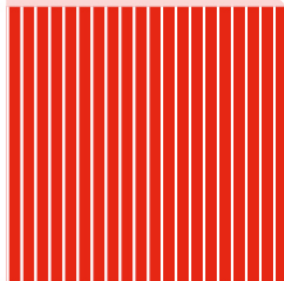
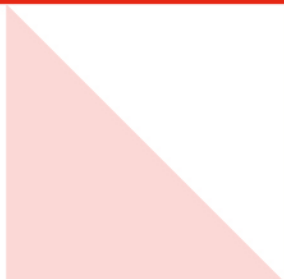
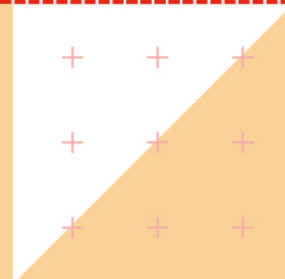
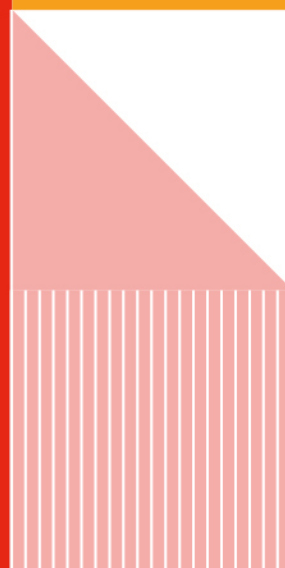


Rapport de projet

« Chador » : Notre application de chat décentralisé

Evan Rozière
Gwenaël Robert
4IR – Groupe A1



Sommaire

I.	Diagrammes UML et Conception	5
1.	Diagrammes de cas d'utilisation	5
2.	Diagramme de classes	6
3.	Diagrammes de séquence	7
i.	Envoi d'information à d'autres utilisateurs (Pseudo, IP, ...)	7
ii.	Réception d'informations d'un autre utilisateur (Connexion, Déconnexion, Mise à jour de pseudo).....	8
iii.	Envoi d'un message	9
iv.	Réception d'un message	10
v.	Gestion de la fenêtre (Réduire fenêtre, Afficher une conversation).....	11
4.	Diagramme de structure composite.....	12
5.	Diagramme de déploiement	13
6.	Maquette de l'interface graphique	14
i.	Fenêtre de login :	14
ii.	Fenêtre de chat	14
II.	Choix des technologies utilisées.....	15
1.	Base de données.....	15
2.	GUI	15
3.	Outil d'automatisation de gestion de projet	15
4.	Format des messages.....	15
5.	Résumé de l'architecture de l'application	15
III.	Tests de l'application.....	16
IV.	Installation et déploiement de Chador	16
1.	Installation des fichiers.....	16
2.	Démarrage de l'application.....	16
V.	Manuel d'utilisation.....	17
1.	Installer l'application	17
2.	Se connecter	17
3.	Utiliser le chat	18

Introduction

Dans ce rapport nous allons voir les différentes étapes de la conception de notre projet dans le cadre de l'UF « Conception et Programmation avancées ». Nous avons eu comme consigne de créer une application de chat décentralisé avec un cahier des charges bien défini au préalable.

Dans un premier temps, nous détaillerons les diagrammes de conception qui nous ont servi de base pour le développement de notre application de chat : « Chador »

Dans un second temps, nous listerons les différentes architectures et technologies que nous avons retenues pour notre application.

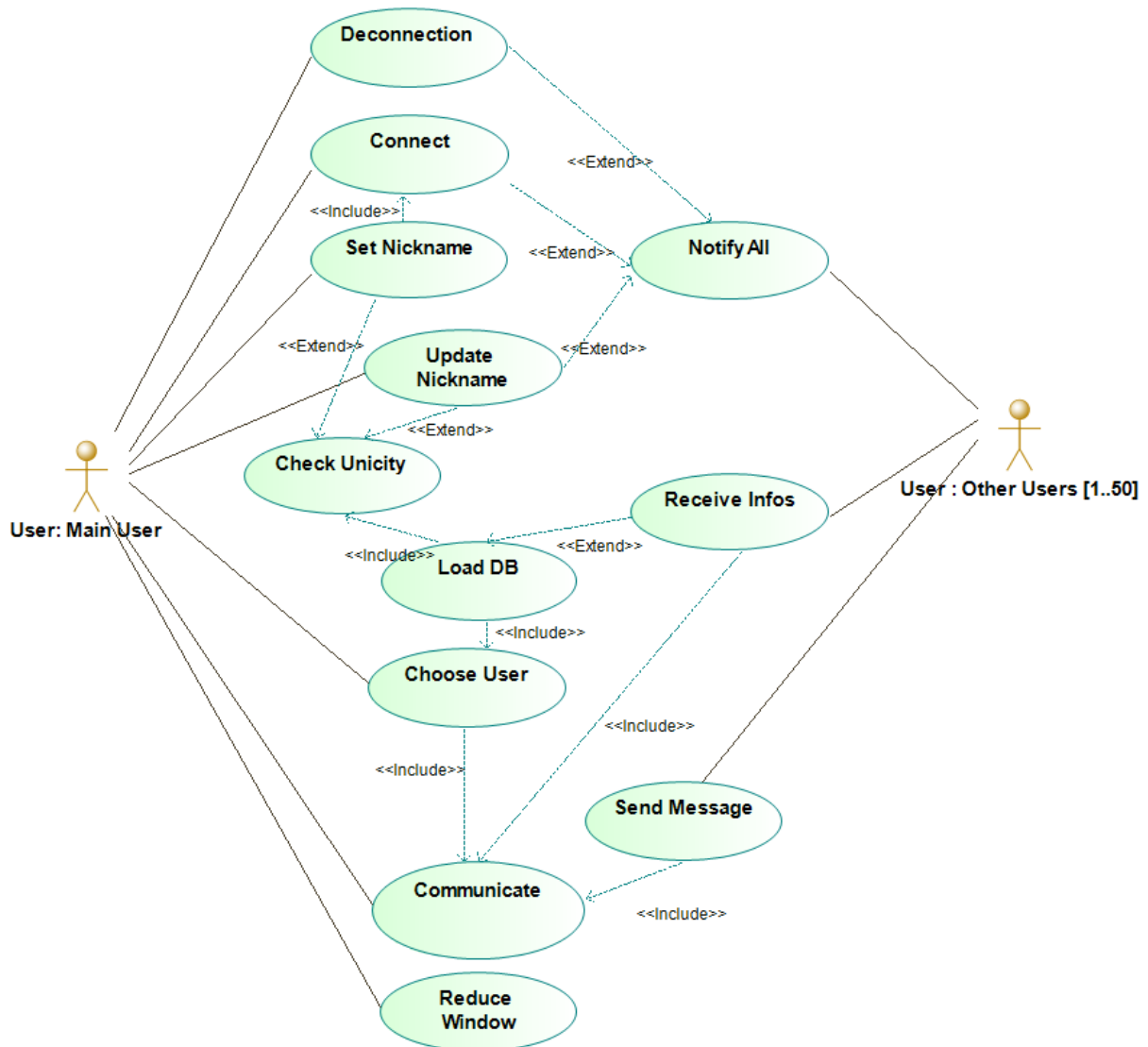
Ensuite, nous verrons nos procédures de test permettant d'assurer que nos fonctions développées fonctionnent bien.

Puis, nous expliquerons comment procéder à l'installation et au lancement de l'application sur une machine.

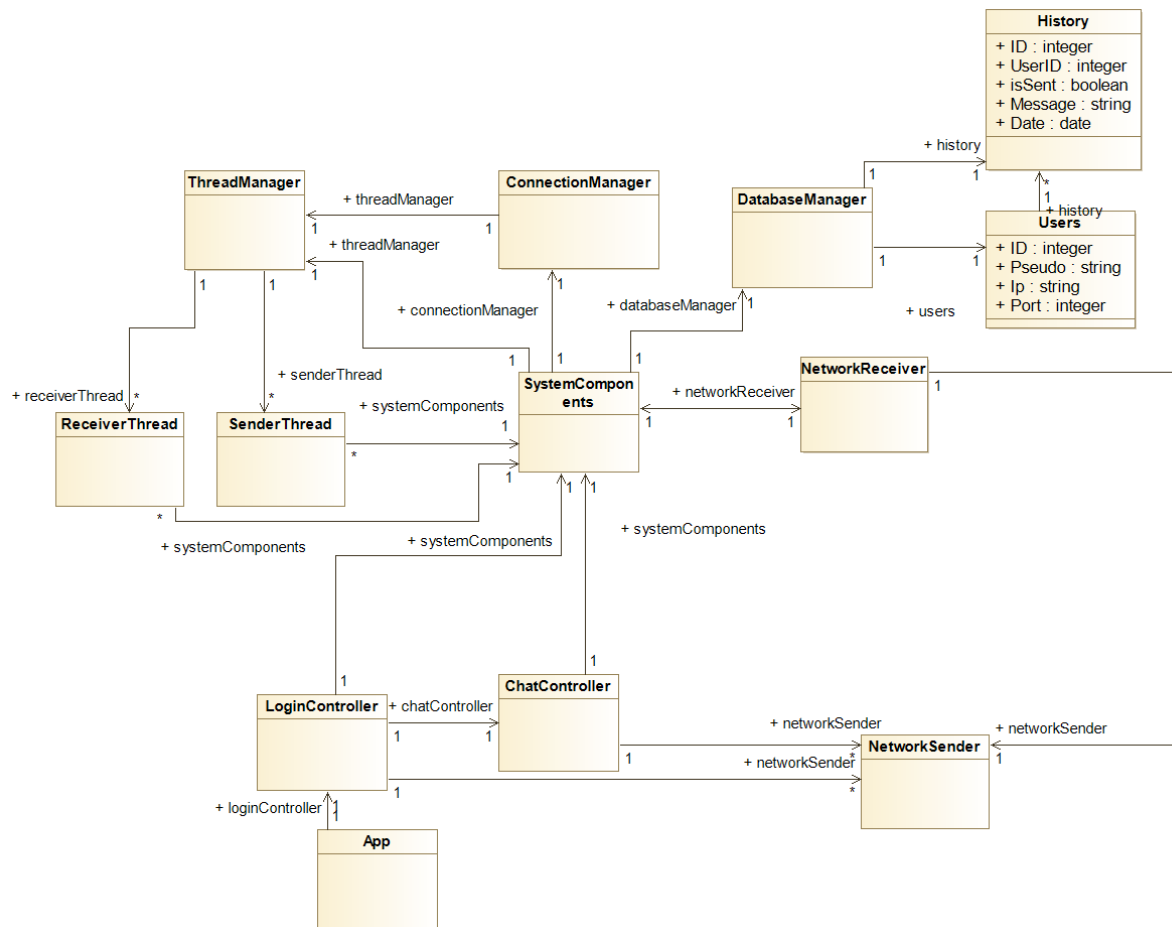
Pour finir, nous mettrons à disposition un petit manuel d'utilisation pour expliquer le bon fonctionnement de notre application à un utilisateur novice.

I. Diagrammes UML et Conception

1. Diagrammes de cas d'utilisation

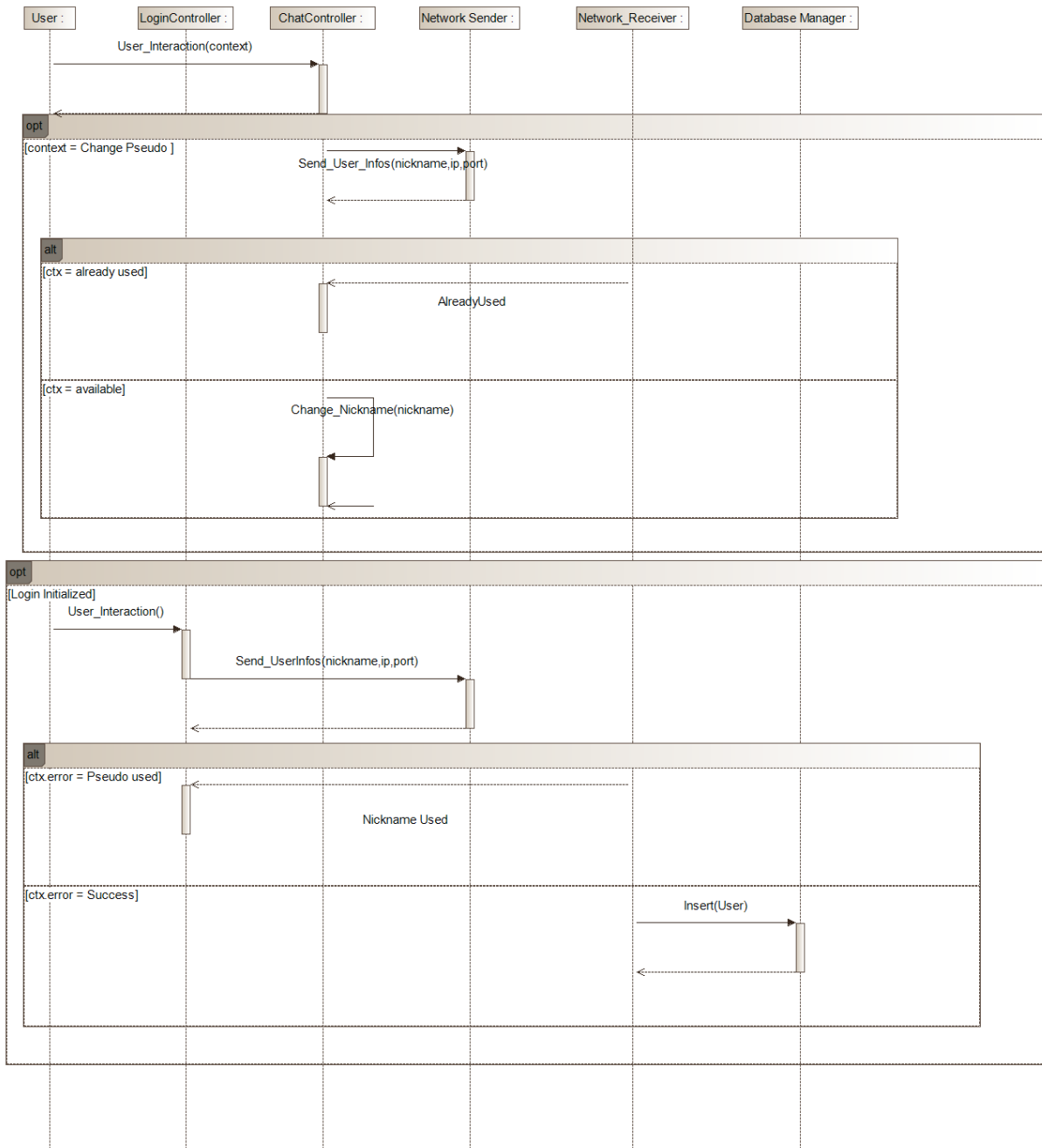


2. Diagramme de classes

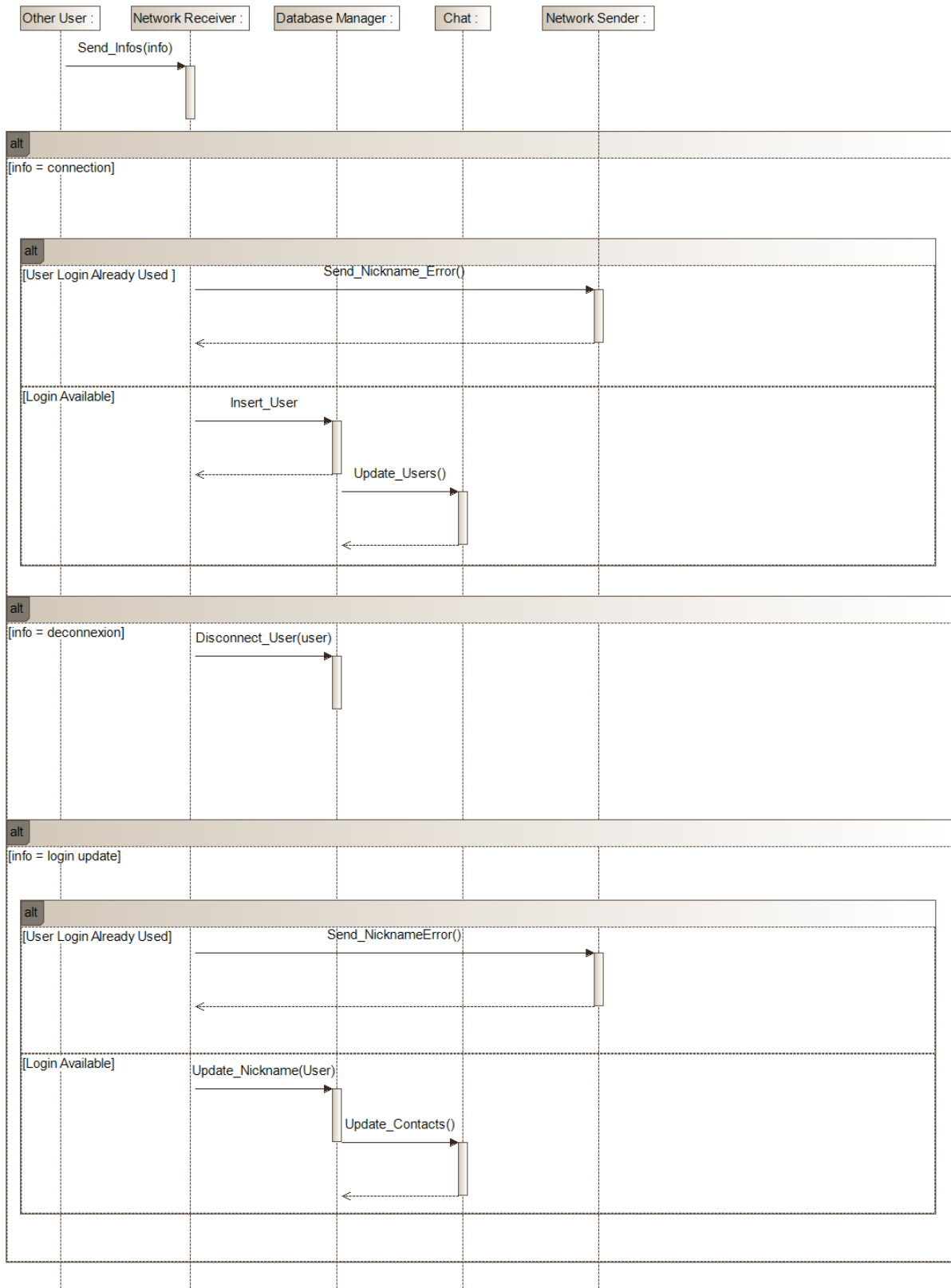


3. Diagrammes de séquence

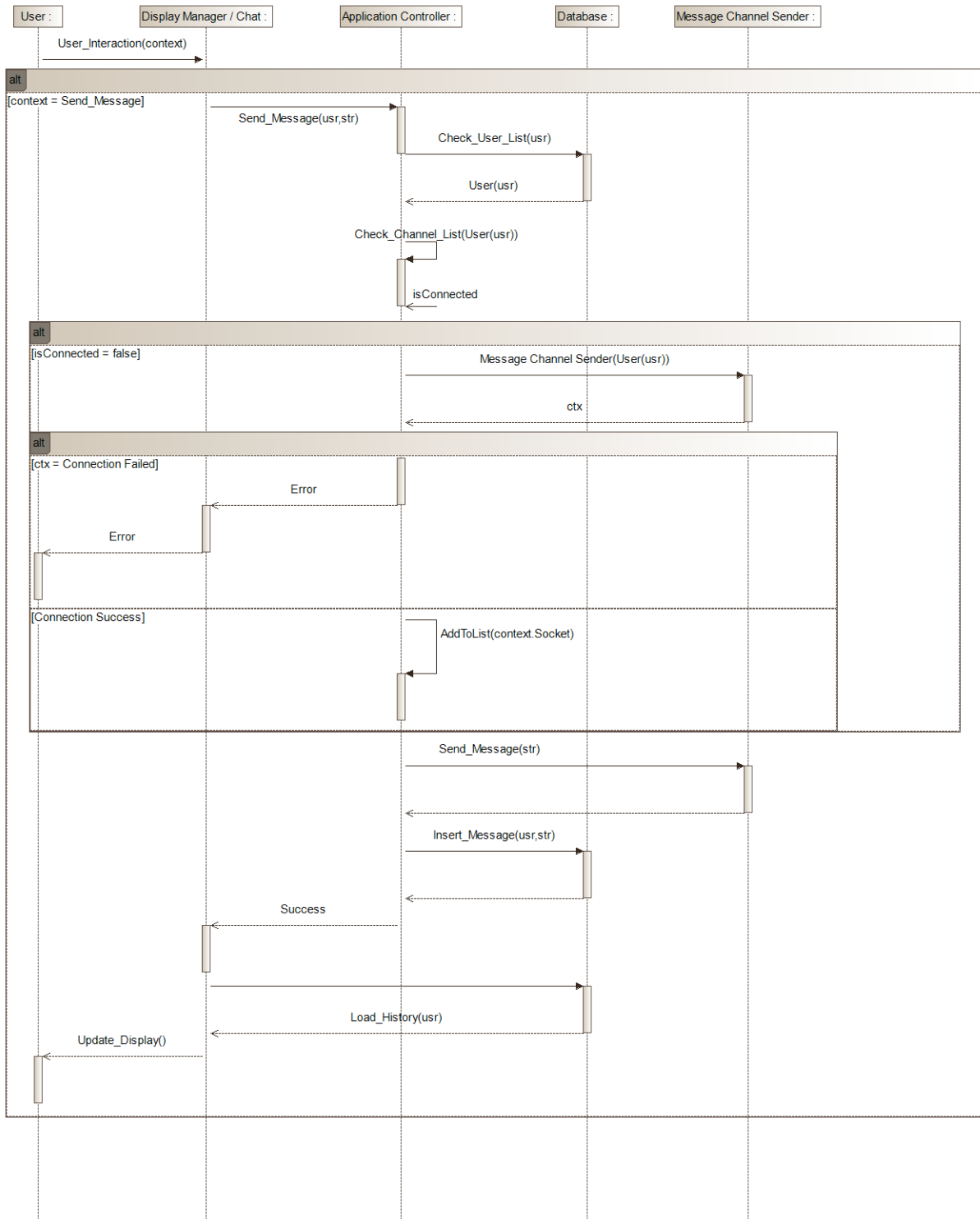
i. Envoi d'information à d'autres utilisateurs (Pseudo, IP, ...)



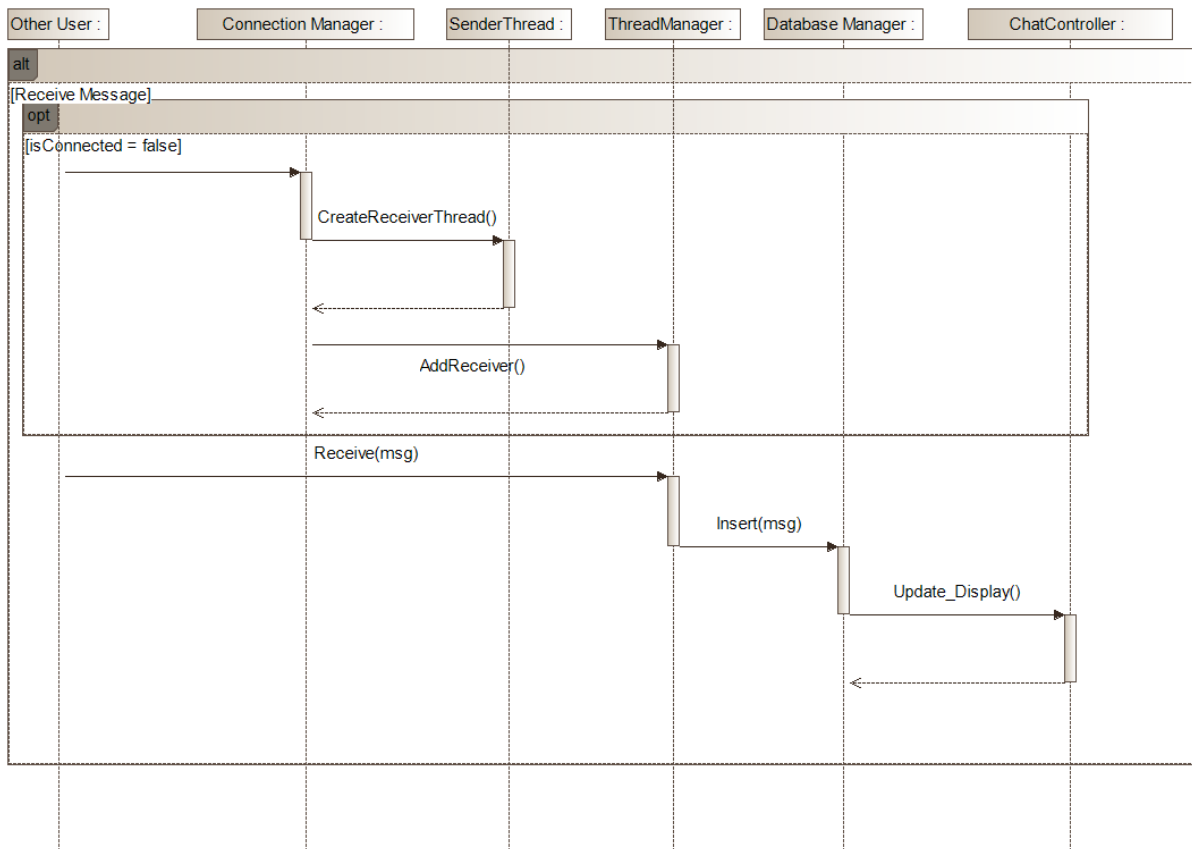
ii. Réception d'informations d'un autre utilisateur (Connexion, Déconnexion, Mise à jour de pseudo)



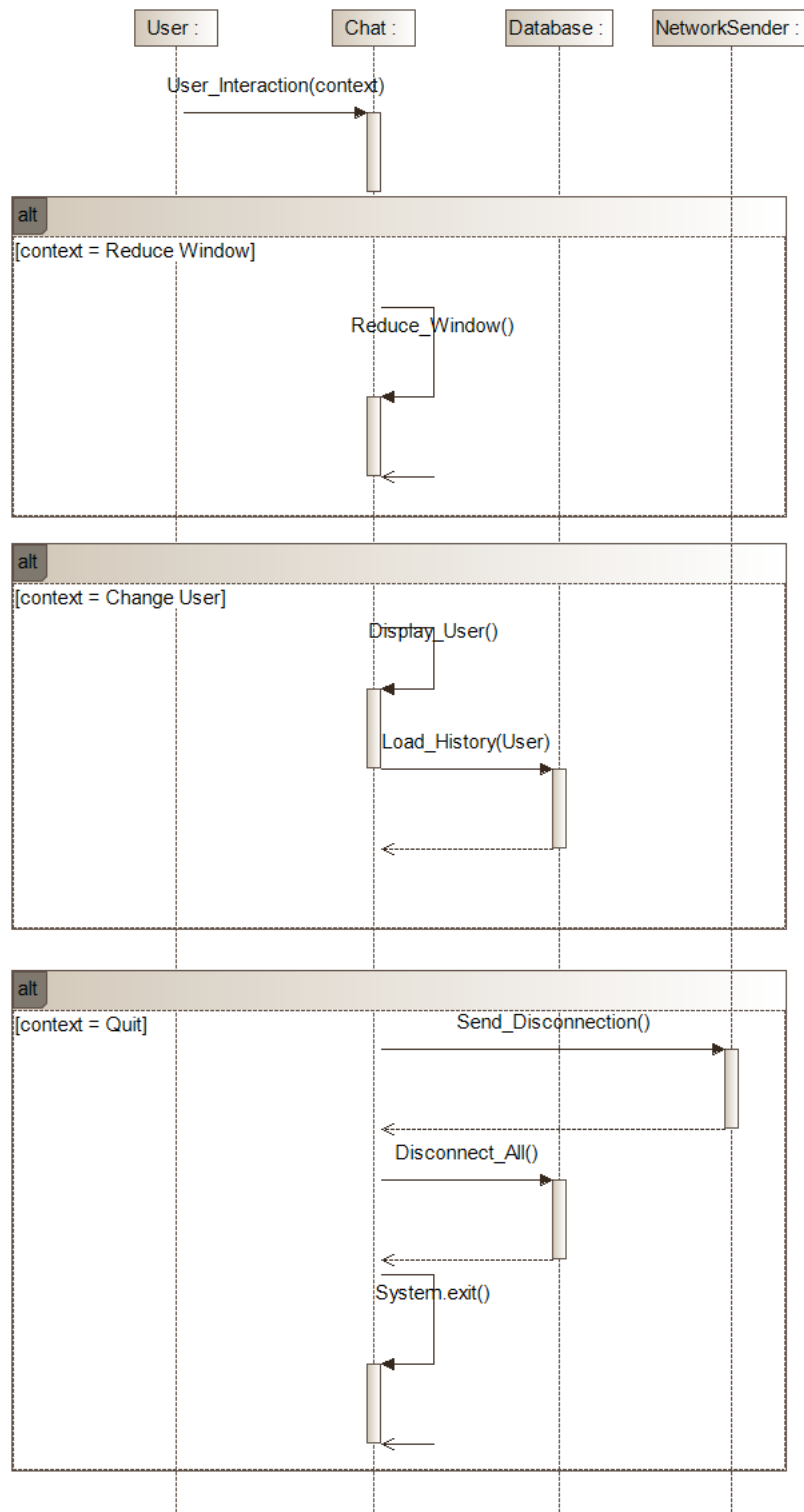
iii. Envoi d'un message



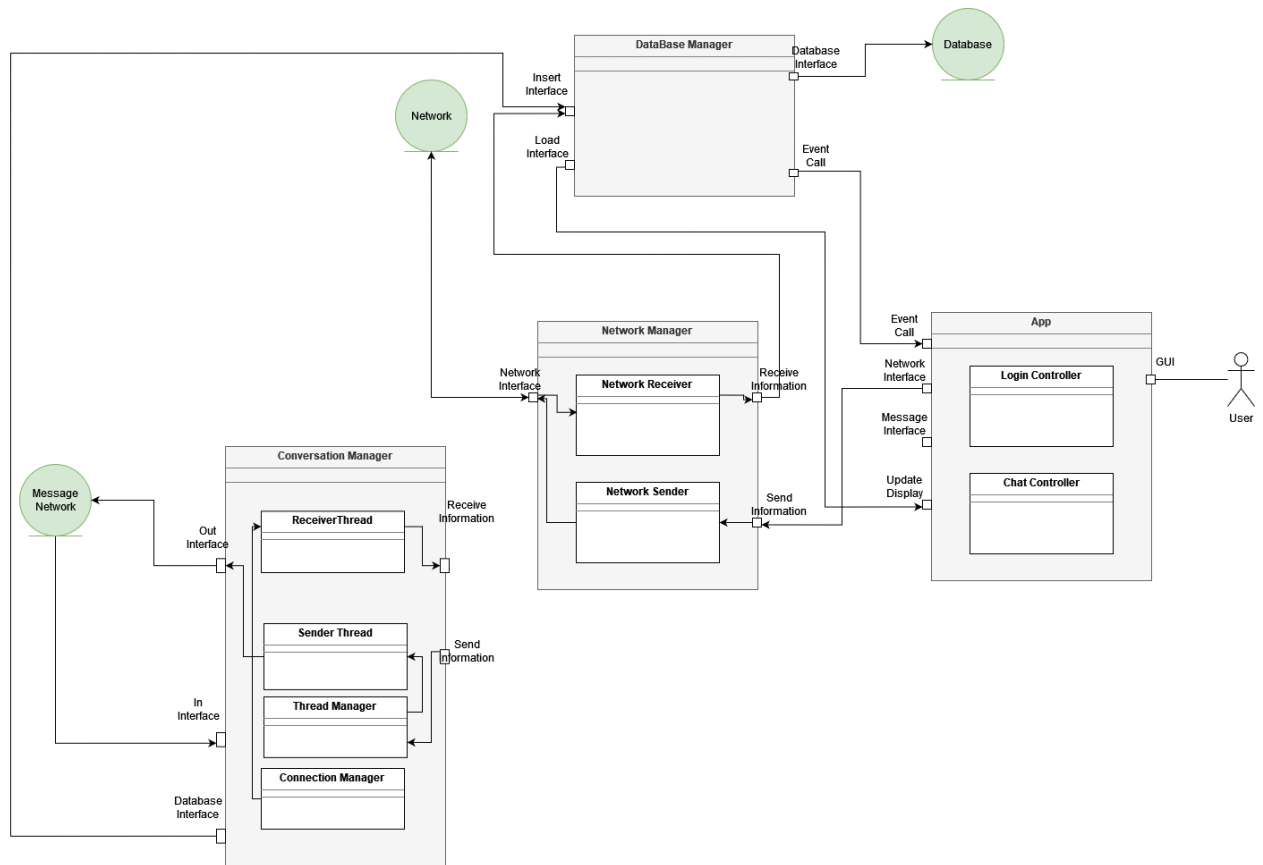
iv. Réception d'un message



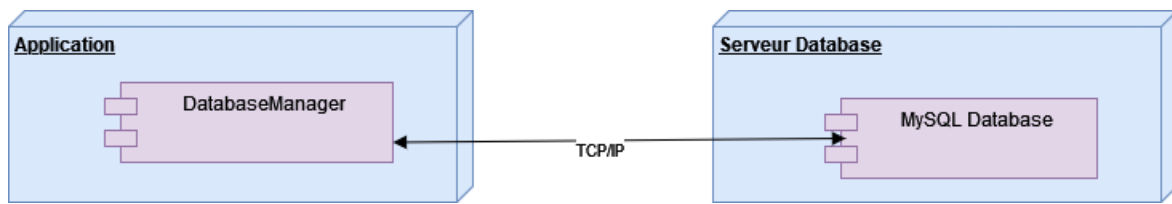
v. *Gestion de la fenêtre (Réduire fenêtre, Afficher une conversation)*



4. Diagramme de structure composite



5. Diagramme de déploiement



6. Maquette de l'interface graphique

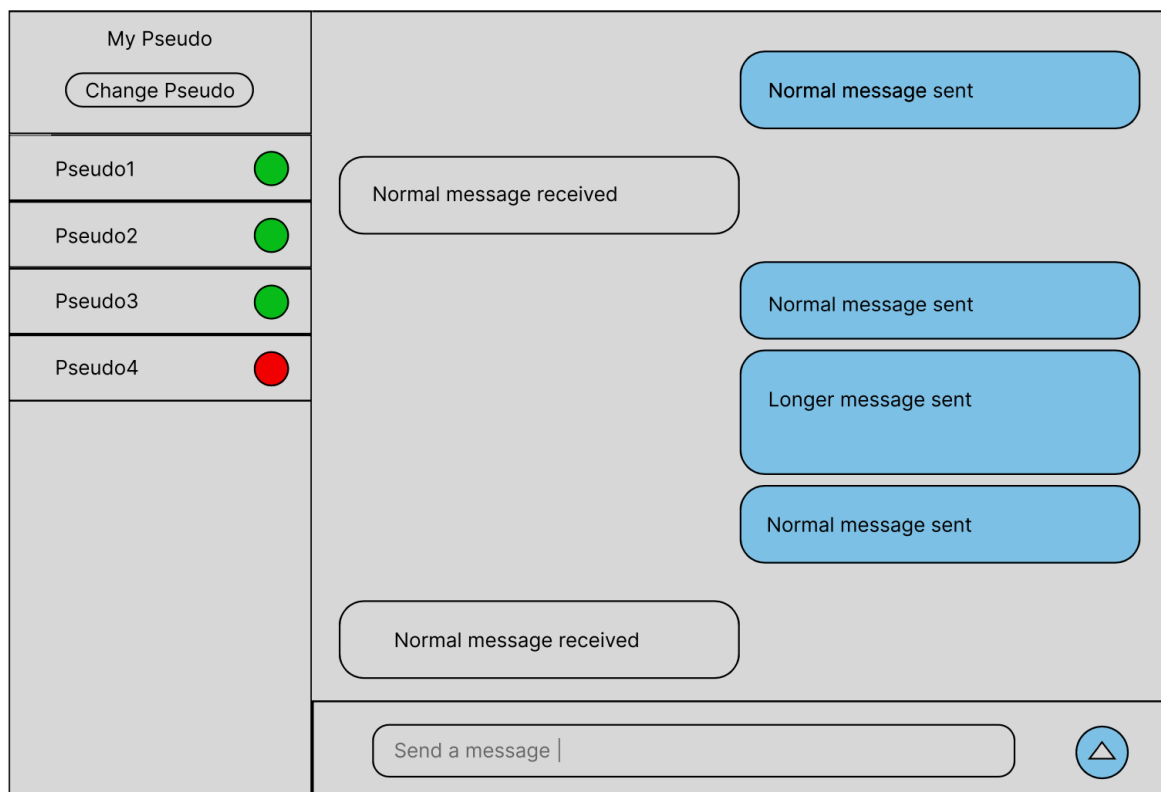
Les maquettes des interfaces graphiques ont été faites sur le logiciel Figma pour servir de modèle mais ne sont pas représentatives de la version finale.

i. Fenêtre de login :



A login window mockup with a light gray background. At the top, the text "Welcome !" is displayed in a large, bold, black font. Below it, the text "Choose your username below :" is in a smaller, regular black font. Underneath is a rectangular text input field. At the bottom, there is a rounded rectangular button with the text "→ Ready to chat".

ii. Fenêtre de chat



A chat window mockup with a light gray background. On the left side, there is a sidebar with a header "My Pseudo" and a "Change Pseudo" button. Below the header is a list of four pseudo names: "Pseudo1", "Pseudo2", "Pseudo3", and "Pseudo4". Each name is followed by a circular status indicator: green for Pseudo1, Pseudo2, and Pseudo3, and red for Pseudo4. The main chat area on the right contains several message bubbles. On the left side of the main area, there are two "Normal message received" bubbles. On the right side, there are three "Normal message sent" bubbles and one "Longer message sent" bubble. At the bottom of the chat area, there is a text input field with the placeholder text "Send a message |" and a blue circular button with a white upward-pointing triangle icon.

II. Choix des technologies utilisées

1. Base de données

Pour stocker nos données, nous avons choisi d'utiliser une **base de données SQL Lite**. C'est une base de données **légère en mémoire** qui est **très facile d'utilisation** et **facile à déployer**. Sur notre application, chaque utilisateur possède sa propre base de données en local qui lui permettra de recharger les conversations, trouver les utilisateurs connectés ainsi que toutes les informations associées (adresse IP, pseudo, ...).

2. GUI

Pour ce qui est de l'interface graphique, nous avons décidé d'utiliser la librairie **JavaFX**. Cette librairie est très utilisée dans le monde du développement d'application java pour la **richesse des fonctionnalités** qu'elle propose. De plus, JavaFX nous semblait plus adaptée car elle permet de faire **des « belles » interfaces graphiques simplement** grâce à l'outil Scene Builder ce qui est à nos yeux important pour une application de messagerie.

3. Outil d'automatisation de gestion de projet

Nous avons retenu **Maven** pour comme outil d'automatisation de gestion de projet. Notre choix c'est naturellement porté vers ce dernier car nous avons eu l'occasion de travailler avec auparavant pendant les TP/TD. L'utilisation du fichier POM est très pratique pour **gérer les différentes librairies sans contraintes** et pour le **déploiement de l'application** (cf. Partie IV).

4. Format des messages

Pour simplifier le format des messages dans le réseau, nous avons choisi d'utiliser la librairie **Gson**. Cette librairie permet **de convertir des objets au format JSON**. Les fonctions UDP et TCP attendant un string, cela est beaucoup plus pratique pour envoyer des messages ou des informations.

5. Résumé de l'architecture de l'application

Notre application s'organise avec les 3 grandes parties comme vu dans l'UML : le réseau, la base de données, et l'interface graphique.

Le réseau reçoit les messages et informations des autres utilisateurs qui seront stockés dans la base de données. Les messages et utilisateurs disponible sur le réseau de la base de données seront affichés sur l'interface graphique pour l'utilisateurs. (Réseau → Base de données → Interface Graphique)

L'utilisateur qui souhaite envoyer un message se servira de l'interface graphique pour choisir un utilisateur et saisir un message. Le message sera ensuite envoyé par le réseau puis stocké dans la base de données. (Interface Graphique → Réseau → Base de données).

III. Tests de l'application

Nous avons fait des tests unitaires pour l'application en utilisant **Junit**. Ces tests servent à vérifier le bon fonctionnement de plusieurs points importants :

- Les fonctions de la base de données
- Les envois de message UDP broadcast

Ces tests s'effectuent sur l'ordinateur en « local », c'est-à-dire sans autres machines. Pour pouvoir lancer les tests depuis l'ordinateur d'un utilisateur, il est possible de lancer la commande « *mvn test* » depuis un terminal en étant sur le répertoire ChatbotFX.

IV. Installation et déploiement de Chador

Chador a l'avantage d'avoir été développé avec Maven ce qui permet de faire fonctionner cette dernière sans trop de difficulté.

1. Installation des fichiers

Il suffit d'écrire les commandes suivantes dans un terminal en se plaçant dans le dossier désiré :

- `clone https://github.com/Newglear/Projet_Java.git`

2. Démarrage de l'application

Une fois les fichiers installés sur l'ordinateur, il ne reste plus qu'à lancer dans le même terminal Chador. Pour cela, il faut écrire les commandes suivantes :

- `cd "Projet_Java/ChatBotFX"`
- `mvn compile`
- `mvn javafx:run (et c'est tout !)`

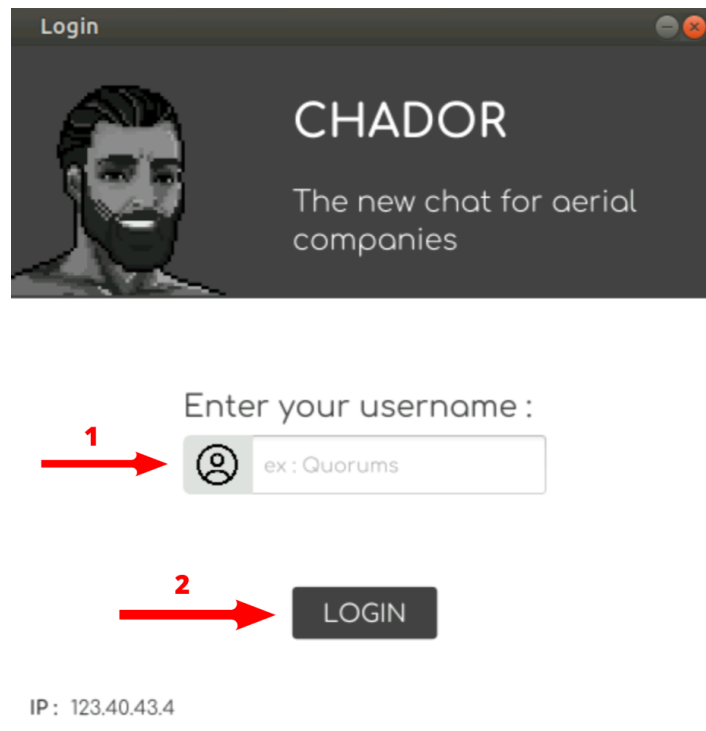
Attention ! Avant de d'écrire ces commandes, il faut penser à vérifier que Maven soit bien installé sur l'ordinateur. Par exemple sur linux, il suffit d'écrire la commande « *sudo apt install maven* ».

V. Manuel d'utilisation

1. Installer l'application

Cf. *Partie IV* qui explique en détail toutes les étapes à faire

2. Se connecter



Enter your username :

1

2

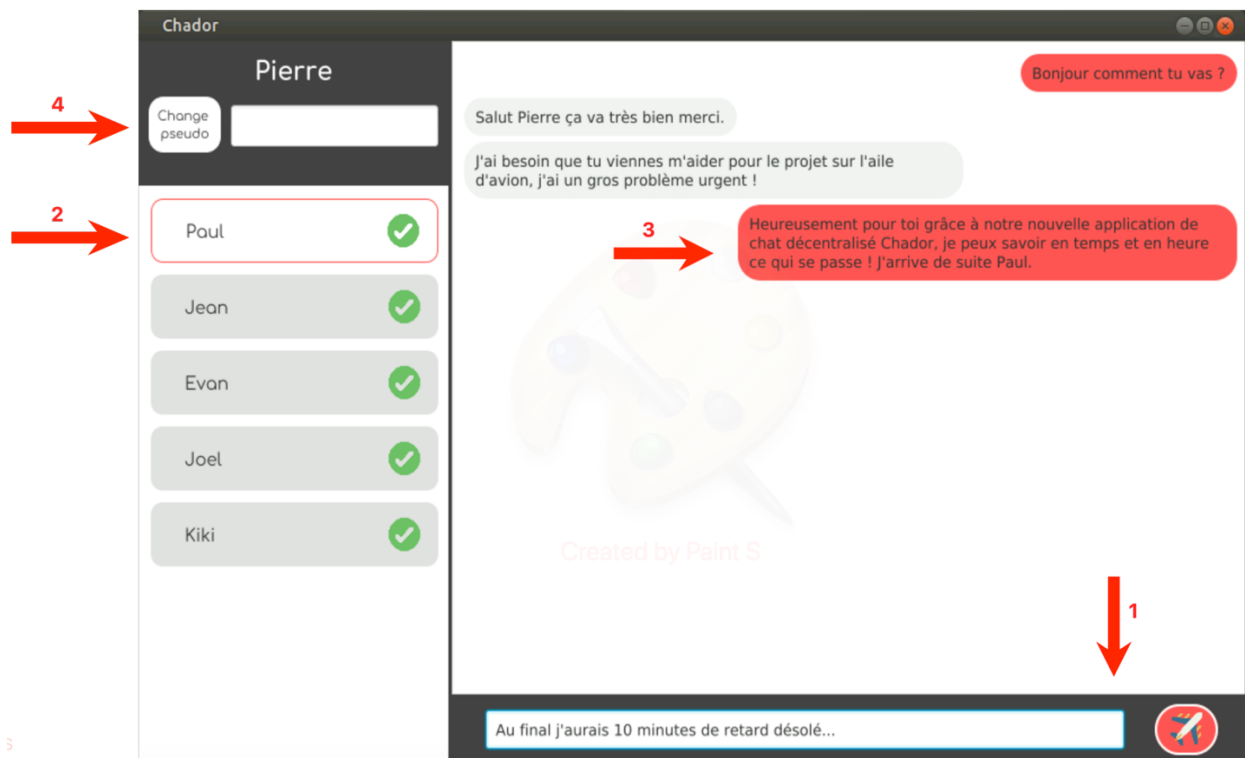
LOGIN

IP: 123.40.43.4

1. Zone de saisie du pseudo. Une fois le pseudo écrit, il est possible d'appuyer sur entrée pour continuer et soumettre son pseudo
2. Il est également possible de soumettre le pseudo choisi en appuyant sur le bouton login

Si le pseudo choisi est déjà utilisé, un message d'erreur s'affichera en dessous de la zone de saisie du pseudo et il sera possible d'en soumettre un nouveau à la place.

3. Utiliser le chat



1. Zone de saisie des messages avec un bouton pour envoyer les messages. Il est également possible d'appuyer sur entrée pour envoyer un message.
2. Zone avec les différents utilisateurs connectés. En cliquant sur l'utilisateur de son choix, cela va afficher la conversation correspondant à cet utilisateur. Il y a également une icône indiquant si l'utilisateur est connecté ou non.
3. Affichage des conversations avec en rouge les messages envoyés et en gris les messages reçus. Ces messages à la conversation de l'utilisateur choisi à l'étape 2. La zone est « scrolable » donc s'il y a trop de message, il suffit de scroller pour remonter.
4. Zone contenant le pseudo de l'utilisateur de l'application. En dessous se situe une zone de saisie permettant de changer de pseudo pendant l'utilisation de l'application. Après avoir choisi le pseudo, il suffit de cliquer sur le bouton changer de pseudo. Si le pseudo est déjà utilisé alors un message d'erreur s'affichera, sinon, le pseudo changera sur le texte au-dessus.