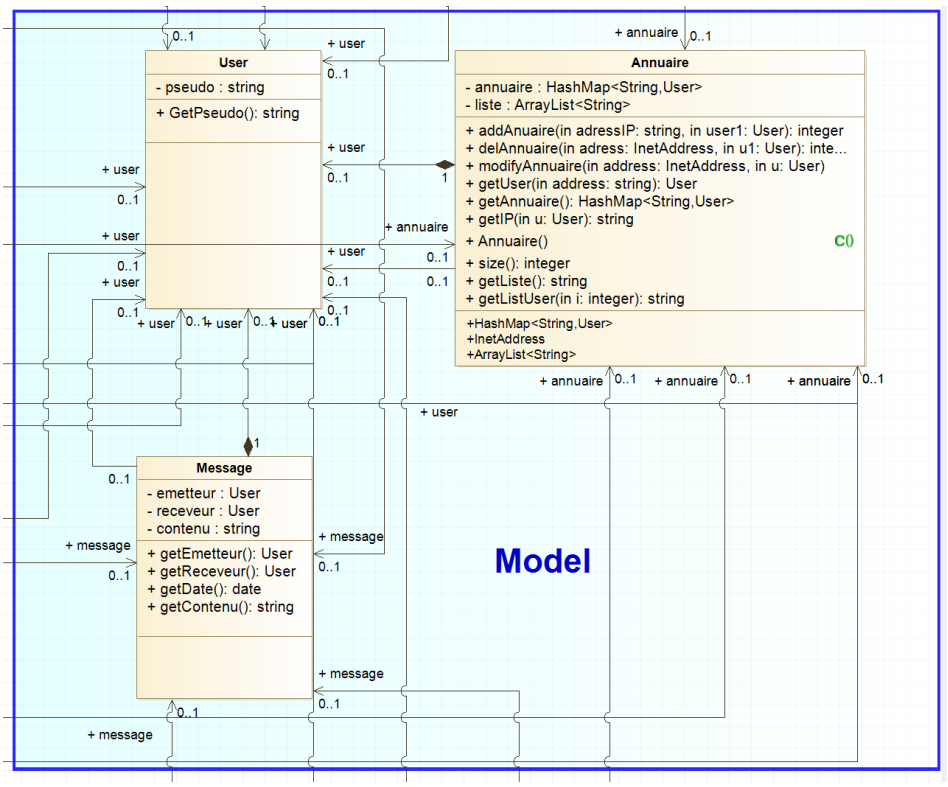

FIGURE 4 – Classe Model réalisée sur Modelio.

La classe **View**

La classe View concerne toute l'interface graphique de notre système de clavardage. Elle est composée de deux classes qui correspondent à nos deux interfaces : une classe InterfaceConnexion et une classe InterfaceManager.

La classe **InterfaceConnexion** implémente l'interface de connexion sur laquelle l'utilisateur se connecte avec son pseudo et son mot de passe. Elle permet à l'utilisateur de se connecter en cliquant sur le bouton "Se Connecter" ou de s'inscrire si l'utilisateur n'a pas déjà crée son compte en cliquant sur le bouton "S'inscrire".

La seconde classe **InterfaceManager** implémente l'interface principale, une fois que l'utilisateur est connecté. A l'intérieur du chat system, l'utilisateur à la possilité de changer son pseudo dans le champ correspondant. Sa fonctionnalité principale est l'envoi et la réception de messages. Pour cela, l'utilisateur peut cliquer sur son destinataire parmi la liste des utilisateurs connectés présentée sous la forme d'une scroll bar. Ensuite, il a simplement à écrire son message dans le champ disponible et de cliquer sur send.

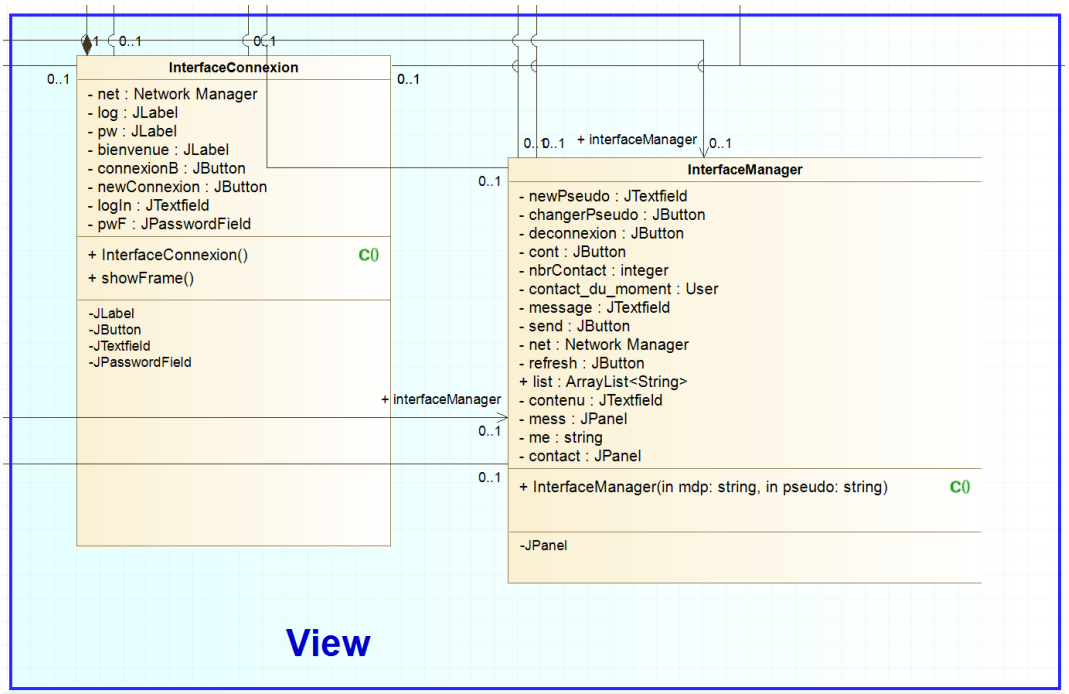

FIGURE 5 – Classe View réalisée sur Modelio.

2.3 Diagramme de séquences

Le diagramme de séquence permet de décrire les messages échangés entre les composants du système de clavardage sur une échelle de temps.

Tout d'abord, l'utilisateur se connecte sur l'interface de connexion en entrant son login et mot de passe. Donc l'InterfaceConnexion est mobilisée en premier via Connexion. L'utilisateur se connecte et une demande est envoyée à la database et précisément au DatabaseManager pour savoir si la connexion est autorisée. Un thread est envoyé pour le ServeurUDP : ainsi le ClientUDP va pouvoir envoyer un message en broadcast à tous les utilisateurs. Ensuite, le serveur du NetworkManager est lancé ainsi que l'InterfaceManager.

La phase de connexion fonctionne de manière similaire, cependant des fonctions d'arrêts sont appelées dans Client UDP et TCP notamment. Un message UDP est envoyé à tous les utilisateurs pour les informer de la déconnexion et enfin le serveur UDP est stopé par l'arrêt du thread.

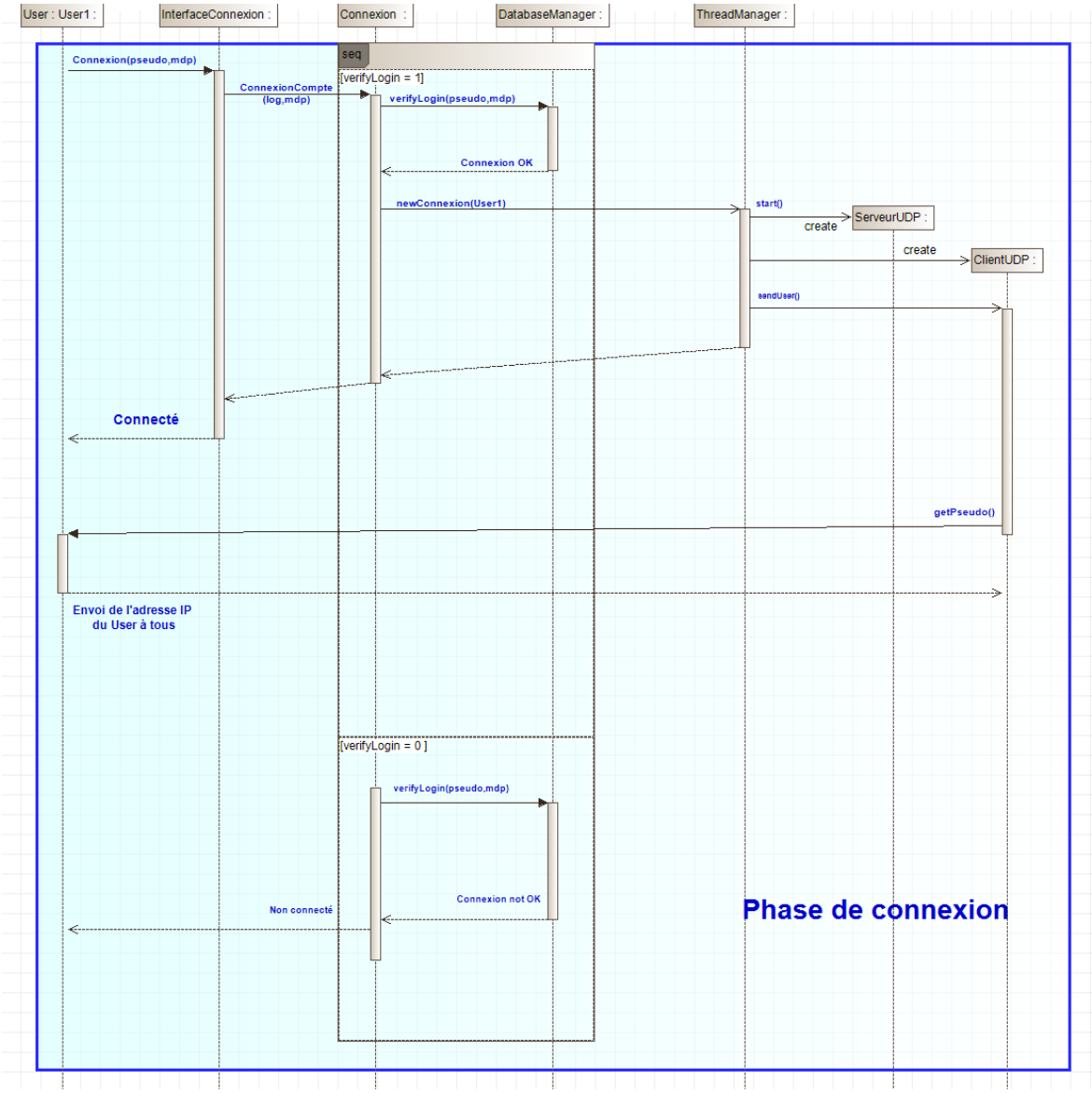

FIGURE 6 – Diagramme de séquence, phase de connexion réalisé sur Modelio.

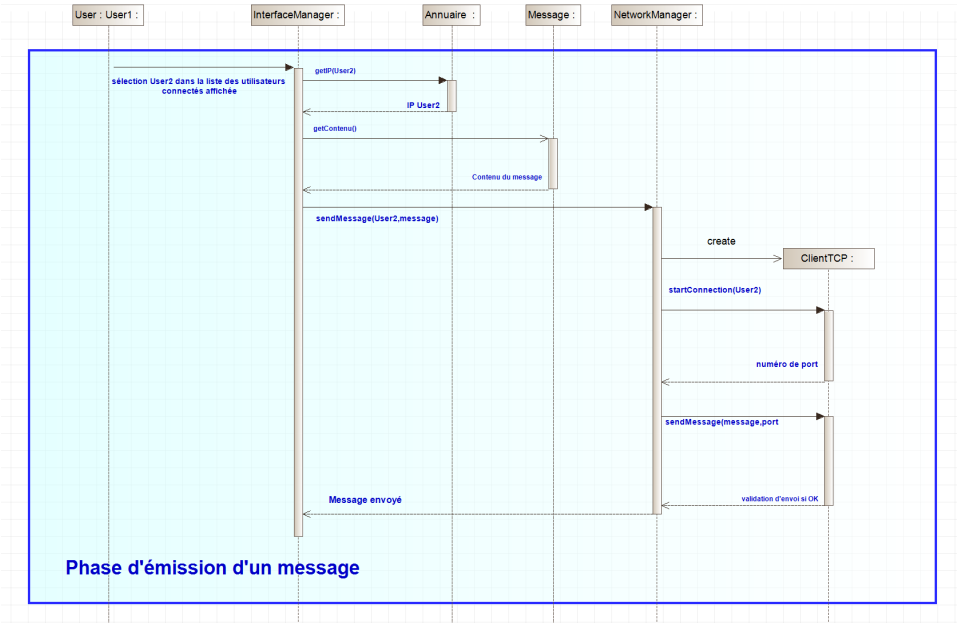

FIGURE 7 – Diagramme de séquence, phase d'émission réalisé sur Modelio.

De même, la phase de réception est très similaire à la phase d'émission donc ne sera pas détaillée ici.

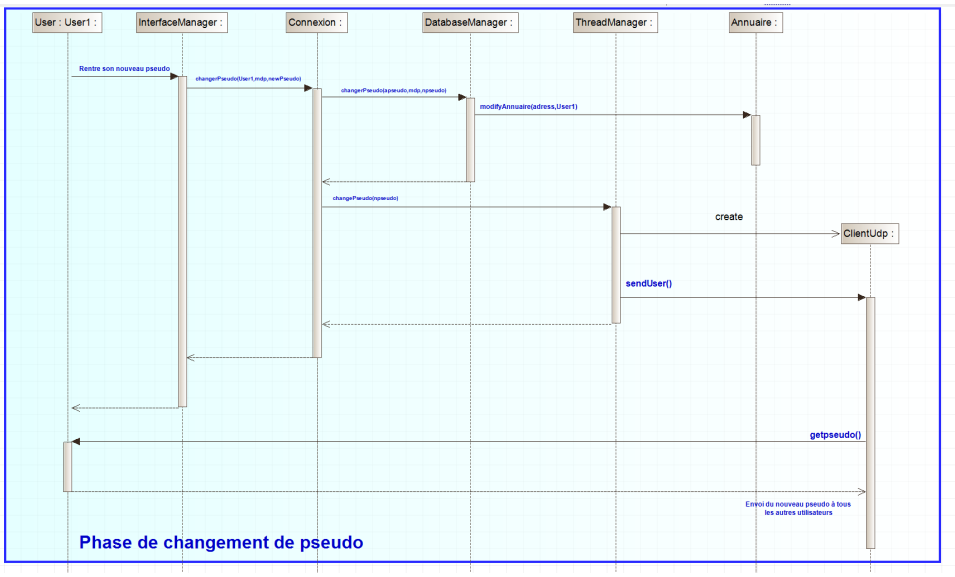

FIGURE 8 – Diagramme de séquence, phase de changement de pseudo d'un utilisateur réalisé sur Modelio.

Nous avons jugé que l'association de diagrammes de séquence, de classe et des cas d'utilisation était suffisante à l'élaboration de la conception du projet.

3 Architecture du sytème et choix technologiques

3.1 Architecture de la base de données

Nous avons décidé d'organiser notre base de données de la manière ci-dessous. Nous avons besoin de stocker les différents User, reconaissables avec leur ID, et de stocker leur pseudo. De plus, nous devons garder en mémoire les messages, différenciables via les deux Users associés et la date du message. Nous devons aussi stocker le contenu pour pouvoir afficher les conversations.
Ensuite, nous voyons que deux Users sont forcément liés à un message, et que nous pouvons avoir entre 0 et n message associé à un User.

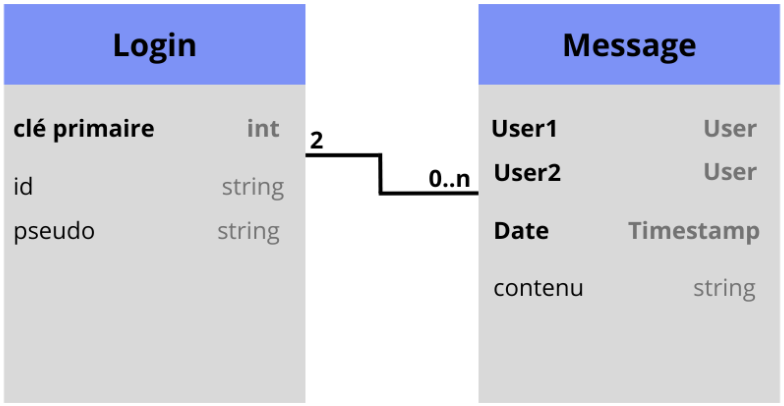

FIGURE 9 – Schéma synthétique de la base de données.

Nous avons adopté une database centralisée car cela nous a été conseillé. En effet, nous n'avions pas suivi les cours de Database donnés plus tôt dans le cursus, donc nous avons opté pour le serveur de l'INSA.

3.2 Architecture des interfaces GUI

Nous avons imaginé notre interface en deux pages. Une première de connexion, et une seconde avec la possibilité de se déconnecter, de changer de pseudo, de choisir un contact et bien sûr d'envoyer un message en voyant l'historique de conversation.

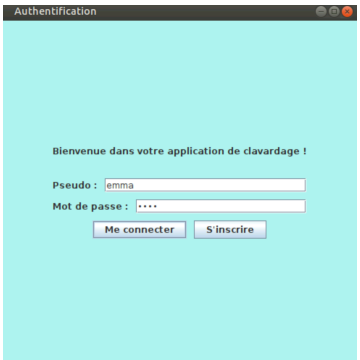

FIGURE 10 – Interface de connexion visualisée via Eclipse

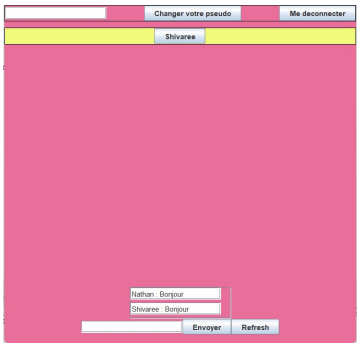

FIGURE 11 – Interface de clavardage avec annuaire de contacts et conversations visualisée sur Eclipse

4 Procédures

4.1 Procédures d'évaluation et de tests

Le but de ces procédures est de tester les fonctionnalités clés de l'interface comme la transmission de messages, la gestion des utilisateurs, la notification de nouveaux messages, etc. Nous avons commencé par créer des comptes d'utilisateurs de test et des scénarios de conversation pour simuler des interactions entre utilisateurs. La deuxième étape était de configurez l'environnement de test . On s'assure que le clavardage est installé et configuré correctement sur la machine de test. En utilisant

des scénarios de conversation différents, on simule des interactions entre utilisateurs. Pratiquement, nous avons créer une classe de test qui est simplement un main qui teste les fonctions crées. La classe affiche à fur et à mesure des messages pour suivre le fonctionnement et l'évolution de notre programme. Ensuite, on enregistre les résultats de chaque test et on les analyse pour identifier les erreurs ou les bugs. Les tests sont répétés régulièrement pour s'assurer que le clavardage fonctionne correctement et de manière stable.

4.2 Procédures d'installation et de déploiement

Le déploiement est le processus de mise en production de l'application sur un serveur. Il permet aux utilisateurs finaux d'accéder au clavardage et d'interagir entre eux. Les utilisateurs finaux pourront utiliser notre interface avec les systèmes d'exploitations où l'installation de l'environnement Java Eclipse est possible (ou tout autre environnement Java).
L'administrateur, lors du déploiement, doit rajouter les drivers utiles (mysql et sqlite) à l'exécution du chat system. Nous fournissons ces drivers sur le dépôt Git sous forme de fichiers compressés que l'administrateur peut récupérer directement.

5 Manuel d'utilisation simplifié

FIGURE 12 – Connexion ou inscription

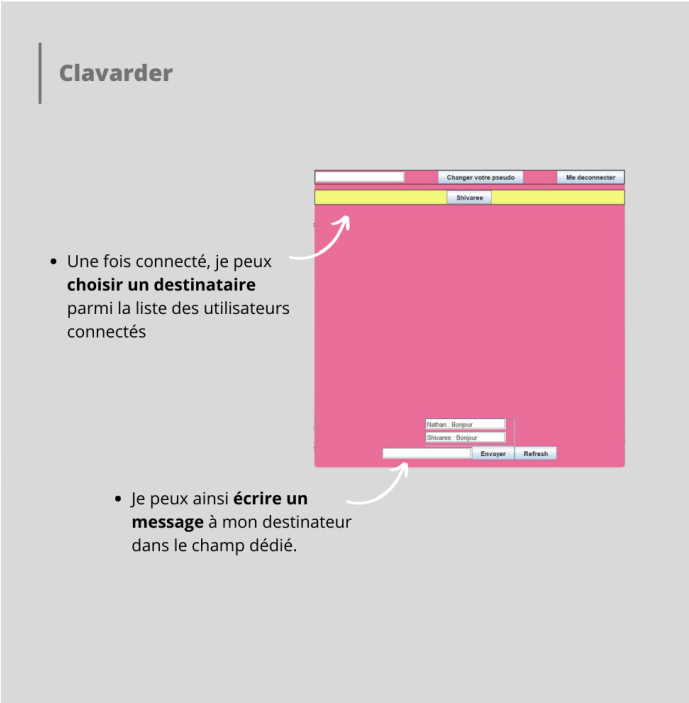

FIGURE 13 – Clavarder

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