

# Rapport POO

Amine Arfa, Sana Rafik



Amine Arfa, Sana Rafik  
4ème année IR groupe B

# Table des matières

- 1-Démarche conseillée installation
  - 1-1 Exécution via le fichier projet.jar
  - 1-2 Importation du projet avec Eclipse
- 2- Déploiement application
  - 2-1 Phase connexion
  - 2-2 Connexion à plusieurs personnes:
  - 2-3 fin session de clavardage
  - 2-4 Base de données:
  - 2-5 Problèmes rencontrés:
- 3- Tests :
- 4- Conclusion

# 1-Démarche conseillée installation

## 1-1 Exécution via le fichier projet.jar

La première façon d'exécuter le projet est d'utiliser le dossier execu, puis compiler le code .sql pour créer une table utilisateur ainsi mettre le lien avec la base de données, puis lancer serveur.jar ainsi client.jar autant de fois que l'on souhaite, sur le terminal il est conseillé d'utiliser la commande `java -jar client.jar` et puis avoir java 11 ou plus installé sur la machine.

## 1-2 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, git clone <https://github.com/aminearfa1/Clavardage>.

Nous avons développé le projet en Java 11 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 2 packages (db, default package qui contient serveur et client) ainsi que la librairie `mysql_connector_java` avec la version 8.0.28, pour éviter tout problème avec la base de données..

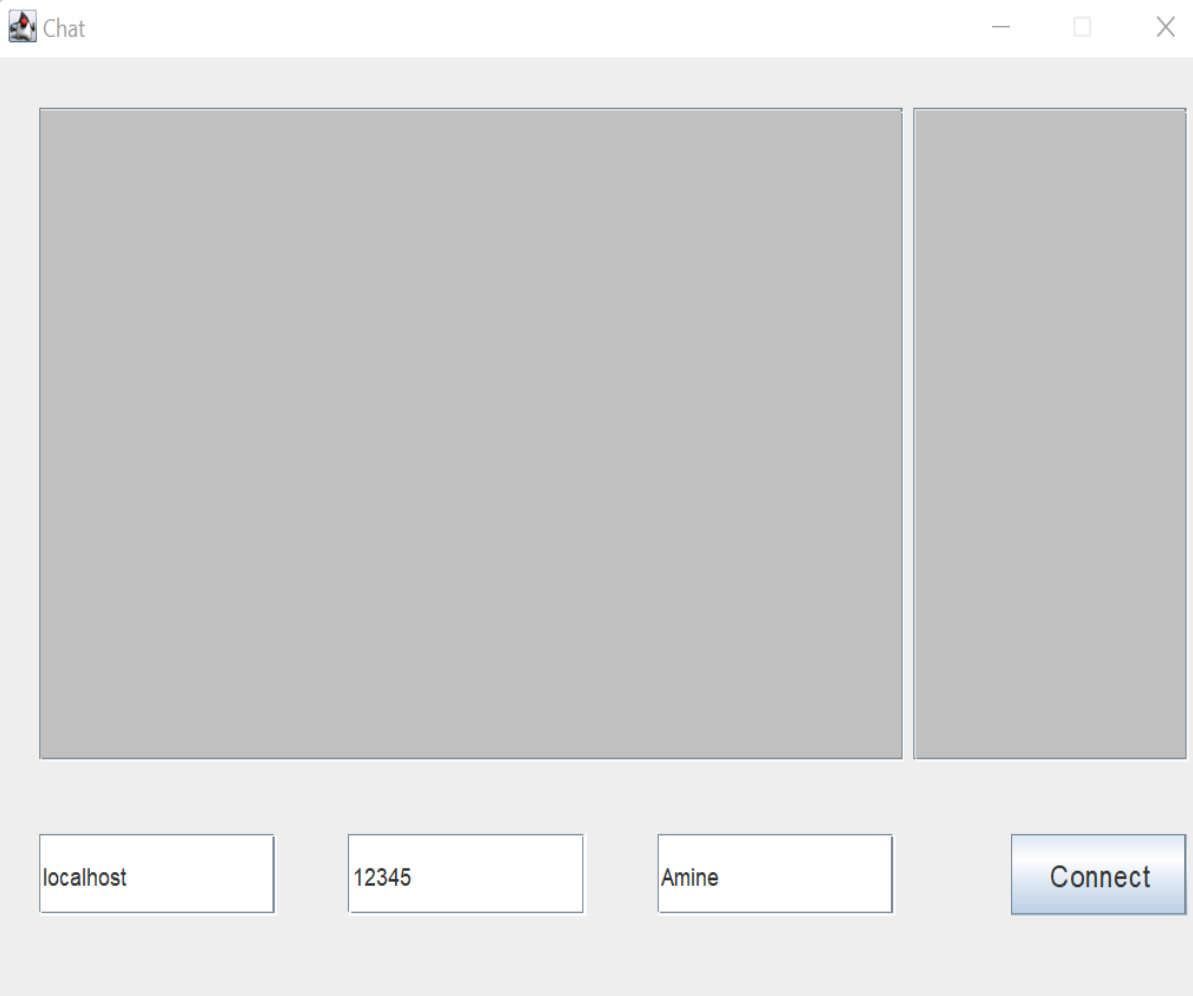
Dans chaque package se trouvent les différentes classes.

Pour pouvoir exécuter le projet il faut se rendre dans le package default et lancer la classe `server` (run as Java Application) puis lancer le tant de client que l'on souhaite de la même façon mais cette fois sur la classe `clientGui.java`.

## 2- Déploiement application

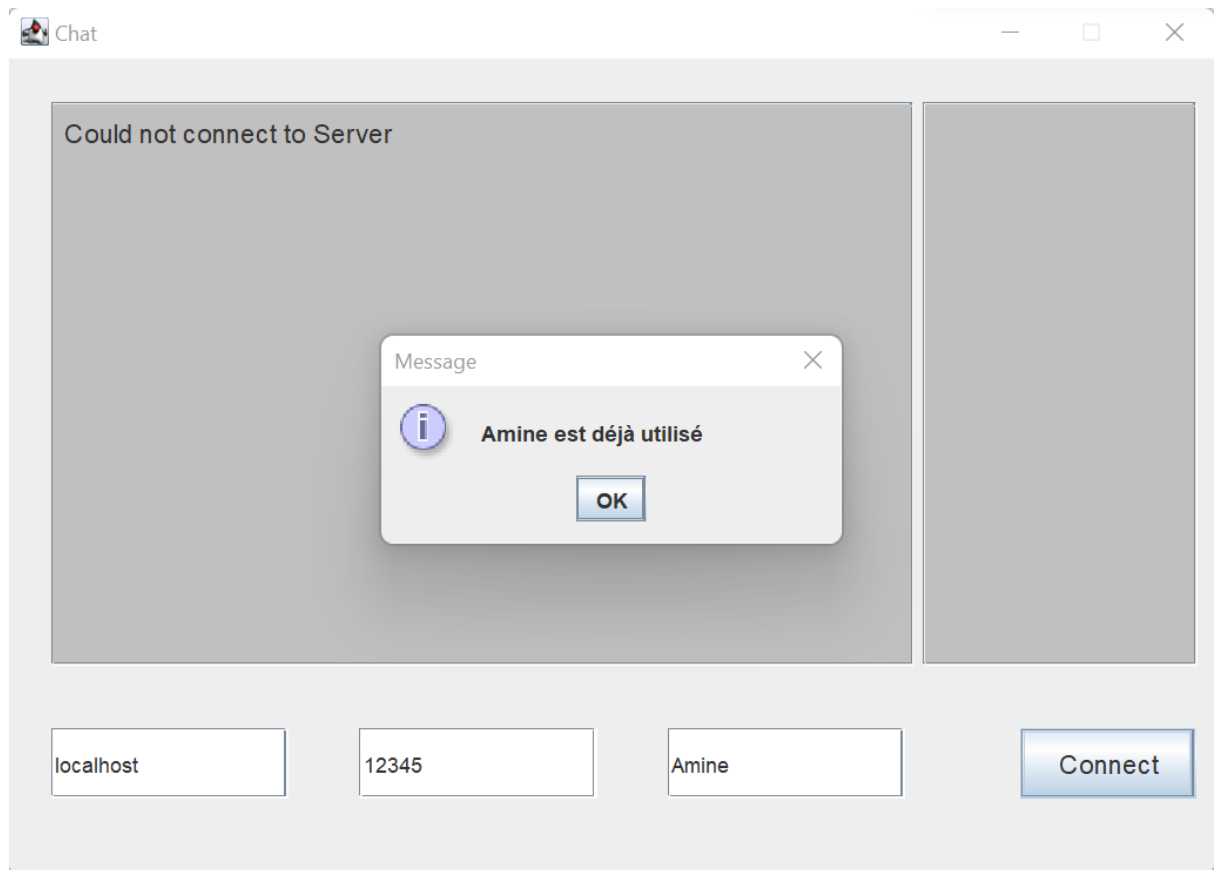
### 2-1 Phase connexion

Dans un premier temps, nous allons demander à l'utilisateur de choisir un pseudo valide qui va lui permettre d'être reconnu parmi les autres utilisateurs.

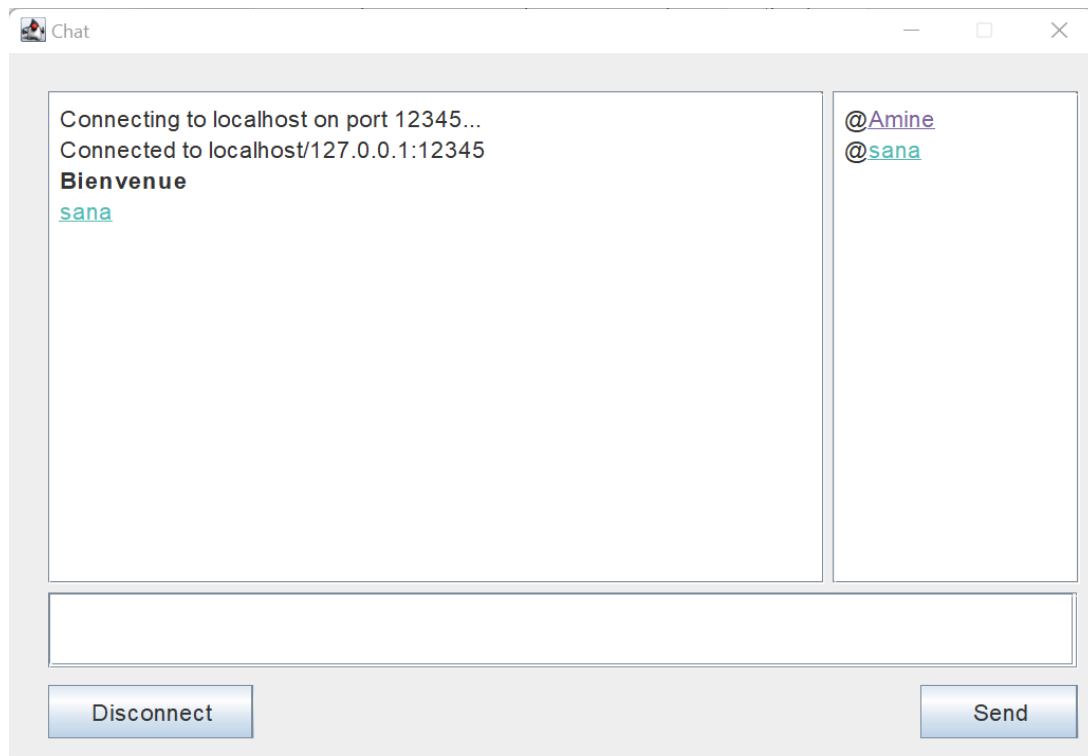


The screenshot shows a web application window titled "Chat". The window has a standard title bar with minimize, maximize, and close buttons. The main content area is divided into two vertical panels, both of which are currently empty and greyed out. At the bottom of the window, there is a form with four input fields and a button. The first field contains the text "localhost", the second contains "12345", and the third contains "Amine". The fourth field is a blue button labeled "Connect".

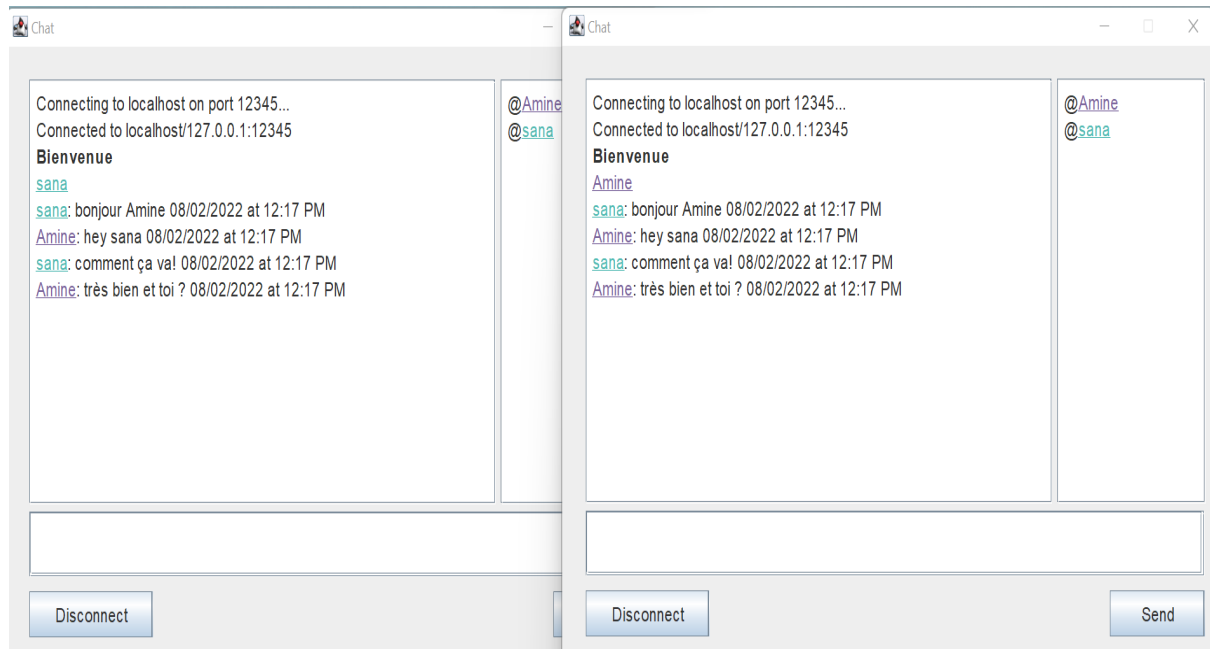
En effet, l'utilisateur peut à n'importe quel moment modifier son pseudo. Par contre, il ne peut pas prendre un pseudo déjà utilisé. Au cas où cela se produit, un message d'erreur est alors affiché.



Une fois le pseudo choisi, l'utilisateur peut donc communiquer avec l'ensemble des utilisateurs en ligne.



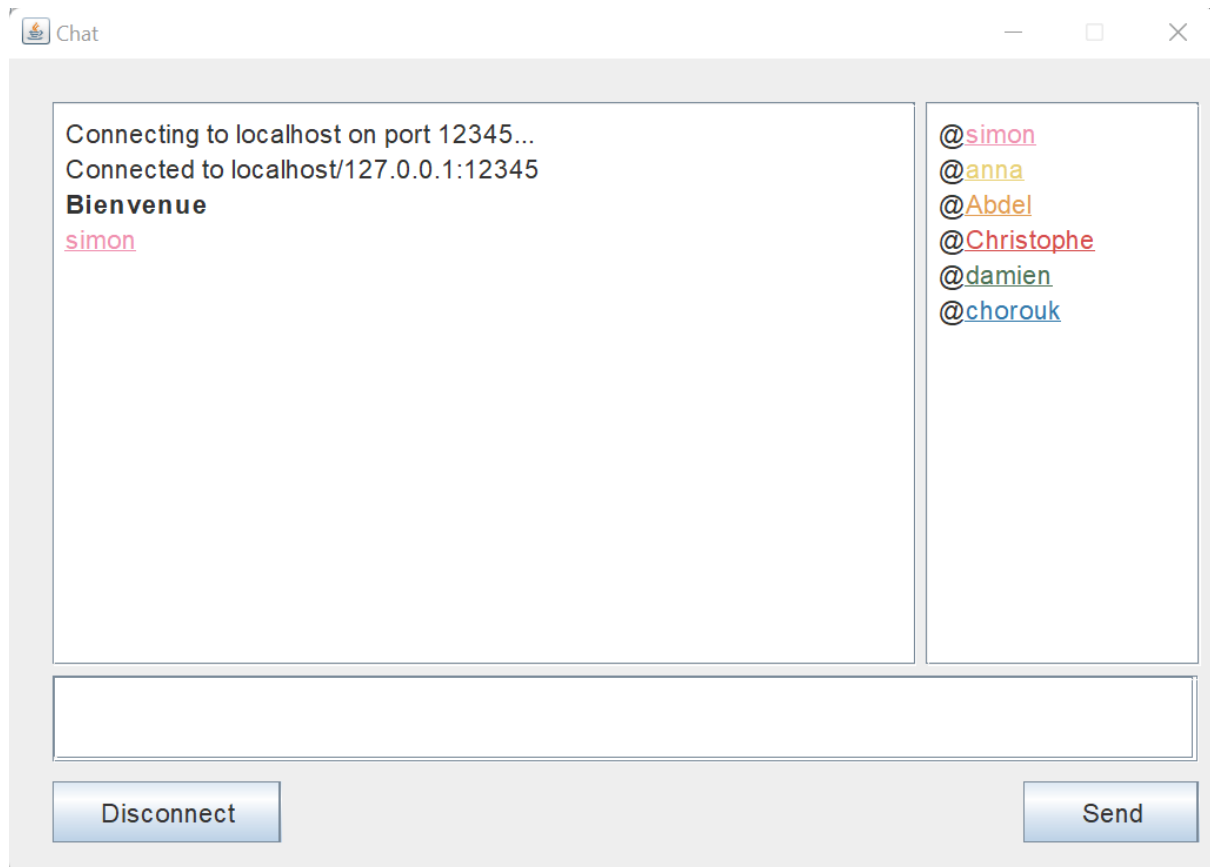
Une barre de menu à droite permet de visualiser les utilisateurs qui sont en ligne et de démarrer une session de clavardage avec eux.



Une fois qu'on choisit l'utilisateur avec qui on souhaite communiquer, on pourrait à ce moment la discuter, les messages sont cependant horodatés.

## 2-2 Connexion à plusieurs personnes:

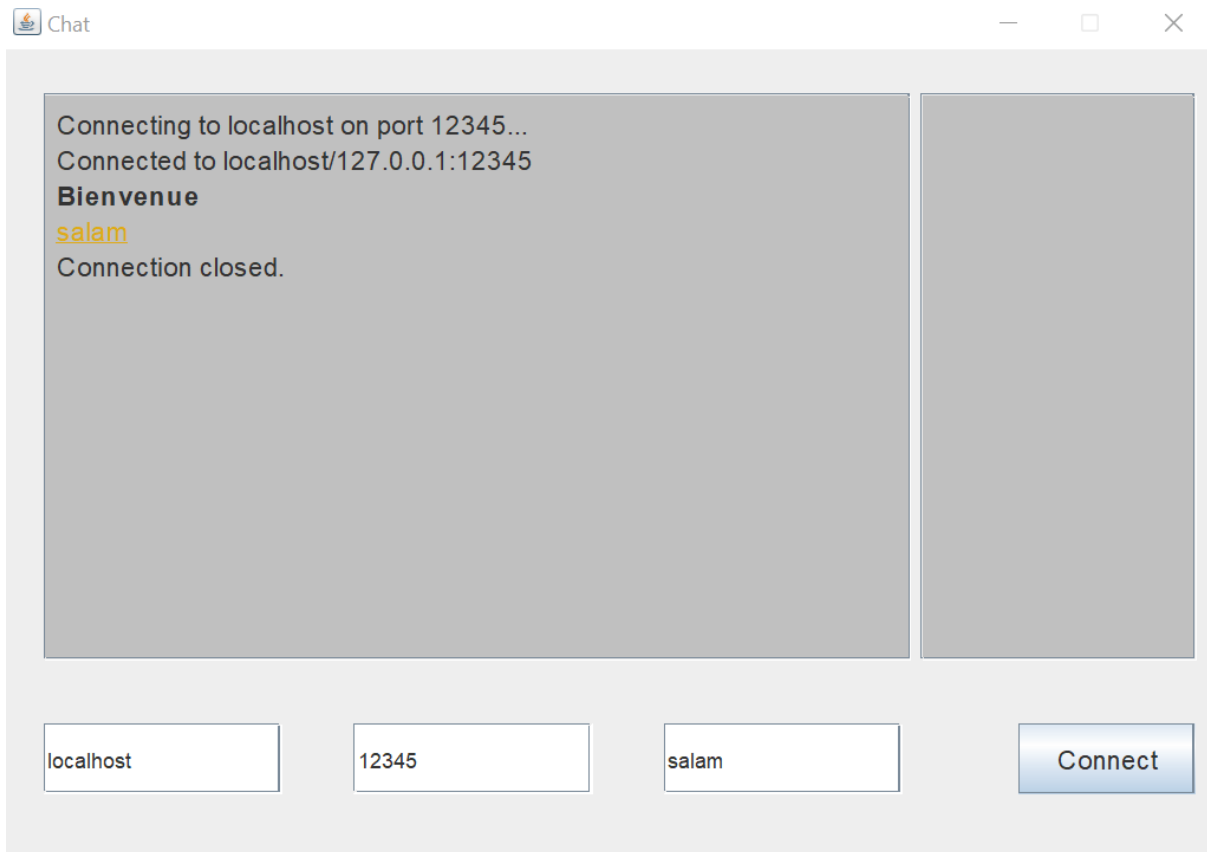
L'utilisateur peut démarrer une session de clavardage avec n'importe quel utilisateur contenu dans la liste des utilisateurs actifs.



## 2-3 fin session de clavardage

Vous pouvez mettre fin à une session de clavardage simplement en fermant la discussion, un message est affiché confirmant la fermeture de connexion.

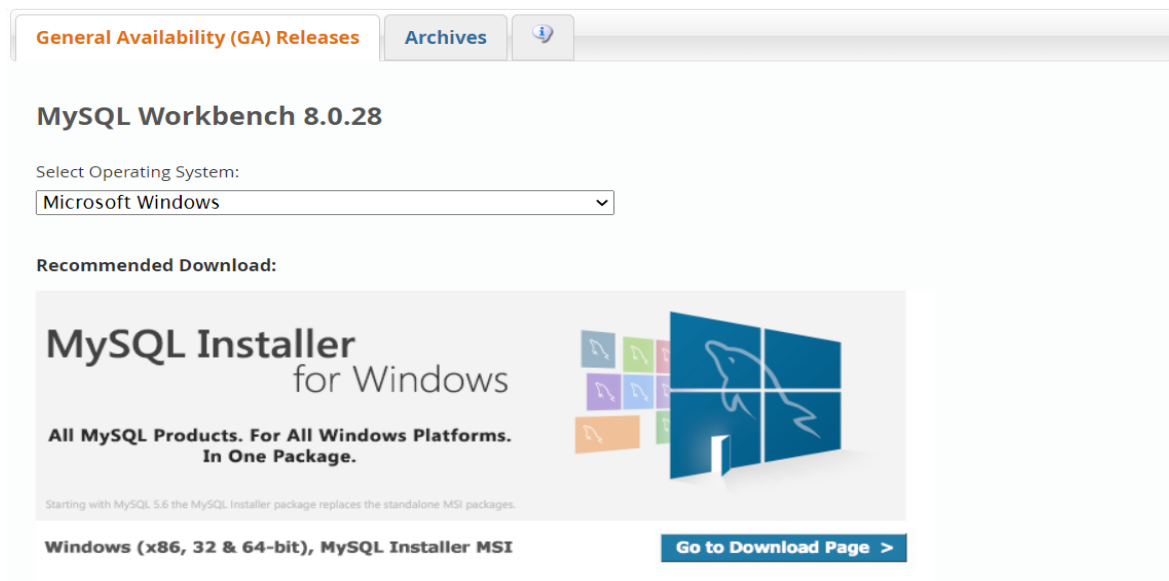
l'utilisateur qui se déconnecte envoie un message UDP en broadcast ce qui permet à tous les autres utilisateurs actifs de le supprimer de leurs listes.



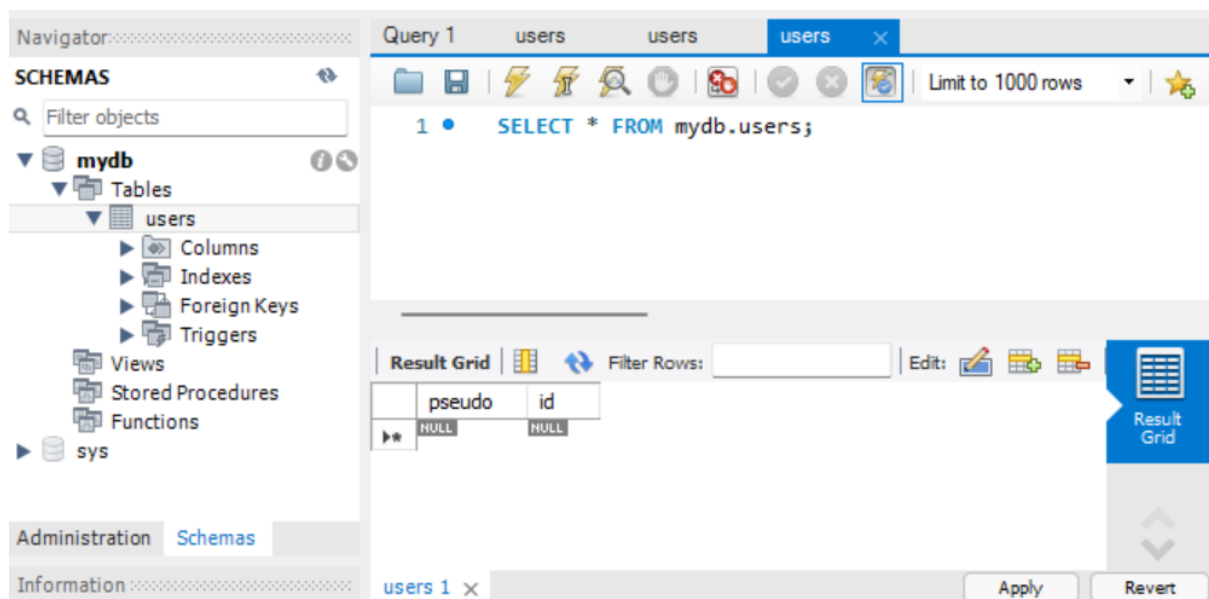
## 2-4 Base de données:

Pour l'implémentation de la base de données, nous avons utilisé MySQL workbench, téléchargeable ici: <https://dev.mysql.com/downloads/workbench/>

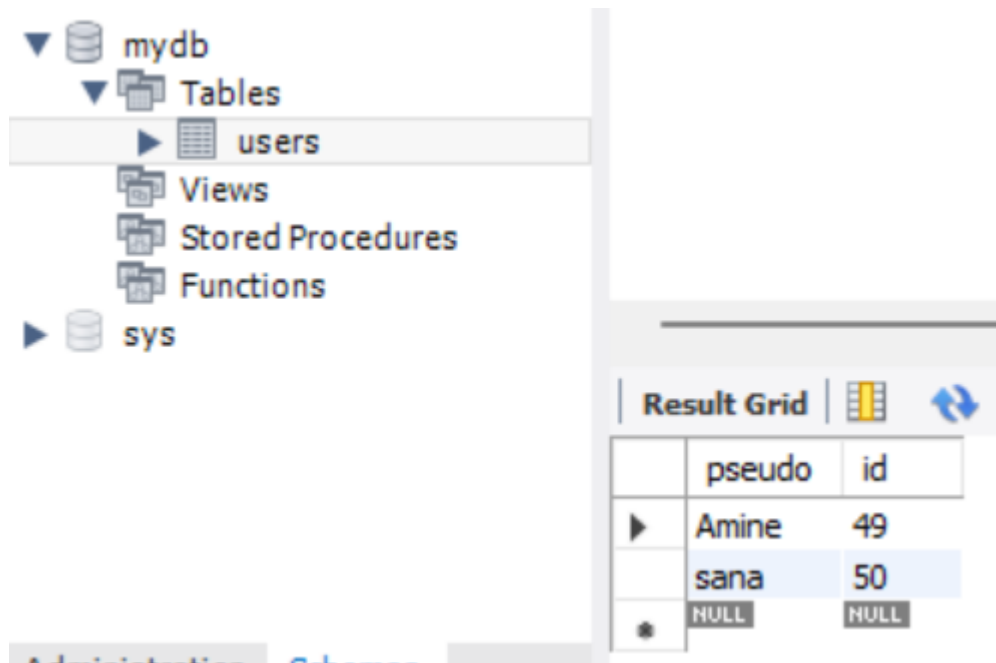




il faudra compiler le fichier users.sql dans le dossier executable pour pouvoir créer une table users vide qui contiendra les futurs utilisateurs.



Une fois cette table créée on pourra laisser tourner le serveur, et lancer autant de clients que l'on souhaite, la capture suivante illustre l'ajout des utilisateurs récemment créer (Amine, et Sana ) avec un id qui s'incrémente.



## 2-5 Problèmes rencontrés:

-Tout au long du projet, nous avons rencontré pas mal de problèmes, tout d'abord nous voulions mettre en place un serveur en ligne, mais nous avons eu des soucis avec l'implémentation de Tomcat, du coup on s'est contenté d'utiliser un serveur en local.

### Gestionnaire d'applications WEB Tomcat

Message :

OK

Gestionnaire

Lister les applications

Aide HTML Gestionnaire

Aide Gestionnaire

Etat du serveur

Applications

Chemin	Version	Nom d'affichage	Fonctionnelle	Sessions	Commandes
/	Aucun spécifié	Welcome to Tomcat	true	0	<div> <div>Démarrer</div> <div>Arrêter</div> <div>Recharger</div> <div>Retirer</div> </div> <div> <div>Expirer les sessions</div> <div>inactives depuis ≥ 30 minutes</div> </div>
/arfa	Aucun spécifié		true	0	<div> <div>Démarrer</div> <div>Arrêter</div> <div>Recharger</div> <div>Retirer</div> </div> <div> <div>Expirer les sessions</div> <div>inactives depuis ≥ 30 minutes</div> </div>

voilà le problème qu'on a rencontré, on n'arrivait pas à lancer une session et on ignorait l'origine du problème.

-En ce qui concerne l'envoi des messages, nous n'avons pas réussi à inclure les autres types de messages(fichiers et images..).

-On n'a pas réussi à créer un système d'authentification avec un login et un mot de passe approprié, ce qui aurait été intéressant surtout si l'application ne nécessite plus de technicien.

-Par contrainte de temps, on n'a pas pu développer une base de données assez élaborée qui permettra de récupérer l'historique des messages et appliquer un traitement de données.

### 3- Tests :

En ce qui concerne la partie de tests, on a essayé de tester chaque fonction écrite de manière individuelle. En effet, on n'a pas développé les tests Junit comme il le faut, mais on a essayé de réaliser la même chose dans le main en appelant les différentes fonctions et en testant leurs comportements pour finalement aboutir aux résultats souhaités.

Nous avons aussi eu recours au mode debug d'Eclipse afin de mieux décortiquer ce qui se passe dans notre code et les parties qui nous posaient problèmes.

Nous avons essayé de mettre des `System.out.println("")` dans l'ensemble de nos fonctions pour voir un peu l'ordre d'exécution des événements. Et pour la phase de réseau, nous nous retrouvions aux salles de l'insa pour tester avec deux ordinateurs.

### 4- Conclusion

Le fait d'allier COO et POO était très bénéfique pour nous, même si nos diagrammes UML initiaux ne correspondent pas exactement à la version finale qu'on a livré. En effet, nous avons mis l'accent sur l'implémentation de l'application de chat et la partie connexion qui nous a énormément pris du temps car on a eu des problèmes côté serveur dont on ignorait l'origine.

Ceci fut aussi la première fois où on apprenait à intégrer une base de données SQL avec Eclipse et gérer les différentes dépendances.

Nous avons compris aussi que développer une application ne revenait pas seulement à écrire des lignes de codes mais à établir une réflexion approfondie sur la partie conception et implémentation et savoir comment établir un lien solide entre les deux parties.

Finalement, nous espérons que notre rendu satisfera vos attentes, et que notre travail et l'ensemble des problèmes rencontrés ont pu présenter une version de projet réaliste de l'expérience qu'on a mené tout au long de ce semestre.