



INSA

INSTITUT NATIONAL
DES SCIENCES
APPLIQUÉES
TOULOUSE

Rapport du projet POO-COO Chat System

HOU Yunan
YANG Yuyuan
IR Groupe B1 – 4^{ème} année
2020/2021




Table des matières

Introduction.....	1
I. Choix d'implémentation	2
I.1. Architecture du code.....	2
I.2. Base de données.....	2
I.3. Serveur de présence.....	3
I.4. Interconnexion réseau	3
I.5. Multithreading.....	4
II. Guide d'installation.....	4
II.1. Récupération les sources et configuration.....	4
III. Guide d'utilisation	5
III.1. Connexion.....	5
III.2. Création de compte.....	6
III.3. Sous-menu du système	7
III.4. Sous-menu de la conversation	8
IV. Test validation	10
Table des annexes – Partie COO.....	14
Annexe 1 : Diagramme de Bode.....	14
Annexe 2 : Cas d'utilisation	15
Annexe 3 : Diagramme de séquence.....	16
Annexe 4 : Structure composite.....	20

Introduction

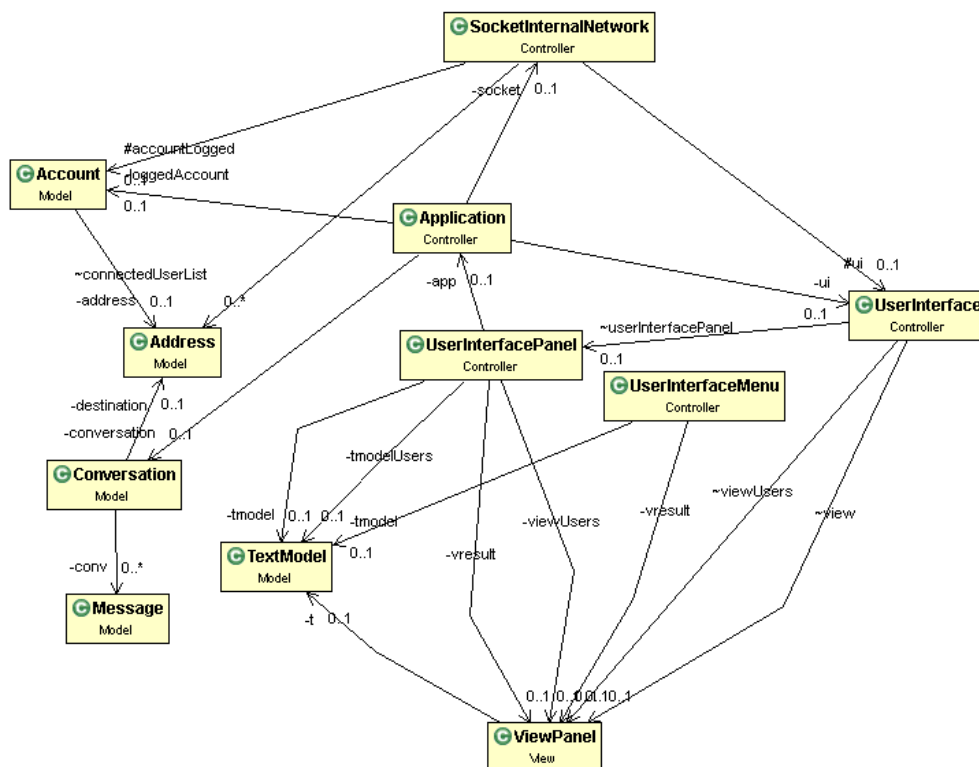
Le système de clavardage distribué interactif multi-utilisateur en temps réel est une application qui contient un serveur, un réseau interne et externe et une base de données. Après l'installation et la configuration, l'utilisateur peut chatter, gérer la connexion, gérer le compte et consulter l'historique des messages. Les fonctionnalités principales sont précisées dans un cahier des charges fourni. Avant le début du développement, les choix de conception sont toujours importants. Tout d'abord, nous allons présenter les différentes conceptions du projet, ensuite, nous allons parler du guide d'installation et d'utilisation, finalement, nous allons présenter les différents tests de validation qui ont été effectués.

I. Choix d'implémentation

Cette partie décrit les principaux éléments d'architecture de notre application.

I.1. Architecture du code

Notre application est développée en suivant l'architecture MVC qui contient des modèles, des contrôleurs et des interfaces graphiques. Les modèles sont utilisés pour présenter les comptes des utilisateurs et les conversations entre ces utilisateurs. Afin de gérer ces modèles, des contrôleurs sont mis en place, qui sont liés avec des vues par des observateurs et qui maintiennent la connexion à l'application, aux bases de données et au serveur de présence. Cela nous a permis de développer en parallèle ces éléments et de les fusionner. Ainsi, chaque membre du binôme pouvait s'atteler de son côté à sa tâche.

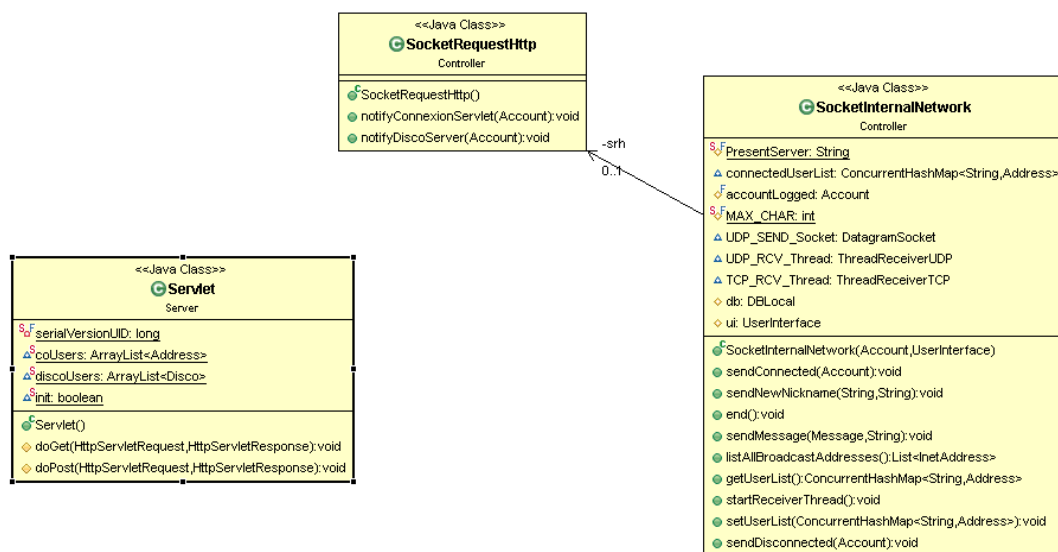


I.2. Base de données

En termes de gestion de données, nous avons choisi la décentralisation de la base de données, qui signifie que chaque utilisateur possède sa base de données locale et la synchronise quand la connexion est établie avec les autres utilisateurs. En plus, dans la partie d'implémentation de la connexion avec la base de données (JDBC), nous avons appliqué des getters et setters selon la norme de JavaBean.

I.3. Serveur de présence

En ajoutant le management de présence, on établit la connexion entre les agents et le serveur avec des Java HTTP Servlets. Les statuts des utilisateurs sont “onlines” et “offlines”. Quand l'utilisateur est connecté sur l'application, son statut va changer en “online” et l'application va informer le serveur que l'utilisateur est connecté. Le même mécanisme est utilisé pour “offline”. La liaison entre les agents et le serveur WEB se fait via des requêtes HTTP GET et HTTP POST. Cela facilite le déploiement du serveur de présence, qui est simplement un servlet qui peut se greffer à un serveur WEB existant qui accepte les servlet JAVA tel qu'Apache Tomcat v9.0.



I.4. Interconnexion réseau

Nous avons choisi de faire la découverte des agents locaux via un broadcast UDP. Cela permet d'atteindre les utilisateurs locaux efficacement. Nous avons choisi le broadcast UDP par rapport à l'IP multicast car, le but étant simplement d'atteindre les utilisateurs locaux, nous n'avons pas vu l'utilité que le paquet “passe” un routeur. En effet, la découverte des utilisateurs externes passe par le serveur de présence. Nous utilisons la même technologie pour signaler qu'un utilisateur change de pseudo.

L'envoi de message passe par un socket TCP/IP, car nous devons nous assurer que le message est correctement reçu par l'autre agent. De plus, TCP nous assure que les messages seront reçus dans l'ordre, ce qui est grandement appréciable pour notre application. Cela évite de vérifier dynamiquement l'ordre des messages. Finalement, cela assure aussi que l'autre agent a reçu le message alors qu'il est encore actif, ce qui évite toute incohérence dans les bases de données locales des agents.

I.5. Multithreading

Afin de profiter des processeurs multicoeurs présent sur la plupart des PC actuels et du parallélisme qu'offre le multithreading, nous avons utilisé un thread main, 2 thread permanent et quelques threads éphémères.

Le thread main contient le contrôleur, il gère donc l'affichage, et l'envoi des messages. Les deux threads permanents contiennent respectivement un serveur UDP et un serveur TCP. Ainsi leur fonctionnement n'altère pas celui du thread main et permet tout de même d'être capable de réagir lorsque l'agent reçoit un message.

Les threads éphémères sont potentiellement créés lorsque le serveur TCP reçoit une connexion. Le serveur crée alors un thread afin que celui-ci puisse gérer la communication et que le serveur puisse retourner dans un état d'écoute. Il est aussi périodiquement nécessaire de créer un thread qui s'occupera de faire une requête auprès du serveur de présence. Ce thread contient donc un client temporaire HTTP.

II. Guide d'installation

Dans cette partie, nous allons présenter l'installation de notre projet, c'est-à-dire le contexte utilisé, ainsi que notre approche générale pour aborder ce projet.

II.1. Récupération des sources et configuration

Tout d'abord, il faut récupérer le code source en ligne, en clonant le git au dessous et l'application se trouve sous le nom "[EasyChat.jar](https://github.com/iattach/Projet_COO_POO)".

https://github.com/iattach/Projet_COO_POO

Spécification de l'application :

- **Nom** : EasyChat.jar
- **Taille** : 10.1 Mo
- **Ports utilisés** : 6666, 6667, 6668, 6669, 6670
- **Version de la JDK** : Java SE Development Kit 11.0.10
- **Commande d'exécution** : java -jar EasyChat.jar
- **Librairies** :
 - ojdbc8.jar
 - sqlite-jdbc-3.25.1.jar

Guide de déploiement :

Pour déployer l'application, il faut que la machine sur laquelle est exécuté le fichier EasyChat.jar possède au moins l'environnement JRE 11, disponible sur le site d'ORACLE. Un fois installé, il suffit de lancer la commande java -jar EasyChat.jar.

Spécification du serveur de présence :

- **Nom du serveur :** <https://srv-gei-tomcat.insa-toulouse.fr/>
- **IP du serveur (Dernier accès : 15/02/2021) :** 10.1.5.2
- **Version du serveur :** Apache Tomcat/9.0.16 (Ubuntu)
- **Version de la JVM :** 11.0.7+10-post-Ubuntu-2ubuntu218.04
- **Adresse du servlet :** <https://srv-gei-tomcat.insa-toulouse.fr/EasyChat/Servlet>
- **Librairie utilisée pour générer le servlet :**
 - servlet-api-tomcat9.jar
 - Model.jar (librairie créée pour ce projet)

Guide de déploiement :

Une fois le serveur TOMCAT9 déployé, il suffit d'insérer le fichier EasyChat.war fourni par l'interface manager de TOMCAT. Si celui-ci n'est pas installé, il suffit d'insérer le fichier EasyChat dans le répertoire WEB racine de TOMCAT. Il faudra alors relancer TOMCAT pour que la modification soit prise en compte. Si vous désirez changer l'adresse du serveur, vous devrez modifier le champ suivant dans le fichier Application.InternalSocket :

```
public class SocketInternalNetwork {  
    protected static final String PresentServer = "https://srv-gei-tomcat.insa-toulouse.fr/EasyChat/Servlet";  
}
```

Il n'y a aucune autre manipulation à faire. Le servlet se met automatiquement à jour avec les différents agents. Pour voir si le servlet fonctionne, taper l'URL suivant : <https://srv-gei-tomcat.insa-toulouse.fr/EasyChat/Servlet> Il retourne la liste des utilisateurs connectés.

III. Guide d'utilisation

Après avoir suivi les étapes d'installation, nous allons expliquer l'utilisation de notre application et les fonctionnalités.

III.1. Connexion

A l'ouverture de l'application nous arrivons sur la page de connexion comme sur l'image ci-dessous. Pour se connecter, il suffit d'entrer son *username* et son *password* dans les champs correspondants, puis de cliquer sur le bouton « Sign in ».

Si le compte n'existe pas, nous arrivons à recevoir un message: « Error of connection : account not found !!! ». Dans ce cas là, nous pouvons cliquer sur « Sign up » pour créer un compte.

Si le binôme *username password* ne correspond à aucun compte existant, la connexion est refusée et un message d'erreur « Error of connection : account not found !!! » apparaît à la place de « Entrez Username/Password ». Dans ce cas, il faut vérifier le mot de passe ou créer un nouveau compte.

III.2. Création de compte

En tant que nouvel utilisateur, pour créer un compte, il suffit de cliquer sur le bouton « sign up » qui nous amène sur la page de création de compte ci-dessous.

Entrez *Username*, *Password* et *Nickname* dans les champs correspondant.

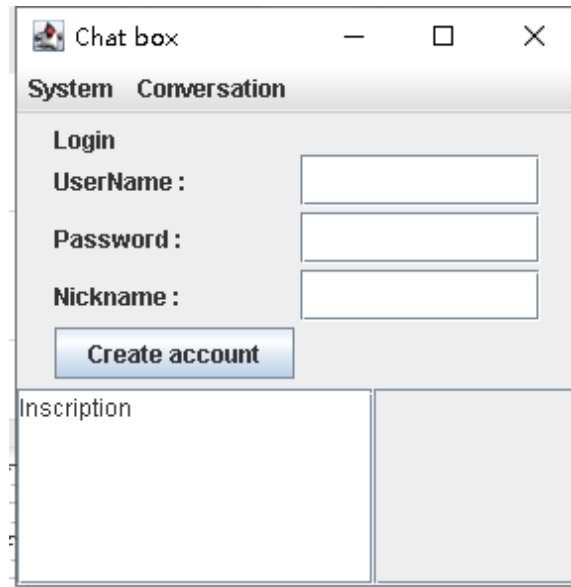
Cliquez ensuite sur le bouton “create account”

Si le compte est déjà existant (username existe déjà pour un autre compte), un message Error: account already exists!!!

Error: username already exists!!!

Error: nickname already exists!!!

Si la création de compte a bien été effectuée, nous revenons automatiquement sur la page de connexion avec un message “Your account has been created successfully !!!”.



Chat box

System Conversation

Login

UserName :

Password :

Nickname :

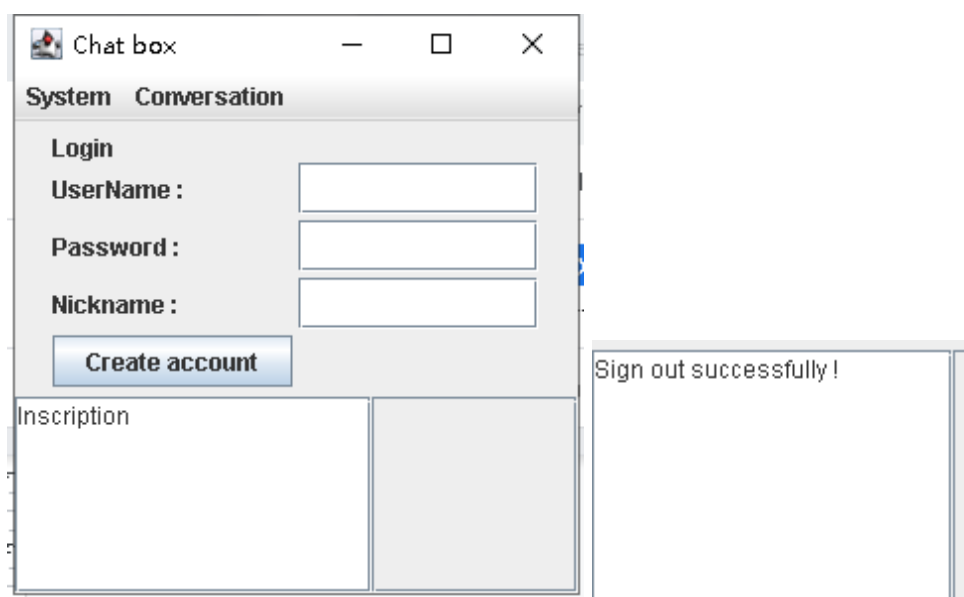
Create account

Inscription

III.3. Sous-menu du système

Déconnexion

Il est possible de se déconnecter de ce système en cliquant sur « Sign out » dans le sous-menu du système. Il affichera un message « Sign out successfully » si la demande de déconnexion est bien prise en compte.



Chat box

System Conversation

Login

UserName :

Password :

Nickname :

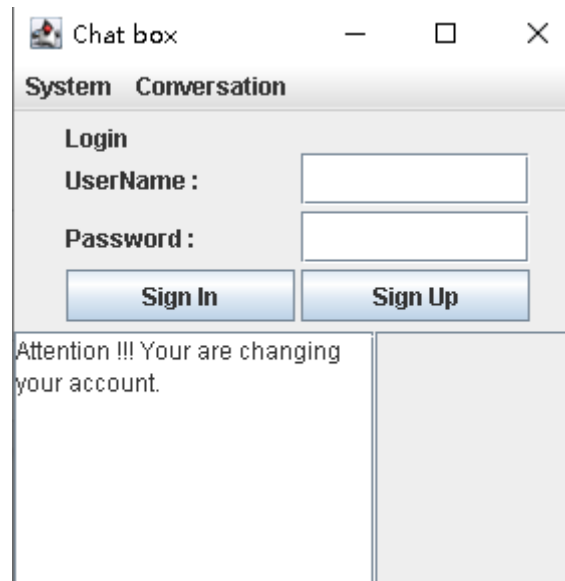
Create account

Inscription

Sign out successfully !

Changement de compte

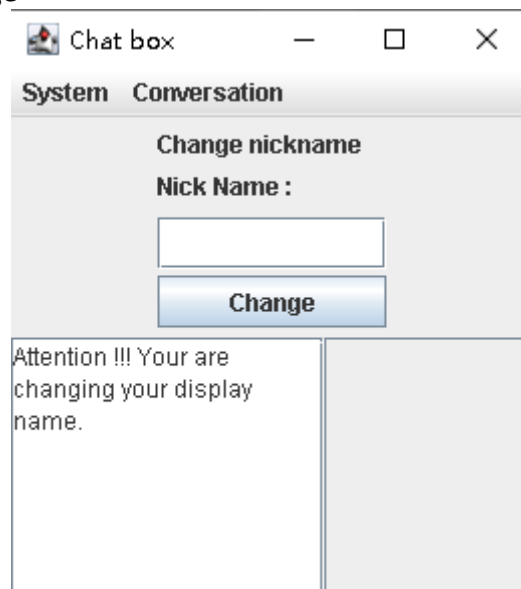
Dans notre sous-menu, il est possible de changer le compte connecté en cliquant « Change account », cette fonctionnalité permet de se déconnecter automatiquement afin d'aller dans la page de connexion pour effectuer une nouvelle connexion d'un autre compte.



The screenshot shows a window titled "Chat box" with standard Windows window controls. It has two tabs: "System" (selected) and "Conversation". Under the "System" tab, there is a "Login" section with labels "UserName :" and "Password :", each followed by a text input field. Below these fields are two buttons: "Sign In" and "Sign Up". At the bottom of the window, there is a message box that says "Attention !!! Your are changing your account." The right side of the window is a large, empty gray area.

Changement de pseudo

Il est aussi possible de changer le pseudo de son compte grâce à la page de changement de pseudo ci-dessous. (accessible via les sous-menu Système -> change nickname). Pour changer son pseudo, il suffit d'entrer le nouveau pseudo dans le champ correspondant puis de cliquer sur "change"



The screenshot shows the same "Chat box" window, but the "System" tab now displays a "Change nickname" section. It has a label "Nick Name :" followed by a text input field. Below the input field is a button labeled "Change". The message box at the bottom now says "Attention !!! Your are changing your display name." The right side of the window remains a large, empty gray area.

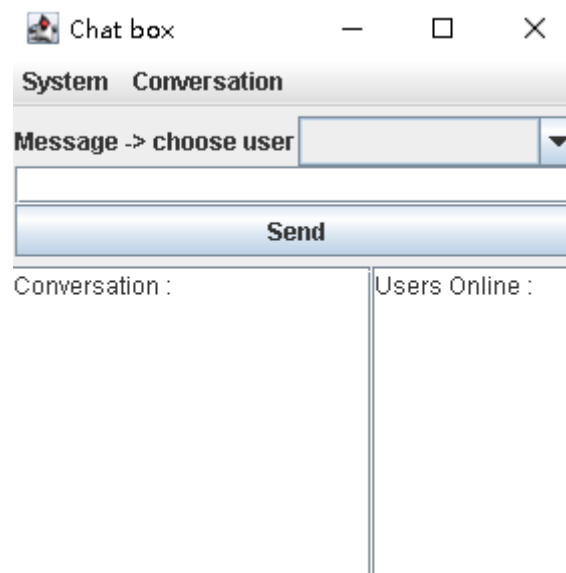
III.4. Sous-menu de la conversation

Conversation

Pour avoir une conversation avec un autre utilisateur, il suffit de cliquer sur son *DisplayName* dans l'onglet « choose user ». On arrive sur la page de conversation grâce à cette action.

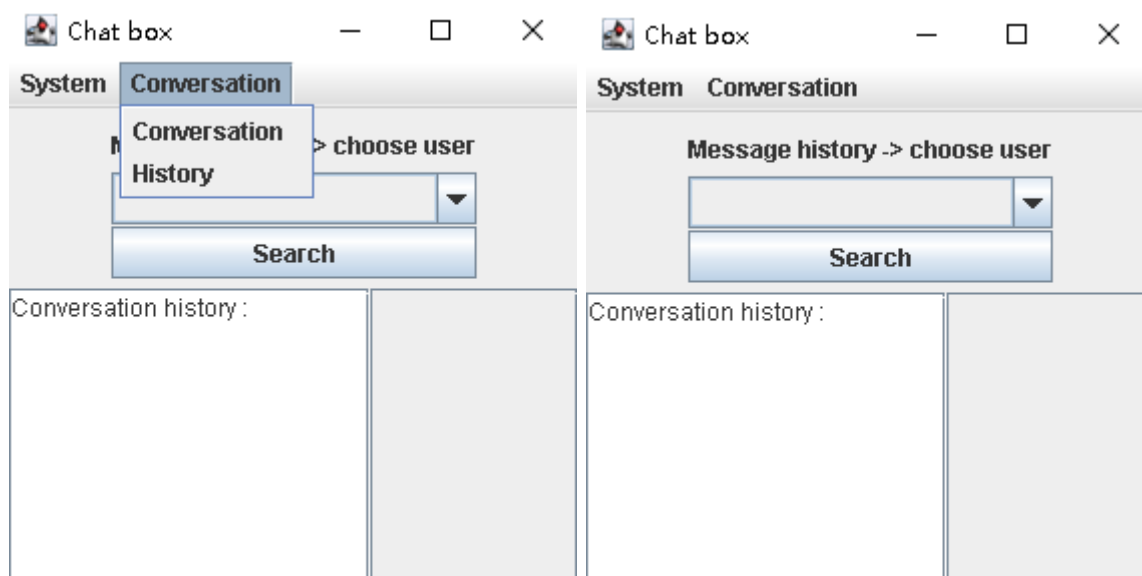
On peut voir facilement les utilisateurs connectés sur la zone *Users Online*, et il est possible de leur envoyer des messages en écrivant dans la zone blanche et en cliquant sur « send ».

De plus, il est possible de voir l'historique de la conversation (dans le sous-menu conversation que nous allons expliquer après) où chaque message est daté.



Historique

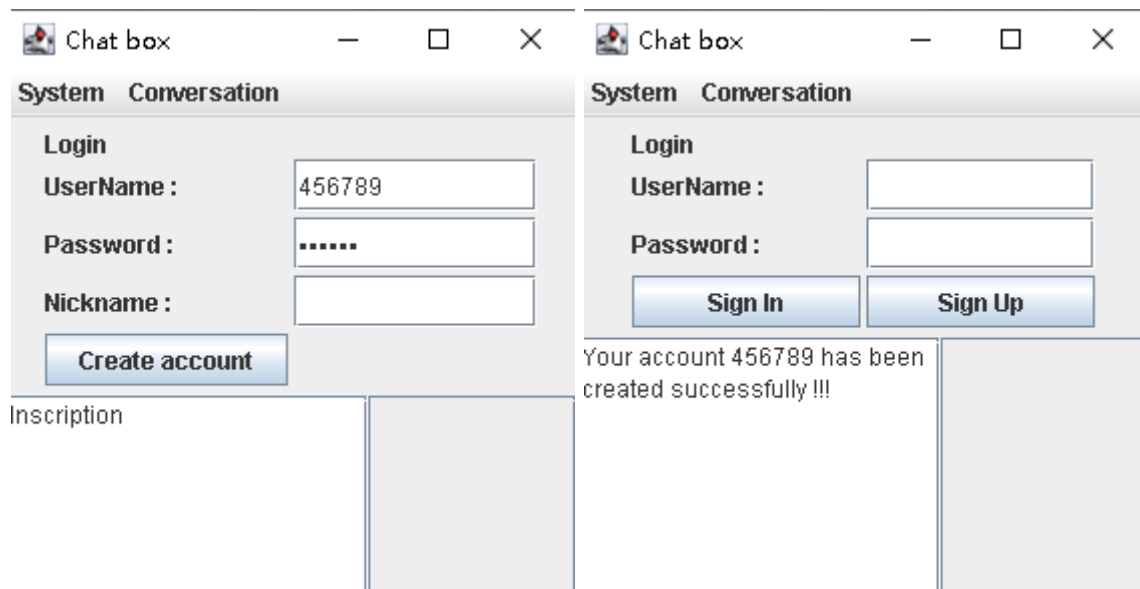
Nous pouvons consulter les historiques de conversation dans le sous-menu de conversation.



IV. Test validation

A cause de la limitation du nombre de postes, nous avons testé sur une machine virtuelle via VirtualBox avec un système d'exploitation Ubuntu, qui est isolée dans un réseau différent, c'est-à-dire un domaine d'adresse différent.

Tout d'abord, on crée un compte qui a pour *Username* "456789" et *Password* "456789", cela va créer directement un compte avec un nickname "456789" par défaut.



The image shows two side-by-side screenshots of a web application window titled "Chat box".

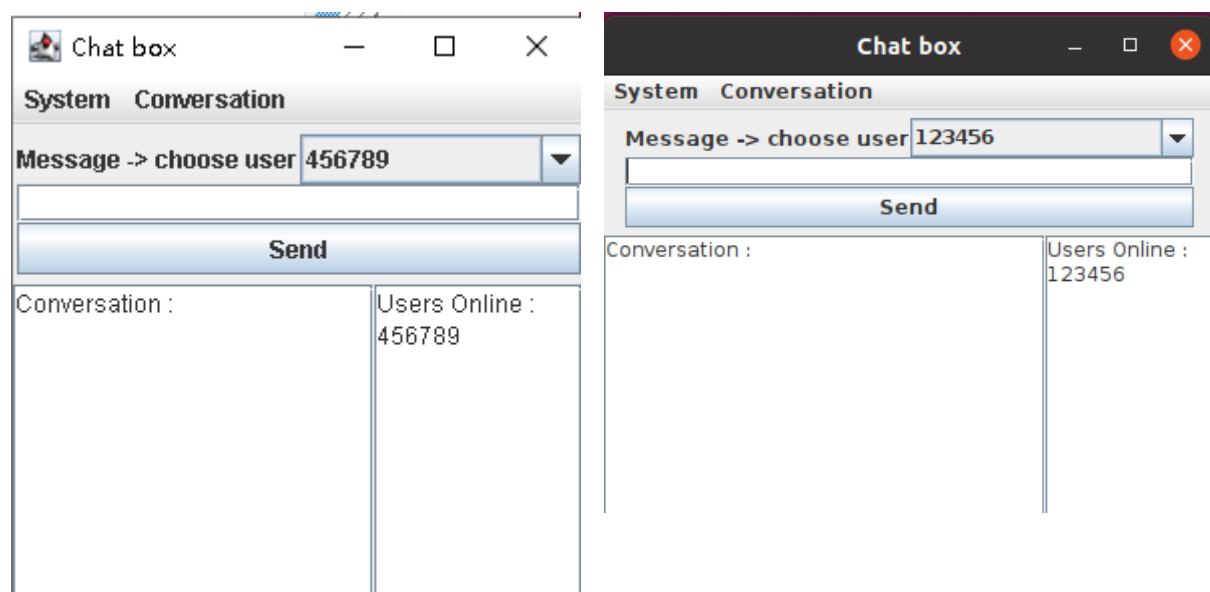
The left screenshot displays the "Login" section with the following fields and buttons:

- Username :** 456789
- Password :** masked with dots
- Nickname :** (empty)
- Create account** button

The right screenshot shows the same "Login" section but with the following changes:

- Sign In** and **Sign Up** buttons are present.
- A message below the buttons states: "Your account 456789 has been created successfully !!!"

Ensuite, nous faisons les mêmes étapes sur l'autre machine avec un compte ayant pour *Username* "123456" et *Password* "123456". On connecte ces comptes sur chaque machine et l'application va automatiquement changer la page vers celle de conversation. Quand les deux comptes sont connectés, les *nicknames* vont être affichés sur les autres machines connectées.

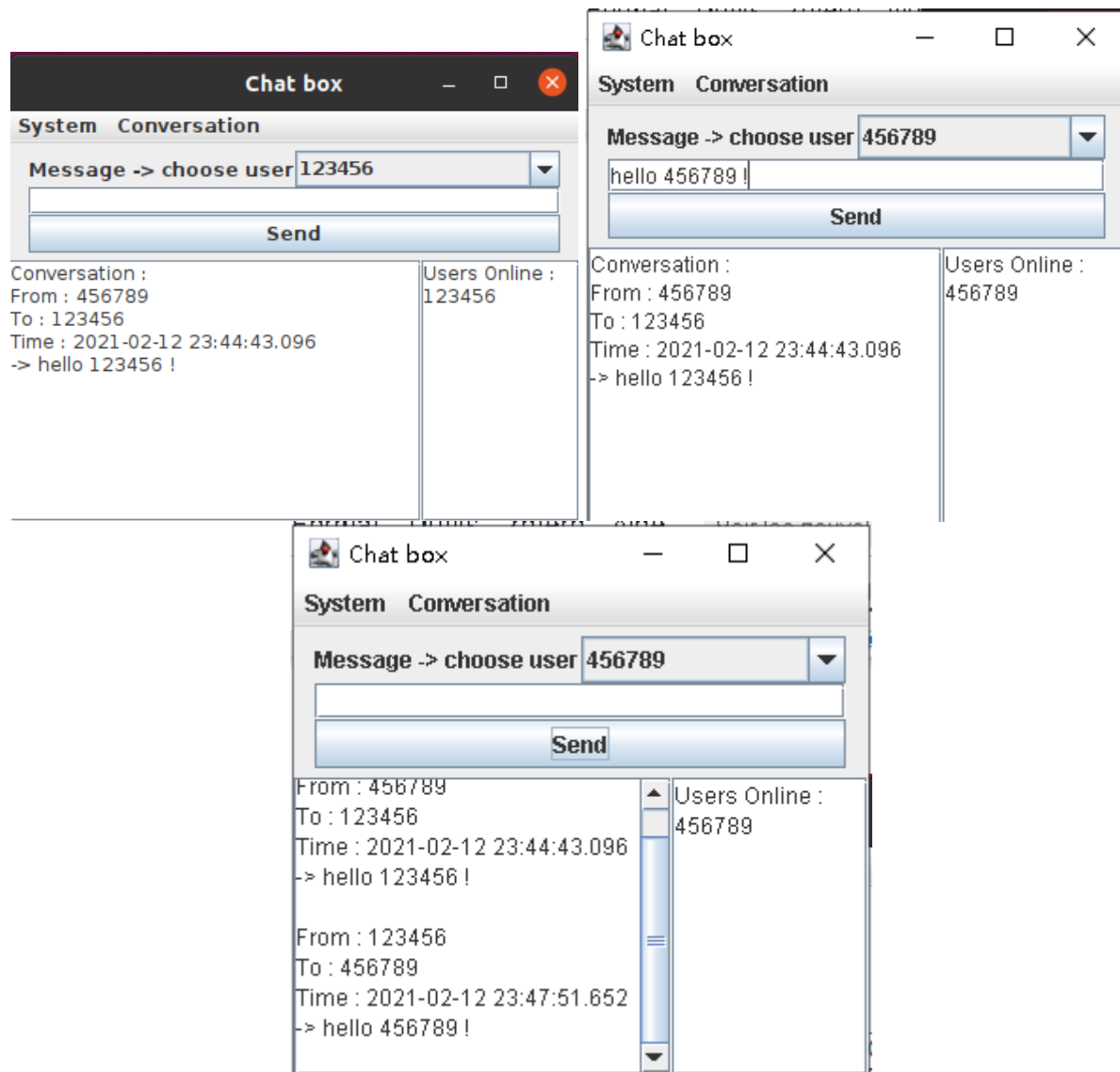


The image shows two side-by-side screenshots of the "Chat box" application window, displaying the conversation interface.

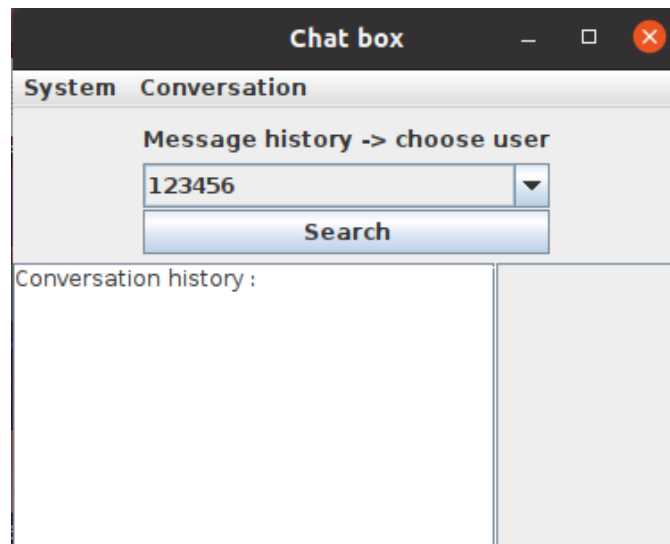
The left screenshot shows the "Message -> choose user" dropdown menu with "456789" selected. Below the dropdown is a "Send" button. The "Conversation" area is empty, and the "Users Online" list shows "456789".

The right screenshot shows the same interface but with "123456" selected in the dropdown menu. The "Users Online" list now shows "123456".

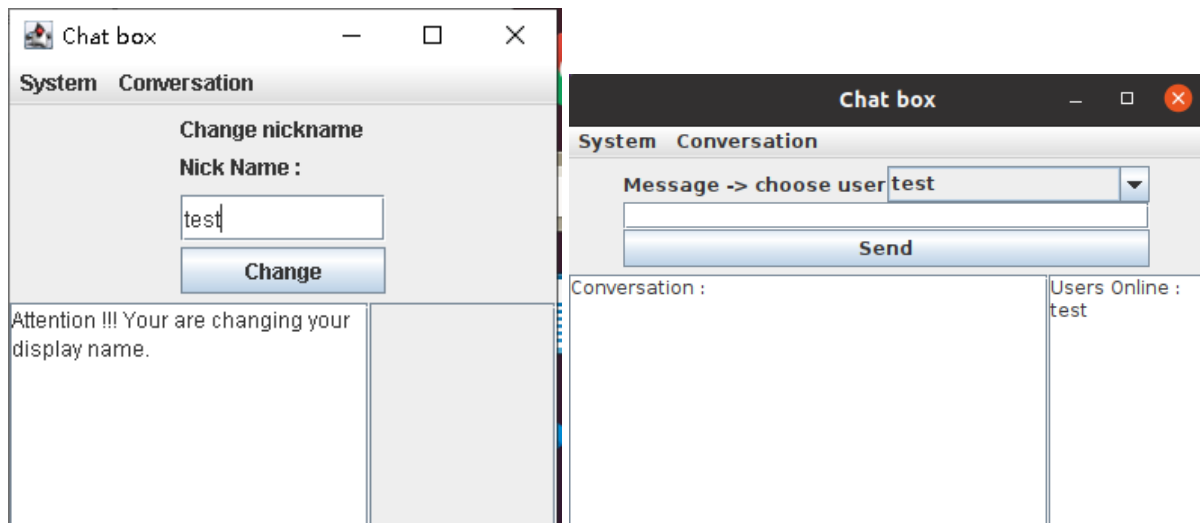
Ensuite, nous allons envoyer des messages entre les deux machines pour tester l'envoi et la réception des messages.



En même temps, l'utilisateur peut visualiser tous les messages qui viennent de l'autre par consultation de l'historique des messages.



Pendant qu'un utilisateur est connecté, nous changeons le pseudo de l'autre utilisateur qui est directement visible par les autres.



A la fin, quand un utilisateur est déconnecté (en cliquant sur “sign out ” dans le sous-menu ou la croix en haut à droite), il n’est plus affiché dans les listes d’utilisateurs connectés des autres utilisateurs.

Chat box

System

Conversation

Message -> choose user

Send

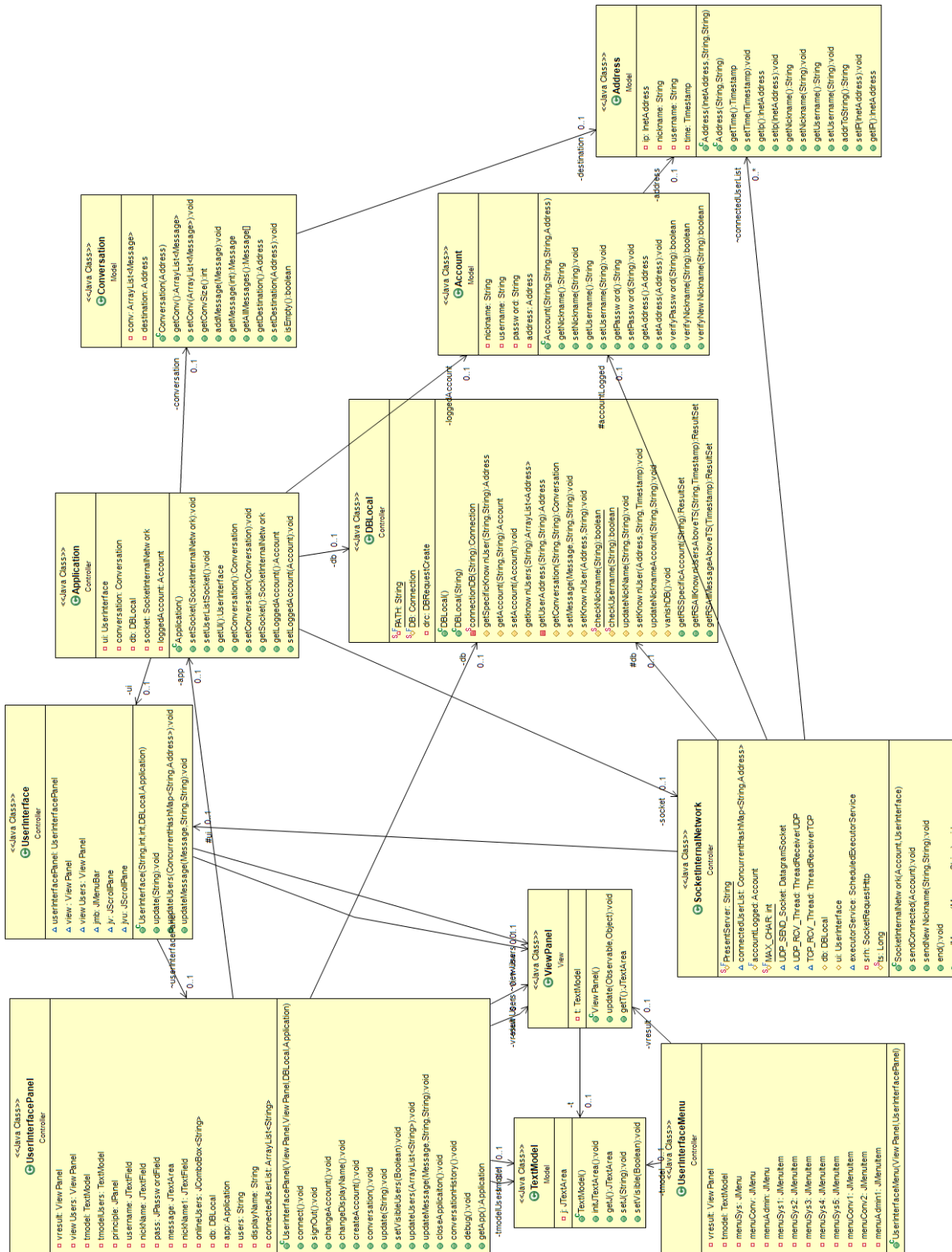
Conversation :

Users Online :

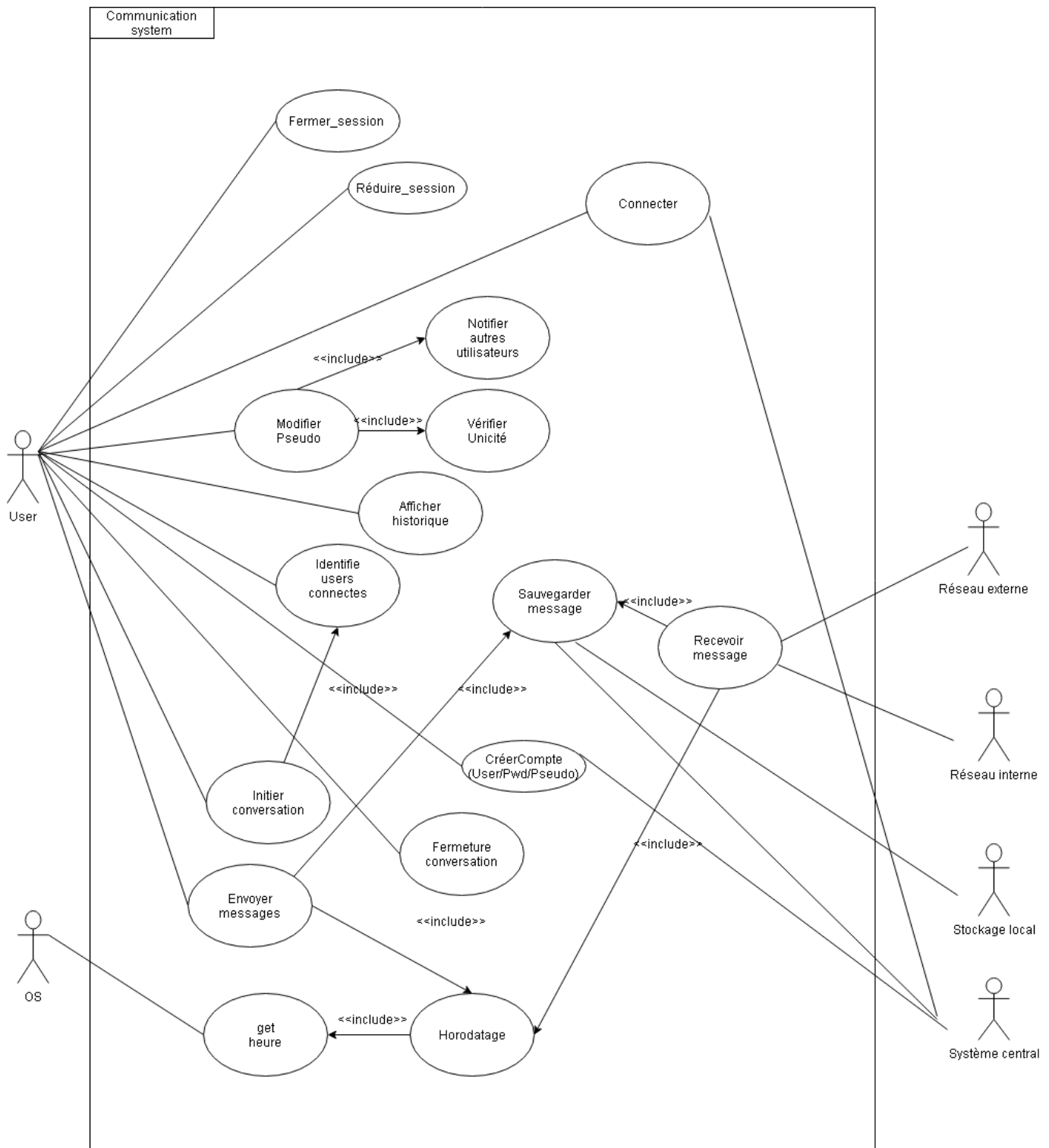
Table des annexes – Partie COO

Nous pouvons également trouver la partie COO sur le lien git où se trouvent tous les diagrammes réalisés.

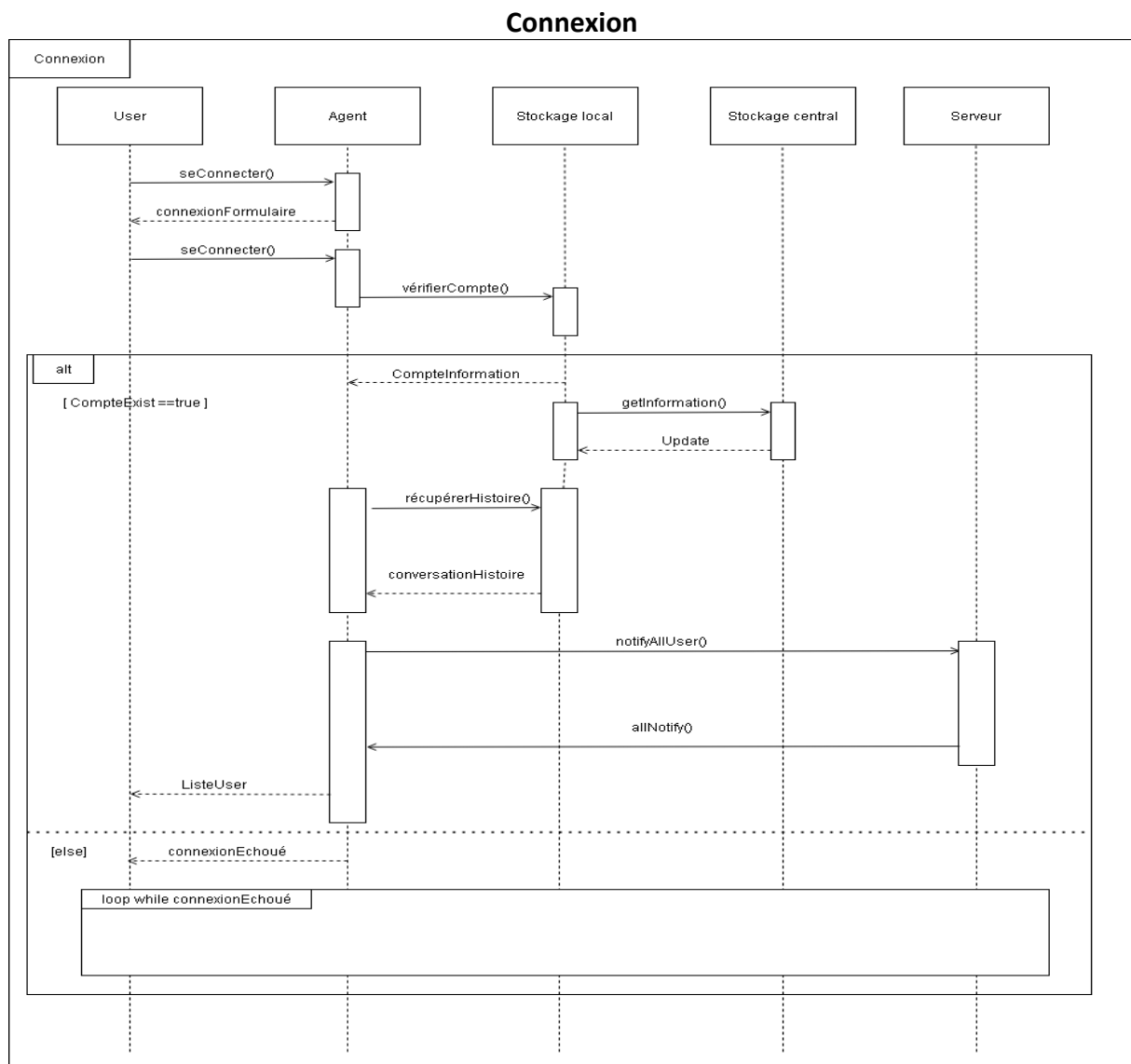
Annexe 1 : Diagramme de Bode



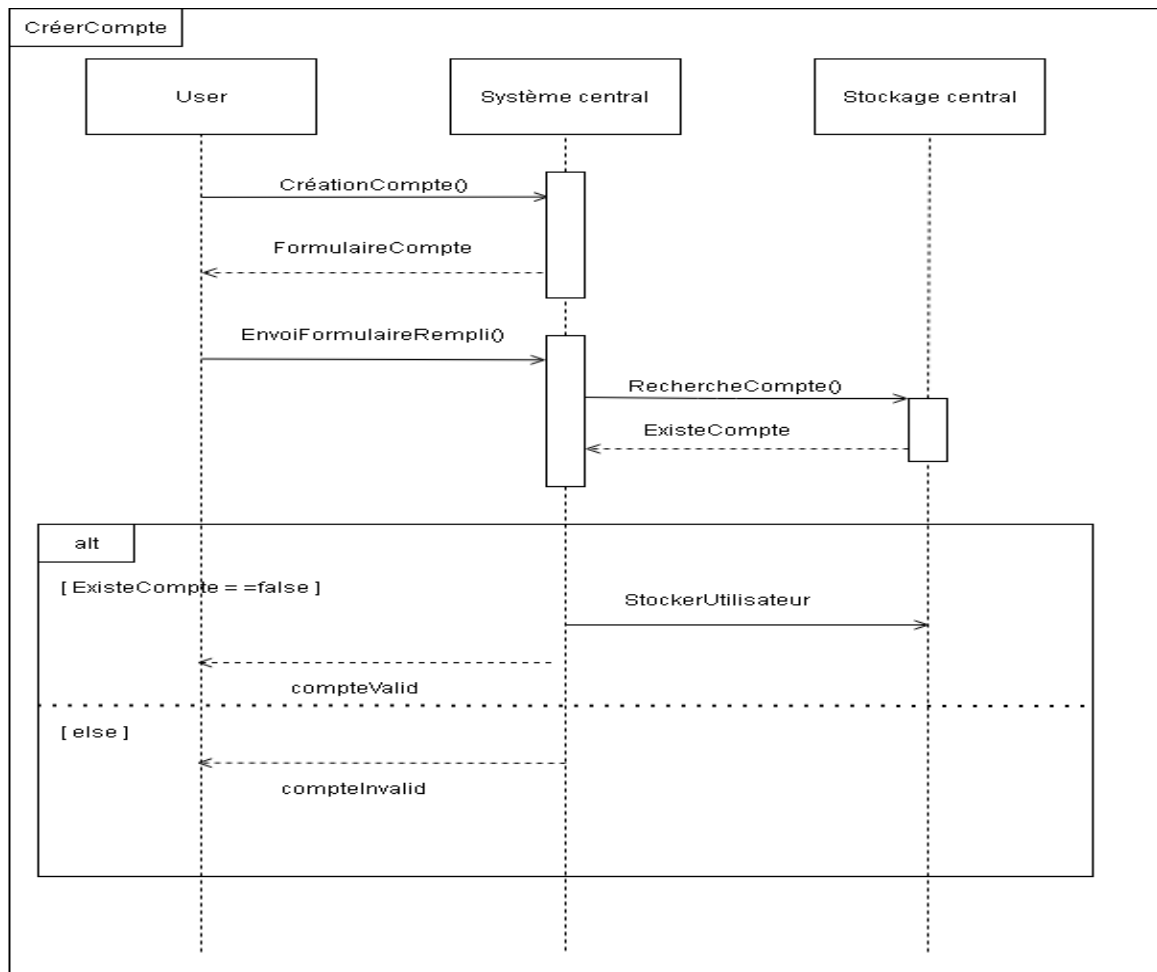
Annexe 2 : Cas d'utilisation



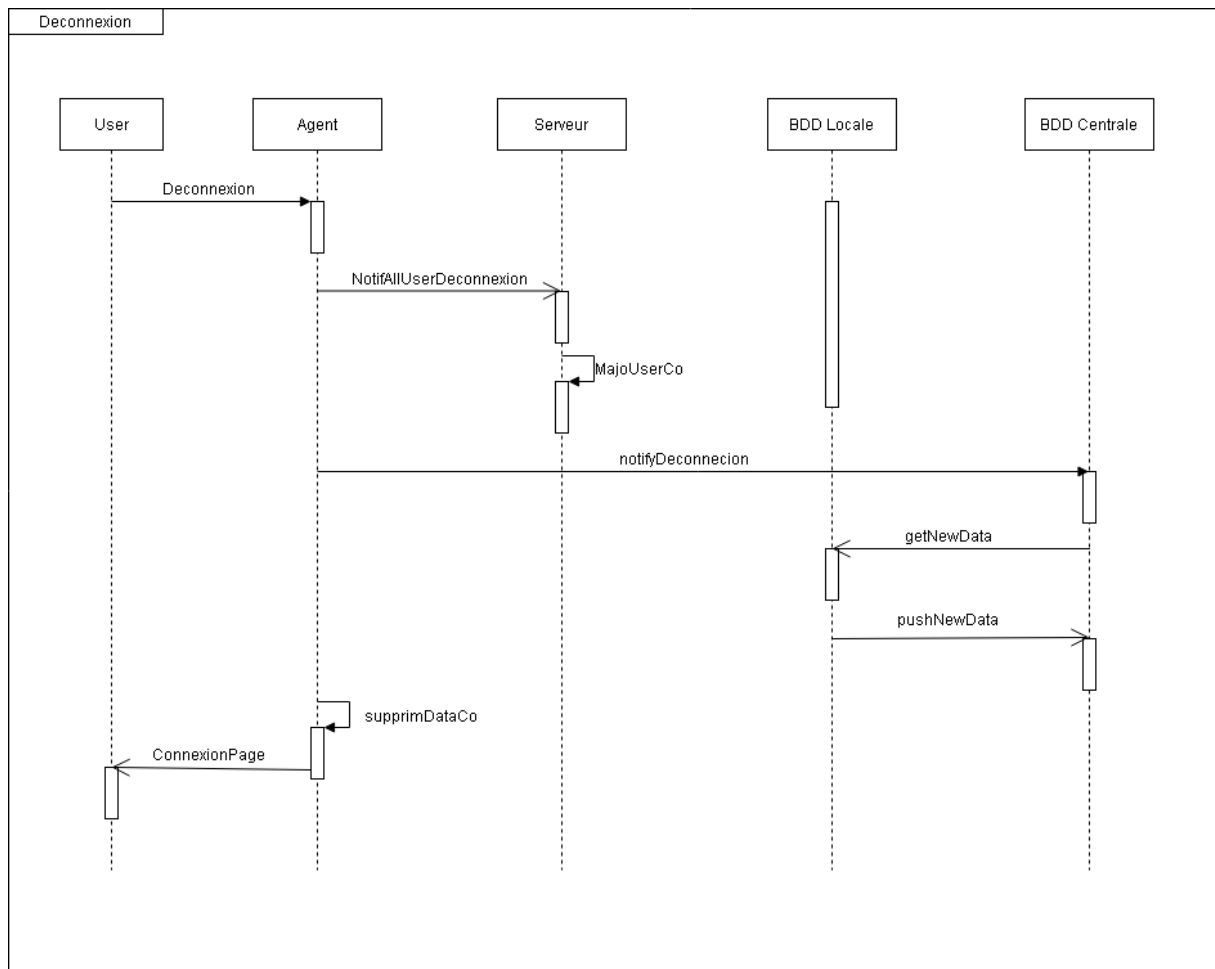
Annexe 3 : Diagramme de séquence



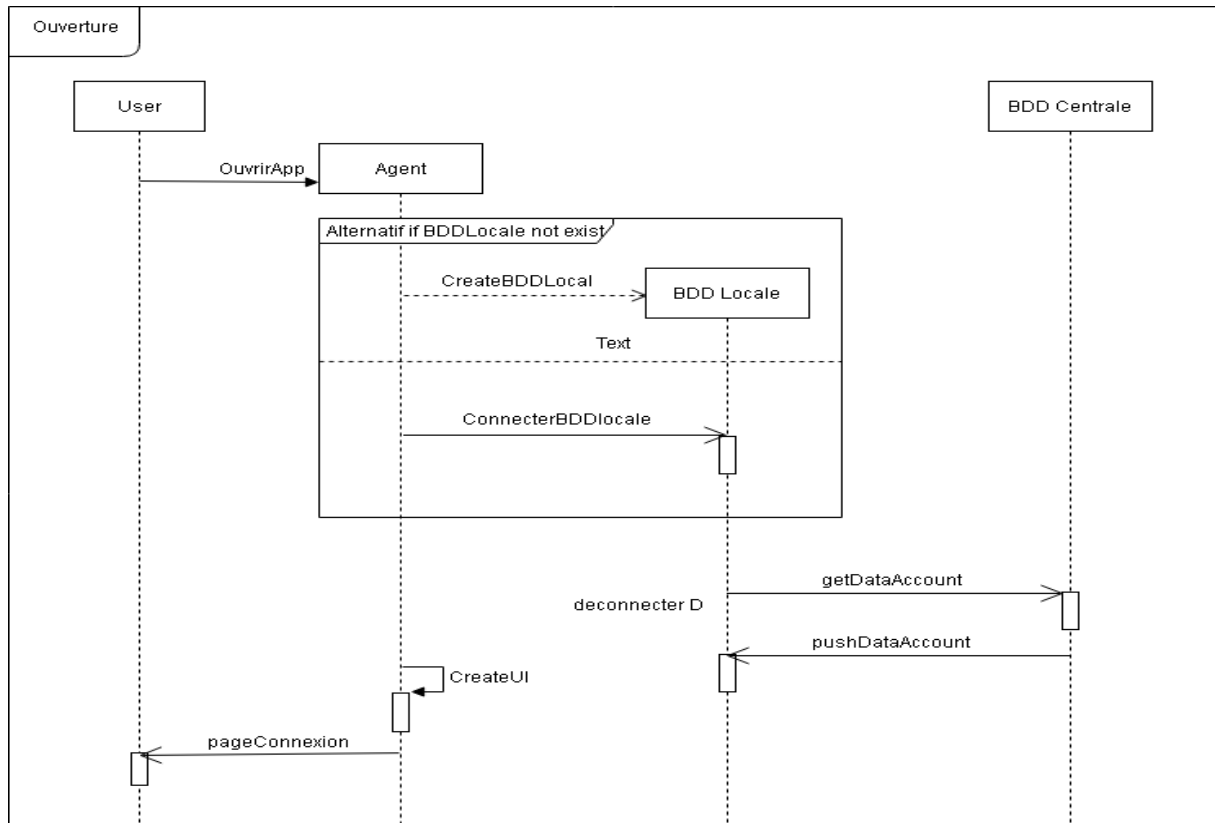
Création du compte



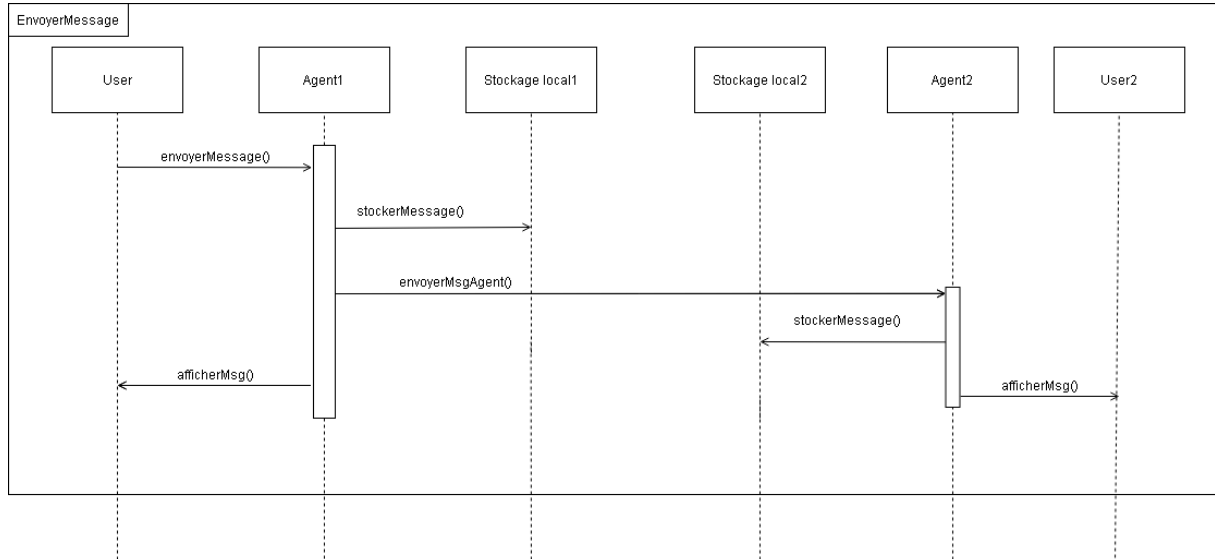
Déconnexion



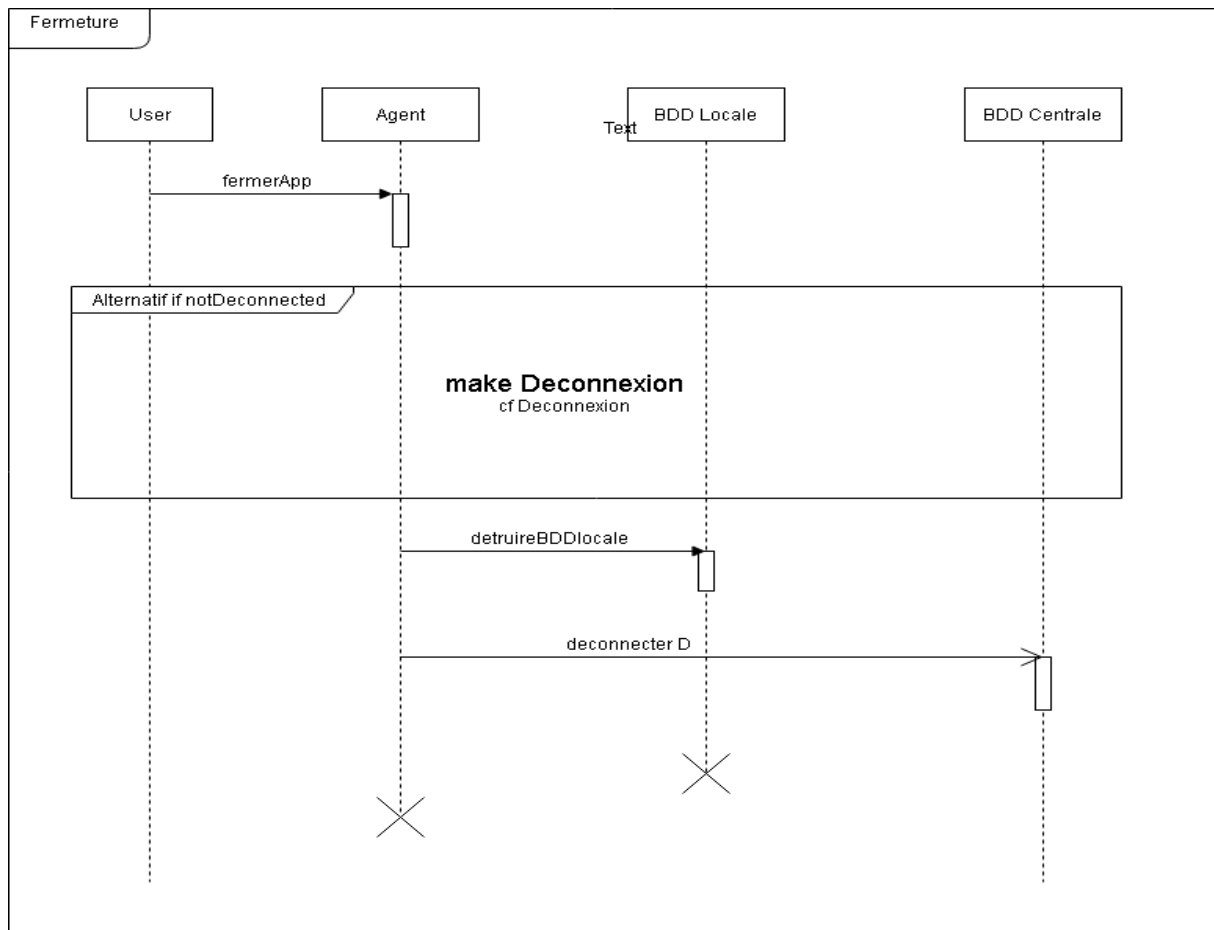
Ouvrir l'application



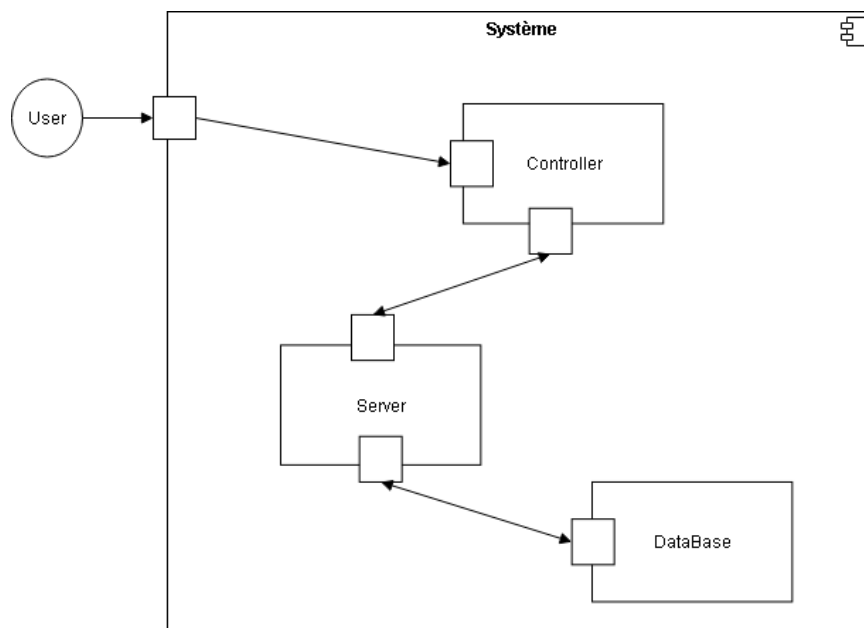
Envoyer de message



Fermer la session



Annexe 4 : Structure composite



INSA Toulouse

135, avenue de Rangueil
31077 Toulouse Cedex 4 - France
www.insa-toulouse.fr



MINISTÈRE
DE L'ÉDUCATION NATIONALE,
DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE