INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Rapport Projet Clavardage

POO COO (2020-2021)

FOUSSATS Morgane
BARNABE Elise
4IR A1

1. Introduction

Le projet que nous devions réaliser dans le cadre de l'UF POO/COO consistait à réaliser un système de clavardage, utilisable par les entreprises. Ce tchat devait permettre aux utilisateurs de pouvoir communiquer avec les utilisateurs étant actifs sur le système. Pour ceci, nous allons présenter dans un premier temps les différentes conditions de déploiement du projet. Puis, nous présenterons un manuel d'utilisation expliquant plus en détail le fonctionnement du tchat. Nous montrerons ensuite les éléments de conception du système sous forme de graphes UML. Enfin, nous exposerons les problèmes que nous avons rencontrés et les tests que nous avons effectués afin de valider les fonctionnalités du tchat.

2. Déploiement du projet

a. Le fichier de configuration

Pour faciliter la configuration du projet nous avons choisi de regrouper les paramètres à initialiser dans un même fichier json (nommé config.json) qui est ensuite utilisé par l'application pour récupérer les différents champs spécifiés. L'administrateur chargé de l'installation devra donc compléter tous les champs de ce fichier.
Détail des champs du fichier config.json :

- "serveur" : concerne les paramètres reliés utilisés par le serveur de présence
  - "actif" : à mettre à 1 si le serveur est actif sinon 0
  - "url" : spécifier l'url du serveur à utiliser, ici nous utiliserons l'url suivante où le code de notre serveur de présence est déployé : "https://srv-gei-tomcat.insa-toulouse.fr/servletBarnabeFoussats/toto"
  - "port" : spécifier le port à utiliser pour le serveur (par ex : 8080)
- "config"
  - "interface" : le nom de l'interface sur laquelle la machine est connectée au réseau local (cf section 2.b pour l'explication de comment trouver l'interface nécessaire).
  - "portSrc" : port sur lequel on va recevoir les messages
  - "portDest" : port sur lequel on va envoyer les messages

b. Retrouver le nom de l'interface réseau à utiliser

La marche à suivre détaillée ici est valable sous Windows car en raison du distanciel nous n'avons pu tester que sous ce système d'exploitation. Pour Linux ou d'autres systèmes d'exploitation il faudra ajuster le protocole (notamment au niveau des lignes de commandes).

Première étape : ouvrir un terminal (tapez cmd dans la barre de recherche). Ensuite il faut taper la commande : ipconfig. Ensuite vous aurez un résultat sensiblement similaire à la figure suivante :

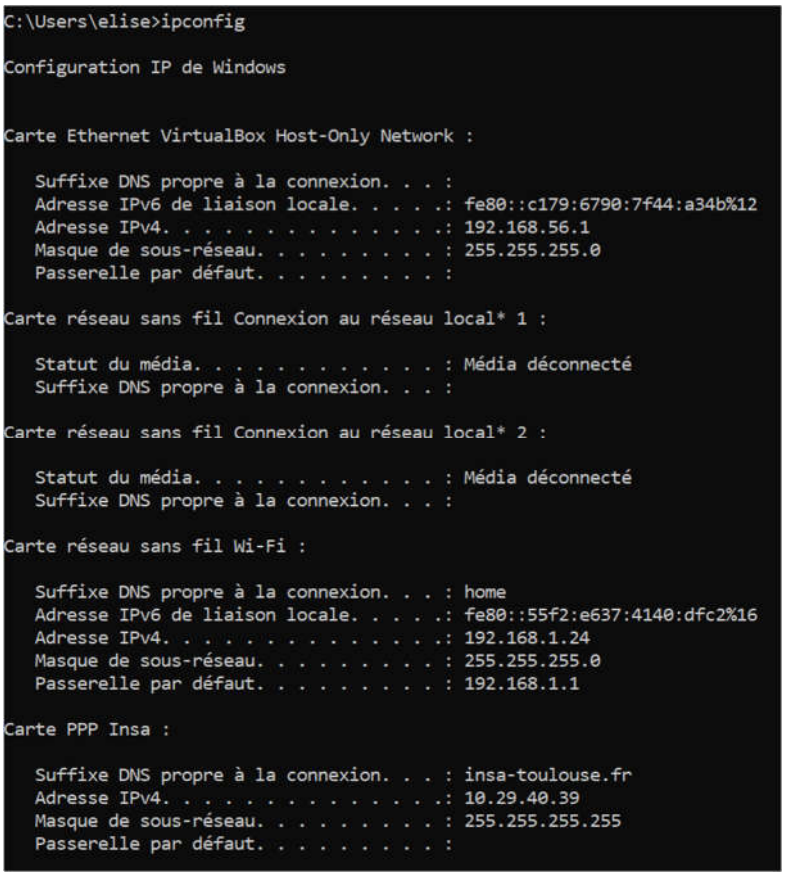

Figure 1 : Terminal après la commande ipconfig

Grâce à cette commande vous pourrez visualiser les différentes interfaces disponibles sur votre machine. Vous devrez identifier celle qui correspond à l'interface reliée au réseau local. Si la machine est reliée via un câble Ethernet cela correspondrait ici à la première interface Carte Ethernet mais si la machine est reliée en Wifi cela correspond par exemple à la Carte réseau sans fil Wi-Fi. Une fois la bonne interface identifiée il faut récupérer l'adresse IPv4 (le noter quelque part).

Ensuite vous pouvez exécuter le fichier interface.jar à l'aide la commande suivante :

java -jar interface.jar

Vous devriez obtenir un résultat similaire à la figure suivante :

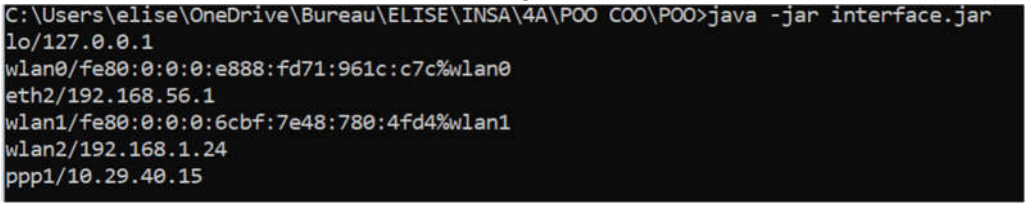

Figure 2 : Terminal après l'exécution de interface.jar

La dernière étape est de repérer l'adresse mémorisée plus haut et de noter le nom de l'interface correspondante. Ce nom-là correspond au champ interface dans le fichier config.json.

c. Exécution via le fichier projet.jar

La première façon d'exécuter le projet est d'utiliser le fichier projet.jar que nous mettons à votre disposition. Il y a trois conditions nécessaires pour que cela fonctionne :

- Être connecté au VPN de l'INSA (notamment pour l'accès à la base de données)
- Avoir le fichier config.json placé dans le même répertoire que le fichier .jar
- Avoir java 11 ou plus installé sur la machine

Si ces trois conditions sont remplies vous pouvez alors exécuter le fichier projet.jar.
Il y a deux manières d'exécuter ce fichier :

- En double cliquant dessus
- En utilisant la ligne de commande java -jar projet.jar

d. Importation du projet avec Eclipse

Il est également possible d'importer le projet sous Eclipse pour avoir accès au code. Nous avons mis le projet sur le dépôt Git et il est donc possible de l'importer. Nous avons développé le projet en Java 11 (et 13) il est donc nécessaire d'avoir une version de Java installée sur votre machine supérieure ou égale à 11.

En observant l'arborescence du projet Eclipse, vous devriez observer 7 packages (config, controller, gui, model, network, server et test) et un répertoire images. Dans chaque package se trouvent les différentes classes.

Pour pouvoir exécuter le projet il faut se rendre dans le package test et lancer la classe Launcher (run as Java Application).

3. Manuel d'utilisation

a. Première connexion : Inscription

Lors du lancement de l'application, vous arriverez sur une fenêtre comme ci-dessous :

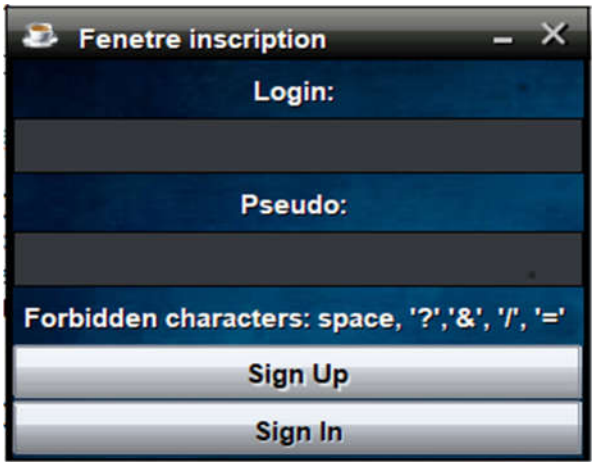

Figure 3 : Fenêtre Inscription

Lors de votre première connexion il vous faudra vous créer un “profil”. Pour cela remplissez les champs Login et Pseudo en respectant les règles indiquées concernant les caractères interdits. Ensuite cliquez sur Sign Up.

Le champ **Login** correspond à votre identifiant que vous utiliserez à chaque connexion à l’application. Le Login n’est pas modifiable et est unique de manière à vous identifier. Lors du choix du login si celui de votre choix est déjà utilisé par un autre utilisateur, l'application vous le notifiera et il faudra donc en choisir un autre pour pouvoir vous connecter.

Le champ **Pseudo** correspond au nom sous lequel les autres utilisateurs vous verront. Le Pseudo est modifiable par la suite à condition que le nom choisi ne soit pas utilisé par un autre utilisateur actif.

b. Connexion

Si ce n’est pas la première fois que vous vous connectez à l’application et que vous avez déjà réalisé une fois l’action décrite dans la section 3.a vous n’aurez donc pas besoin de vous inscrire mais simplement de vous connecter.

Pour ce faire cliquez simplement sur Sign In (lorsque vous êtes sur la fenêtre représentée en Figure 1) et la fenêtre suivante apparaîtra :

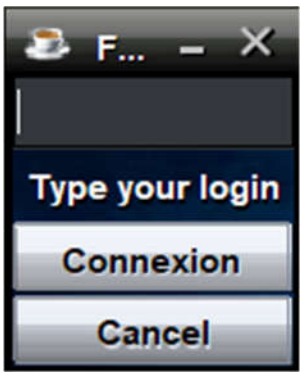

Figure 4 : Fenêtre Connexion

Dans cette fenêtre, vous n’avez qu’à taper le Login que vous aviez choisi lors de votre première connexion de manière à vous connecter.

Si vous accédez à cette fenêtre mais que vous n’avez jamais réalisé les actions de la section 3.a, cliquez sur le bouton Cancel de manière à retourner à l’écran précédent.

c. Discuter avec des utilisateurs actifs

Une fois que vous avez réussi à vous connecter vous arriverez sur la fenêtre suivante :

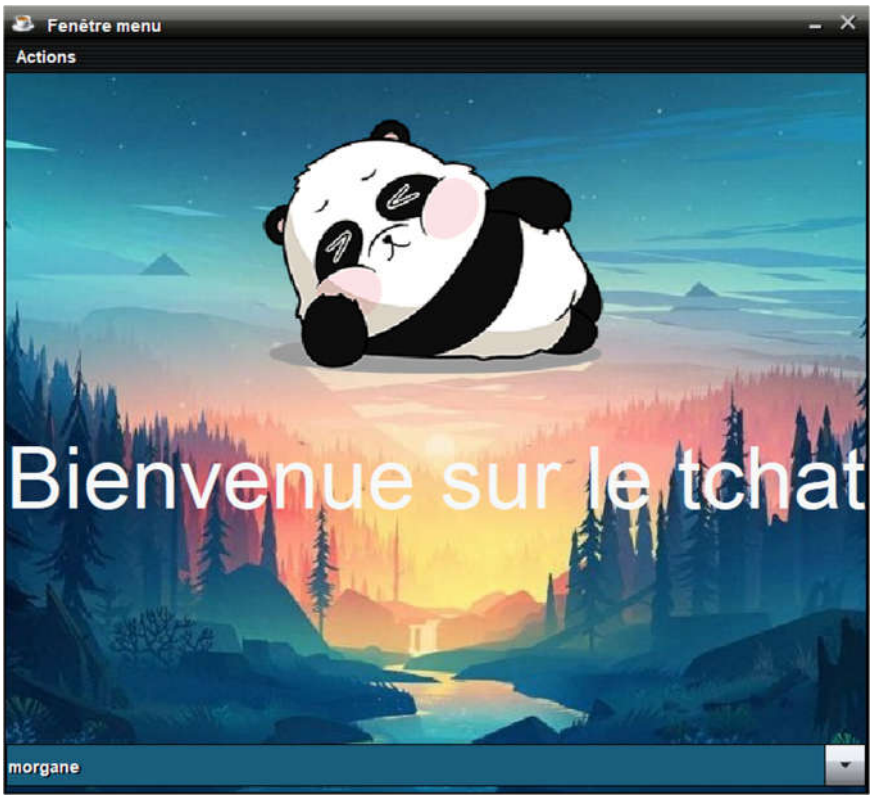

Figure 5 : Fenêtre Menu

La barre de menu déroulante en bas de l'écran permet de visualiser les utilisateurs actifs lorsqu'on clique dessus. Pour ouvrir une fenêtre de discussion avec un utilisateur, il suffit de cliquer sur son nom.

Lorsque vous avez cliqué sur le nom du destinataire une fenêtre de chat comme ci-dessous s'ouvrira.

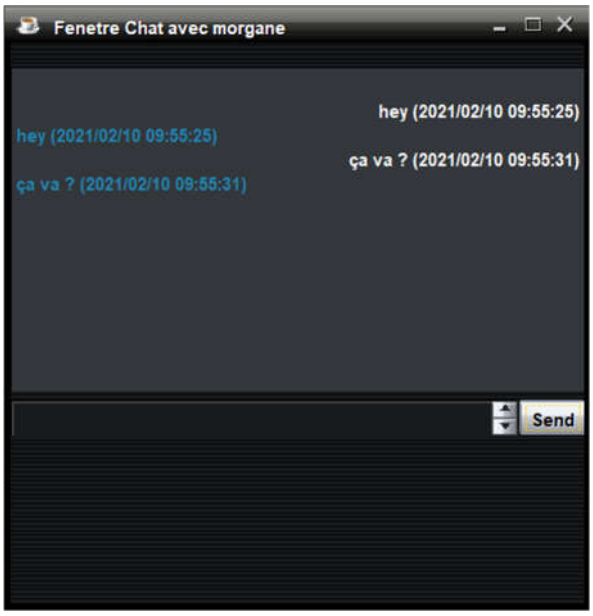

Figure 6 : Fenêtre Chat

Dans le cadre “Enter your message” vous pouvez taper le texte que vous souhaitez envoyer. Pour envoyer le message il faut appuyer sur le bouton Send.

Les messages s’affichent dans le grand cadre blanc au centre. Les messages que vous envoyez s’affichent en bleu à droite de l’écran et les réponses de votre interlocuteur s’affichent en rouge à gauche. Chaque message comporte la date à laquelle il a été envoyé. Toutes les discussions que vous pourrez avoir avec vos interlocuteurs seront sauvegardées dans un historique qui sera affiché à chaque fois que vous reviendrez sur la page de discussion concernée.

d. Changer son pseudo

Pour changer votre pseudo, il vous faut dans un premier temps être sur la Fenêtre d’accueil représentée sur la Figure 3. Une fois sur cette fenêtre il faut cliquer sur le bouton Actions.

Un menu composé de deux lignes va apparaître. La première ligne : Modify Your Pseudo nous intéresse. Cliquez sur cette phrase. La fenêtre suivante s’ouvrira :

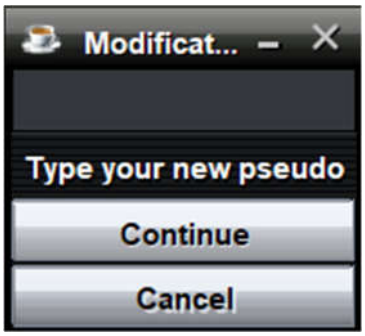

Figure 7 : Fenêtre Pseudo

Dans cette fenêtre vous avez la possibilité de rentrer votre nouveau pseudo. Pour valider votre choix il vous suffit de cliquer sur Continue.

Si le pseudo que vous avez choisi n’est pas disponible (car un autre utilisateur actif l’utilise notamment), une fenêtre s’ouvrira pour vous indiquer qu’il vous faut choisir un autre pseudo.

Si finalement vous ne souhaitez plus changer de pseudo vous pouvez cliquer sur le bouton Cancel de manière à revenir sur la fenêtre précédente.

4. Conception

a. Diagramme de cas d'utilisation

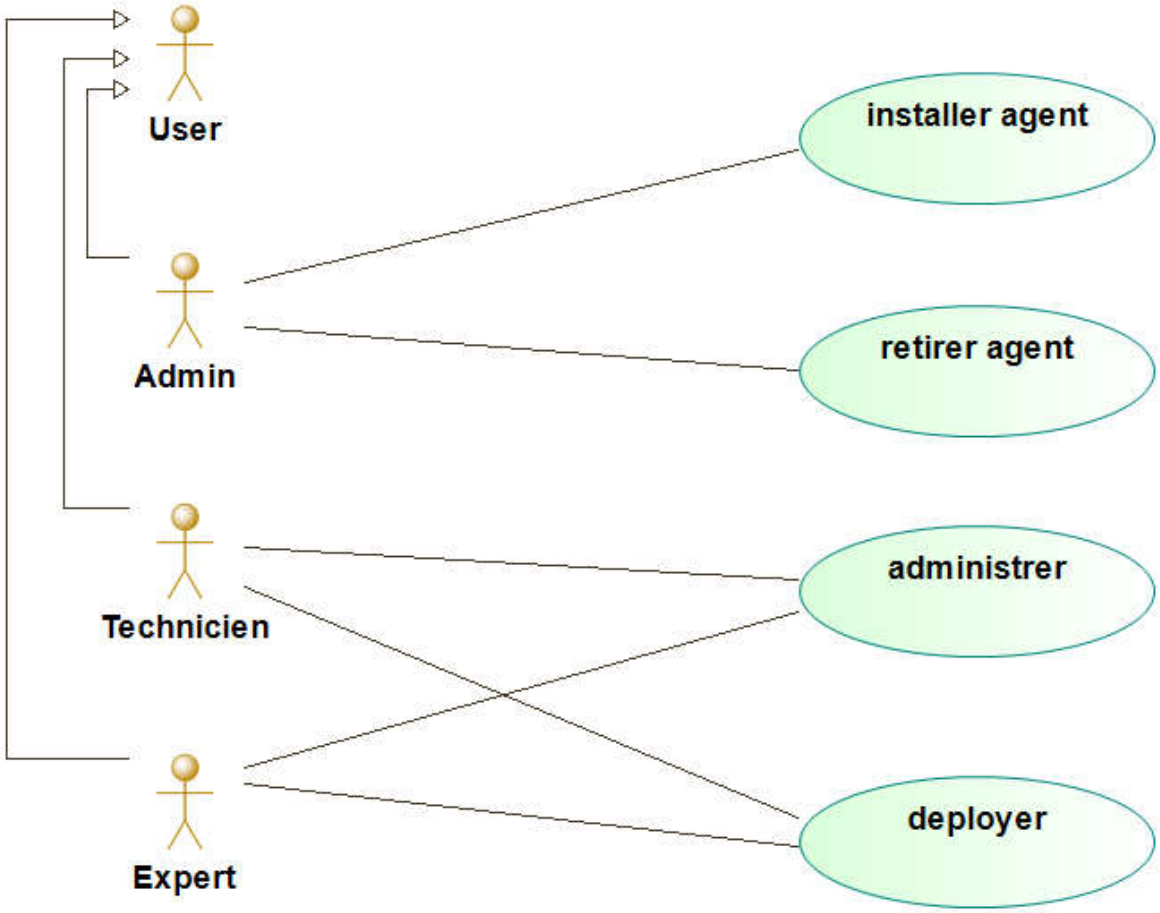

Figure 8 : Diagramme de cas d'utilisation (1)

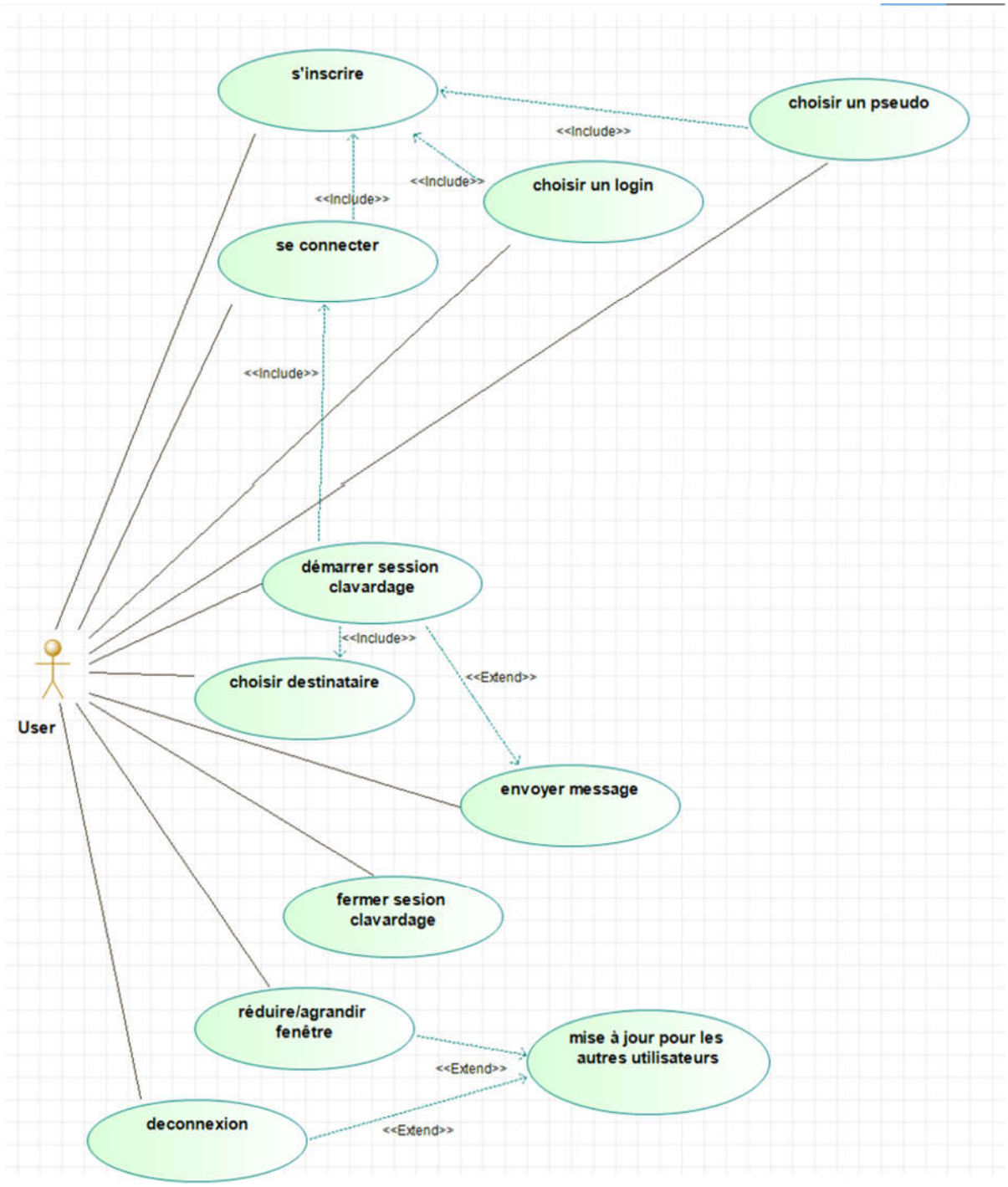

Figure 9 : Diagramme de cas d'utilisation (2)

b. Diagramme de classe

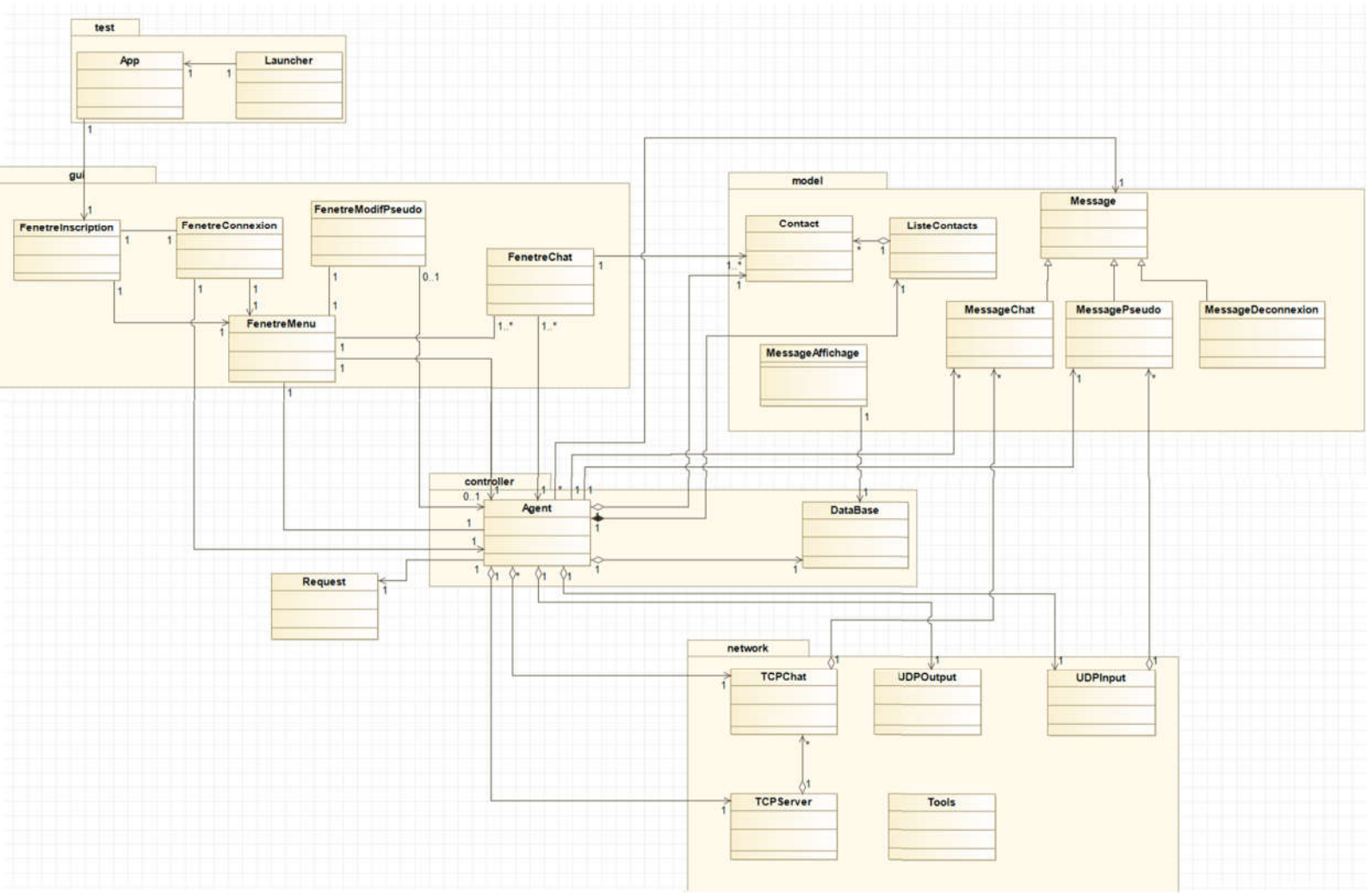

Figure 10 : Diagramme de classe

Pour des raisons de lisibilité, nous avons décidé de ne pas mettre les fonctions des différentes classes, certaines en ayant une dizaine voire plus.

De plus, nous avons utilisé à plusieurs reprises le design pattern “Singleton”. En effet, que ce soit pour la liste d’utilisateurs actifs ou pour la base de données, il était nécessaire de n’avoir qu’une seule version de l’élément. C’est pourquoi nous avons utilisé ce design pattern.

Enfin, nous avions également décidé d’utiliser l’architecture Modèle-Vue-Controller, comme on peut le voir grâce au package. Ce pattern nous a permis une organisation plus simple au niveau des différentes classes à créer, mais aussi au niveau de leurs fonctionnalités.

c. Diagramme de séquence

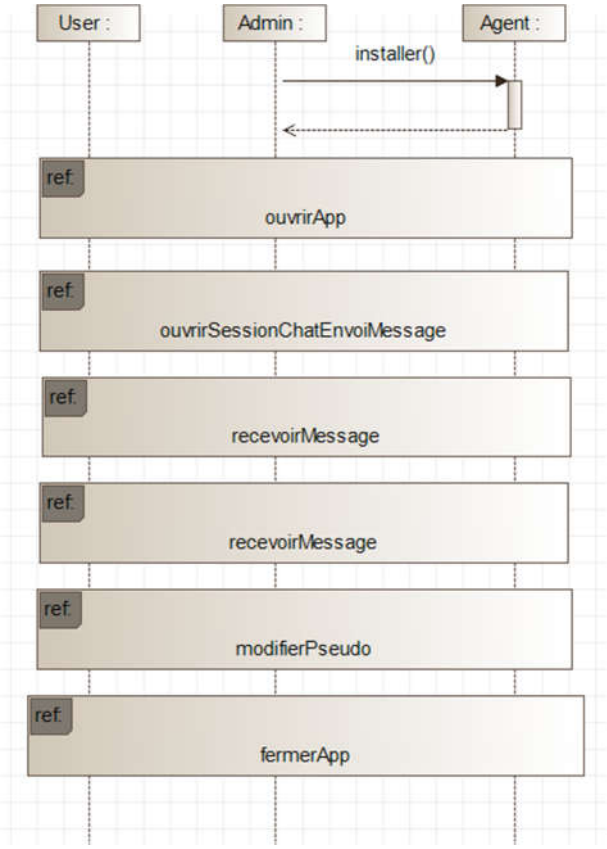

Figure 11 : Diagramme de séquence général

Notre premier diagramme de séquence regroupe toutes les principales fonctionnalités du système, que nous détaillons dans d'autres diagrammes de séquence, chacun dédié à une de ces fonctionnalités.

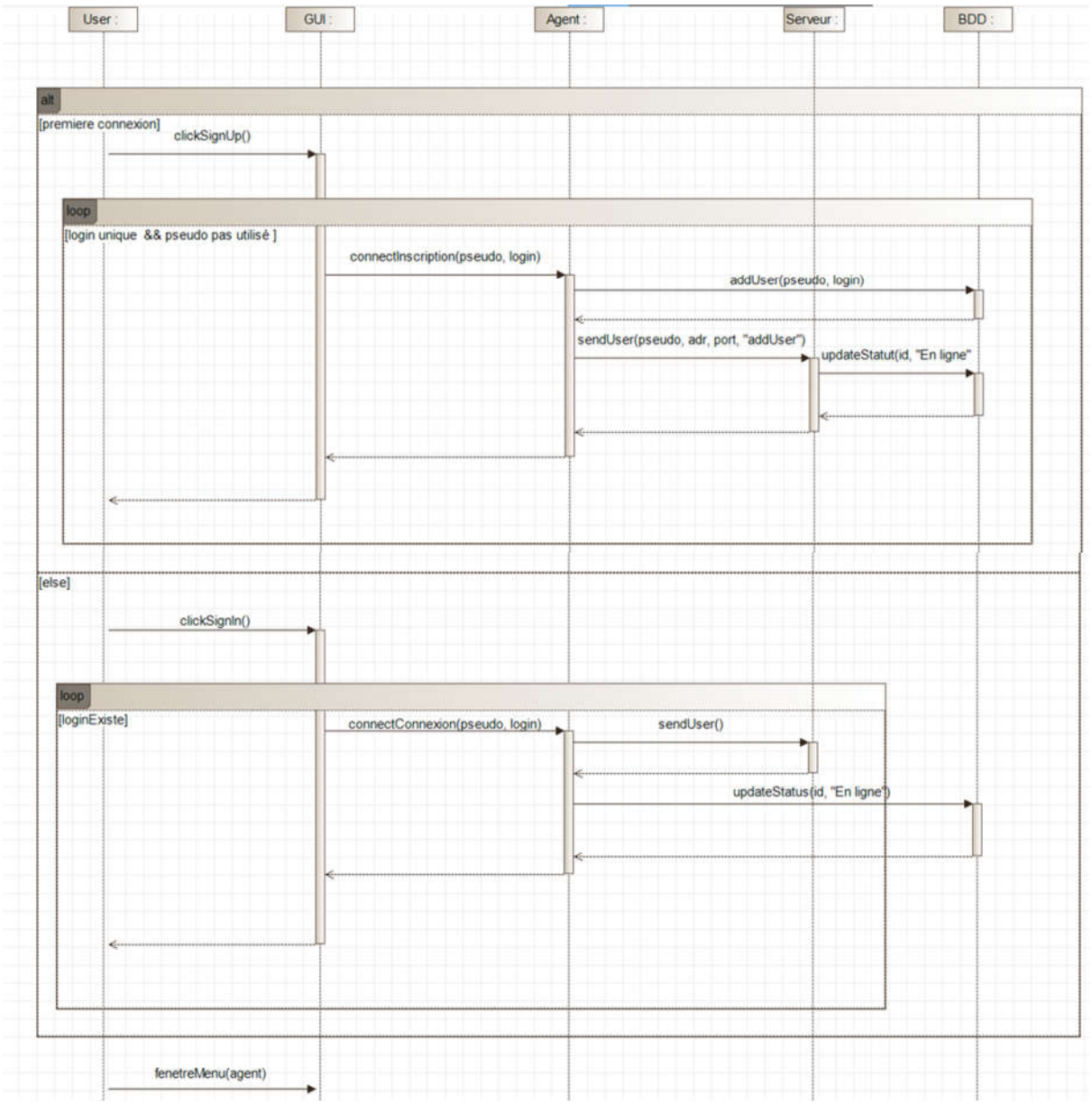

Figure 12 : Diagramme de séquence : Ouvrir App

- Phase de connexion avant la modification du cahier des charges :

Avant que le cahier des charges n'évolue, la phase de connexion fonctionnait de la façon suivante : lorsqu'un utilisateur se connectait, il envoyait un message UDP en broadcast à tous les utilisateurs actifs présents sur le réseau. Ces derniers répondaient en mentionnant leur pseudo. À la réception de tous les messages, A se construisait une liste des utilisateurs actifs en local grâce à tous les pseudos qu'il avait pu recevoir. Ensuite, grâce à cette liste il pouvait choisir son pseudo en vérifiant qu'il n'était pas présent dans la liste. Une fois un pseudo correct choisi, il envoyait un message UDP en broadcast pour notifier tous les autres utilisateurs de son pseudo. Ainsi, les autres utilisateurs, à la réception de ce message pouvaient mettre à jour leurs listes locales respectives.

De même pour la déconnexion, l'utilisateur qui se déconnecte envoie un message UDP en broadcast ce qui permet à tous les autres utilisateurs actifs de le supprimer de leurs listes.

- Phase de connexion après la modification du cahier des charges :

Lorsque le cahier des charges a évolué, imposant l'utilisation d'un serveur de présence, nous avons fait le choix de modifier le déroulement de la phase de connexion. En effet, nous n'avons pas implémenté la fonctionnalité de proxy du serveur. Pour donner une fonction utile à notre serveur, nous avons choisi de centraliser la liste des utilisateurs actifs. Dans notre nouvelle version, les utilisateurs n'ont plus chacun une liste en local mais ils consultent tous la même liste située au niveau du serveur.

Un utilisateur A se connecte et il envoie toujours un message UDP en broadcast aux autres utilisateurs actifs. Pour choisir son pseudo, A envoie une requête au serveur contenant le pseudo qu'il souhaite choisir, ce dernier lui répond qu'il a le droit de l'utiliser ou non. Si la réponse est positive, A est ajouté dans la liste des utilisateurs actifs au niveau du serveur et il se connecte. Les autres utilisateurs actifs, de la même façon, répondent au premier message UDP de A par un message UDP contenant leur pseudo. Ces messages de réponses permettent à A d'ajouter dans la fenêtre menu les personnes actives présentes.

De même pour la déconnexion, l'utilisateur qui se déconnecte notifie le serveur qui le supprime de la liste et il envoie un message UDP en broadcast ce qui permet aux autres utilisateurs d'actualiser leurs fenêtres Menu où les utilisateurs actifs sont affichés.
Remarque : Après la modification de notre code décrite ci-dessus, les messages UDP n'ont plus vraiment d'intérêt car nous aurions pu aussi passer par le serveur pour maintenir l'affichage des utilisateurs actifs dans la fenêtre de menu. Cependant, nous avons choisi tout de même de modifier le code pour montrer que nous avons réussi à faire interagir l'application et le serveur. Pour montrer que nous avions également réalisé la communication udp nous avons donc laissé cette partie en l'utilisant pour maintenir l'affichage de la liste.

Figure 13 : Diagramme de séquence : Fermer App

- Phase d'ouverture d'un tchat et d'envoi du message

Afin de pouvoir envoyer des messages l'utilisateur doit tout d'abord cliquer sur l'utilisateur avec qui il veut discuter dans la liste d'utilisateurs actifs. Une fenêtre de tchat s'ouvrira, ce qui enclenchera la création d'un socket entre les deux utilisateurs, ainsi qu'un tableau dans la base de données afin de stocker les messages échangés. C'est grâce à cette table que l'historique est affiché à chaque fois qu'une conversation est ouverte une nouvelle fois.

Lorsque l'utilisateur clique sur le bouton Send, si c'est la première fois que les deux utilisateurs se parlent c'est à ce moment-là que la table sera créée dans la base de données, sinon le message sera ajouté dans la table déjà existante.

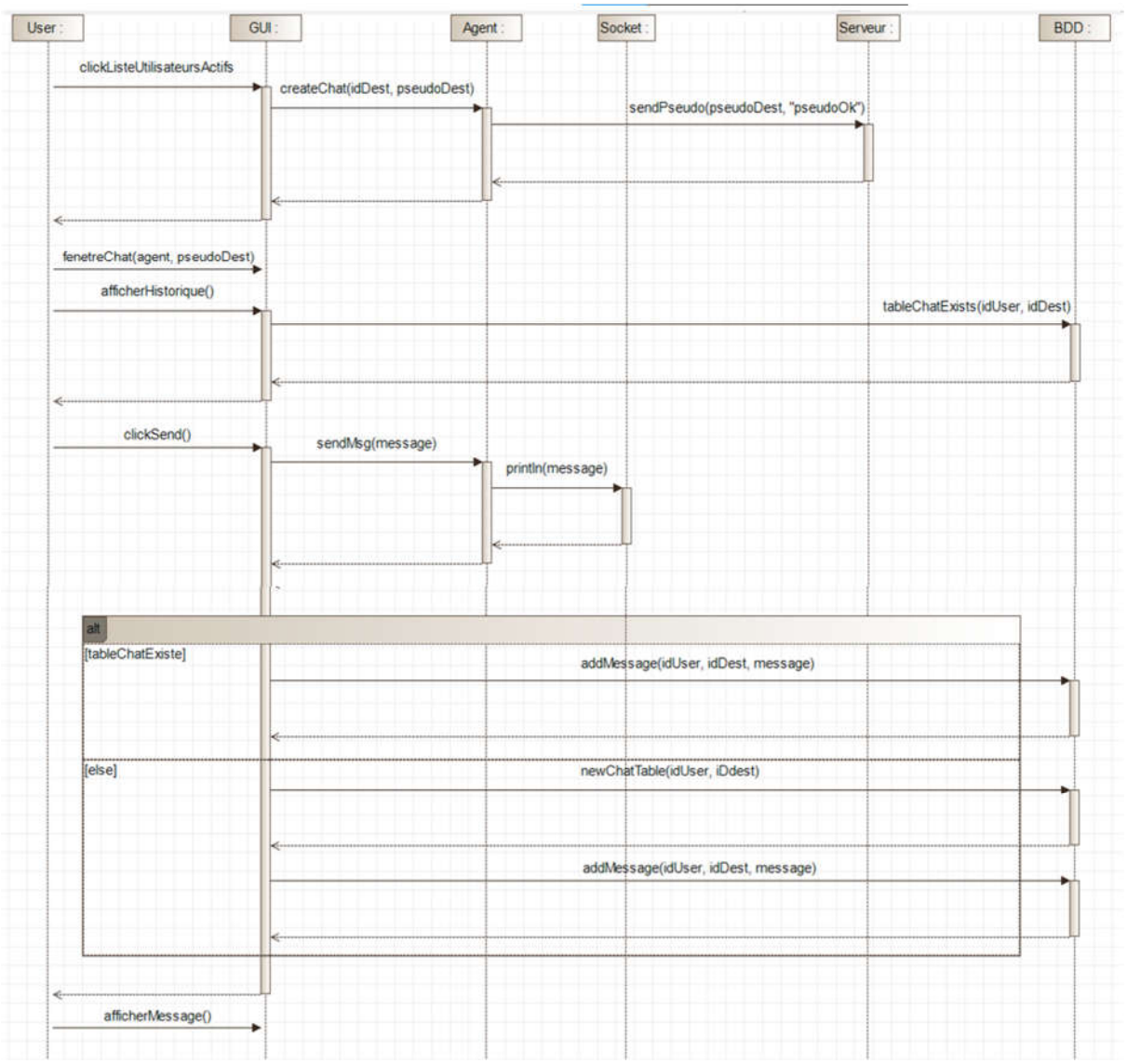

Figure 14 : Diagramme de séquence : Ouvrir Session Chat Envoi Message

- Phase de réception d'un message :

Concernant la réception des messages, elle est plutôt simple. En effet, elle consiste à lire le buffer du socket et à l'afficher dans la fenêtre de discussion. Il n'y pas besoin de l'ajouter au tableau de la base de données puisque cette action sera faite lors de l'envoi de l'autre utilisateur.

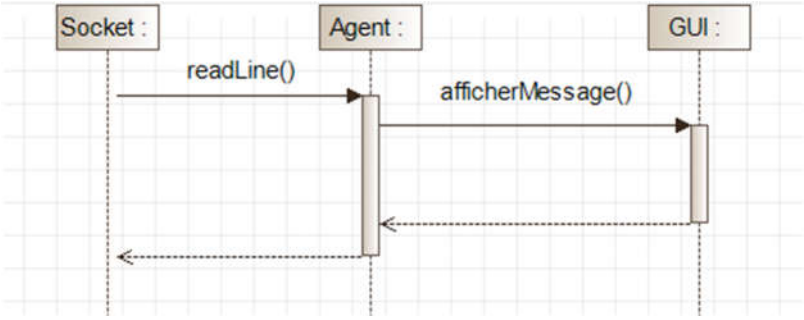

Figure 15 : Diagramme de séquence : Recevoir Message

- Phase de modification du pseudo :

Afin de modifier le pseudo de l'utilisateur, ce dernier doit cliquer sur l'onglet concernant cette tâche. Ensuite il doit rentrer un pseudo qui n'est pas utilisé. C'est l'agent qui se charge de changer le pseudo, que ce soit dans le serveur ou dans la base de données. De plus, une fois la modification effectuée, la liste des utilisateurs est mise à jour automatiquement et c'est donc le nouveau pseudo qui est affiché.

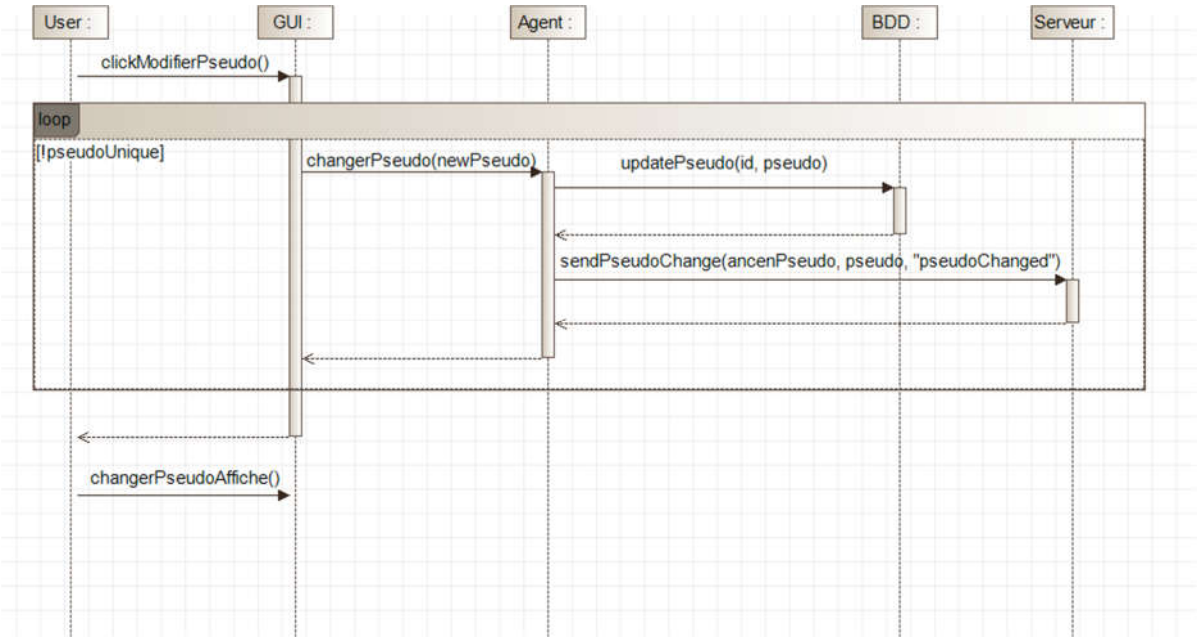

Figure 16 : Diagramme de séquence : Modifier Pseudo

d. Diagramme d'état

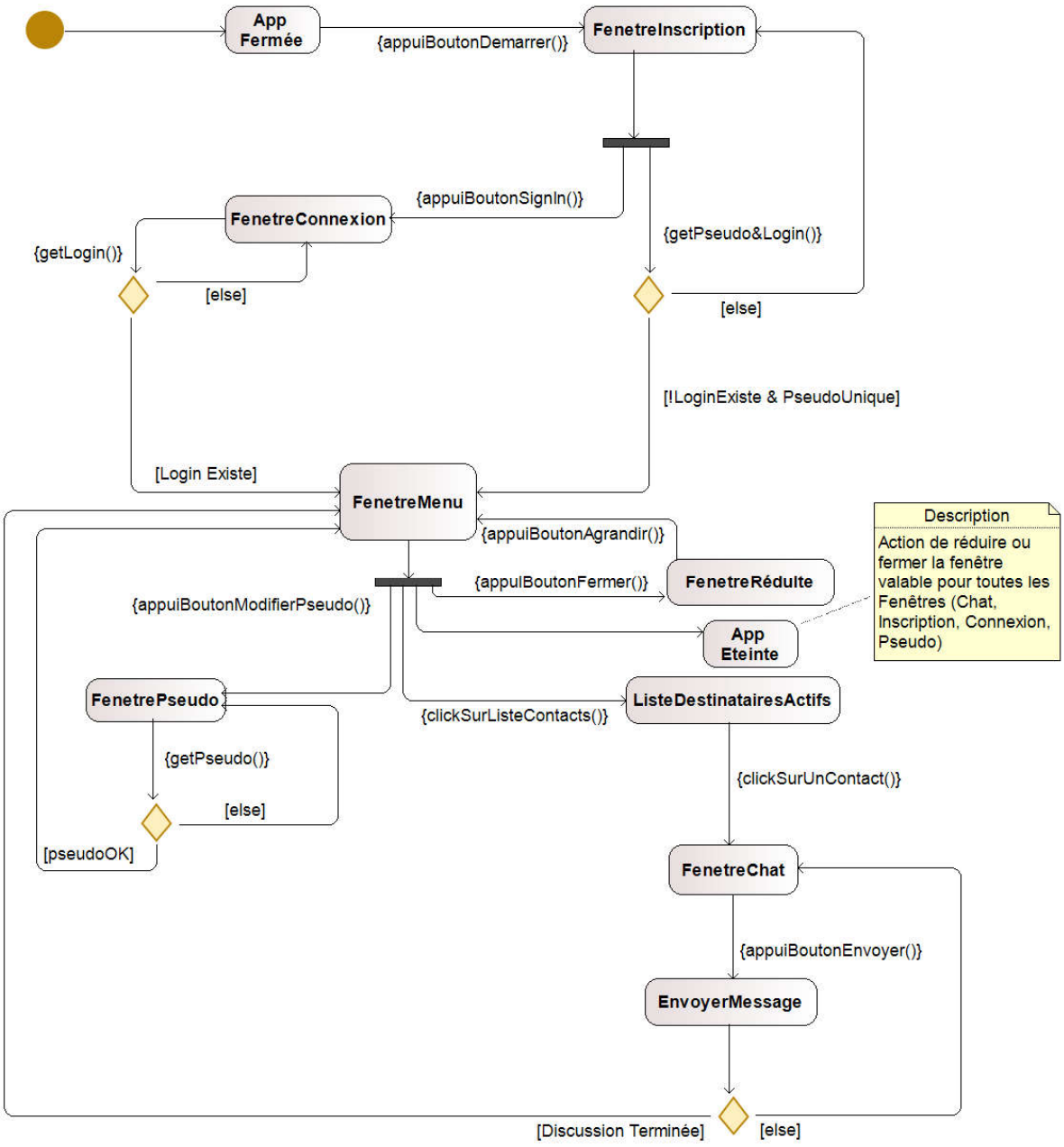

Figure 17 : Diagramme d'états

5. Problèmes

Lors de la réalisation du projet nous avons rencontré de nombreux problèmes. Les principales sources de ces derniers étant le travail à distance et le manque de connaissances techniques.

a. Envoi uniquement de texte

Le cahier des charges spécifie que l'application doit pouvoir permettre l'envoi de messages textes mais également de fichiers, d'images et de schémas.

Pour commencer nous avons décidé de réaliser uniquement l'envoi de messages textes car rajouter d'autres formats de messages représentait pour nous une difficulté supplémentaire. Nous avions décidé de nous pencher sur la question plus tard lors de l'avancement du projet.

Malheureusement nous avons rencontré beaucoup de problèmes qui nous ont retardés c'est pourquoi nous avons finalement décidé de renoncer à cette fonctionnalité par manque de temps.

b. Usage du Servlet

Pendant la première moitié de la période destinée à la réalisation du projet nous avons opté comme choix d'implémentation de maintenir une liste d'utilisateurs actifs pour chaque personne connectée sur le réseau (voir section 4).

Lorsque le cahier des charges a évolué nous avons appris qu'il fallait mettre en place un serveur de présence permettant à des utilisateurs externes de se connecter et d'utiliser l'application. La mise en place du Servlet nous a demandé beaucoup de temps car c'est une technologie nouvelle et que nous n'avions jamais manipulé.

Nous avons eu des difficultés à comprendre comment relier le Servlet à notre projet. Nous avons renoncé à implémenter la partie proxy du serveur car principalement nous avons manqué de temps. Pour que notre Servlet ait une utilité, nous avons décidé de migrer la liste des utilisateurs actifs sur le serveur de manière qu'elle soit centralisée. Cependant nous sommes conscientes que ce n'était pas forcément le comportement attendu.

c. Environnement de test

Enfin, une de nos principales sources de problème est venue du fait que nous avons dû réaliser le projet à distance. En termes de développement cela n'a pas posé de problème car nous avons utilisé des moyens de communication ainsi que git pour échanger le code.

En revanche, pour ce qui est des tests, cela nous a posé beaucoup de problèmes. En effet, pour tester tout ce qui implique les messages, la communication entre utilisateurs, la connexion ou l'inscription (notamment l'envoi de messages en UDP). Comme notre application ne fonctionne que si les machines se trouvent sur le même réseau, nous devions nous voir en présentiel. Se réunir pour travailler sur le projet n'a pas forcément été très aisé compte tenu de la situation sanitaire et des différentes mesures gouvernementales (couvre-feu notamment).

Finalement, nous n'avons testé notre application qu'avec deux utilisateurs en simultané. Nous ne pouvons pas garantir le fonctionnement de l'application avec plus de deux utilisateurs.

d. Problème lors de la déconnexion

Lors des dernières semaines du projet, nous avons fait face à un problème que nous ne sommes pas parvenus à résoudre.

Le problème est le suivant : A et B se connectent à l'application et discutent sur un chat. Leur discussion se termine et B décide de quitter le chat et de se déconnecter. Pendant ce temps-là, A reste actif. Ensuite, B se reconnecte et ouvre un chat avec A. Dans cette situation, B reçoit les messages de A qui s'affichent dans sa fenêtre de chat mais A ne reçoit pas les messages que B lui envoie.

Nous avons identifié ce problème en testant différents scénarios qui pouvaient se produire. Nous avons essayé de le résoudre en cherchant les sources du problème mais aucune de nos solutions n'a fonctionné.

6. Tests

Concernant la phase de tests nous avons réalisé différentes choses. Dans un premier temps, à chaque fois que nous avons développé une fonction, nous avons testé son fonctionnement de manière individuelle. Nous n'avons pas écrit des tests unitaires JUnit à proprement parler mais nous avons réalisé l'équivalent dans un main en appelant les fonctions et en observant les comportements et les résultats.

Ensuite, lorsque nous avons rencontré de plus gros problèmes nous avons beaucoup utilisé le mode debug d'Eclipse pour mieux comprendre le comportement de notre code et identifier les parties du code qui pouvaient poser des problèmes.

Pour identifier les endroits dans le code qui posaient des problèmes nous avons mis des *System.out.println("")* dans différentes fonctions pour identifier la synchronisation entre les événements et leur ordre d'exécution.

Enfin, pour tester en réseau nous nous retrouvions pour tester avec deux ordinateurs sur le même réseau en utilisant le VPN de l'INSA.

7. Conclusion

Le but de ce projet était d'implémenter un système de clavardage utilisable par les employés d'une entreprise. Nous avons réussi à construire un système permettant d'échanger des messages textuels au sein d''un même réseau, tout en assurant une certaine autonomie du système concernant la mise à jour des données des utilisateurs.

Cependant, certains problèmes subsistent, malgré de nombreux essais pour les résoudre. Le contexte et l'impossibilité de pouvoir tester notre code en vrai, ont rendu ces problèmes très durs à résoudre, ce qui explique en grande partie pourquoi ils sont toujours présents dans notre version finale.

Malgré tout ceci, notre système de clavardage possède les fonctionnalités les plus importantes d'un tchat, le rendant utilisable par les utilisateurs.

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