Rapport Projet POO

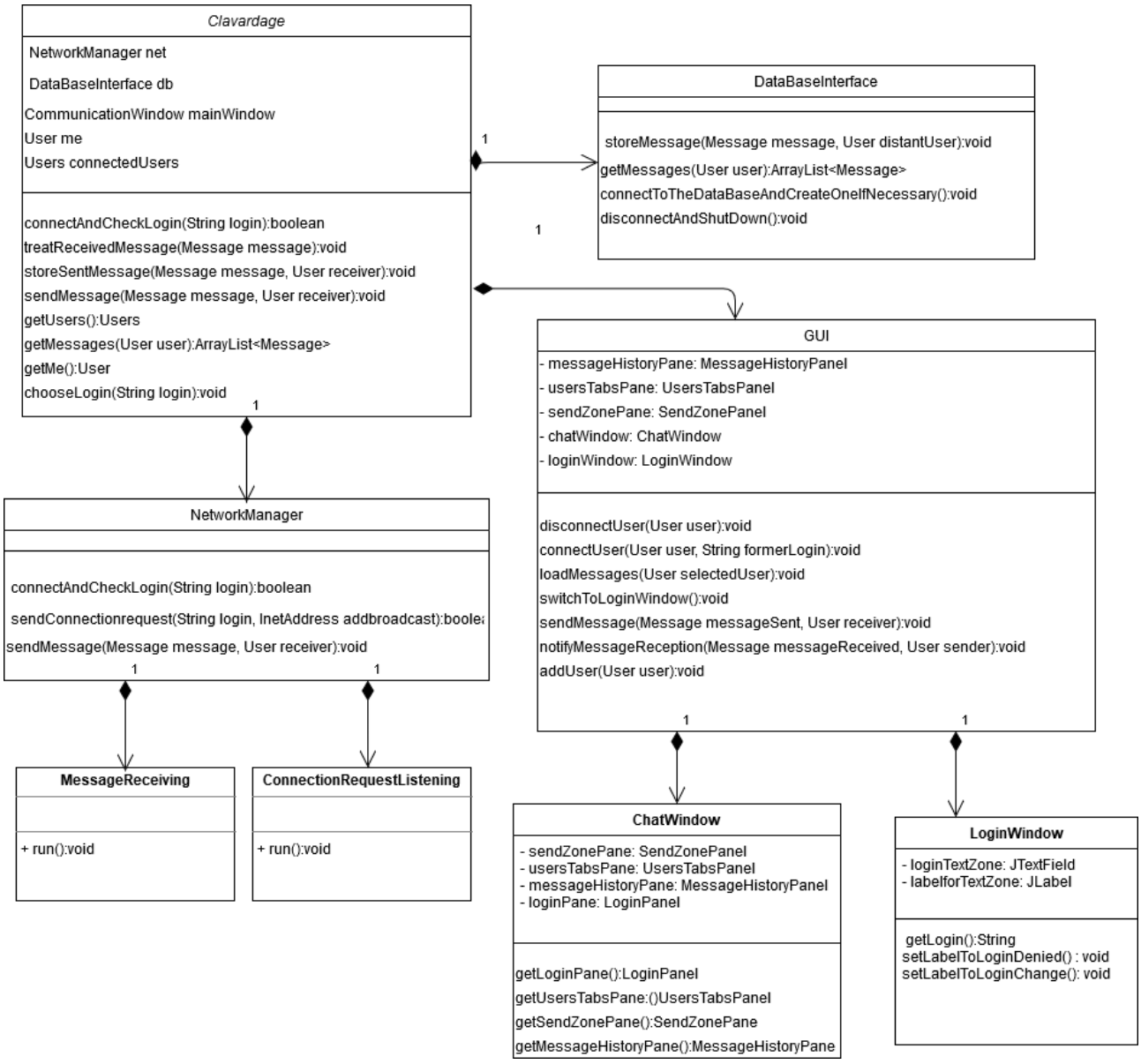
Durant ce projet, nous avons développé un système de chat qui respecte un cahier des charges précis.
Les cas d'utilisations traités sont résumés dans le diagramme des cas d'utilisation ci-dessous.

I- Différents choix de conception

1) Respect du MVC

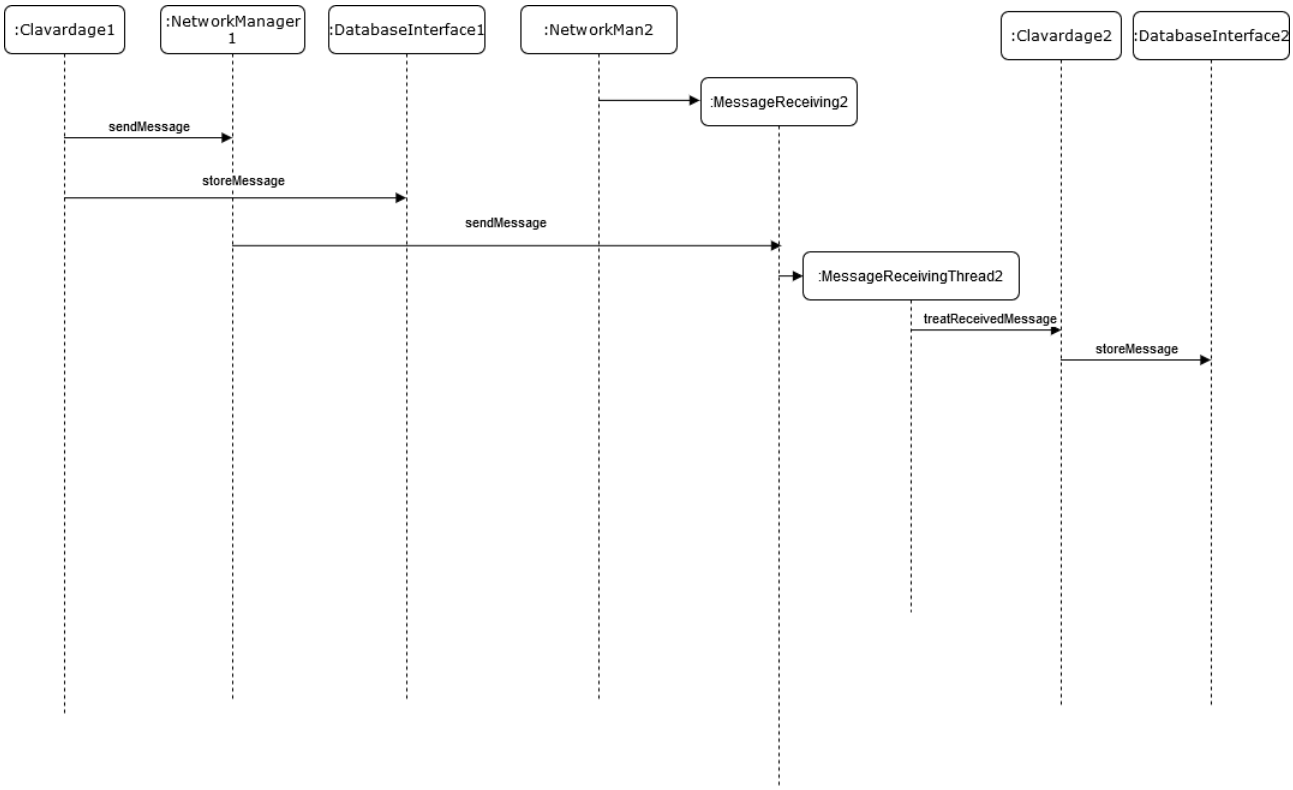
Dans l'optique de respecter la spécification MVC, notre projet se compose de trois classes principales chacune implémentant un module de la MVC :
-Clavardage pour le Controller
-DataBaseInterface pour le Model
-GUI pour la vue

La dernière classe principale est le NetworkManager qui permet les communications réseaux entre les utilisateurs. Les interactions entre ces quatre classes sont résumées par le diagramme de classe ci-après. Pour plus de détails, vous pouvez également consulter la javadoc dans le dossier POO_Clavardage/doc.



2) Gestion du réseau

Pour la partie réseau, les demandes de connexion se font en UDP alors que les échanges de message sont réalisés en TCP. UDP a été choisi pour la connexion puisqu'il était plus facile de broadcaster les demandes en UDP plutôt que de créer des groupes multicast en TCP. De plus le risque de pertes étant minimes sur un réseau local, cela ne pose pas de problèmes à l'utilisation d'UDP.
Deux classes actives sont dédiées à la réception des messages entrants : MessageReceiving et ConnexionRequestListening. Ces deux classes sont en permanence à l'écoute de potentiels messages ou requêtes de connexion qui pourraient arriver. On est ainsi assuré de n'en louper aucune.
L'envoi d'un message est schématisé par le diagramme de séquence ci-dessous :



3) Base de données

Pour stocker les messages échangés, nous avons choisi d'utiliser une base de données Apache Derby. Nous nous sommes portés sur ce choix car c'est une base de données implémentée en Java et de petite taille. Nous avons préféré une base de données locale car cela nous semblait plus en accord avec le cahier des charges qui exigeait un chat décentralisé.

II- Manuel d'utilisation

1) Lancement du programme

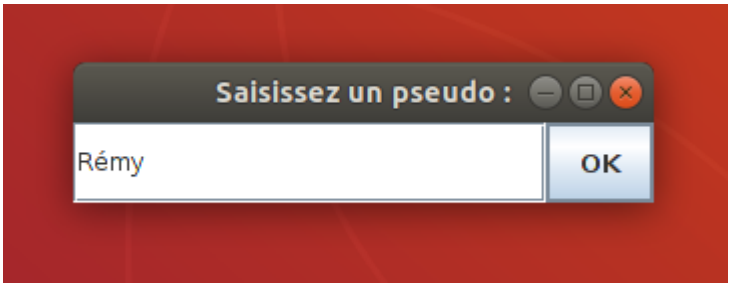
La manière la plus simple de procéder pour lancer le système de clavardage est d'utiliser l'exécutable jar.

Il suffit pour cela de lancer un terminal dans le dossier qui contient le fichier .jar et d'entrer la commande : java -jar projet_poo_clavardage .jar.

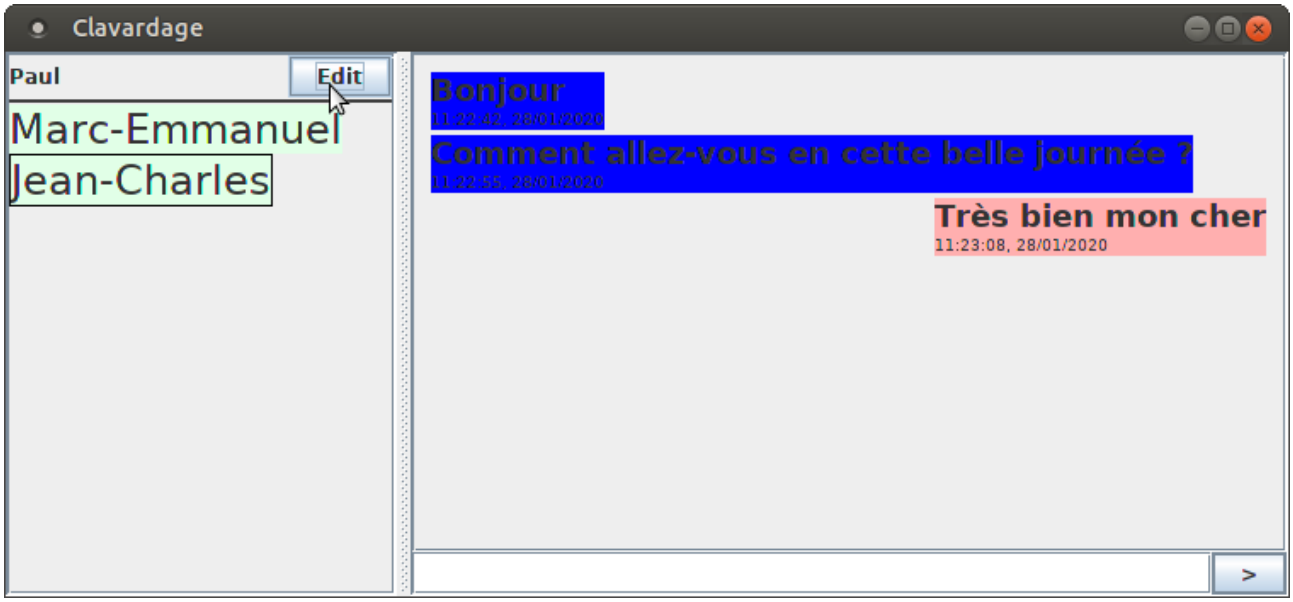
On peut également lancer le programme depuis l'interface d'Intel jdea mais il ne faut pas oublier d'inclure le path de la librairie derby pour la gestion de la base de données.

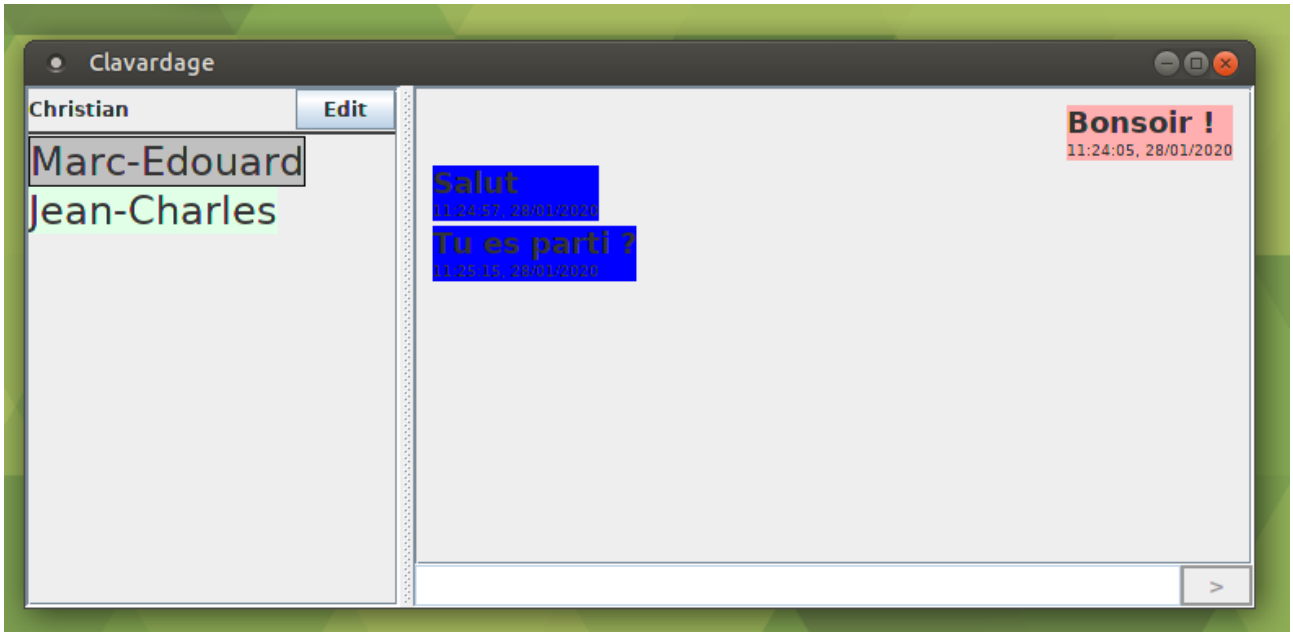
2) Utilisation du système de clavardage

Une fois la base de donnée initialisée, La fenêtre de connexion s'ouvre. Il vous faut alors entrer votre pseudo.

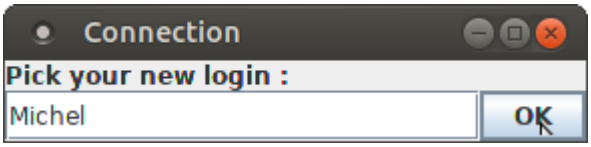


Si le pseudo est valide, le fenêtre de chat s'ouvre. Vous pouvez alors entamer une discussion avec tous les utilisateurs connectés (affichés en vert sur l'interface). Si vous le souhaiter, vous pouvez également consulter l'historique des messages échangés avec les autres utilisateurs, qu'ils soient connectés ou non.





Enfin il vous est possible de changer votre pseudo en choisissant l'option edit dans la fenêtre de chat.



III- Réalisation des tests

Nos premiers tests ont visé à valider l'implémentation des quatre classes principales de notre programme.

1) Test du NetworkManager

Pour tester le réseau, nous avons vérifié que nous pouvions échanger des messages en TCP et en UDP. Pour cela, nous essayions de transmettre un message d'un PC vers un autre PC situé sur le même réseau. Nous avons également testé le fonctionnement du broadcast en vérifiant que le message broadcasté atteignait au moins deux PC sur le réseau.

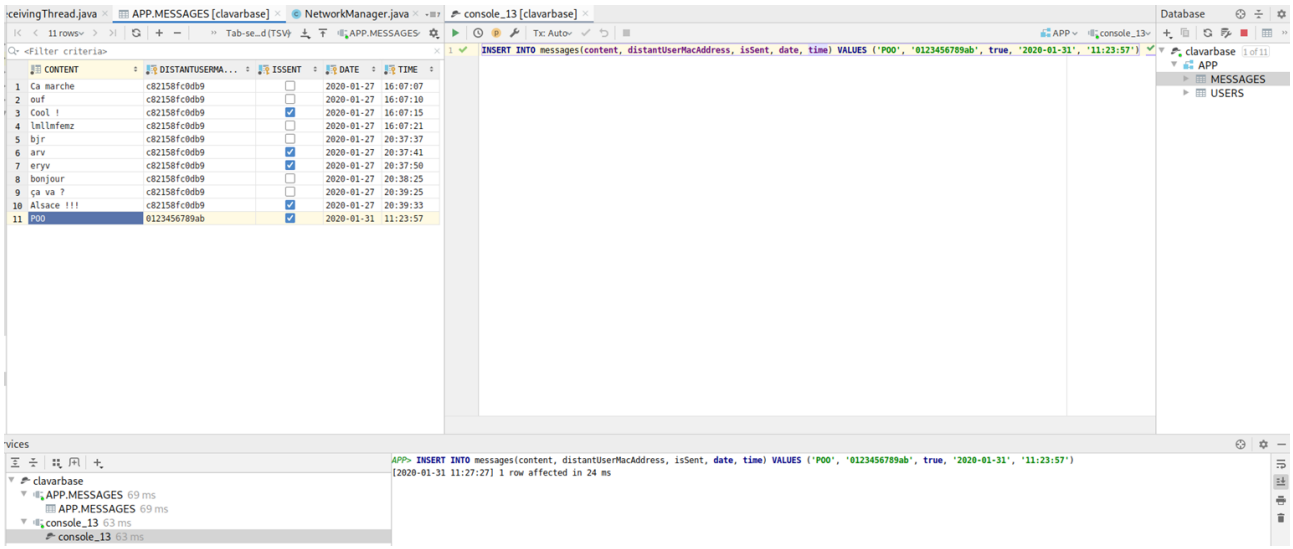
2) Test du GUI

Pour nous assurer que notre interface graphique fonctionnait bien, nous avons vérifié que sa visualisation était conforme à ce que nous attendions : la liste des utilisateurs à gauche, l'historique des messages en haut à droite et la zone de texte en bas à droite. Nous avons également testé qu'il était possible d'afficher un message dans le système de chat en l'entrant dans la zone de texte.

3) Test de la base de données

Pour le test de la base de données, nous avons entré artificiellement des messages entre deux utilisateurs dans la base et nous avons vérifié que nous pouvions les récupérer. De plus nous avons

utilisé l'outil de visualisation de base de données d'Intel JDEA qui permet à la fois d'entrer des commandes SQL mais aussi de regarder les entrées dans les tables (voir image ci-dessous).
Enfin vu que l'interface graphique était opérationnelle, nous avons pu lier les deux et constater qu'en cliquant sur un utilisateur sur l'interface graphique, on arrivait bien à charger l'historique de conversation et à l'afficher dans la zone prévue à cet effet.



4) Test final

Enfin une fois toute l'application développée, nous avons testé son bon fonctionnement, en connectant trois utilisateurs sur notre application. Les captures d'écran du manuel d'utilisation sont issues de cette session de test. Nous nous sommes assurés que les messages parvenaient à leur destinataire quel que soit le poste. Nous avons vérifié que le changement de pseudo le changement de pseudo fonctionnait correctement.