INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Département de Génie Électrique & Informatique

Rapport Java

Projet Clavardage

January 27, 2023

Students :

Matthis	RAVEAU	raveau@insa-toulouse.fr
Patrick	OSORNIO-GLEASON	osorniog@insa-toulouse.fr

Introduction

Dans ce rapport, nous allons expliquer le processus de conception qui nous a conduit à réaliser nos différents diagramme UML ainsi que les choix faits dans le cadre de la réalisation du Projet Java Clavardage.

Ainsi, notre rapport sera accompagné premièrement des diagrammes UML (qu'il est possible de voir avec d'avantages de détails dans le repository Git car par soucis de place nous n'avons pas pu absolument tout inclure de manière succincte), puis nous élaborerons sur nos choix d'architecture et technologiques pour finir avec les procédures d'évaluation et de tests.

A la suite de tout ceci, nous avons inclus la procédure d'installation de notre projet ainsi qu'un concis manuel d'utilisation à l'attention des utilisateurs.

Contents

1 Diagrammes UML

Nous allons voir dans cette section l'ensemble des diagrammes UML que nous avons pu produire et qui nous ont aidés dans la création de notre application.

1.1 Identification des acteurs

1.1.1 Primaires

Les acteurs principaux sont l'utilisateur du service de message et l'administrateur.

1.1.2 Secondaires

L'autre utilisateur avec lequel communique le premier utilisateur peut être considéré comme un acteur secondaire.

1.2 Diagramme des cas d'utilisation

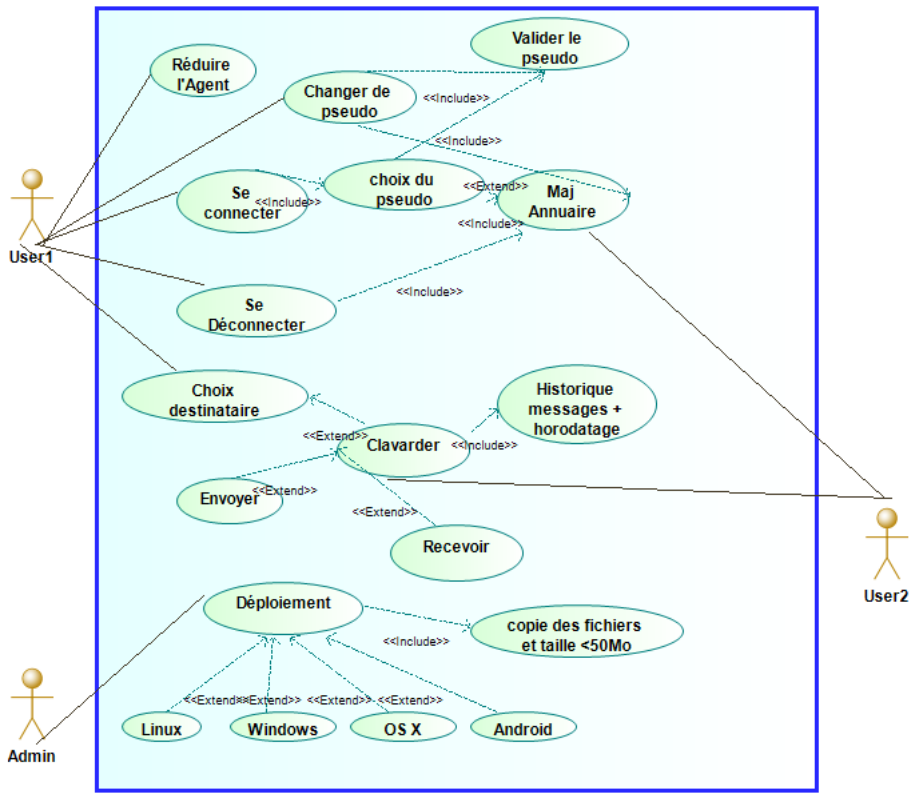


Figure 1: "Figure 1 : Diagramme des cas d'utilisation"

On peut voir ci-dessus le Diagramme des cas d'utilisation avec les acteurs présentés auparavant ainsi que toutes les actions que chacun peut réaliser. Notre implémentation est fidèle à ce diagramme et tous les cas d'utilisation du cahier des charges respectées.

1.3 Diagramme des classes

1.3.1 Schéma

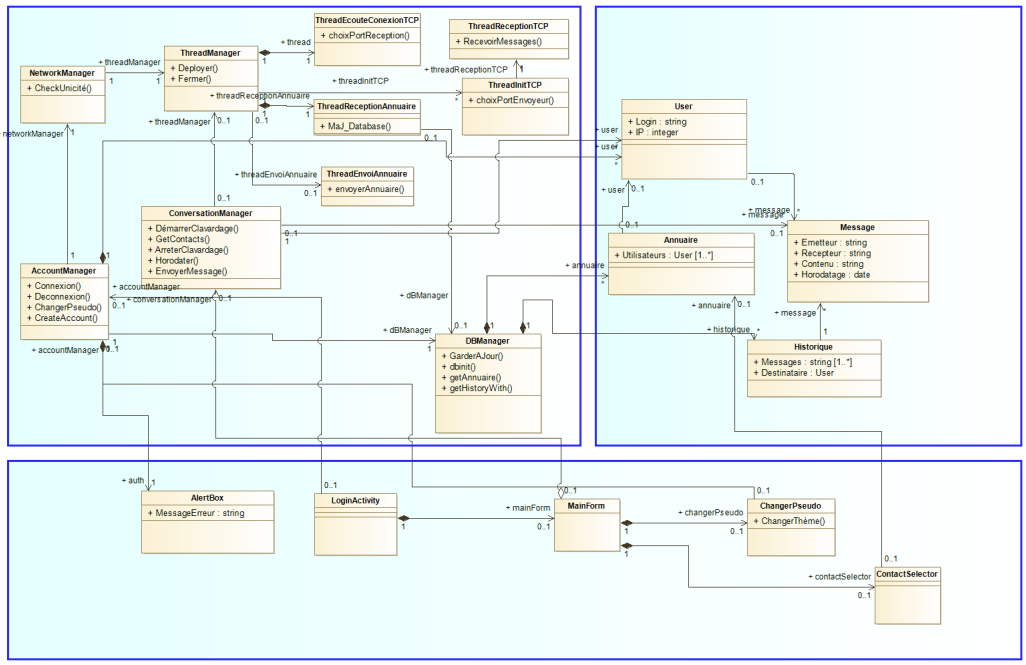


Figure 2: "Figure 2 : Diagramme des classes"

1.3.2 Détails et typage

Nous avons choisi d'utiliser une architecture MVC (Modèle Vue Contrôleur) avec les modèles en haut à gauche, les contrôleurs en haut à droite et Vue en bas.

1.4 Diagramme de séquence

Pour le diagramme de séquence nous avons choisi d'expliquer les situations suivantes.

1.4.1 Connexion/Déconnexion

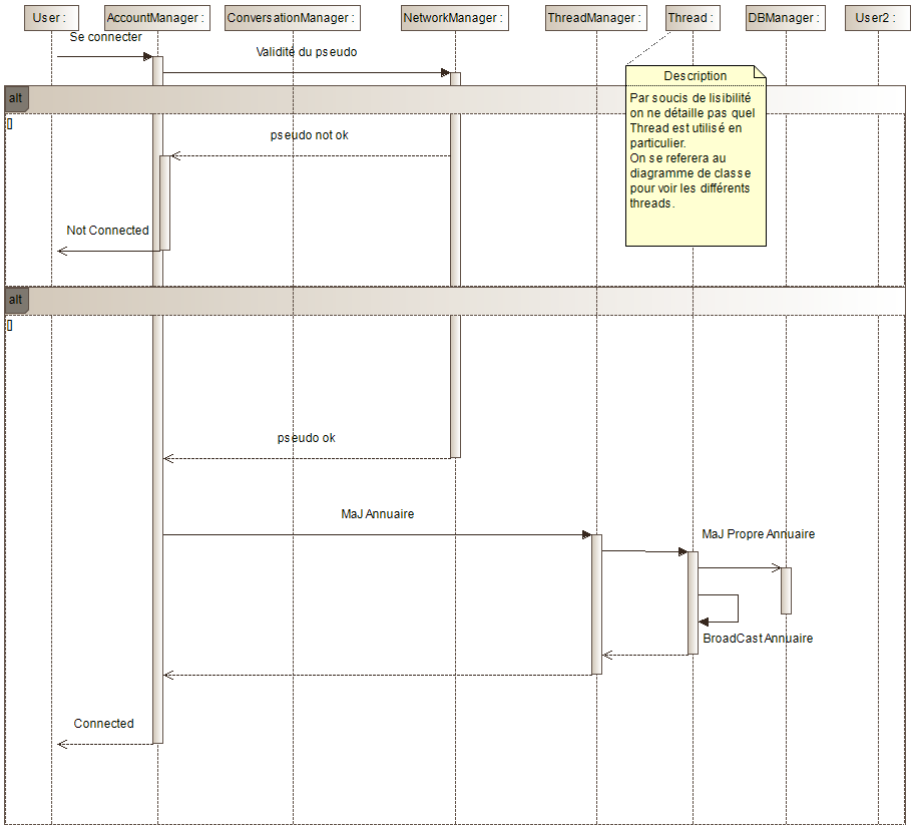


Figure 3: "Figure 3 : Diagramme de séquence Connexion avec Validation de pseudo"

Pour la validation du pseudo, soit il est valide, soit il ne l'est pas. C'est-à-dire que soit quelqu'un dans notre database possède déjà ce pseudo ou alors il est libre. Lorsqu'il est valide, l'annuaire est mis à jour et on le broadcast, sinon une alert box qui nous invite à réessayer est levée.

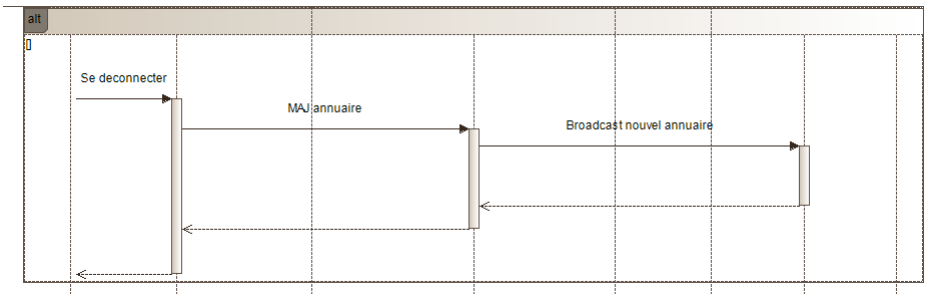


Figure 4: "Figure 4 : Diagramme de séquence Déconnexion"

Lors d'une déconnexion, on se retire de l'annuaire qui est mis à jour et on le broadcast. Aussi nous devons fermer les threads lancés auparavant en ayant prévenu les personnes avec qui nous discutions au préalable de la fermeture de leur session de clavardage avec nous.

1.4.2 Envoi Message

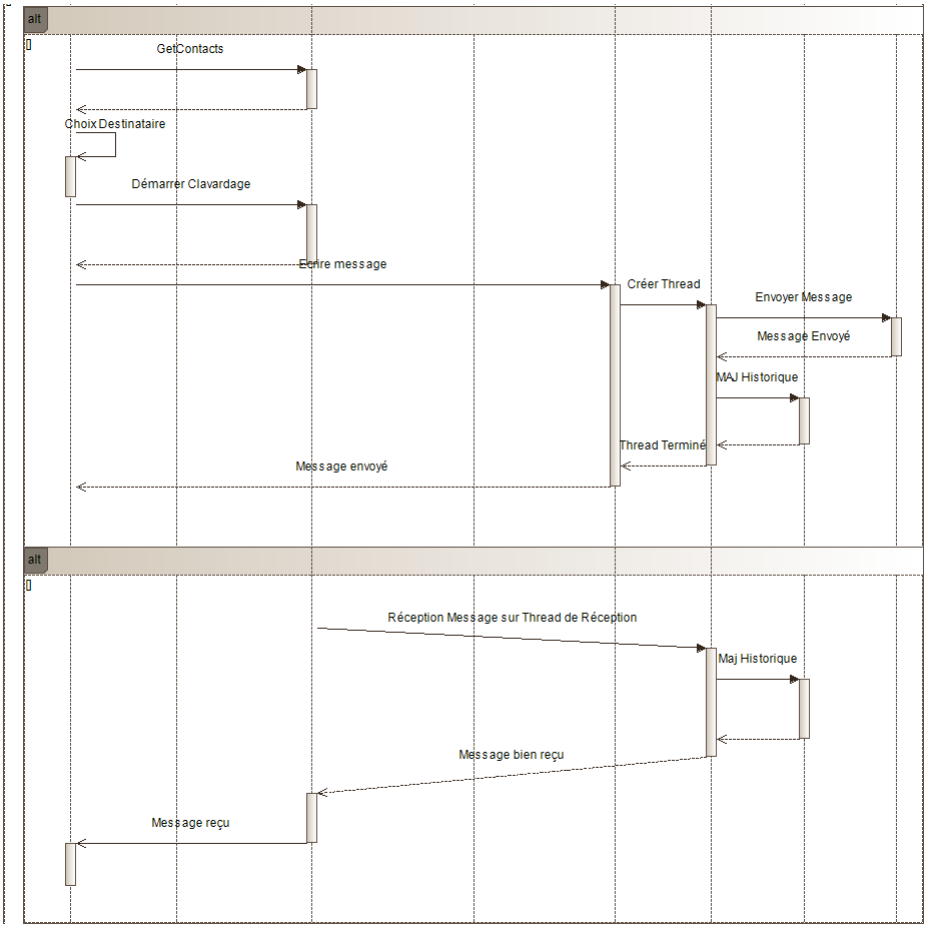


Figure 5: "Figure 5 : Diagramme de séquence envoi message"

Lors de l'envoi d'un message, le manager des conversations nous renvoie la liste des contacts parmi lesquels on peut initier une conversation, on crée une thread d'initiation à la connexion pour pouvoir communiquer avec le contact choisi et pour chaque envoi et réception on rajoute le message horodaté à l'historique. Pour la réception d'un message, un thread permettant que quiconque puisse initier la connexion avec nous est lancé et tourne à tout instant ; un autre thread de réception de message est instancié lorsque quelqu'un initie la connexion avec nous ou si nous même souhaitons communiquer avec un autre utilisateur.

1.4.3 Changement de pseudo

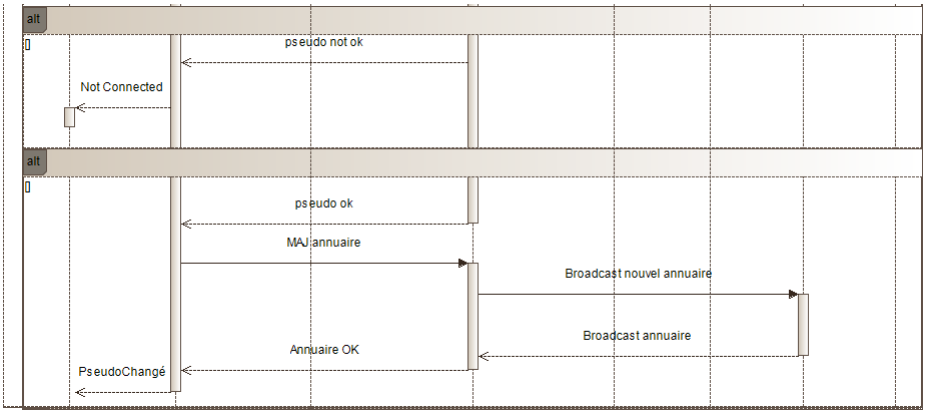


Figure 6: "Figure 6 : Diagramme de séquence changement de Pseudo"

Le changement de pseudo est un cas particulier de la connexion au final et fonctionne presque de la même manière.

1.5 Diagramme de composites

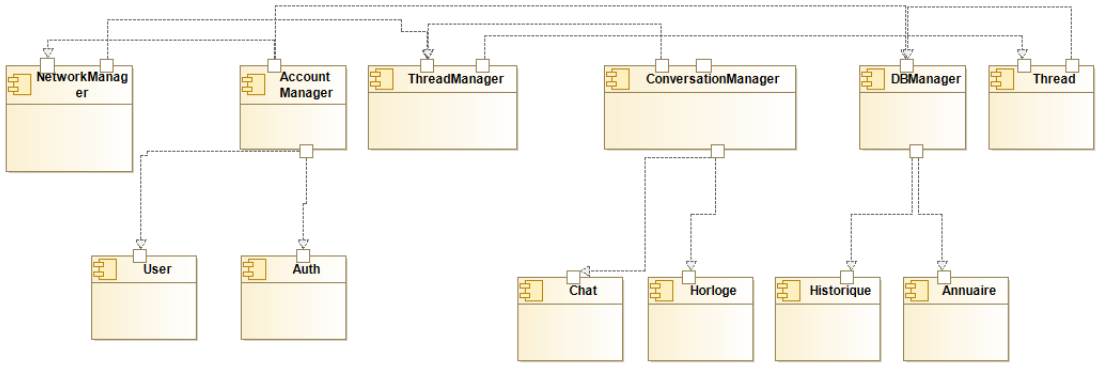


Figure 7: "Figure 7 : Diagramme de composites"

On peut voir l'interconnexion entre les différentes classes avec au-dessus les contrôleurs et en bas la vue et les modèles.

1.6 Diagramme de déploiement

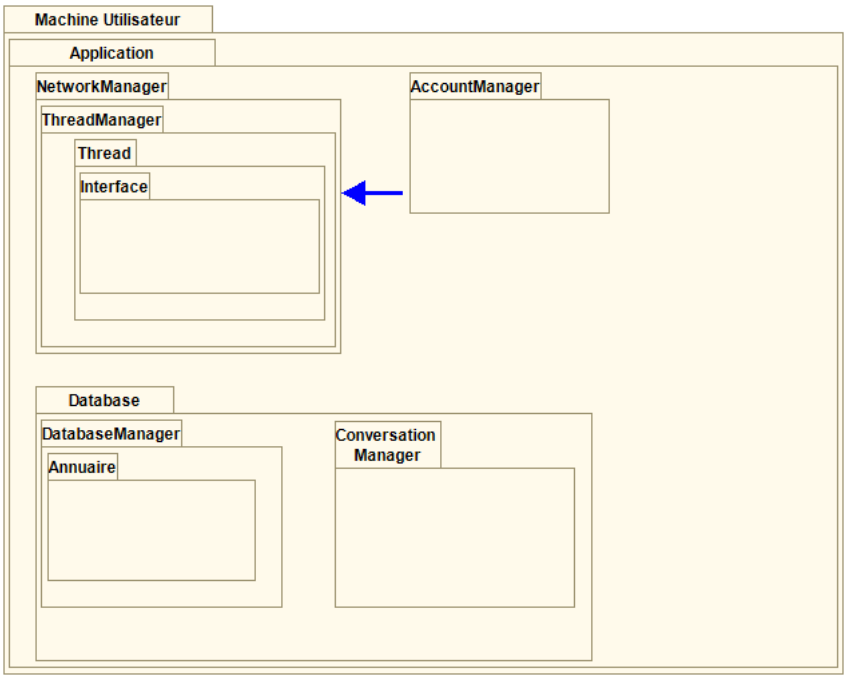


Figure 8: "Figure 8 : Diagramme de déploiement"

Le diagramme de déploiement indique l'imbrication de nos modèles et contrôleurs.

1.7 Schéma de la BDD

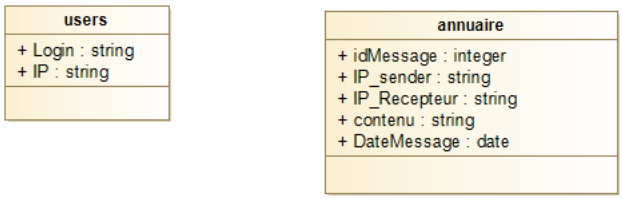


Figure 9: "Figure 9 : Schéma de la BDD"

On détaillera dans les parties suivantes les choix faits pour notre database.

1.8 Diagramme des composants

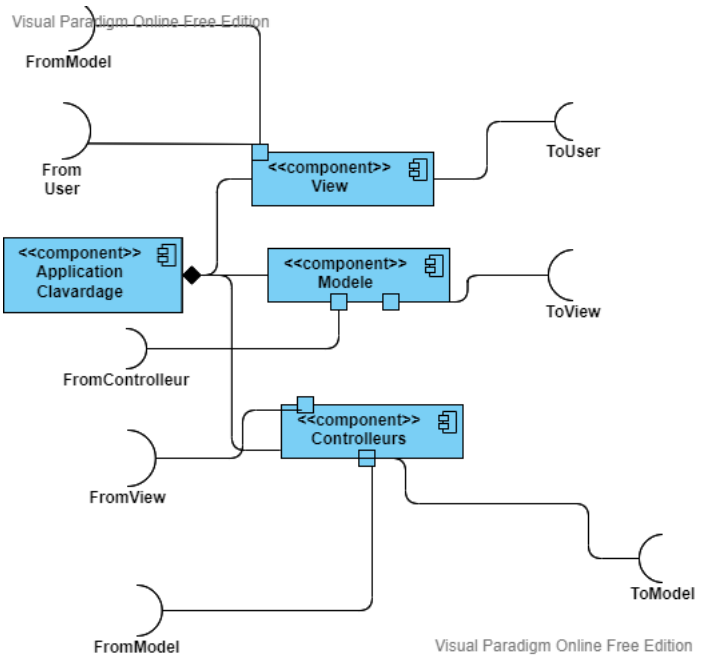


Figure 10: "Figure 10 : Diagramme des Composants"

Diagramme des composants fait en plus du plan prévu initialement spécifiant les interactions entre MVC.

2 Choix d'architecture et technologiques et procédures de tests

Dans cette section nous allons discuter des quelques choix que nous avons pu faire sur notre projet qui peuvent différer d'un projet à l'autre.

2.1 Choix d'architecture et technologiques

Dans un premier temps, nous avons choisi de séparer la tâche en deux en commençant à créer une interface graphique simpliste pour pouvoir effectuer des tests d'intégration assez vite et tôt tandis que d'un autre côté nous commencions directement la base de données pour prévoir l'affichage de message et d'annuaire sur le GUI très rapidement.
Nous avons décidé d'implémenter Maven dès le début, vu qu'on avait besoin de travailler à deux, en changeant de salle parfois, ceci nous a permit de gérer les dépendances dont on avait besoin, notamment JDBC pour la base de données. De plus, le procès d'instalation est simplifié. Nous avons fait le choix pour notre base de données de suivre un modèle décentralisé ; c'est à dire que nous souhaitions que chaque utilisateur de l'application ait sa propre base de données locale. Cela implique beaucoup de communications entre chacun des clients mais cela permet de ne pas avoir de serveur central.

Pour la mise en place de cette BDD nous avons choisi de l'implémenter avec sqlite et pour que chaque utilisateur puisse informer les autres de sa présence et récupérer les autres utilisateurs en ligne à tout instant nous avons prévu un système de broadcast de la database en UDP sur le sous réseau sur lequel l'application est utilisée.
Il s'agit d'une des hypothèses que nous avons suivi où nous avons décidé que l'application sera uniquement utilisable sur un unique sous-réseau ce qui semble à notre sens ne poser aucun problème quant au cahier des charges.

Pour les messages entre utilisateurs nous avons fait le choix du protocole TCP qui assure un mécanisme de reprise des pertes de tel sorte à ce qu'aucun message ne trouve pas son destinataire. L'établissement d'une connexion TCP se fait via un thread qui tourne à tout instant de reception et qui va pouvoir établir un autre thread sur un port d'écoute particulier que seul le destinataire des messages connaîtra.

Par ailleurs, l'historique des messages est également stocké de manière locale et on n'a bien entendu accès uniquement aux messages qui nous concernent.
Chaque message contient l'information nécessaire pour restituer la conversation avec des requêtes simples. Un message est composé de qui émet le message, qui le reçoit,le contenu du message, l'horodatage et un id de message unique.

Enfin, pour le choix du système d'authentification, nous n'utilisons pas de mot de passe. L'utilisateur peut choisir n'importe quel pseudo qui n'est pas encore utilisé , même si son pseudo est différent du dernier qu'il a utilisé. Cela est dû au fait que nous utilisons l'IP du sous-réseau comme facteur d'authentification et que chaque machine qui utilise l'application aura sa propre identité dans l'annuaire.

2.2 Procédures d'évaluation et de tests

Nous avons divisé l'application en plusieurs classes Java bien distinctes que nous avons testées de manière unitaire avant même de les relier entre elles et de les intégrer au GUI. Tout d'abord nous avons mis en place et tester la base de données avec ajout de messages entre deux utilisateurs,

suppression de message, changement de pseudo, test d'unicité de pseudo et bien d'autres.

Il fallait aussi faire en sorte que le broadcast marche bel et bien, que toute les machines le reçoivent et que l'on puisse répondre avec le protocole UDP comme prévu. S'assurer que la BDD fonctionne n'était pas le plus difficile. Il ne resterait plus qu'à s'occuper de son bon fonctionnement une fois intégrée à l'interface graphique. Pour tester l'interface graphique, on a construit une base de données avec quelques contacts et messages, ceci nous a permit de voire chaque changement immédiatement.

Maintenant, il a fallu aussi tester le protocole TCP mis en place et les nombreux threads qui l'accompagnent. Par conséquent on a d'abord vérifié que le scan des ports et l'établissement de connexion entre les deux utilisateurs se fait bien avant de rajouter un troisième utilisateur et de vérifier que cela n'impacte pas la connexion avec les deux premiers et que le troisième puisse malgré tout initier la connexion avec n'importe lequel des deux autres.

Avec l'intégration faite, on a pu commencer la partie la plus importante du test de l'application telle quelle. On a donc tester toutes les fonctionnalités : la connexion avec un pseudo non utilisé, la connexion avec un pseudo utilisé qui n'est pas autorisée, la déconnexion, le changement de pseudo vers un pseudo non pris, le changement de pseudo vers un pseudo pris qui ne fonctionne pas, l'initiation d'un clavardage vers un membre de l'annuaire, la réduction de l'agent, la communication avec différents utilisateurs avec qui la communication est initiée, la déconnexion avec des communications lancées, le changement de pseudo pendant une session de clavardage, l'envoi de message vers un utilisateur qui a changé de pseudo qui ne fonctionne pas directement, l'affichage de l'historique qui fonctionne malgré le changement de pseudo, etc...

En somme, notre application a pu passer une très grande batterie de test qui respecte la totalité du cahier des charges. Néanmoins, d'éventuels bugs peuvent apparaître et nécessite le redémarrage de l'application mais elle fonctionne la grande majorité du temps.

3 Installation et utilisation de l'application de clavardage

Nous allons maintenant nous intéresser à l'aspect pratique de notre application avec la manière très simple de l'installer sur une machine (dans notre cas sous Linux sur le réseau de L'INSA avec par exemple une quelconque machine du GEI) ainsi que son fonctionnement très basique.

3.1 Procédure d'installation de l'application avec l'exemple du PC Linux sur le réseau de l'INSA

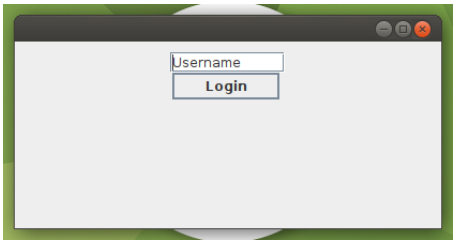
L'installation de l'application est très simple ; il suffit de lancer eclipse jee et de choisir l'option qui permet de récupérer un projet depuis un lien git (git clone). Depuis cette option, il faut indiquer l'url de notre repo git : https://github.com/posornio/4aJava.git et choisir la branch main. Toutes les dépendances et extensions vont se télécharger d'elles-même et l'application sera prête à l'utilisation.

Pour cela il suffit de "Run as Java application" le fichier java ActivityLogin.java et l'application est lancée et utilisable telle quelle.

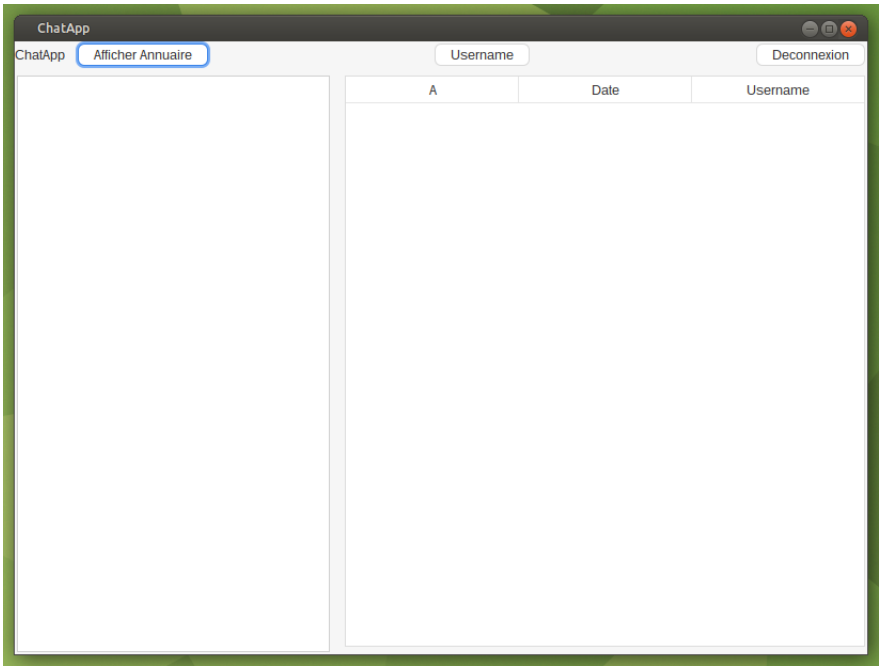
3.2 Manuel d'utilisation de l'application

On considère que l'installation est complétée et que l'application est lancée comme expliqué précédemment.

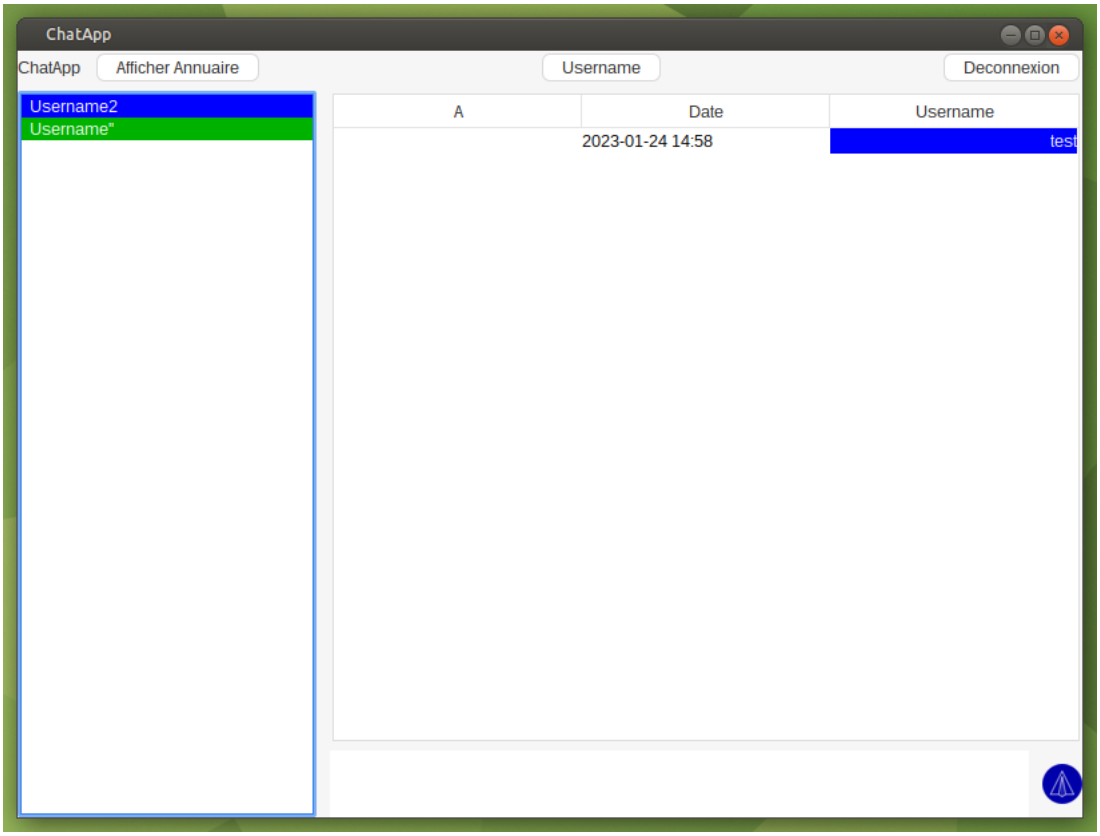
On peut donc débuter par la connexion une fenêtre telle que celle-ci va apparaître :



On peut alors rentrer le pseudonyme que l'on souhaite utiliser et, si il est disponible, nous allons pouvoir accéder au coeur de l'application, le MainForm :



D'ici, on a accès à plusieurs fonctions, tout en haut à gauche d'abord on peut accéder à l'annuaire des personnes connectées sous forme d'une liste.
Pour communiquer avec n'importe lequel d'entres eux, il suffit de cliquer dessus et nous sommes renvoyés au Mainform avec cette fois à droite de l'écran une connexion établie sur laquelle on peut cliquer pour faire apparaître au milieu l'historique de clavardage avec l'utilisateur choisi ainsi que de débloquer la possibilité de lui envoyer et de recevoir des messages. Cela se présente alors tel quel :



La conversation sélectionnée aura un fond bleu, les conversations avec des messages non lus auront un fond vert et les autres auront un fond blanc. Aussi, nous pouvons changer de pseudo avec le bouton dont le nom est votre pseudo au milieu en haut du MainForm. Il est également possible de changer le thème pour passer en Dark/Light mode grâce à un simple clic. Enfin on peut se déconnecter en un simple clic avec le bouton Déconnexion qui ferme l'application. Pour se reconnecter il suffit de suivre la dernière étape de la procédure d'installation de l'application.

Alors notre historique sera bien sauvegardé mais les clavardages seront tous fermés. Les autres utilisateurs sont alors bien évidemment mis au courant de notre déconnexion, nous n'apparaissons plus dans leur annuaire.

Conclusion

En conclusion, notre application répond au cahier des charges mais nous avons du pour cela revenir à plusieurs reprises sur les diagrammes que nous avions fait avant le début du projet. Nous avons eu quelques problèmes auxquels nous ne nous attendions pas du tout qui nous ont retardés mais nous pouvons rendre aujourd'hui une application à la hauteur de ce que nous désirions fournir.