

# Rapport Projet : POO

## « Chat System Project »

*Système de clavardage distribue interactif  
multi-utilisateur temps réel*

Étudiant :

BAH Thierno Amadou - 4IR-SC

BENCHEHIDA Yacine - 4IR-SC

Enseignant :

Sami Yanguï

Cours :

I4IRIL11-COO-POO

# Table des matières

Introduction .....	3
I- Choix d'implémentations.....	4
a) <i>Choix techniques</i> .....	4
1) <i>Base de données</i> .....	4
2) <i>Serveur</i> .....	4
b) Choix sur la signalisation entre les clients de l'application :.....	4
c) <i>Choix conceptuels</i> .....	4
II- Guide d'installation et de configuration .....	5
a) <i>Installation et configuration de la base de données : Projet concerné chatjpa</i> .....	5
b) Compilation et génération de l'exécutable pour l'API gérant la base de données <i>chatjpa.jar</i> .....	5
c) <i>Installation du client</i> .....	6
III- Manuel d'utilisation de l'application.....	6
a) <i>Page d'accueil</i> : .....	6
b) <i>Envoi d'un message textuel</i> .....	7
IV- Points à améliorer .....	8

# Introduction

Dans le cadre de notre 4<sup>ème</sup> année en informatique et réseaux, il nous est proposé un projet de 5 mois nous permettant de mettre en pratique nos connaissances et nos compétences au travers d'un cahier des charges ayant pour finalité la conception et le développement d'un système de clavardage.

Il s'agit d'un système de communication qui sert de support aux équipes et groupes de l'entreprise afin de leur permettre d'accroître leur efficacité naturelle. Le système de communication pourra supporter des messages à caractère textuel, des messages de type image et tout type de documents ( PDF, Word, programme informatique ...).

## I- Choix d'implémentations

### a) Choix techniques

#### 1) *Base de données*

D'après le cahier des charges, le système de stockage pour l'application de Chat System doit impérativement être décentralisé. Nous avons décidé donc de mettre en place un serveur *PostgreSQL* qui va gérer la base de données. Nous avons également utilisé une API pour manipuler la base de données : JPA (Java Persistence API).

L'avantage de cette approche, c'est sa scalabilité. En effet cela permet un traitement asynchrone entre le serveur et la base de données, ce qui améliore donc considérablement le temps de réponse du serveur aux clients.

#### 2) *Serveur*

Le serveur permet l'inscription d'un nouvel utilisateur, l'authentification, le choix du nom d'utilisateur, de garder temporairement les utilisateurs connectés, et également de sauvegarder sur la base de données, les messages textuels envoyés entre les utilisateurs, (les fichiers et images sont stockés localement chez les clients). Ainsi, chaque utilisateur connecté annonce régulièrement sa présence au serveur, et en retour il récupère les autres utilisateurs connectés.

### b) Choix sur la signalisation entre les clients de l'application :

Chaque utilisateur authentifié, annonce sa présence au serveur qui, en retour lui transmet la liste des utilisateurs connectés. Chaque utilisateur contacte régulièrement le serveur pour annoncer qu'il est encore connecté et redécouvre les utilisateurs encore actifs.

L'échange de tout type de messages (textuel, image ou fichier) entre les clients de l'application s'effectue en utilisant TCP.

### c) Choix conceptuels

Pour utiliser l'application, on doit posséder un compte protégé par un mot de passe. Un utilisateur souhaitant modifier son pseudonyme ou son mot de passe devra se connecter d'abord se connecter sur le Chat System avec son ancien login ainsi que son ancien mot de passe. En effet, la modification du pseudonyme sera répertoriée à tous les autres clients de l'application.

## II- Guide d'installation et de configuration

A l'état actuel de notre application, elle n'est pas encore utilisable, car nous avons rencontré un problème lors de la création du **.jar** de notre serveur qu'on n'arrive pas encore à fixer. Cependant nous allons dans les lignes qui suivent expliquer la procédure qu'on aurait mis en place pour le déployer.

Tout d'abord, vous il faudra aller chercher le code source en ligne, en clonant le git suivant :

```
git clone https://github.com/tabha/clavardage-Amadou-Yacine
```

### a) *Installation et configuration de la base de données : Projet concerné chatjpa*

En utilisant un script bash, l'administrateur installe le serveur **postgresql** et il crée un compte utilisateur avec un login et un mot de passe. Ensuite on crée la base de données et l'ensemble des tables dont a besoin l'application. On adapte le fichier META-INF/persistence.xml, qui contient les configurations de connexion de l'API JPA, sur la base de données ainsi créée. Dans ce fichier, il faudra renseigner les informations nécessaires pour la connexion.

```
<properties>
  <property name="javax.persistence.jdbc.url" value="jdbc:postgresql://localhost:5432/DATABASE_NAME"/>
  <property name="javax.persistence.jdbc.user" value="USER_NAME"/>
  <property name="javax.persistence.jdbc.driver" value="org.postgresql.Driver"/>
  <property name="javax.persistence.jdbc.password" value="PASSWORD"/>
</properties>
```

### b) *Compilation et génération de l'exécutable pour l'API gérant la base de données chatjpa.jar*

On mettrait également à disposition de l'administrateur, un script bash permettant de compiler correctement le projet **chatjpa** en incluant toutes les dépendances nécessaires, notamment le driver **jdbc**, ou encore **EclipsLink** (API utilisé pour le JPA).

Une fois que le jar est généré correctement et la base de données lancée, on pourra donc compiler le projet du serveur en incluant le **chatjpa.jar** comme dépendance.

Le serveur utilise **Tomcat** 8.5, et une servlet qui permet de gérer les requêtes clients.

Il faudra donc copier le dossier **chatServer** contenant l'ensemble des fichiers du serveur dans le dossier **webapps** du dossier d'installation de Tomcat.

On mettra également un fichier de configuration nous permettant de configurer le serveur comme, son adresse IP ainsi que le numéro de port sur lequel il se met en attente de requête.

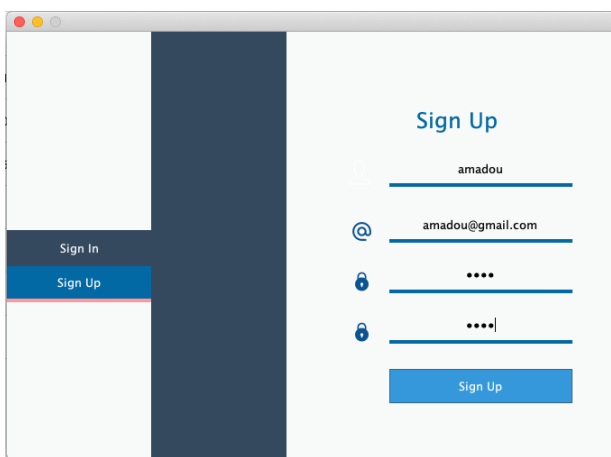
Une fois que le serveur est configuré correctement : les clients pourront à présent se connecter et échanger.

### c) Installation du client

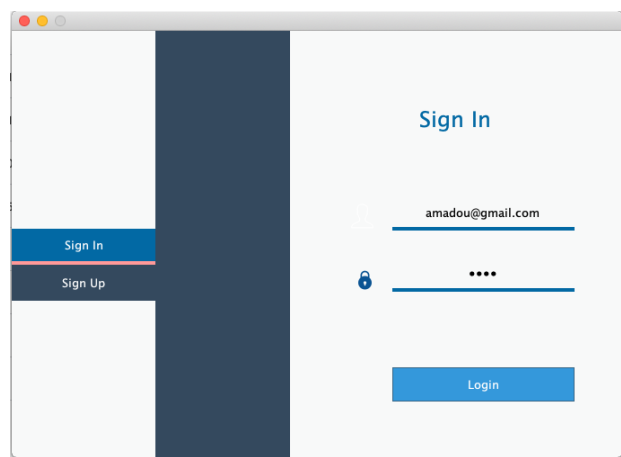
A la différence du serveur **le client est prêt**. Il faut donc télécharger l'exécutable **chatClient.jar** qui se trouve sur le git. IL faudra créer un dossier d'installation où il faudra également mettre le fichier **setting.ini** qui va définir les paramètres d'utilisation de l'application notamment, l'adresse du serveur distant et son numéro de port.

## III- Manuel d'utilisation de l'application

### a) Page d'accueil :

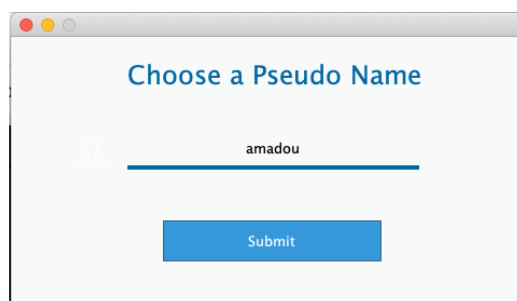
A screenshot of a web application window titled "Sign Up". The window has a dark blue sidebar on the left with two buttons: "Sign In" and "Sign Up". The "Sign Up" button is highlighted in blue. The main content area is white and contains a "Sign Up" heading. Below the heading are four input fields: a text field with "amadou", an email field with "amadou@gmail.com", a password field with "\*\*\*\*", and a confirm password field with "\*\*\*\*". Each field has an icon to its left: a person for the name, an '@' for email, and a lock for passwords. At the bottom is a blue "Sign Up" button.

Inscription

A screenshot of a web application window titled "Sign In". The window has a dark blue sidebar on the left with two buttons: "Sign In" and "Sign Up". The "Sign In" button is highlighted in blue. The main content area is white and contains a "Sign In" heading. Below the heading are two input fields: an email field with "amadou@gmail.com" and a password field with "\*\*\*\*". Each field has an icon to its left: an '@' for email and a lock for password. At the bottom is a blue "Login" button.

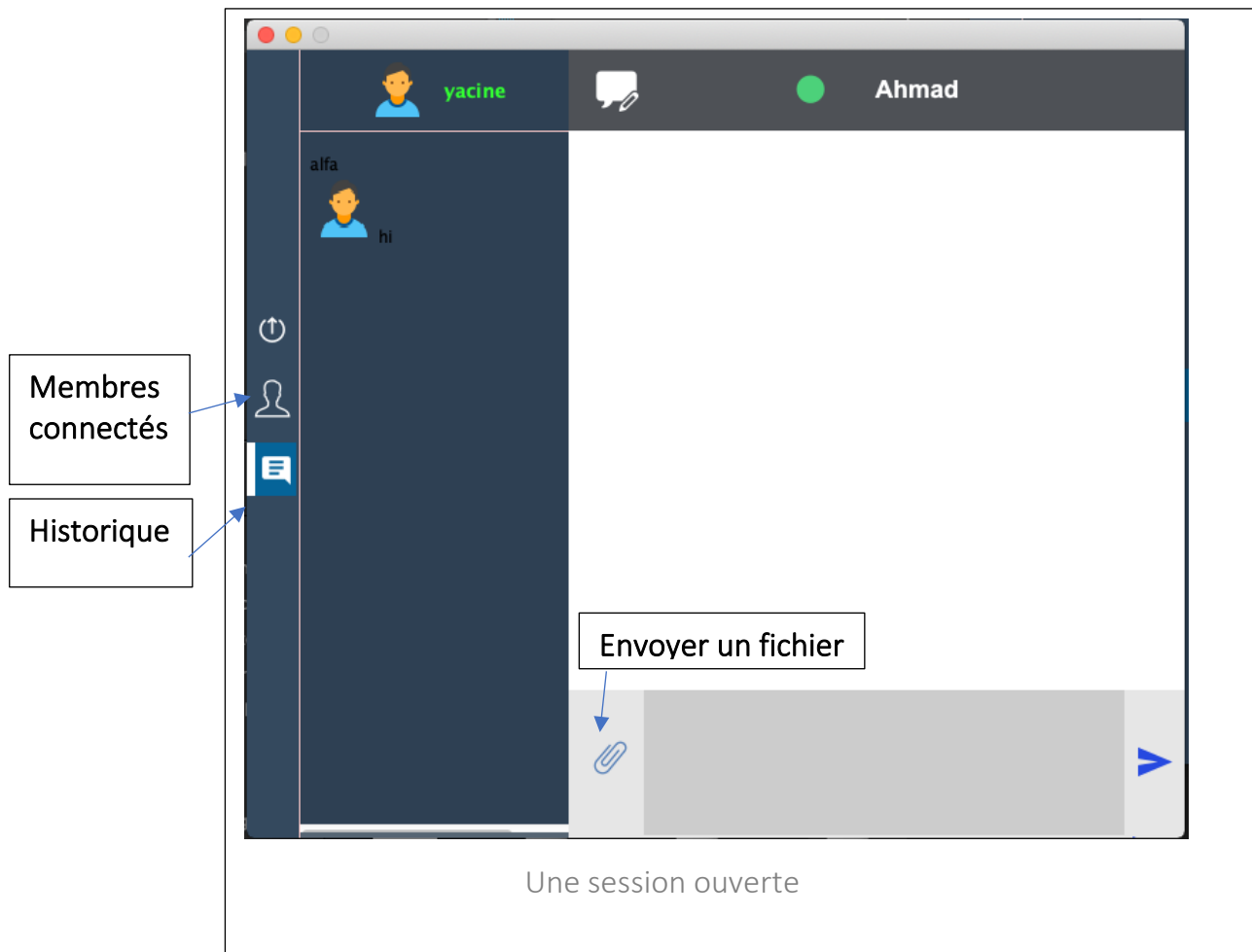
Connexion

Une fois que vous êtes authentifié, on vous demande de choisir un pseudo pour la session nouvellement ouverte.

A screenshot of a web application window titled "Choose a Pseudo Name". The window has a white background. At the top is the heading "Choose a Pseudo Name". Below it is a text input field with "amadou". At the bottom is a blue "Submit" button.

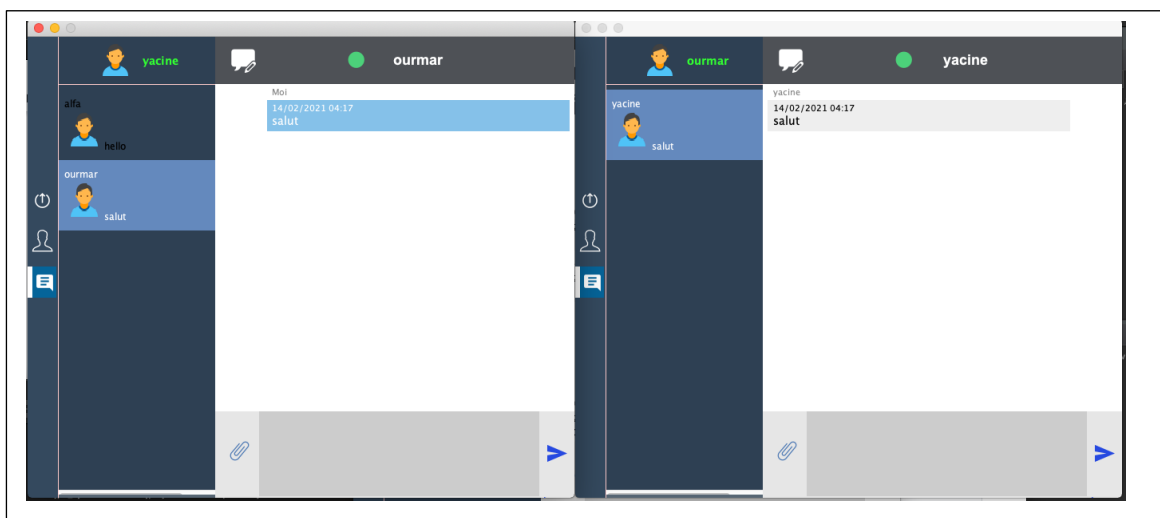
Choix du Pseudo Name

Si le nom choisit est valide on démarre une session, sinon on demande à l'utilisateur de choisir un autre pseudo.



### b) Envoi d'un message textuel

Lorsqu'on envoie un message, on envoie le message au serveur distant pour l'enregistrer dans la base de données, ensuite on contacte l'utilisateur connecté par TCP, puis on lui envoie le message. Pour les fichiers, on transmet uniquement le nom du fichier au serveur, ensuite on adresse le fichier à l'utilisateur connecté qui l'enregistrera chez lui localement.



Sur l'image ci-haut on voit deux sessions ouvertes qui sont connectées au système, et qui s'envoient un message de salutation. Lors de leur prochaine connexion, ils pourront consulter leur historique.

#### IV- Points à améliorer

En effet, le projet que l'on a rendu n'est pas encore déployable mais fonctionnel localement. Cela est étant dû à la mise en place du serveur qui a beaucoup de dépendances.