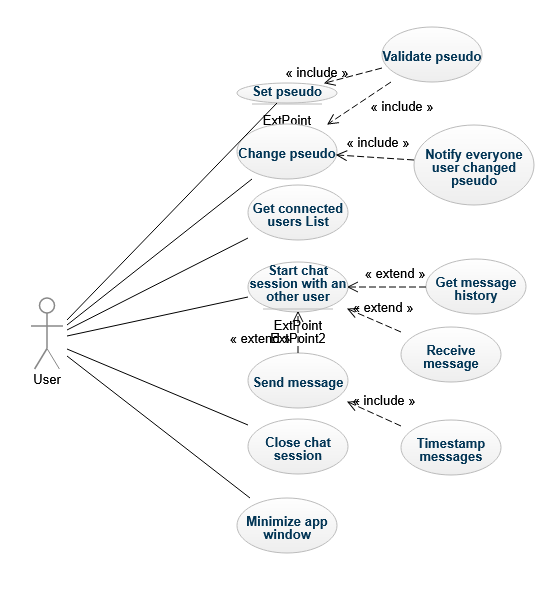
## **Rapport projet Chat-App**

Clément VIGAND

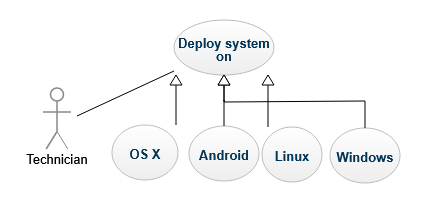
Antoine ALAVERDOV

# Conception et diagrammes réalisés

## Diagramme des cas d’utilisation

 Pour réaliser les diagrammes de cas d’utilisation, nous nous sommes basés sur le cahier des charges fourni par le client. En analysant ses différentes demandes, nous avons modélisé les interactions possible entre les différents acteurs et le système

En ce qui concerne la partie utilisation de l’application : nous avons défini un acteur : l’user. C’est lui qui va agir directement sur le système, l’utilisateur va pouvoir se choisir un pseudo, le changer, voir la liste des autres utilisateurs connectés, démarrer une session de chat avec eux, leur envoyer des messages, en recevoir, fermer une session de chat, et minimiser la fenêtre de l’application.



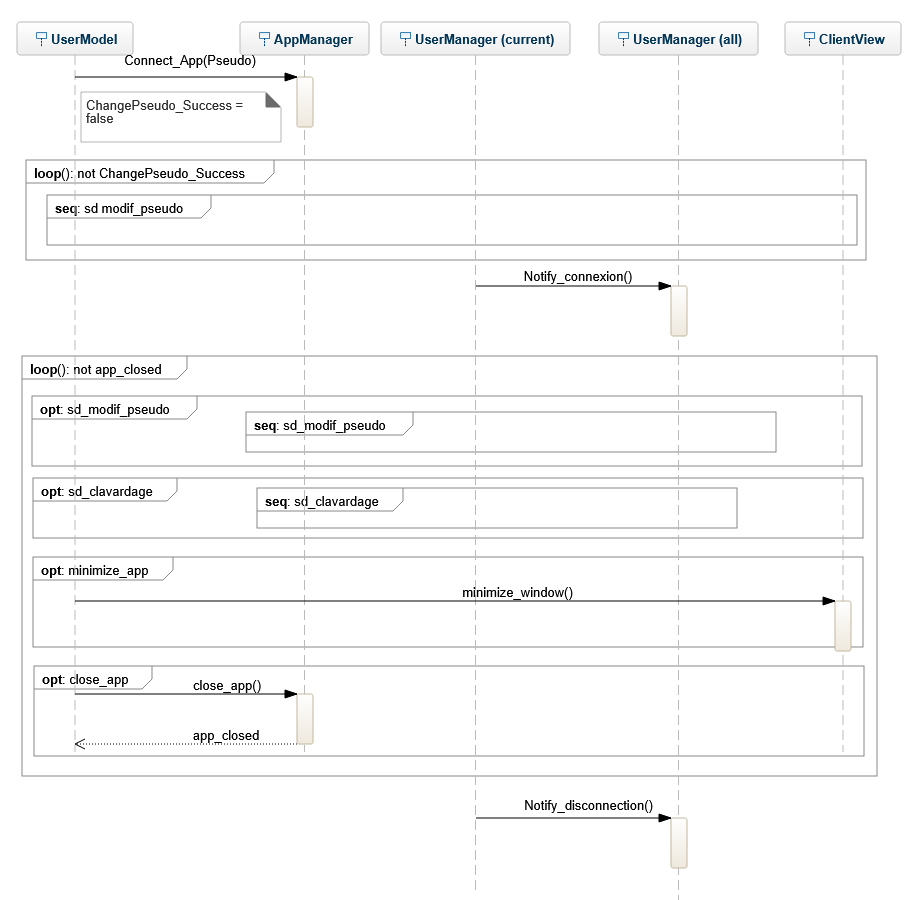
Pour la partie de l’installation de l’application sur les machines du réseau du client, nous avons défini un seul acteur : le technicien, qui doit pouvoir déployer l’application sur les machines du client.

## Diagramme de séquence

Les différents diagrammes qui suivent permettent une meilleure compréhension des échanges qui auront lieux entre les composants du système. Ainsi, il est plus facile de visualiser le comportement des différentes phases qui interviennent au cours du cycle de vie de l’application.

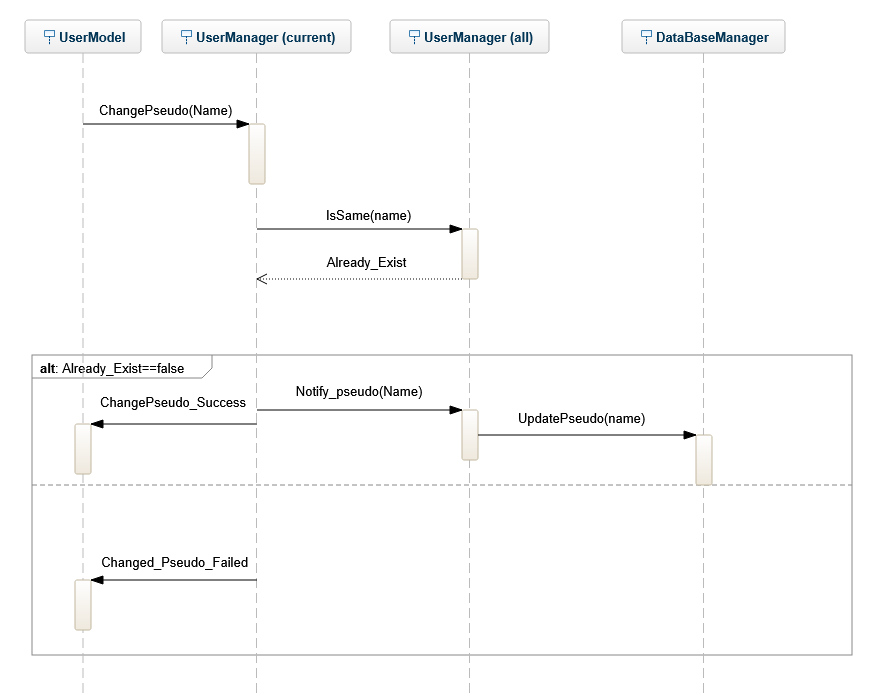
### Diagramme récapitulatif

Le diagramme ci-dessous récapitule les différentes fonctionnalités qui doivent être réalisées par l’application. Quand l’utilisateur va se connecter il va choisir un pseudo, on doit vérifier que celui-ci n’est pas déjà utilisé. Si c’est le cas, alors on notifie aux autres usagers notre connexion. Tant qu’on n’arrête pas l’application, on doit pouvoir changer notre pseudo, ouvrir des session de chat avec les autres utilisateurs, minimiser la fenêtre et enfin pouvoir fermer l’application.



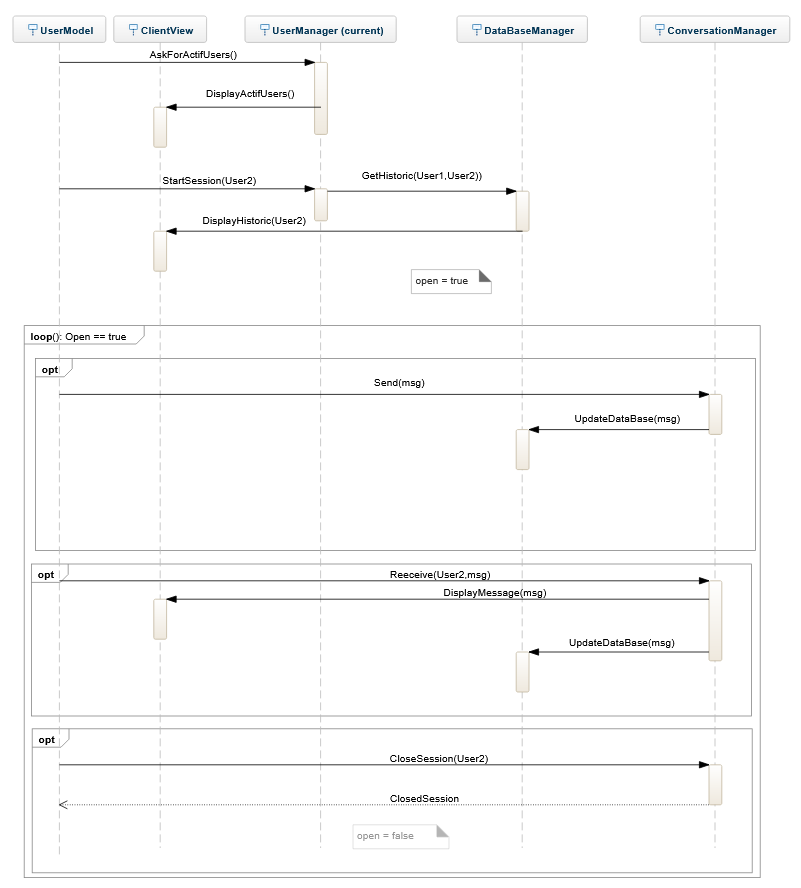
### Phase de changement de pseudo

Un utilisateur doit pouvoir changer de pseudo quand il le souhaite. Quand il en fait la demande, il devra choisir un nouveau pseudo, le système vérifiera qu’il n’est pas déjà attribué a un autre utilisateur et si ce n’est pas le cas alors on modifie le pseudo actuel par le nouveau choisi et on notifie aux autres usagers ce changement, sinon on demande a l’utilisateur de rentrer un autre pseudo.



### Phase d’envoi de message

On récupère la liste des utilisateurs actifs, ça sera la liste de nos possibles destinataires. Lorsqu’un destinataire est sélectionné, on récupère d’abord l’historique de communication entre l’utilisateur et le destinataire. Ensuite, tant que la conversation est active, on a la possibilité d’envoyer et de recevoir des messages, et bien sur ensuite de fermer la conversation.



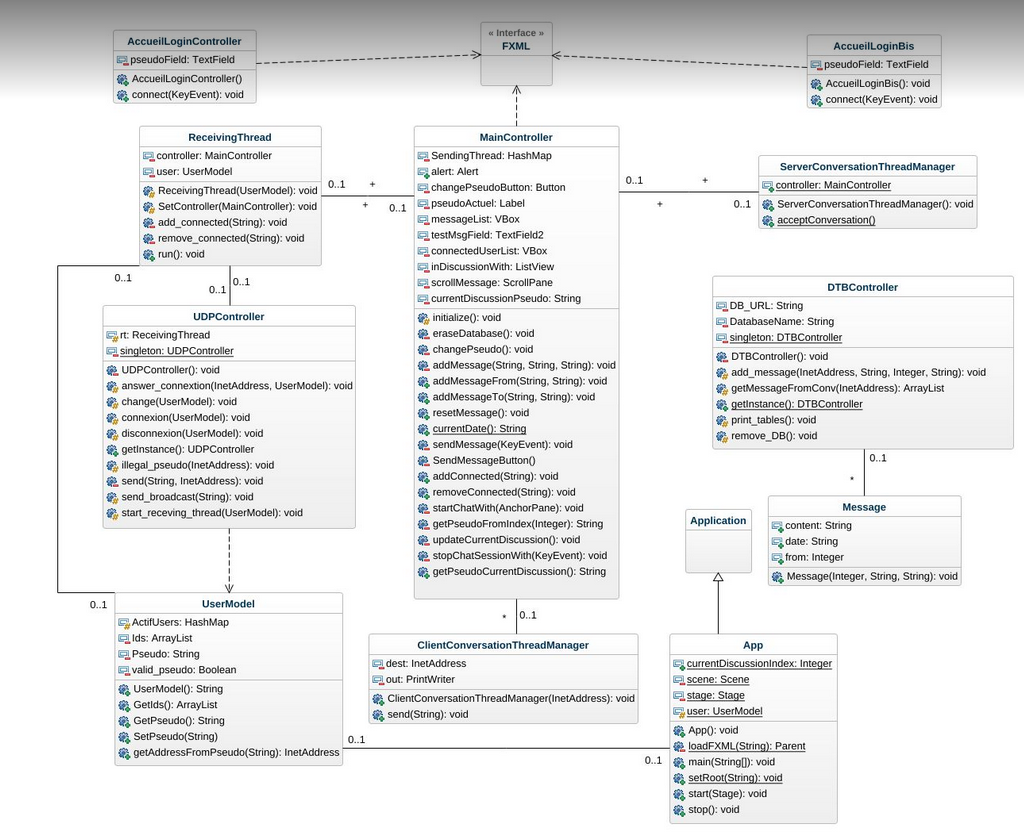
## Diagramme de classe

Les diagrammes de classes ont pour but de représenter les dépendances entre les différentes classes du projet. Dans notre diagramme nous avons adopté une architecture MVC (Modèle Vue Contrôleur). Nous avons mis deux diagramme de classe dans cette section, le premier correspond au diagramme de classe réalisé en amont du projet et le second correspond à la version finale du projet

### Première version

# Architecture du système et choix technologiques

### Version finale



# Architecture du système et choix technologiques

Pour l’implémentation du GUI, nous avons fait le choix d’utiliser JavaFX pour sa simplicité et sa facilité d’utilisation. Nous avons aussi utilisé l’application « SceneBuilder » pour réaliser les interfaces visibles par l’utilisateur.

Pour gérer les bases de données, nous avons opté pour la technologie SQLite pour les mêmes raison que précédemment : sa simplicité et sa facilité d’utilisation

# Procédure d’évaluation et de tests

Nous avons essayé de couvrir toutes les situations possibles lors de nos phases de tests, pour ce faire, nous sommes partis de la situation la plus simple et avons imaginé plusieurs scenarii qui pourraient causer des problèmes.

Nous avons donc testé le cas le plus simple : deux utilisateurs A et B se connectent, ils peuvent chacun voir que l’autre utilisateur est connecté et voir son pseudo. Les deux peuvent aussi s’envoyer des messages. Ensuite, on change le pseudo de A et on regarde si le pseudo a bien été modifié sur la vue de B. Maintenant on déconnecte A, B ou les deux, et on les reconnecte, et on peut voir que l’historique des messages échangés entre A et B est bien présent.

Maintenant à trois utilisateurs A, B et C. On les connecte et on vérifie qu’ils apparaissent tous comme étant en ligne. Les trois utilisateurs s’envoient des messages, changent de pseudo, se déconnectent, se reconnectent

# Procédure d’installation et de déploiement

Une fois que vous avez récupérer l'ensemble du projet depuis github, vous pouvez récupérer le fichier

Chat\_App-0.0.1-SNAPSHOT-jar-with-dependencies.jar

qui se situe dans

Chat-App/POO/Chat\_App/target.

Depuis un terminal, depuis le dossier où vous avez mis le fichier .jar ou bien depuis

Chat-App/POO/Chat\_App/target

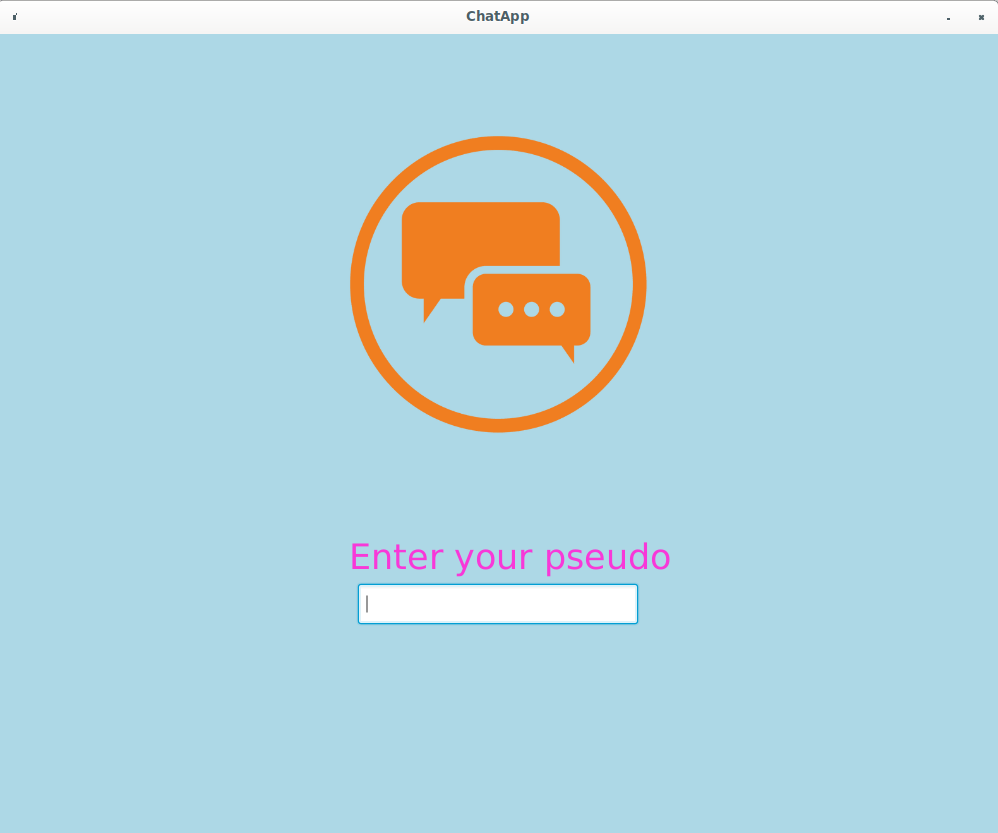
si vous ne l'avez pas déplacé, pour lancer l'application, lancez la commande :

java -jar Chat\_App-0.0.1-SNAPSHOT-jar-with-dependencies.jar

Vous pouvez maintenant utiliser l'application en choisissant un pseudo et communiquer avec d'autres pc sur le même réseau local ayant l'application lancée.

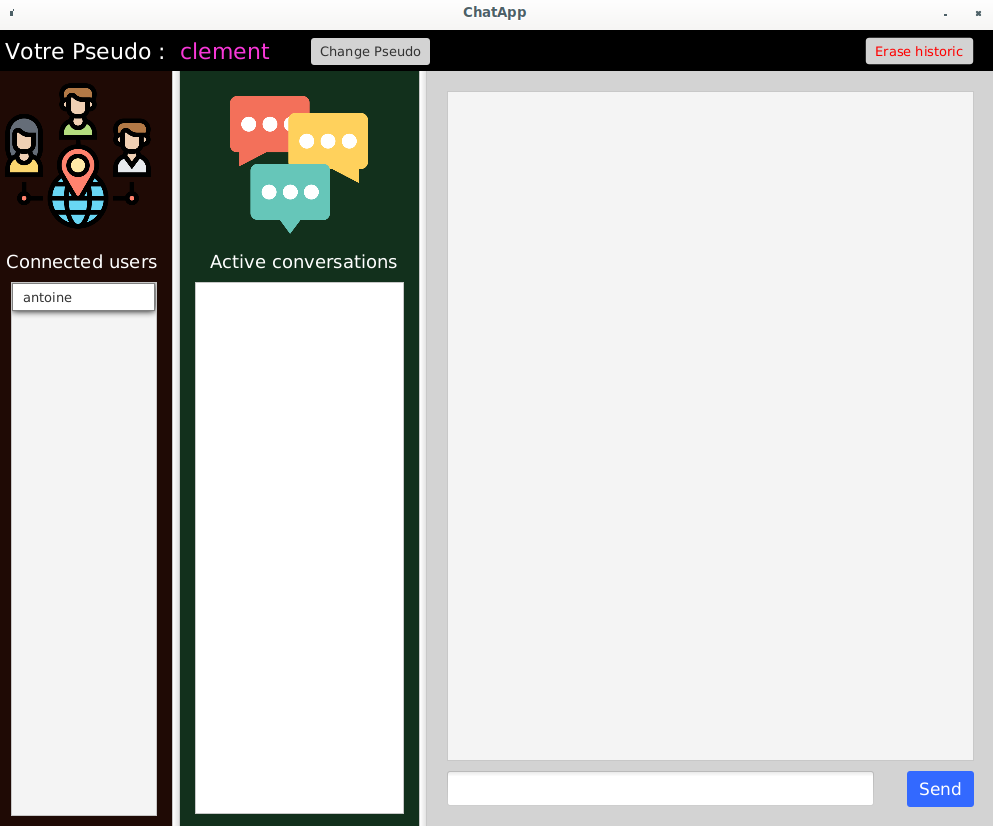
# Manuel d’utilisation

Lorsque vous lancez l’application, vous arrivez sur la page d’accueil qui vous demande d’écrire un pseudo.

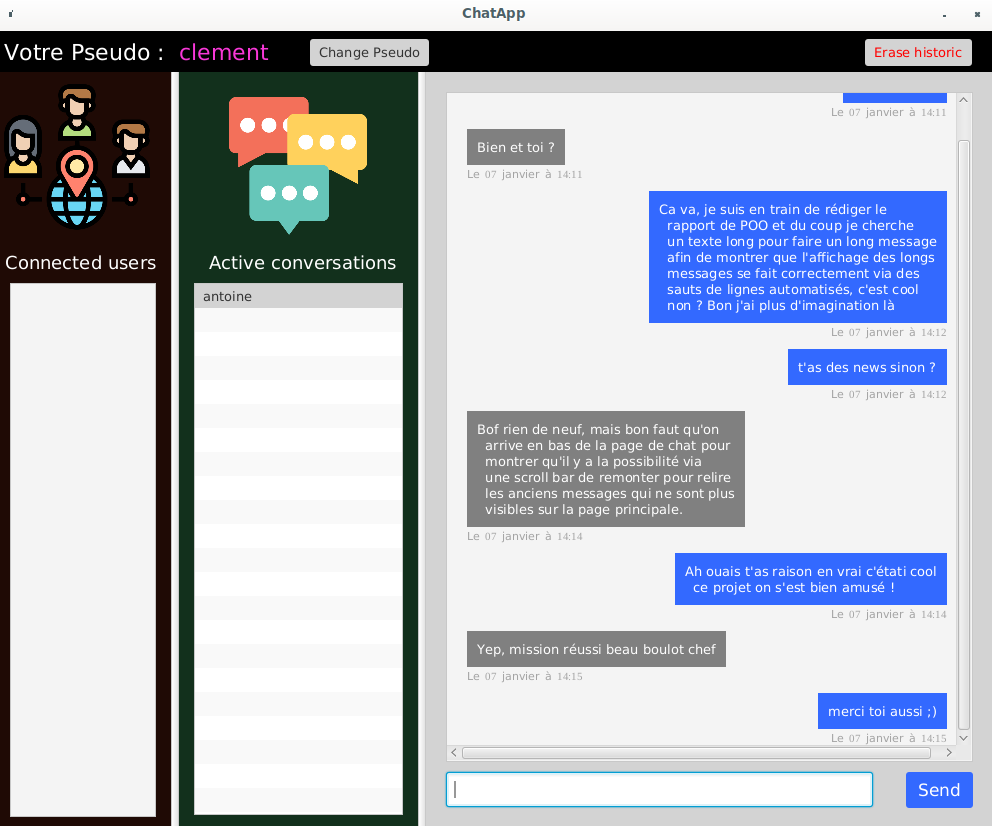


Vous devez choisir un pseudo qui sera celui que verront les autres utilisateurs connectés. Le pseudo ne doit contenir qu’un seul mot et doit être différent de celui des autres utilisateurs connectés. Dans le cas contraire vous serez à nouveau dirigé vers la même page d’accueil.

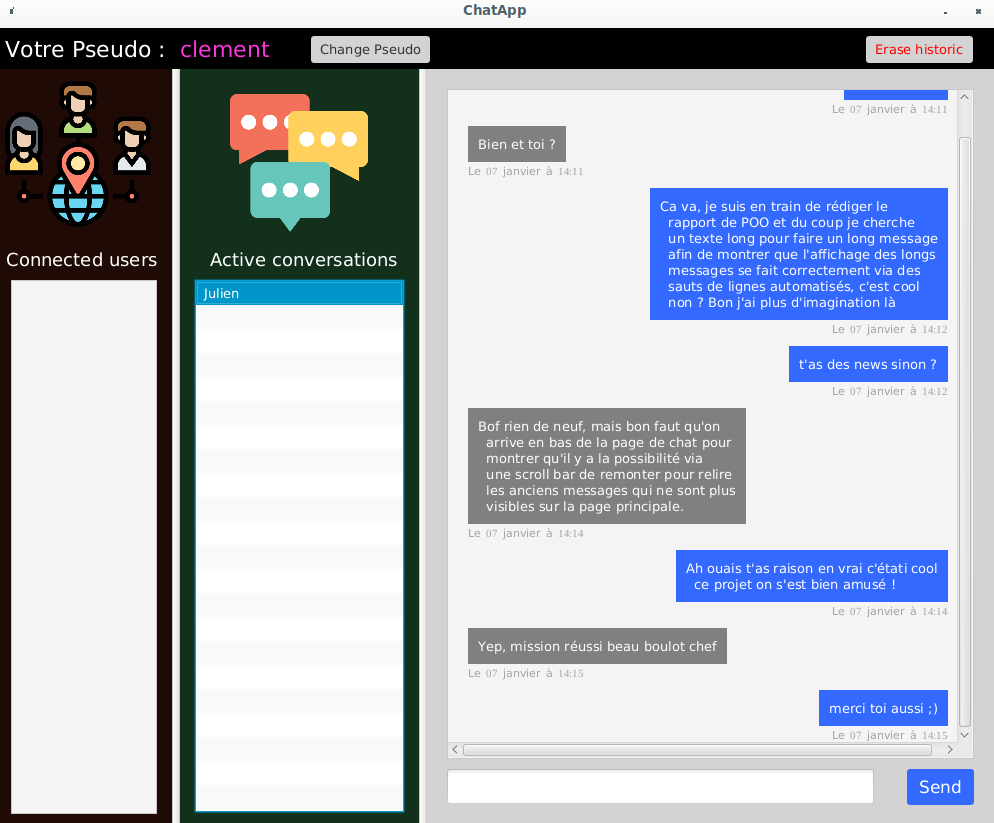
Une fois que vous avez appuyez sur entrer en ayant sélectionné un pseudo valide, vous arrivez sur la page principale de l’application avec sur la gauche une liste des utilisateurs connectés, à côté la liste des utilisateurs sur lesquels vous avez cliqué une fois car vous souhaitez discuter avec eux, votre pseudo en haut ainsi qu’un bouton permettant de changer de pseudo et un autre permettant de supprimer l’ensemble de l’historique de toutes les conversations.



Lorsque vous êtes sur cette page vous pouvez décider de sélectionner un utilisateur de la liste « Connected users » pour le faire passer dans la liste « Active Conversations ». De même vous pouvez sélectionner un utilisateur dans « Active Conversations », appuyer sur SUPPR, et le faire retourner dans « Connected users ». Cela permet lorsqu’il y a un grand nombre d’utilisateurs connectés de retrouver facilement et plus rapidement ceux à qui je suis en train de parler. Une fois un utilisateur dans la liste « Active Conversations », vous pouvez le sélectionner et l’application vous affiche l’ensemble des messages échangés avec cet utilisateur et vous pouvez maintenant utiliser la barre en bas pour écrire des messages et lui envoyer via le bouton Send ou la touche ENTRER.



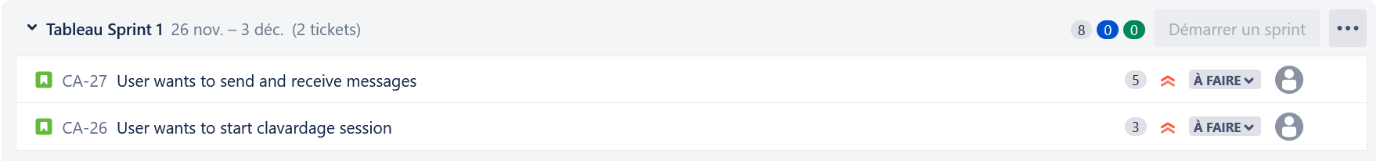
Lorsque vous décidez de changer de pseudo, les autres utilisateurs auront aussi la mise à jour de votre pseudo mais l’historique des messages restera le même car le pseudo ne défini pas un utilisateur mais bien son adresse IP. Ainsi vous pouvez changer de pseudo indéfiniment et garder les mêmes conversations.

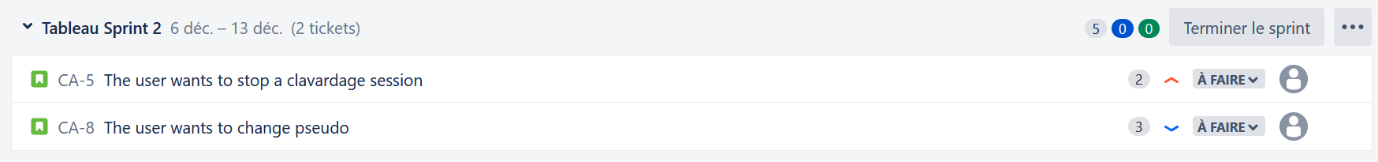


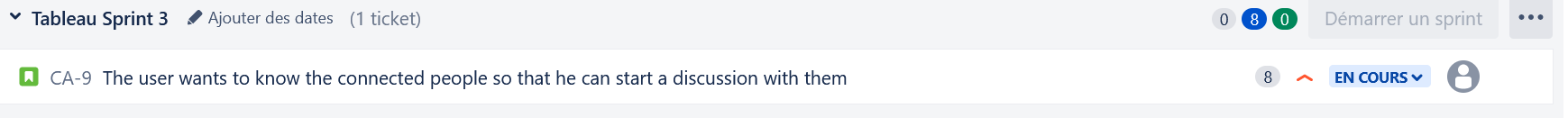
Lorsque vous vous déconnectez en appuyant sur la croix, les autres utilisateurs ne vous verront plus à l’écran et ne pourront plus vous parler.

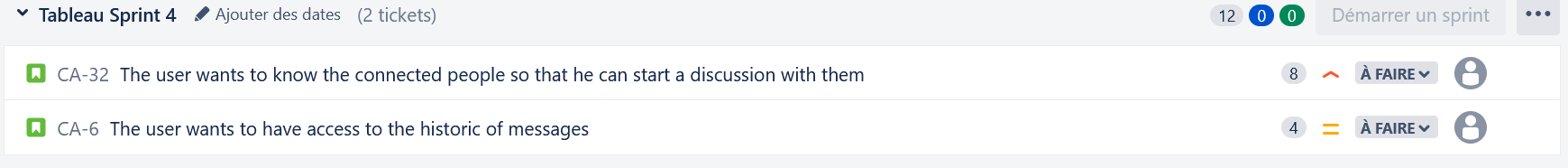
# Utilisation de Jira

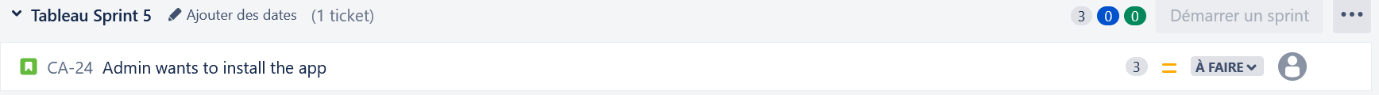
Afin de mieux gérer l’avancement de notre projet, nous avons utilisé Jira pour rédiger un Backlog composé de sprints d’une semaine afin de savoir en permanence ce que l’on devait développer. Ces sprints sont composés de user stories accompagnés de tâches pour le développeur. Nous avons préféré faire des sprints plus léger en début et fin de projet afin de pouvoir prendre du temps pour mieux appréhender le projet et répondre à des problèmes inattendus qui peuvent apparaître en toute fin de projet.











Tout au long du projet nous avons dû réarranger notre backlog en fonction de notre avancement et des besoins que nous avons identifié en cours de projet, par exemple la user story « the user wants to know the connected people so he can start a discussion with them » a pris plus de temps que prévu, nous avons donc dû la répartir sur deux sprints différents.

# Utilisation de git

Lors de ce projet, nous avons utilisé git pour faciliter le travail en parallèle ainsi que la gestion des versions grâce à l’historique des modifications. Afin de travailler en même temps sans conflits, nous avons créé une branche de travail chacun, au début il s’agissait d’une pour le front et une pour le back. Dès que nous avions assez avancer chacun de notre côté, nous avons faits des merges entre nos branches et créé chacun une nouvelle à partir du merge afin de continuer notre travail à partir de la version la plus avancée de notre projet. De temps en temps des bugs apparaissaient lors des merges, il nous a donc fallu les annulé grâce au revert pull request afin de corriger séparément les problèmes et effectuer à nouveau le merge.