Zhong Brandon
Vendeville Léo

4IR-B1

SYSTÈME DE CLAVARDAGE DISTRIBUE INTERACTIF MULTI-UTILISATEUR TEMPS REEL

Rapport de conception

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

27 Janvier 2023

Table des matières

Table des matières **2**

I. Introduction **3**

II. Diagrammes d'utilisation **4**

III. Diagrammes de Séquence **5**

1. Connexion 5
2. Echange 6
3. Déconnexion 8

IV. Diagrammes de Classe **9**

V. Les méthodes et outils d'implémentations **10**

1. La base de donnée 10
2. Les méthodes réseaux 10
3. Les interfaces graphiques 11
4. Diagramme de déploiement 11

VI. Procédure de tests **12**

VII. Manuel d'utilisation **12**

1. Connexion 12
2. Echange de messages 12
3. Changement de pseudo 12
4. Déconnexion 12

I. Introduction

Dans le cadre de l'UF COO-POO, nous avons reçu un projet de réalisation d'un système de chat afin de pouvoir mobiliser nos connaissances. Dans la suite du rapport, nous allons vous expliciter notre manière d'avoir conçu et développé ce système de chat, en respectant le cahier des charges fourni.

Dans un premier temps, nous vous montrerons les différents diagrammes réalisés lors de la conception du projet, c'est-à-dire la traduction des besoins du client pour mieux découper le projet.

Ensuite, après avoir fait une version initiale des diagrammes, nous avons commencé à implémenter les différentes fonctions qu'on a repéré dans le cahier des charges.

Nous présenterons aussi dans cette section les différents outils de développement utilisés pour ce système de clavardage.

II. Diagrammes d’utilisation

Nous avons commencé par modéliser un diagramme d’utilisation en résumant toutes les exigences décrites dans le cahier des charges du projet de chat.

Dans ce diagramme, nous avons pu distinguer plusieurs grandes fonctionnalités qui regroupent différentes exigences: la connexion, l’échange, la déconnexion.

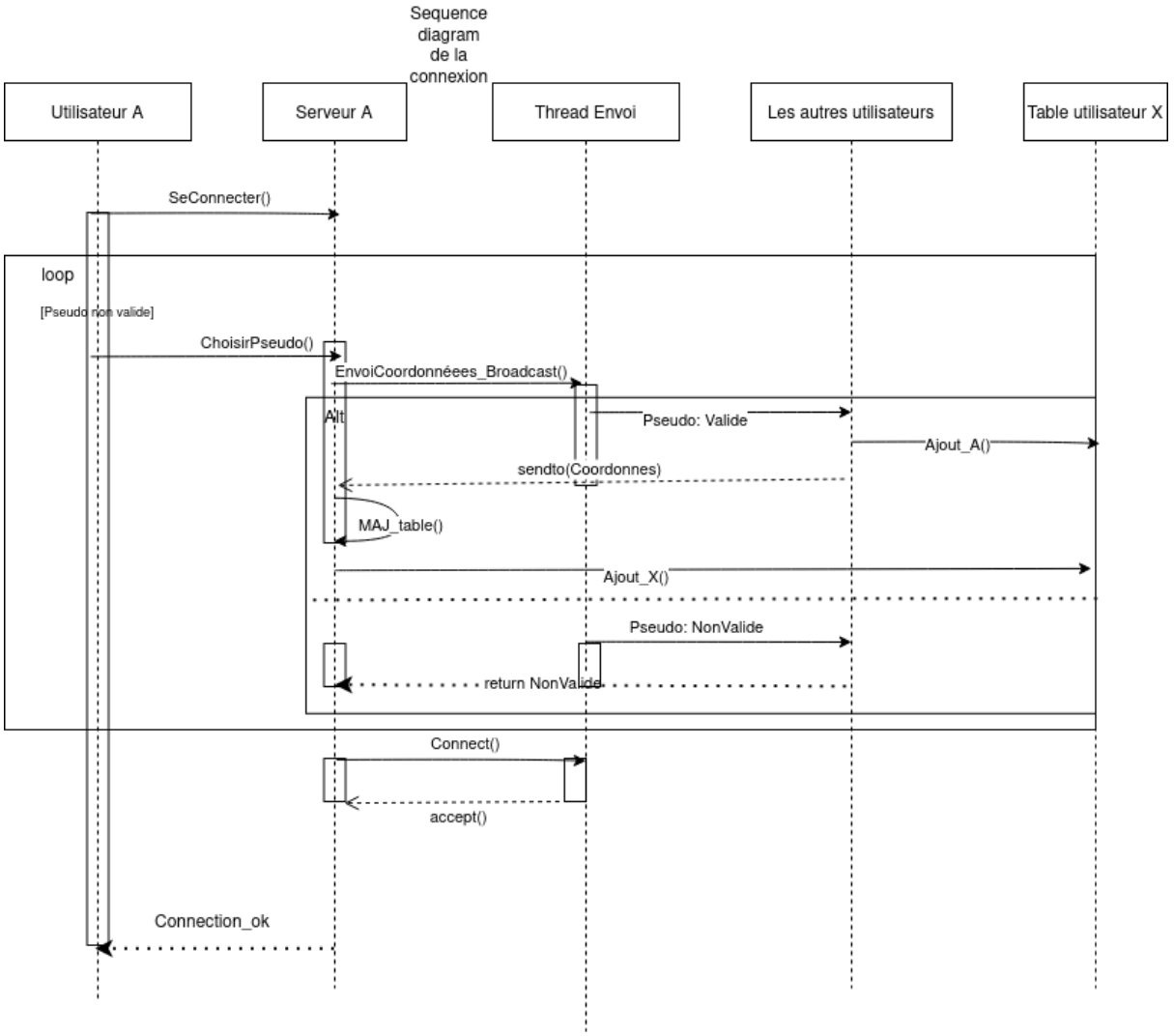
La connexion nous permettra de choisir un pseudo et de rentrer dans le réseau pour échanger.

L’échange qui va permettre aux utilisateurs de partager leurs idées.

La déconnexion qui va annoncer qu’un utilisateur part.

III. Diagrammes de Séquence

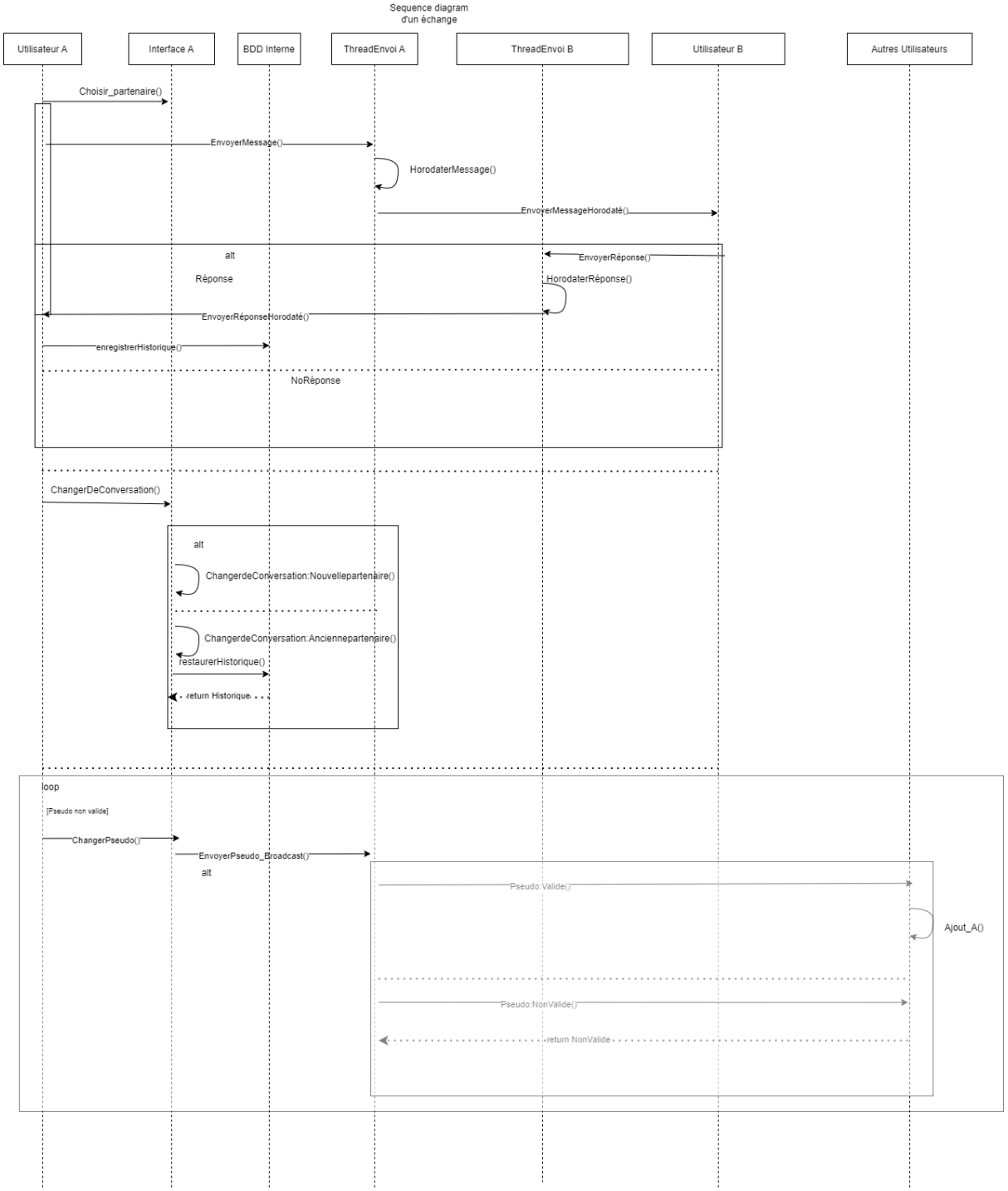
1. Connexion



Pour se connecter, un utilisateur doit rentrer un pseudo valide (il est demandé en boucle tant qu'il ne l'est pas). Lorsque ceci est fait, chaque utilisateur met à jour sa table d'identifiants en ajoutant le nouvel entrant, et le nouvel entrant met à jour la sienne en enregistrant tous ceux qui lui ont envoyé leur coordonnées.

Tout ceci est fait en UDP puisqu'il n'y a pas encore de connexion. Puis l'utilisateur se connecte en TCP à n'importe quel autre utilisateur pour lancer / reprendre une session de clavardage.

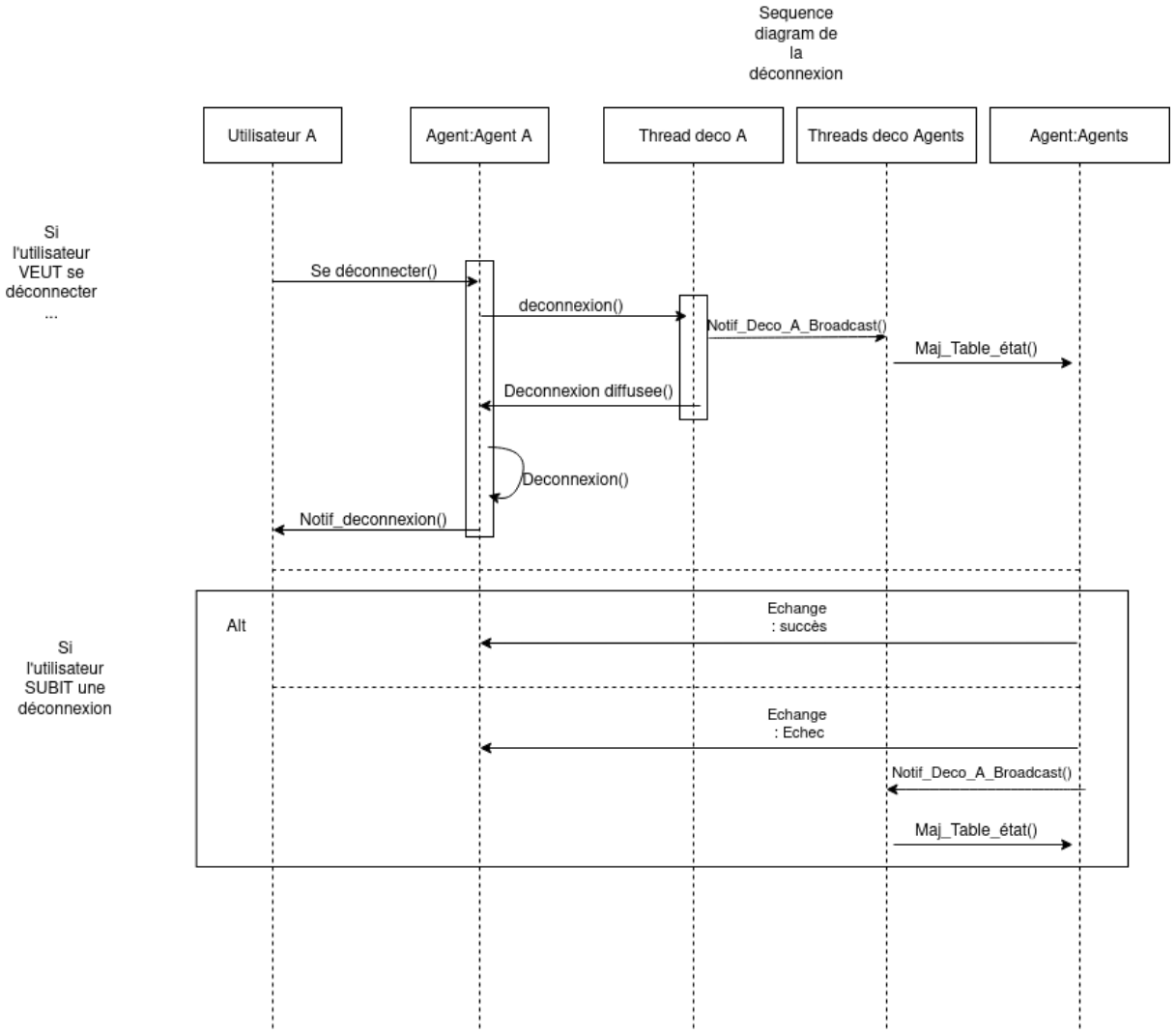
2. Echange



- La méthode Choisir_Partenaire() qui fait l'utilisateur répond à l'attente de la fonctionnalité d'utilisation du cahier des charges [CdC-Bs-9] en demandant de démarrer une session de chat avec un utilisateur du système qu'il choisira dans la liste des utilisateurs.
- Les méthodes HorodaterMessage() et HorodaterReponse() qui sont faits par le ThreadEnvoi répondent l'attente de la fonctionnalité d'utilisation du cahier des charges [CdC-Bs-1] en demandant de horodater des messages.

- La méthode ChangerDeConversation() qui est faite par l'utilisateur répond à l'attente de la fonctionnalité d'utilisation du cahier des charges [CdC-Bs-14] en demandant d' afficher l'historique des messages quand l'utilisateur démarre une nouvelle session de chat avec quelqu'un avec qui il avait déjà parlé.
- La méthode ChangerPseudo() qui est faite par l'utilisateur répond à l'attente de la fonctionnalité d'utilisation du cahier des charges [CdC-Bs-17] en demandant de notifier aux autres utilisateurs le changement du pseudonyme et aussi [CdC-Bs-18] en demandant de ne pas finir la session de chat en cours en changeant de pseudonyme.

3. Déconnexion

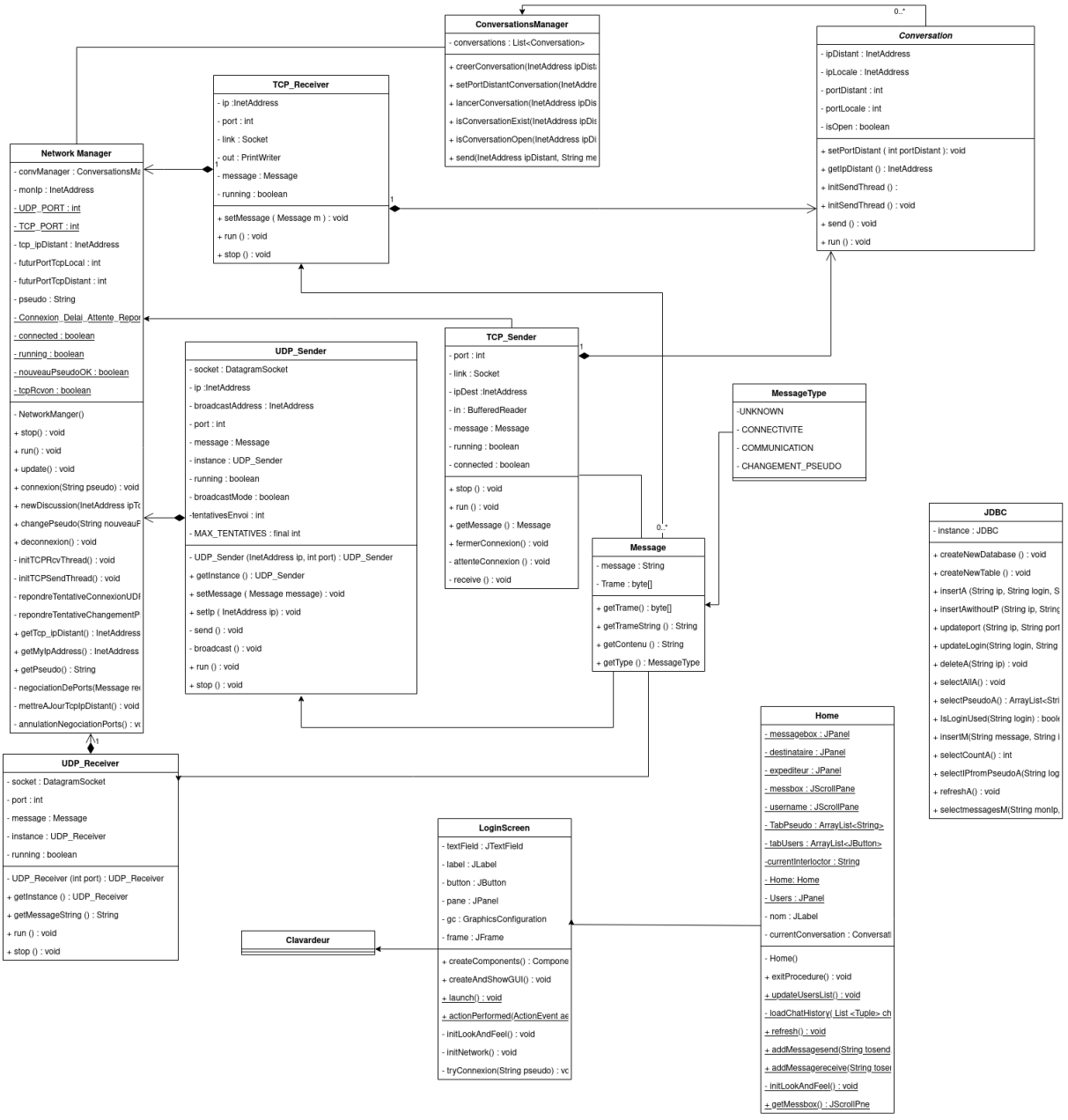


Pour la déconnexion, nous avons distingué 2 cas distincts :

- Le cas où l'utilisateur **choisit** une déconnexion
  il en informe par un broadcast tous les autres utilisateurs et se déconnecte.

- Le cas où l'utilisateur **subit** une déconnexion
  il finit par être découvert inactif par un agent qui notifiera tous les autres de sa déconnexion.

IV. Diagrammes de Classe



Nous avons décidé de faire une classe *Utilisateur* qui stockera un pseudonyme définissable à la connexion et modifiable.

Nous avons géré les messages avec deux threads d'envoi et de réceptions eux-mêmes liés chacun à un TCP handler et un UDP handler. De cette manière, on pourra envoyer / recevoir des messages par TCP (lors d'une session) ou UDP (pour la connexion) à tout moment.

La base de donnée est représentée par un Historique auquel on ajoute constamment les messages envoyés et reçus. Cet historique peut être enregistré (en local) et restauré.

V. Les méthodes et outils d'implémentations

Dans cette partie, nous allons vous présenter les outils d'implémentations ainsi que les méthodes utilisées pour répondre au projet.

1. La base de donnée

Nous avons décidé que les utilisateurs auront tous une base de données individuelle pour enregistrer les informations nécessaires : la table d'annuaire, la table de messages.

Dans la table d'annuaire, on insérera le pseudo associé au IP, et une colonne port pour la communication tcp.

La table messages, quant à lui, donnera les deux IPs en question, le contenu du message et la date.

Pour gérer cette base de données, nous avons choisi le Sqlite3, connecté en utilisant JAVA Eclipse et le driver JDBC.

Après avoir connecté l'utilisateur avec la BDD, nous avons aussi implémenté des fonctions pour manipuler ces tableaux lors de la connexion, échange et déconnexion.

Annuaire		
IP	Login	Port

Messages				
id	Message	ip1	ip2	Date

2. Les méthodes réseaux

Pour se connecter au réseau et échanger des messages, nous avons utilisé les protocoles suivants : TCP et UDP.

Nous avons utilisé UDP pour plusieurs cas :

- la première connexion, pour que le nouvel utilisateur s'annonce à tous les autres, récupère leur pseudo et leur demande au passage confirmation pour l'utilisation de son pseudo. Il y a un envoi broadcast puis des réponses unicast sur l'adresse IP de l'envoyeur.
- Le changement de pseudo : l'utilisateur qui veut changer de pseudo demande aux autres (de manière similaire à la connexion) si son pseudo est valide pour eux et pour les prévenir du changement.

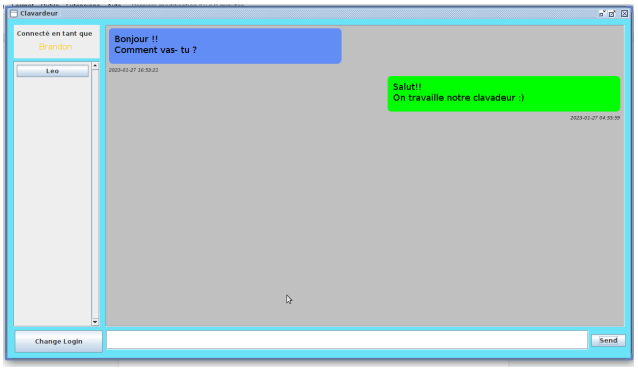
- la déconnexion, pour laquelle l'utilisateur envoie un broadcast qui prévient les autres de sa déconnexion avant de fermer sa session. Il n'y a pas de réponse attendue.

TCP est utilisé pour :
- la négociation de ports pour débuter une conversation
- chaque thread de discussion.

Pour cela, nous avons utilisé la librairie dans java.net NetworkInterface permettant de gérer des paquets réseaux, comme nous l'avions appris en cours de programmation java avancée.

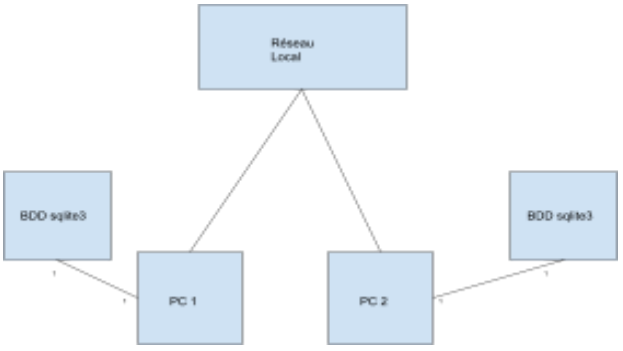
3. Les interfaces graphiques

Nous devons aussi développer des interfaces graphiques pour que l'utilisateur puisse manipuler les fonctions nécessaires, tels que l'interface de connexion et l'interface de chat. Pour le design de ces interfaces, nous avons pris JAVA SWING. A l'aide des WindowsBuilders, nous avons pu assembler les différents composants pour former une interface de chat opérationnel.



4. Diagramme de déploiement

Pour notre application, le déploiement se fait ainsi :



Chaque utilisateur se connecte sur un PC avec un IP fixe. Les PCs auront leur propre BDD qui va être complétée ou supprimée selon la situation. Et pour échanger, les PCs doivent passer par le réseau local.

VI. Procédure de tests

Nous avons réalisé quelques tests unitaires concernant les classes qui étaient testables automatiquement, comme la classe Message. Nous avons eu beaucoup de mal à tester les autres tests autrement qu'à la main, étant donné que toutes les classes sont reliées au networkManager et qu'il faut avoir 2 machines pour le faire fonctionner. Les tests sont valides cependant. Nous ne sommes pas non plus parvenus à les faire exécuter par jenkins, nous avons un bug récurrent sur cette plateforme depuis le début et nous ne sommes pas parvenus à nous en défaire, rendant les tests automatiques à chaque push impossibles.

Malgré cela, nous avons choisi d'agir méthodiquement pour tester notre application : à chaque push, nous testions à deux machines si l'application fonctionnait selon les fichiers que nous avions touchés. A la fin de grosses modifications dans le code, nous refaisions également des tests plus globaux pour vérifier d'éventuels effets de bord. Nous effectuions des tests avec des valeurs différentes, testions des situations peu communes (caractères spéciaux, envoi de longs textes, envois répétitifs…).

Nous avons au préalable testé chacune de nos classes séparément dans des "main" que nous avions mis dans ces classes-là : par exemple, dans JDBC (concernant la bdd) ou encore Home, LoginScreen… Puis pour tester le réseau, nous nous sommes appuyés sur une version simplifiée en console d'une application pour pouvoir tester des échanges réseaux ; cette version se trouve dans la classe ClavardeurTest.

VII. Manuel d'utilisation

1. Connexion

Pour se connecter, vous devez choisir un pseudo valide:

- Pas de caractères spéciales
- Disponible

2. Echange de messages

Un échange se réalise de la manière suivante:

- Choisir un destinataire
- Taper votre texte
- Envoyer votre message
- Recevoir des messages entre temps

3. Changement de pseudo

Pour effectuer un changement de pseudo:

- Cliquer sur le bouton change
- Choisir un nouveau pseudo valide
- Valider votre pseudo

4. Déconnexion

Il suffit de fermer l'application, les autres seront notifiés de la déconnexion.