Rapport POO-COO

Thibault Poncetta & Youssef Amari – 4 IR 2020/2021

AGENT CHAT

Sommaire :

I – Architecture du projet

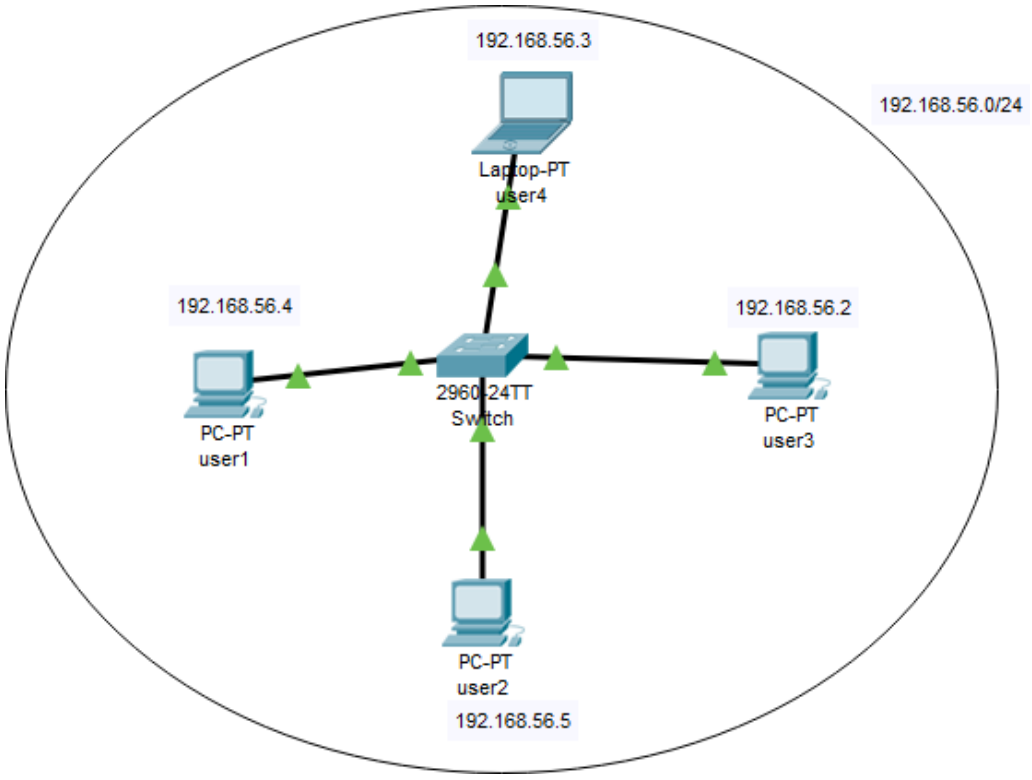
Afin de tester différentes étapes et fonctionnalités du projet, nous avons créer une architecture spécifique au projet.

Le dévelopement de l'UI s'est faite avec NetBeans, sous Java 11

A) Réseau local

Nous avons choisit de définir le réseau de « l'entreprise » avec la plage réseau suivante : ***192.168.56.0/24***

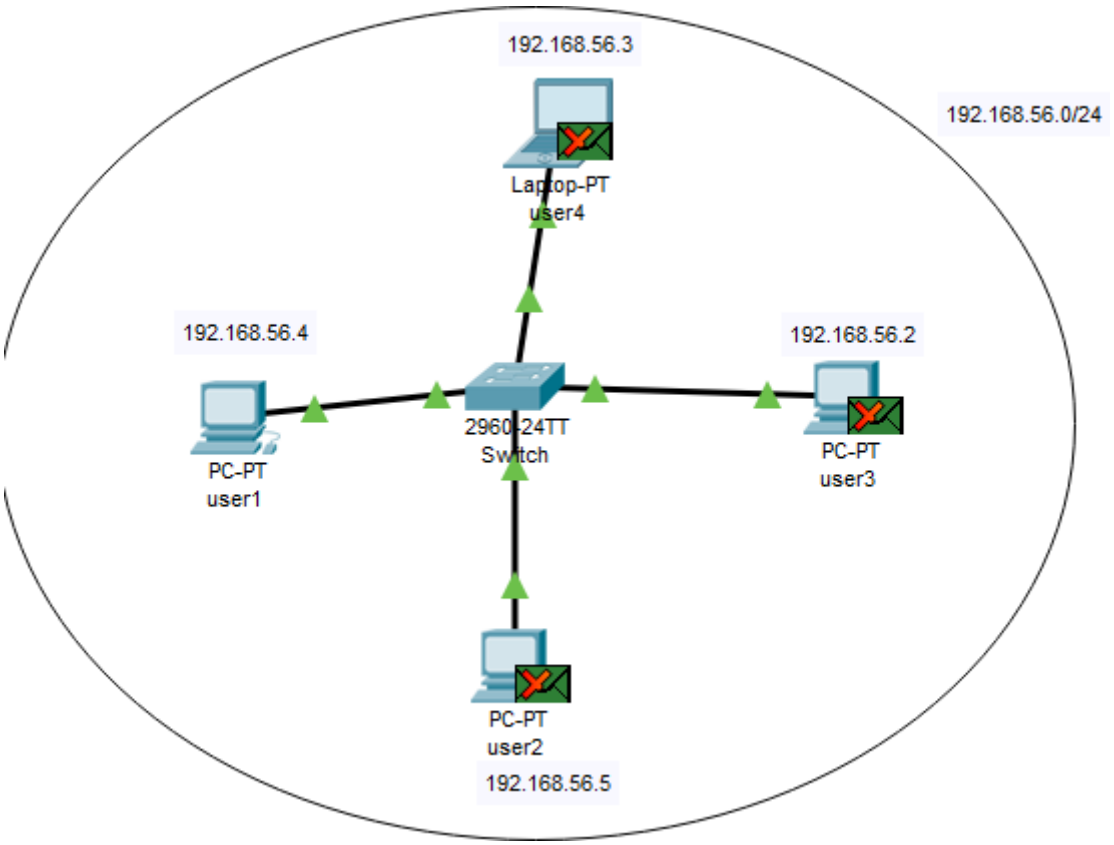
Toute machine n'appartenant pas à ce réseau sera considéré machine externe par le logiciel.



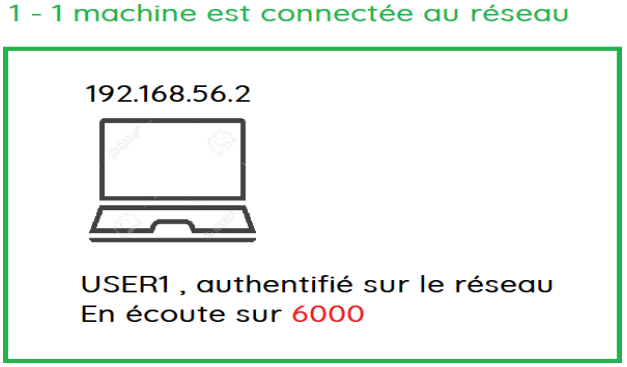
A-a) Authentification des utilisateurs.

Pour la méthode client , se réréfer à Manuel.pdf

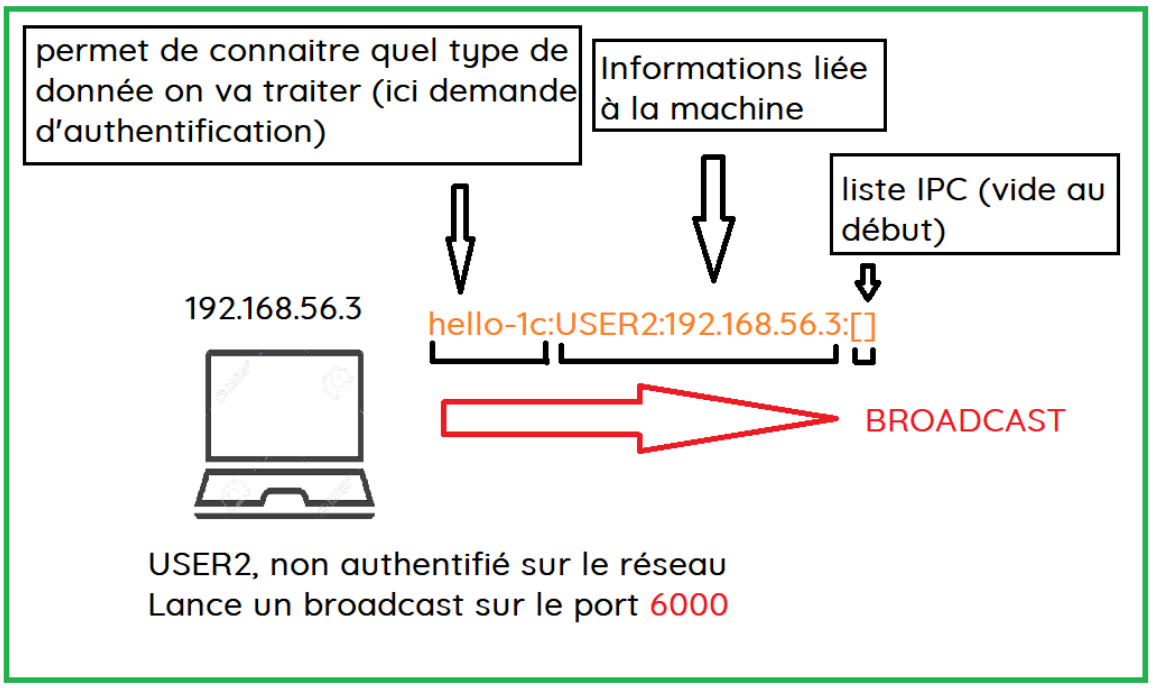
Supposons un réseau initialement null, la 1ère machine machine qui va se connecter au réseau sera automatiquement admise au réseau, car elle est la première.

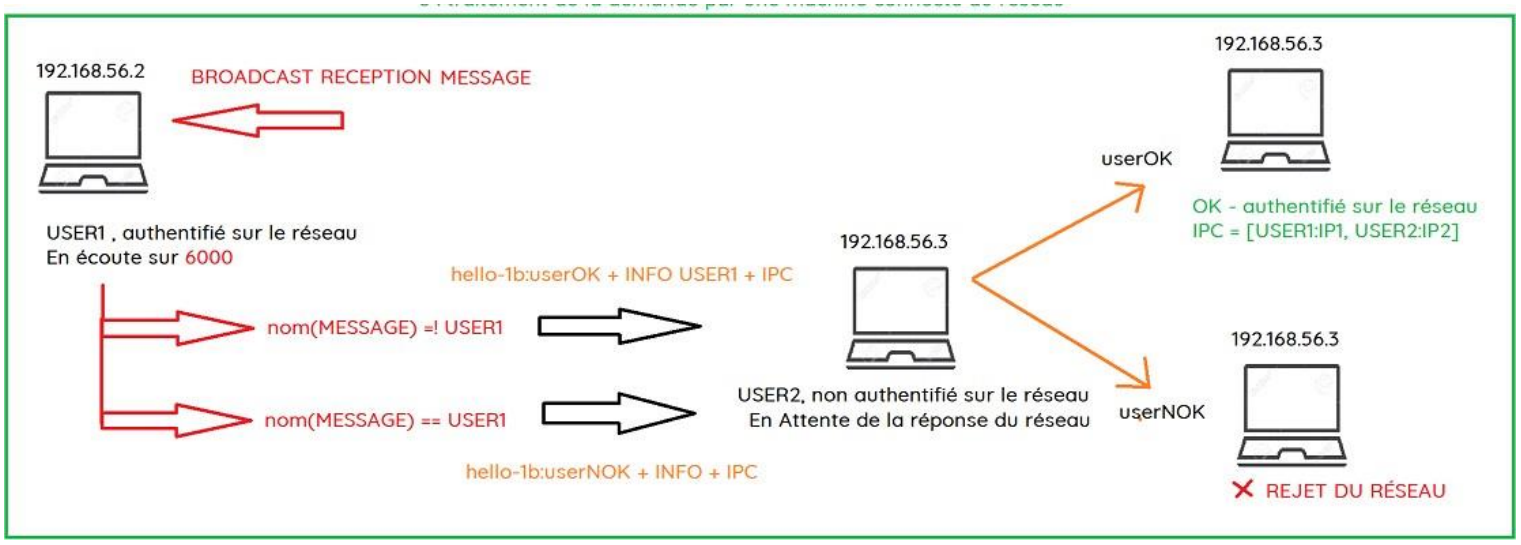


Procédure d'authentification : (UDP)



2 - Une deuxième machine broadcast ses informations





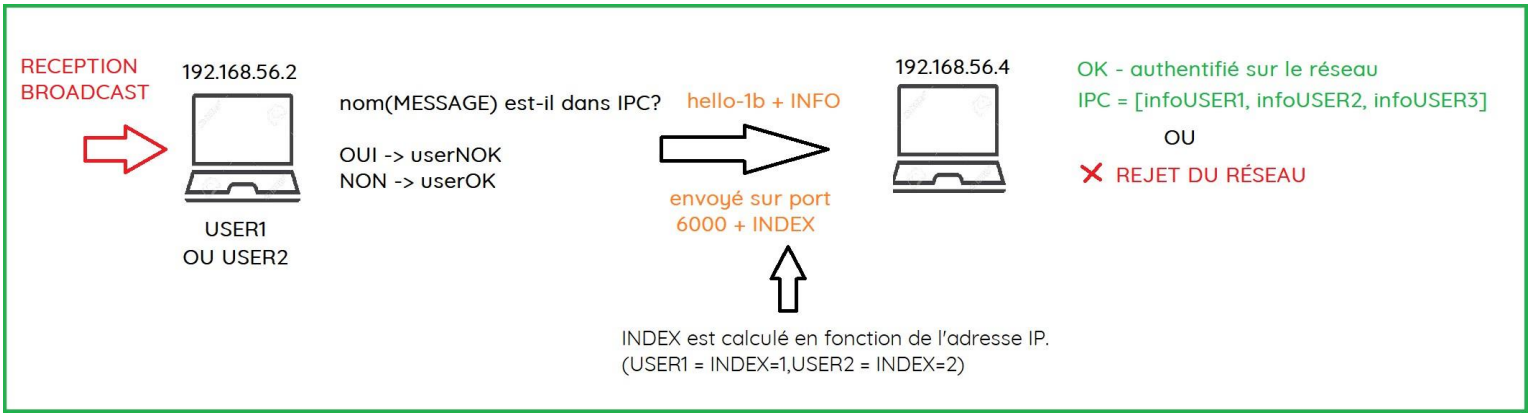
Quand USER1 reçoit la demande d'intégration au réseau de USER2 , si les noms n'ont pas de collisions (nom(USER1)= ! nom(USER2)), alors USER1 répondra de façon positive à USER2. USER1 ajoute également les infos de USER2 à son IPC local (Liste des utilisateurs du réseau) puis l'envoi dans le message de retour.

USER2 recevra ainsi un IPC avec les informations de USER1 ainsi que ses propres informations.

L'IPC est sauvegardé localement dans .**cache/userlist**

Ainsi, en suivant cette procédure on peux authentifier une nouvelle machine sur le réseau.

Dans le cas de plusieurs machines, on doit pouvoir écouter plusieurs réponses après le broadcast, ainsi on refait la même procédure mais on ouvre plusieurs port d'écoute pour accueillir les différentes réponses.



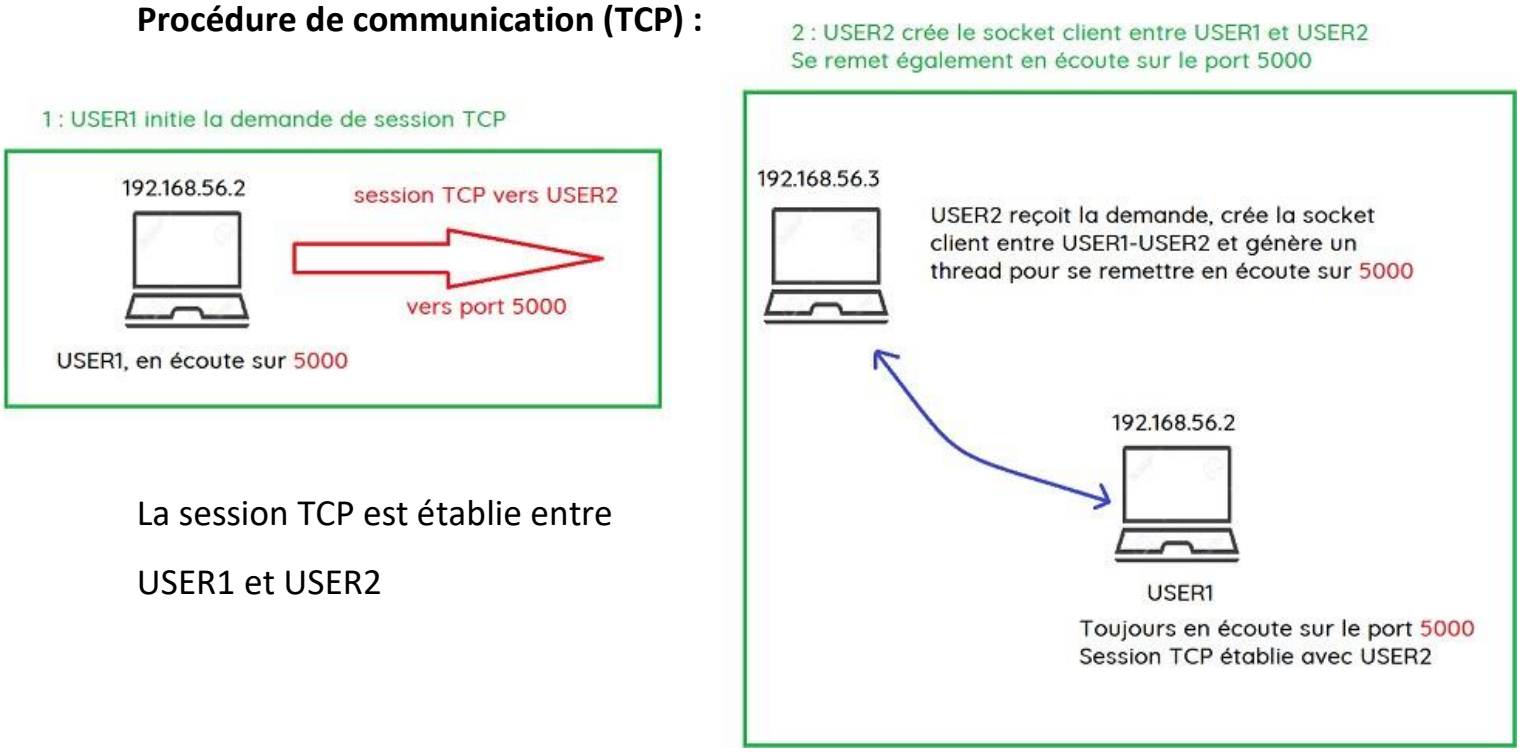
A-b) Communication des clients.

Supposons maintenant que notre liste IPC est formé , identique sur les 3 machines du réseau.

Les 3 noms sont affichés sur l'UI de chaque client.

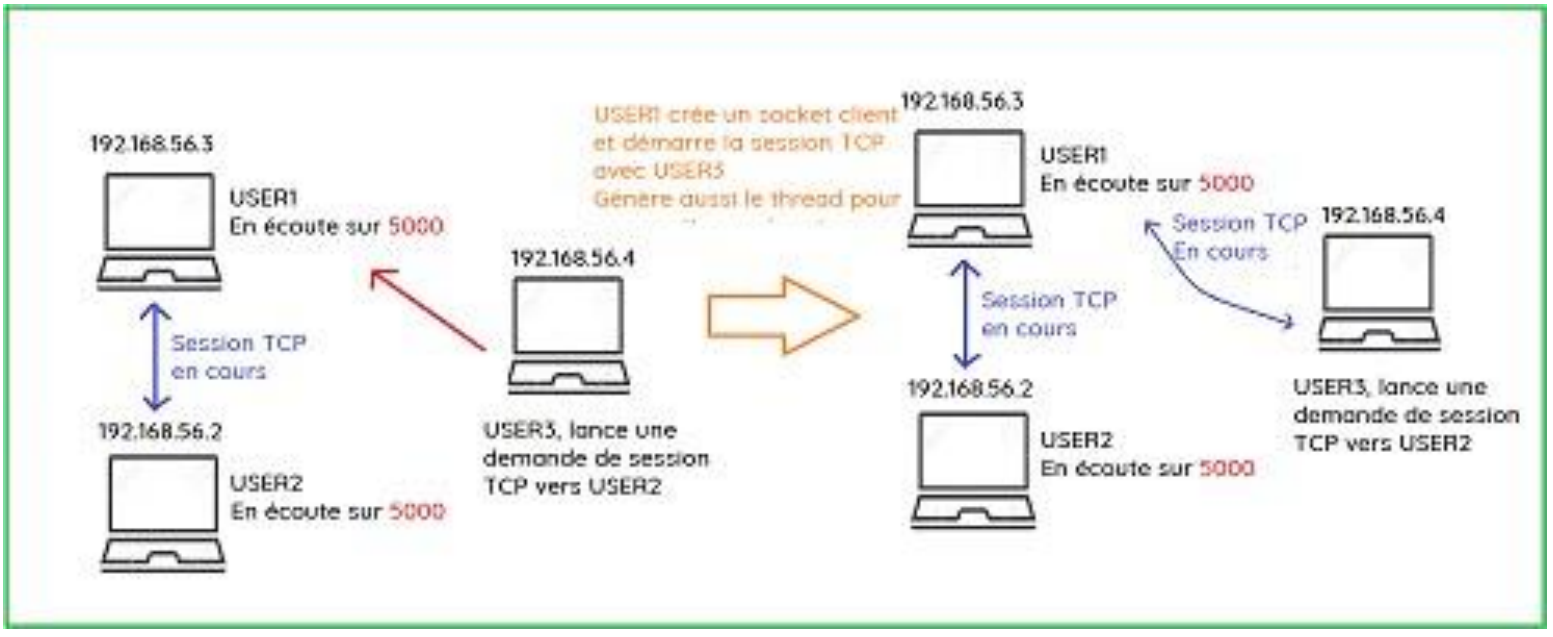
USER1 souhaite parler à USER2. USER3 souhaite également parler à USER2.

Procédure de communication (TCP) :



La session TCP est établie entre

USER1 et USER2



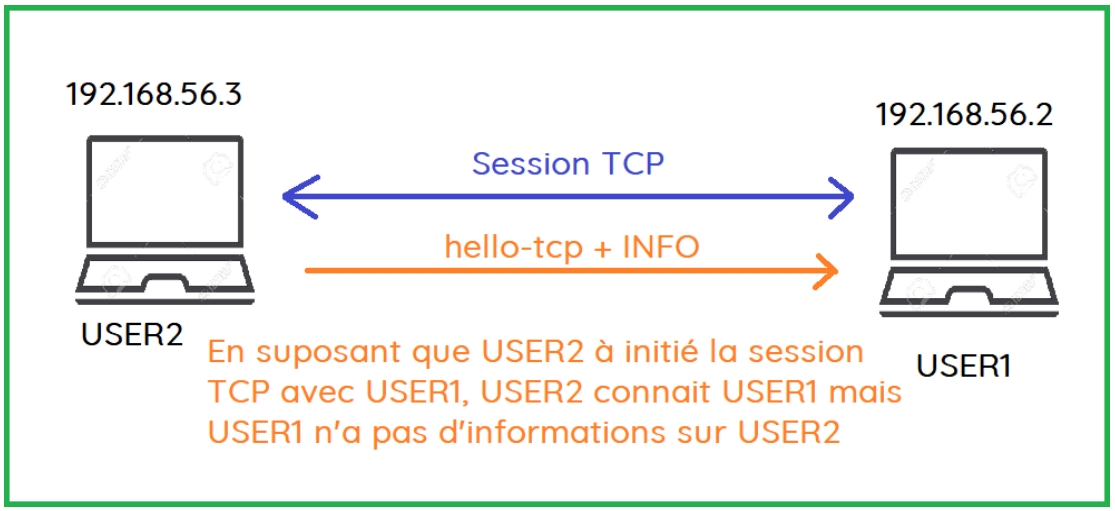
A-b-i) Transmission de fichiers :

Les fichiers sont encodés par base64 et envoyé au destinaire sous la forme d'un message. Un message avertit le destinaire de la reception du fichier (décodé à l'arrivée) dans le dossier : **/AgentChat/download**

A-b-ii) Sauvegarde des messages :

Nous avons choisit de sauvegarder l'historique des messages dans une base de données dans un serveur externe au réseau local (Serveur privé dans l'internet).

Nous utilisons la techonologie mySQL.



Parmi les informations qui seront transmisent dans le hello-tcp, cela inclut le nom et l'id de session.

L'id de session est un entier généré par USER2, il est sauvegardé dans un fichier **.cache/sessions** ou sont stockées la liste des couples :
emmeteur :destinataire :sessionID.

Ainsi, USER1 sauvegardera également le couple
emmeteur :destinaire :sessionID avec l'id de session fourni par USER2.

Cet id de session est nécéssaire, car tout les messages envoyés dans la session TCP entre USER2 et USER1 seront sauvegardés dans la base de données, associés avec l'id de session.

Ainsi au démarrage d'une nouvelle session, le logiciel va regarder le COUPLE emmeteur :destinaire et si il existe déjà un couple éxistant dans le fichier sessions local, on va parcourir la base de donnée pour récupèrer les anciens messages et les afficher.

```
root@localhost:~# mysql -u webchat -p webchat
Enter password:
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 172
Server version: 8.0.22-0ubuntu0.20.04.3 (Ubuntu)
```

Nous nous connectons à notre base de donnée webchat avec l'utilisateur webchat. (Nous avons créer un utilisateur qui permet de transmettre des données à l'exterieur du serveur, qui est par défault interdit).
L'ouverture du port 3306 sur le firewall est bien sur nécéssaire.

```
mysql> select * from message_history;
+-----------+-----------------------------------------------------------+
| sessionid | message                                                   |
+-----------+-----------------------------------------------------------+
|      8382 | 30/01/2021 18:25:25 : Youssef > bonjour                   |
|      8382 | 30/01/2021 18:25:26 : Youssef >                           |
|      8382 | 30/01/2021 18:25:37 : Youssef > hello                     |
|      8382 | 30/01/2021 18:25:47 : Mirasio > salut                     |
|      8382 | 30/01/2021 18:26:22 : Mirasio > je t'ai envoyé une image  |
+-----------+-----------------------------------------------------------+
5 rows in set (0.00 sec)
```

Les Messages sont stockée dans la table **message_history**, dont la structure est int sessionid , varchar(100) message

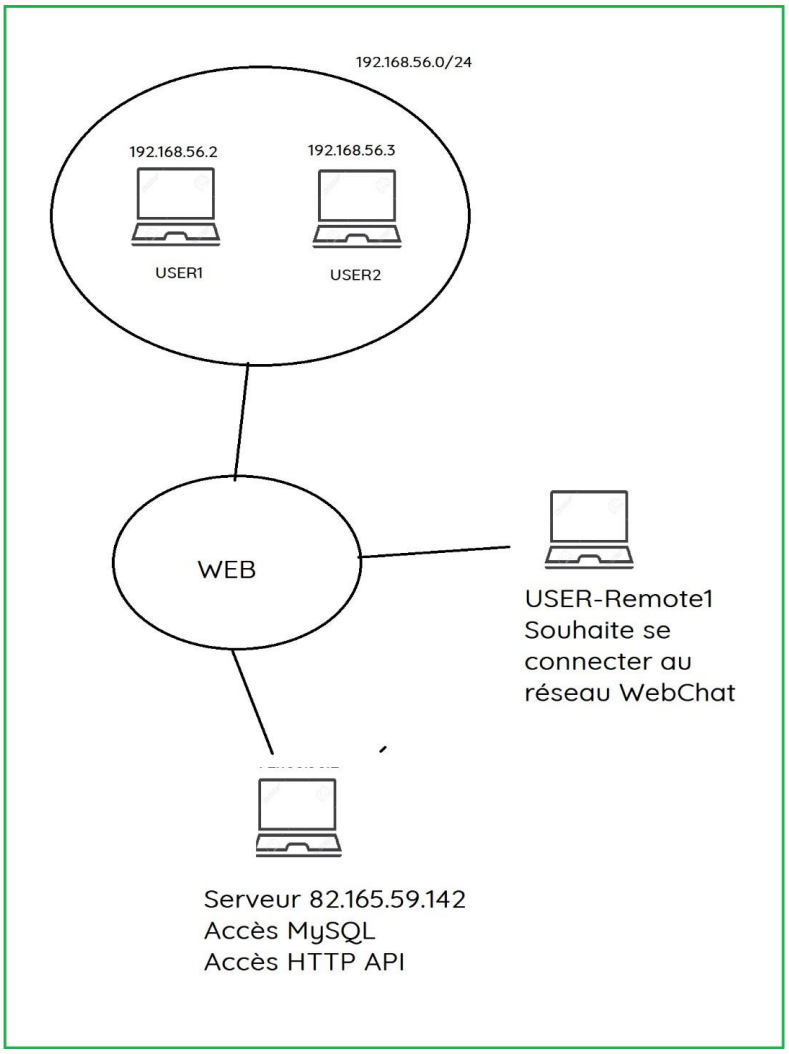
B) Machine Externe au réseau local

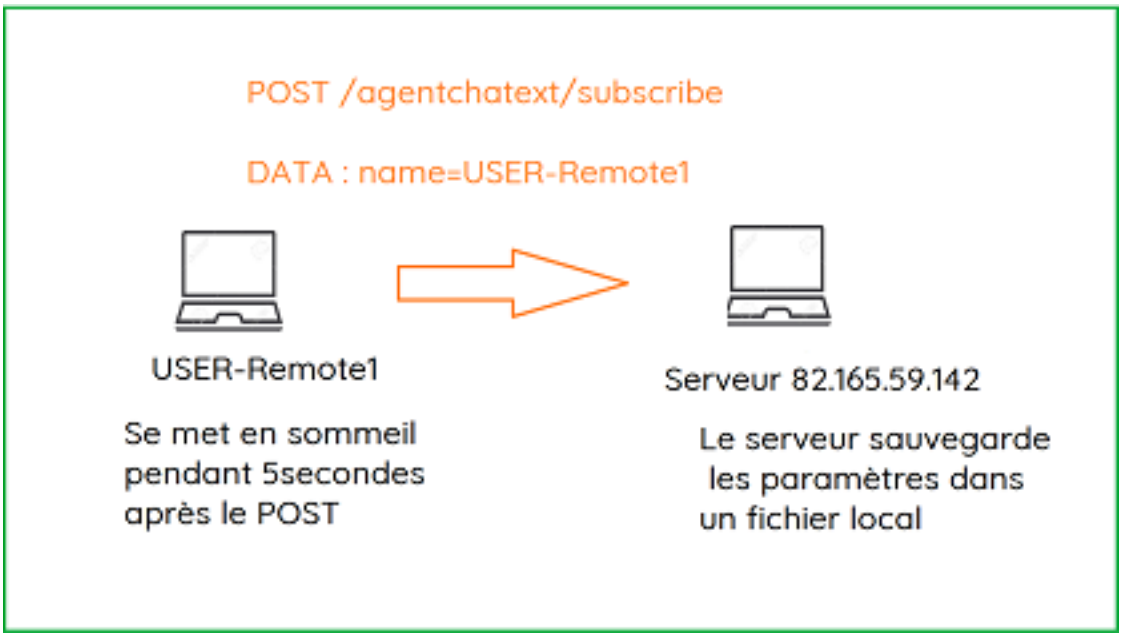
Notre servlet est installé sur notre serveur privé sur internet, et dispose d'un serveur **tomcat.** Nous avons créer une API avec plusieurs endpoint , dont l'utilisation est expliqué dans la procédure de découverte des utilisateurs.

Nous utilisons une classe utilitaire dans la servlet pour servir de logger pour la servlet. (Log4J ou d'autre logger ne marchait pas)

Les exceptions, Info sont sauvegardées dans **/tmp/errors et /tmp/info**

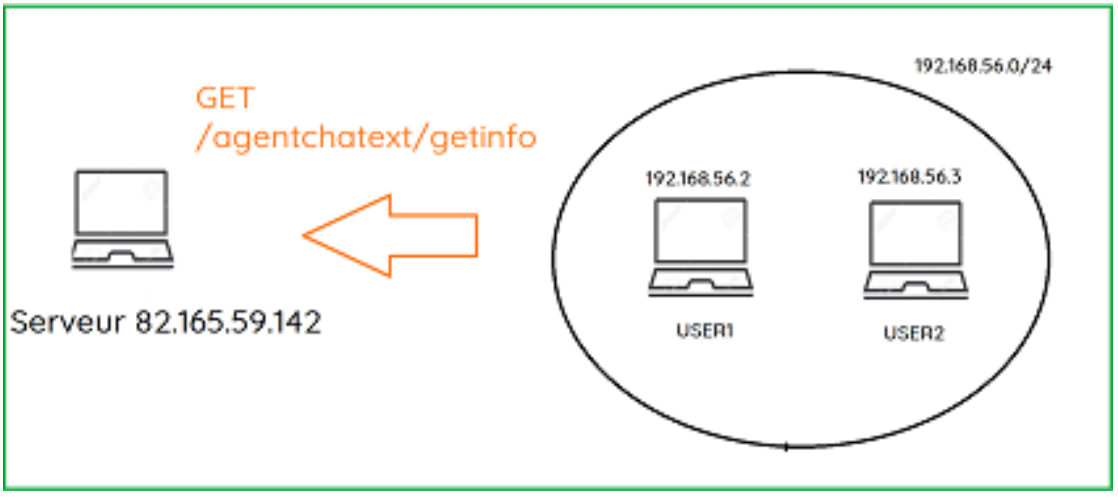
Procédure découverte des utilisateurs : (API HTTP)





L'utilisateur distant envoie son nom au serveur, qui va sauvegarder cette valeur dans un fichier local **(/tmp/param**).

De façon périodique, les machines du réseau local vont demander aux serveurs si des utilisateurs distant souhaitent s'authentifier :

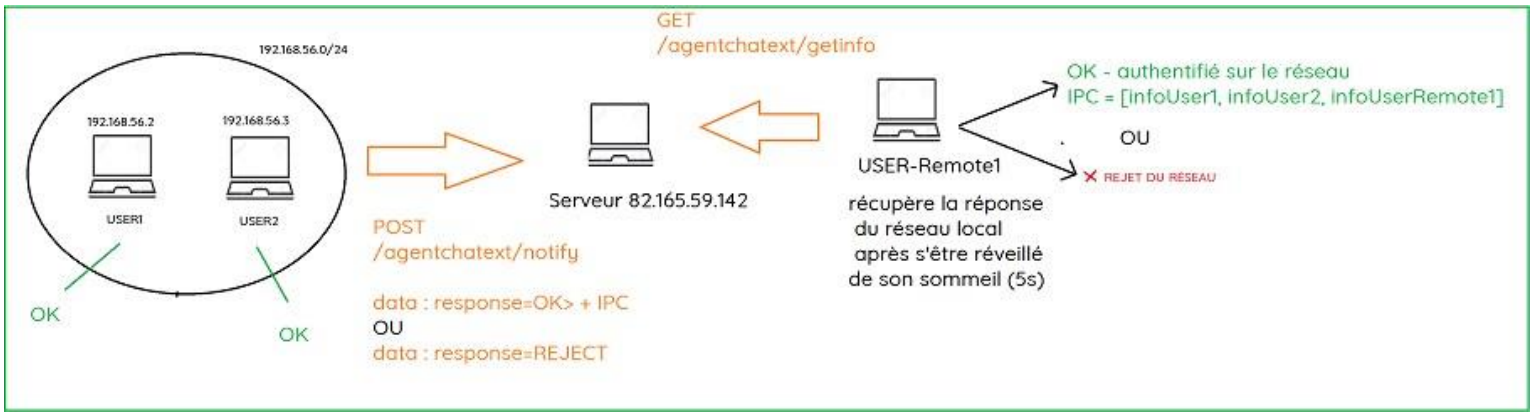


L'API /getInfo fournit le contenu du fichier /tmp/param du serveur, nous récuperons le paramètre que l'on souhaite coté client.

L'appel du getInfo se fait périodiquement (toute les 3s)

Pour éviter de refaire le traitement au prochain appel, le paramètre name est défini à null juste après la récuperation du nom de la machine distante par les machines du réseau local.

Cette procédure asynchrone est obligatoire, car le réseau local n'est pas routable directement depuis l'internet.



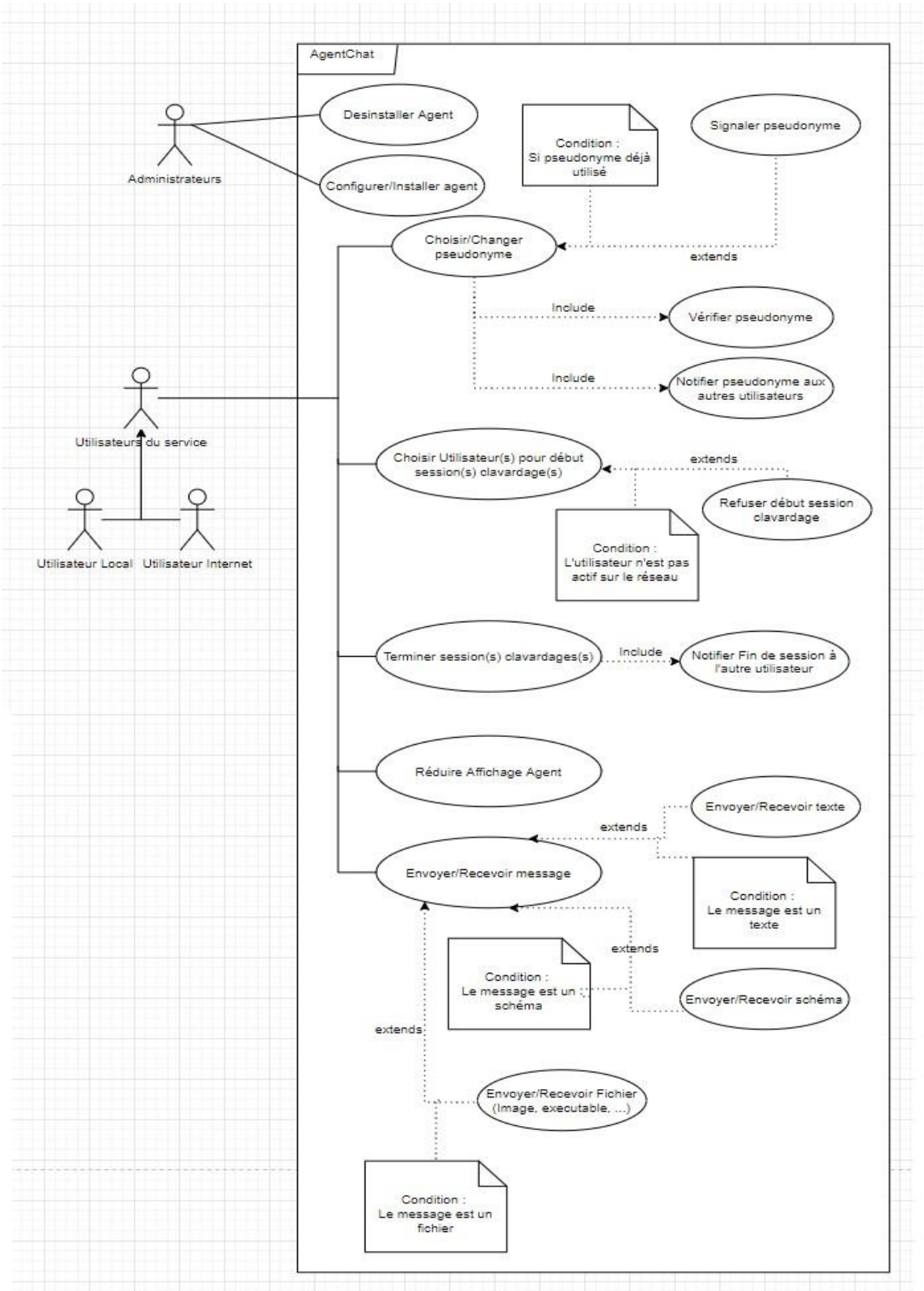
Une fois que le nom est arrivé dans les machines du réseau local, la procédure est exactement la même que pour le réseau local de base, il y'a vérification du nom dans la liste IPC, et les machines vont renvoyer leur réponse au serveur (toutes les machines auront les même réponse car leur IPC est identique).

La réponse est sauvegardé dans le serveur dans /tmp/param.

La machine distante va ensuite récupèrer la réponse après ses 5secondes de sommeil (sommeil déclanché après le premier /subscribe), puis elle pourra ensuite soit être accepté sur le réseau et donc avoir l'ipc du réseau local (si la machine est accepté, ses informations sont ajoutées dans l'iPC du réseau local), soit se faire rejeter et devra recommencer la procédure avec un nom différent.

II – Diagrammes COO

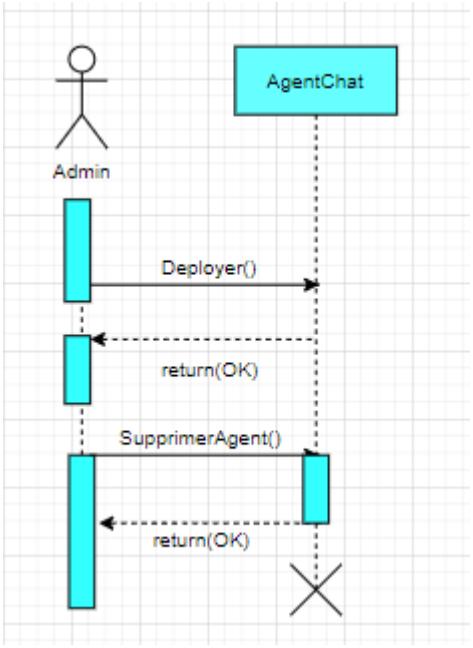
A) Use case diagram



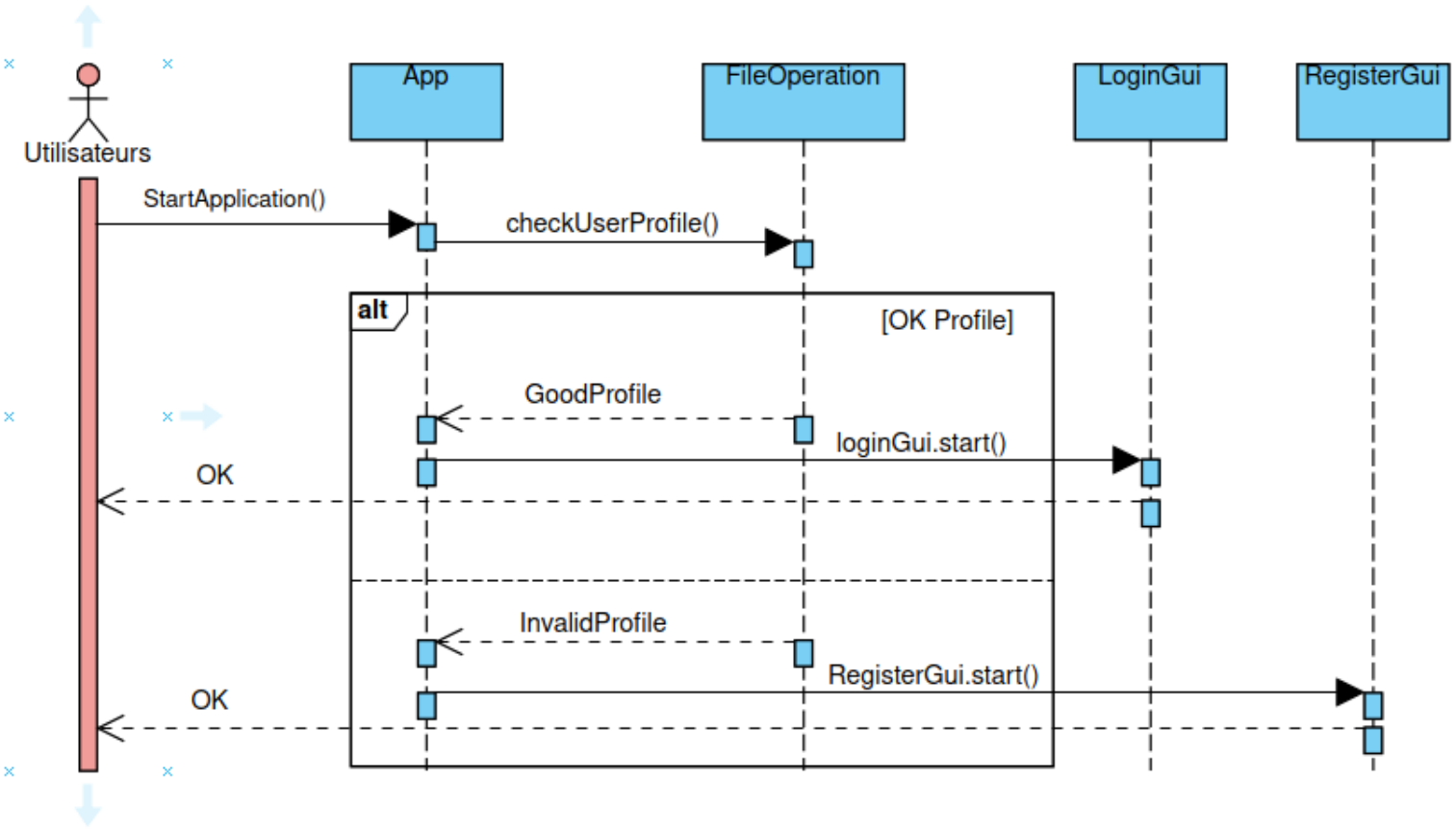
B) Sequence diagrams

La conception étant fait en amont de la programmation, l'ajout de différentes fonctionnalités ainsi que la modification du cahier du charge en cours du projet nous ont menés à refaire la conception. Le diagramme le plus important étant le diagramme de séquence et de classe, nous avons refait ces diagrammes en priorité et les autres sont restés inchangés.

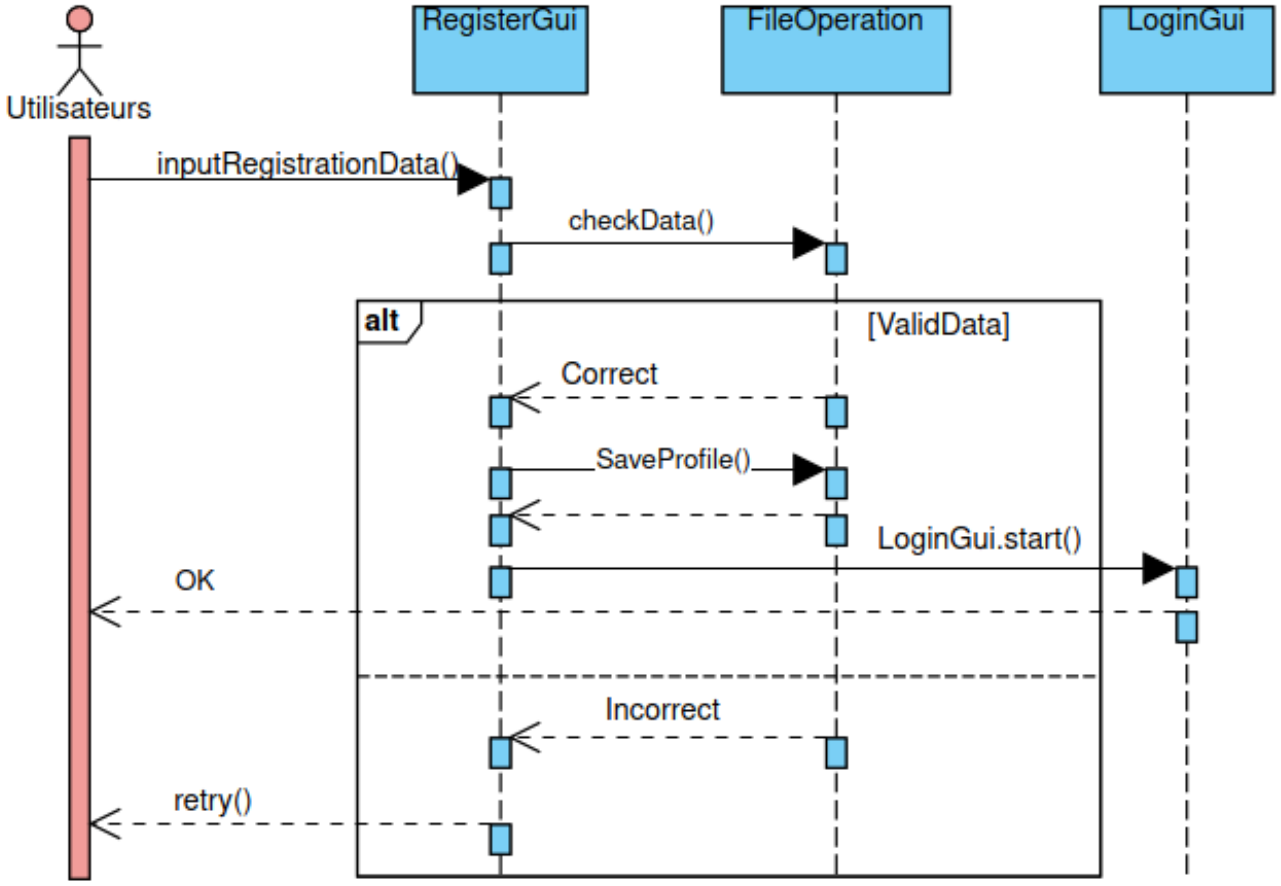
Déploiement / Suppression :



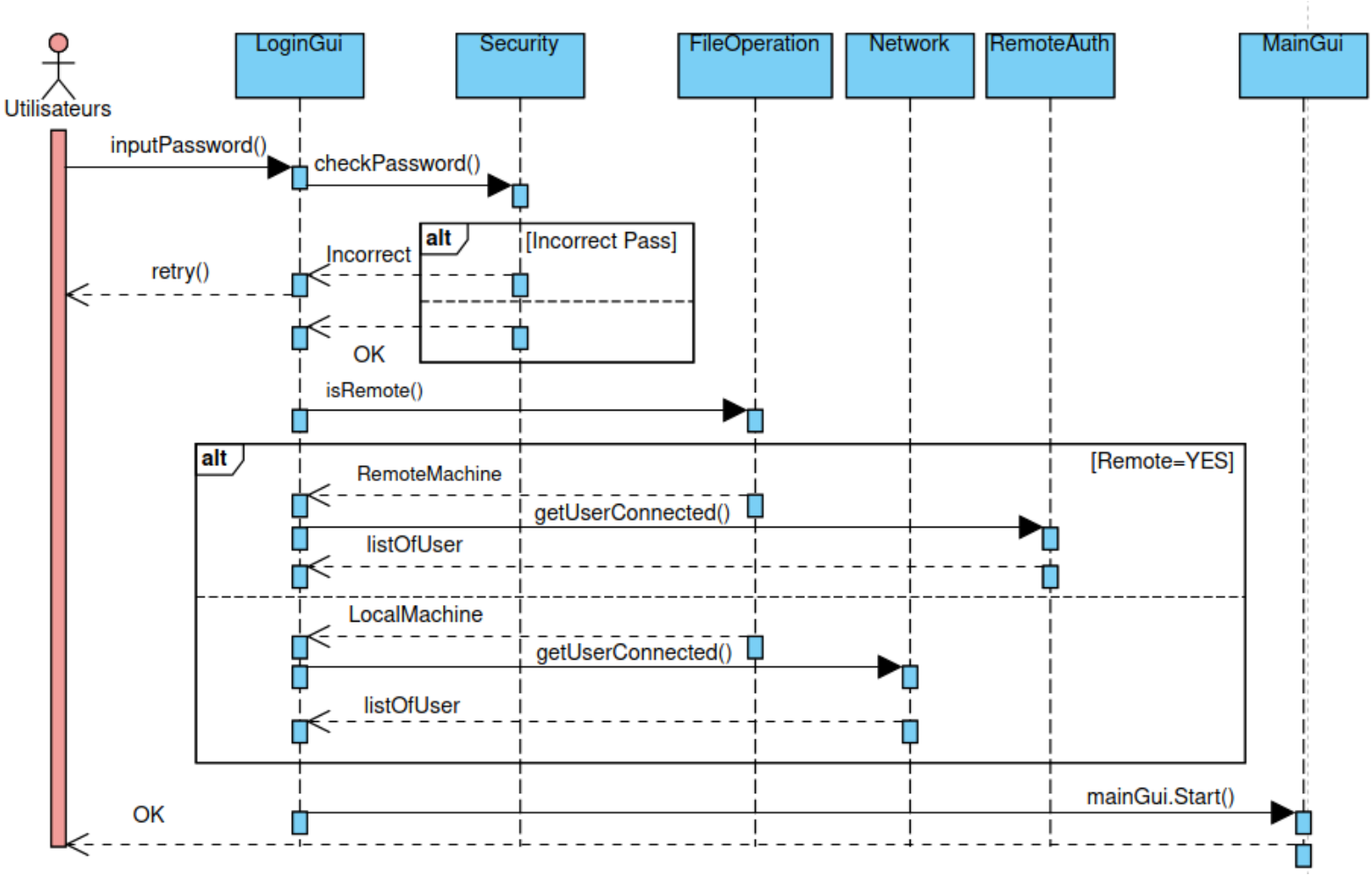
Démarrage Application :



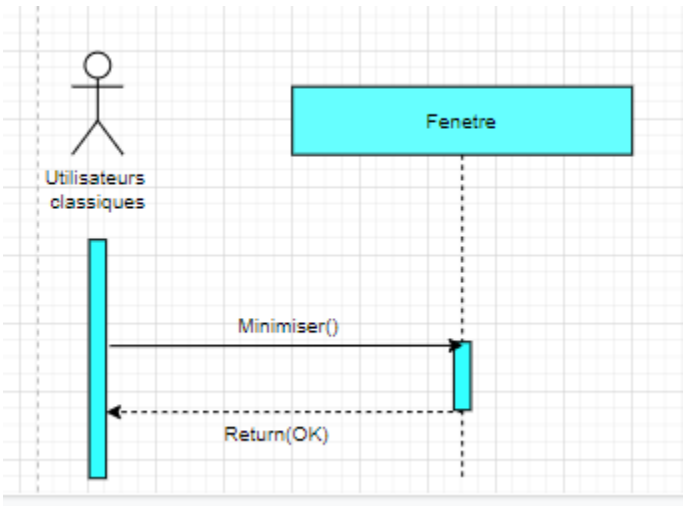
Inscription Utilisateurs :



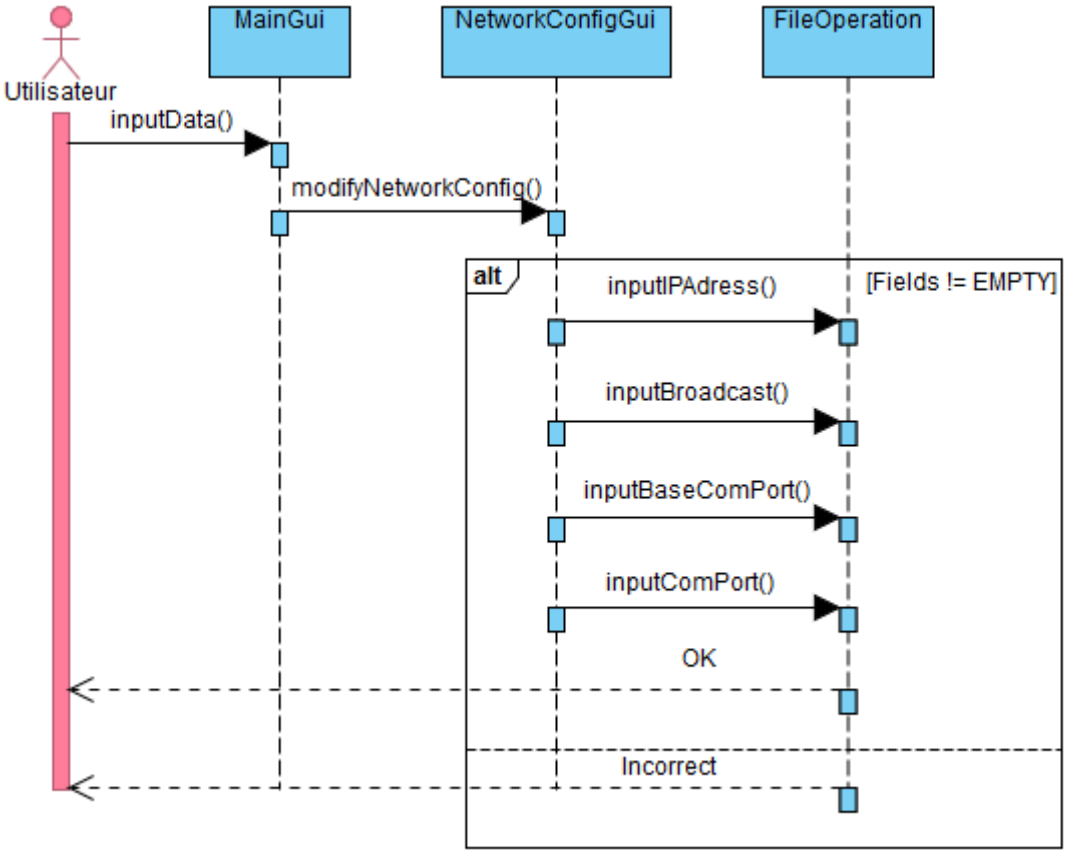
Authentification des utilisateurs Machine Side :



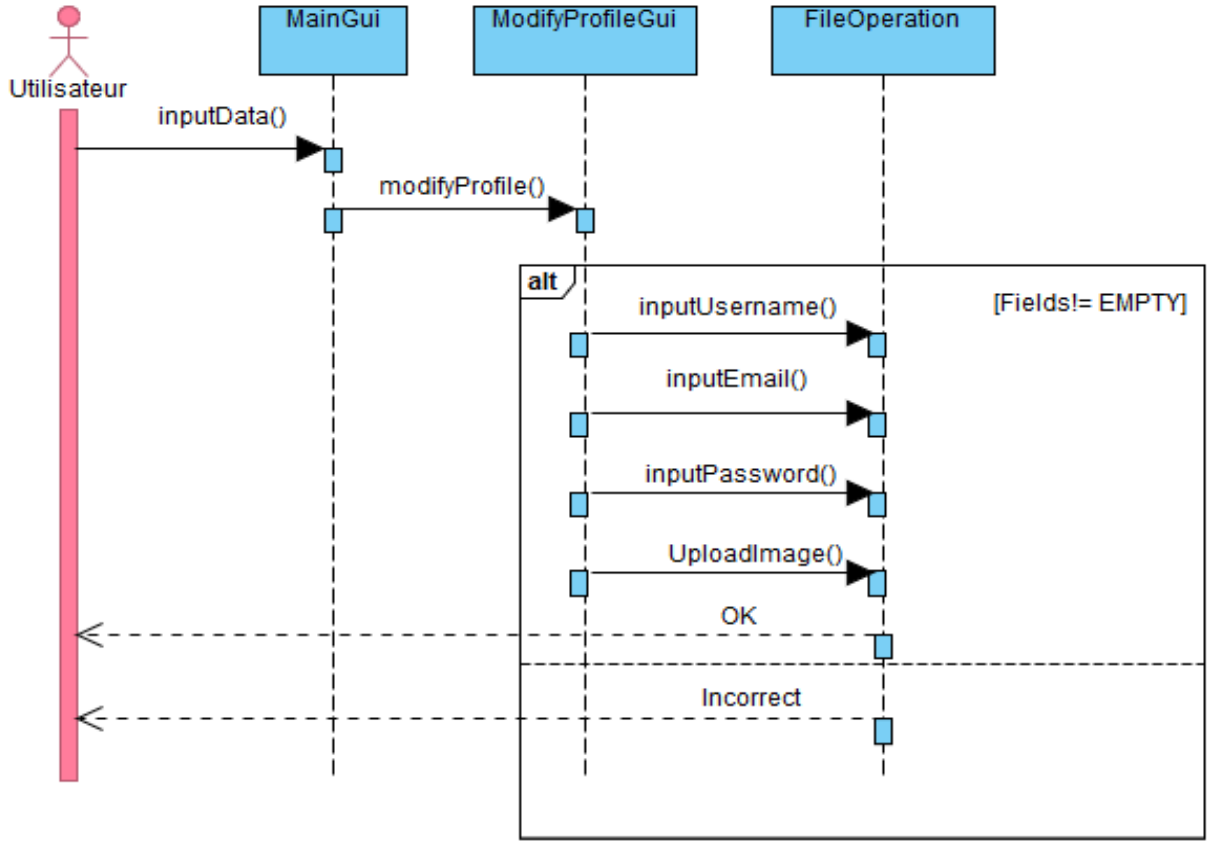
Gestion de la fenêtre :



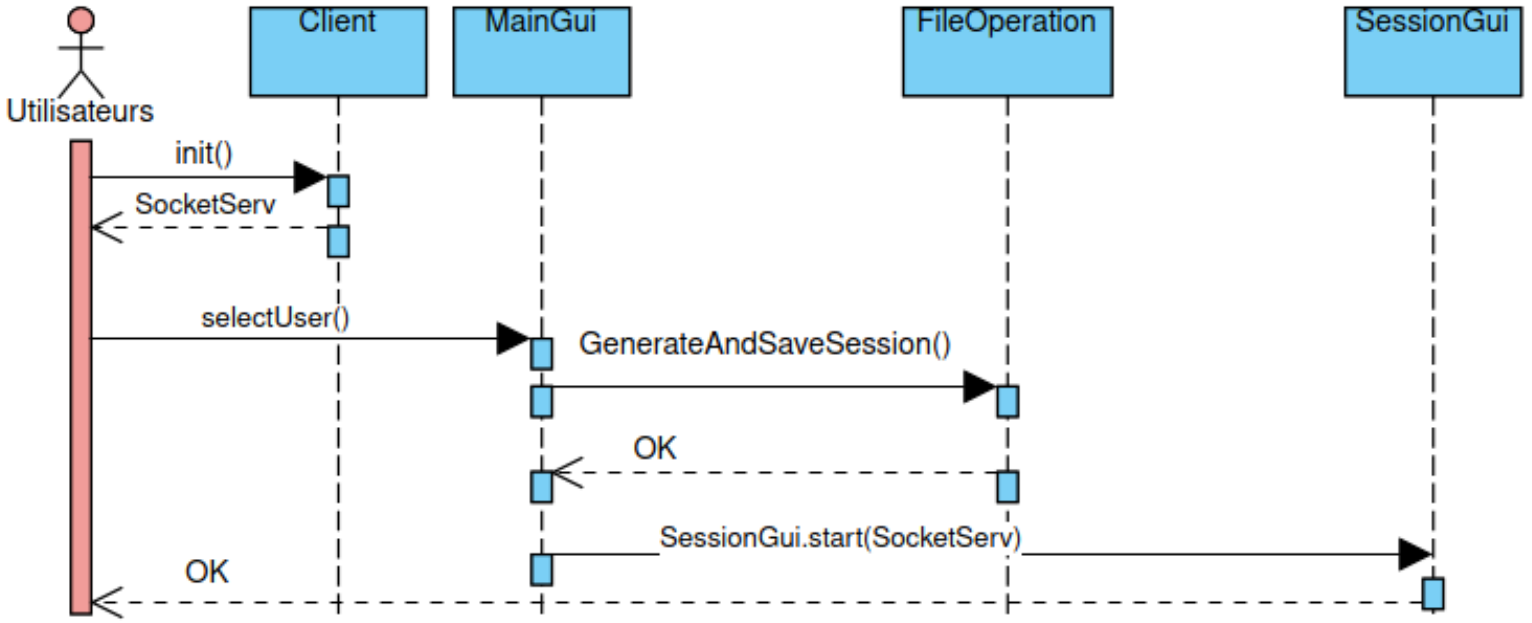
Gestion de la config réseau :



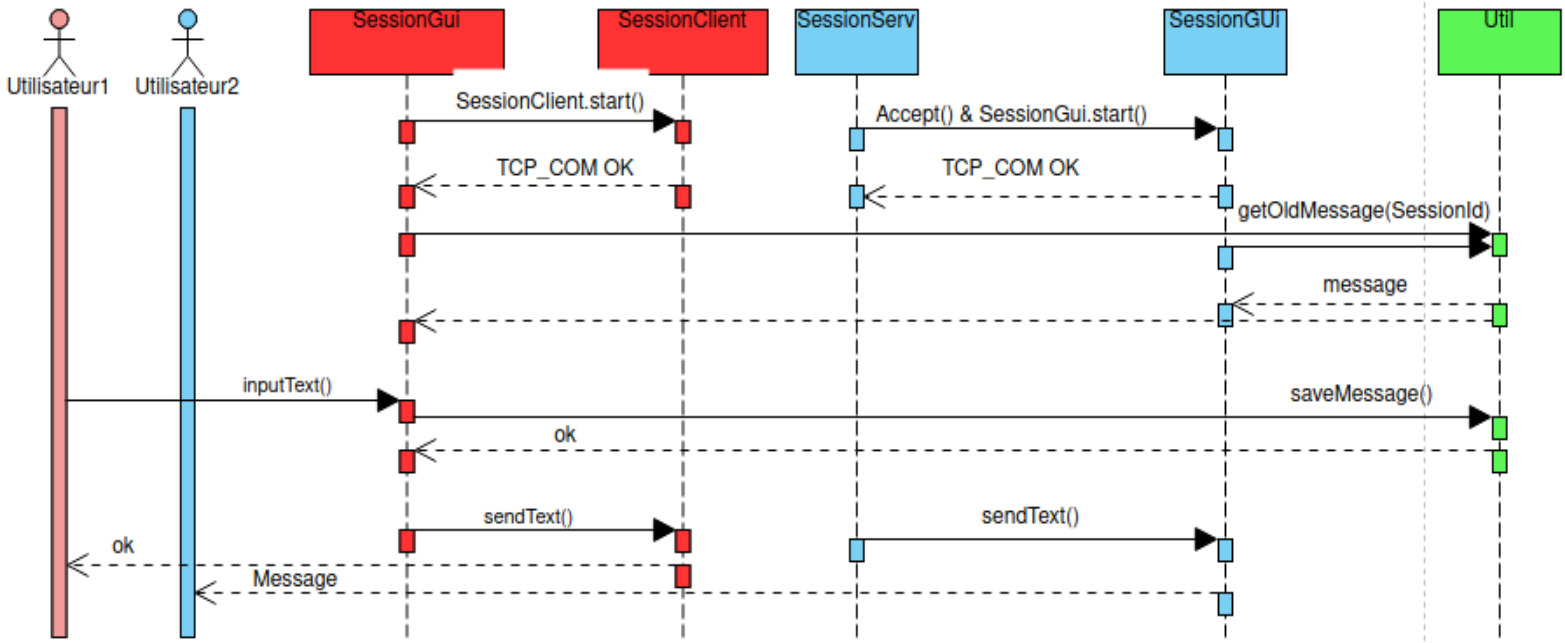
Gestion du profil utilisateur :



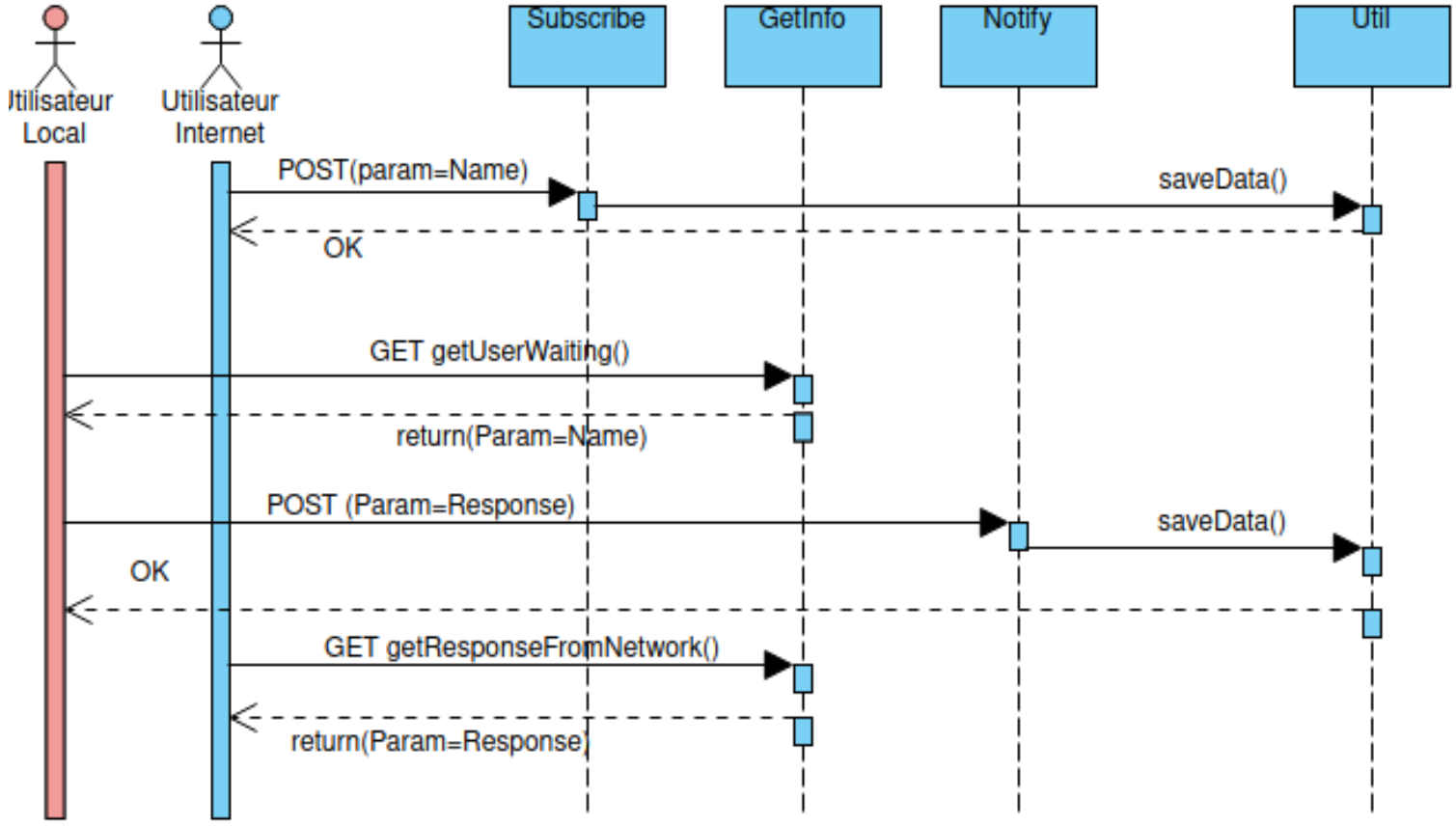
Gestion du démarrage des sessions de chat :



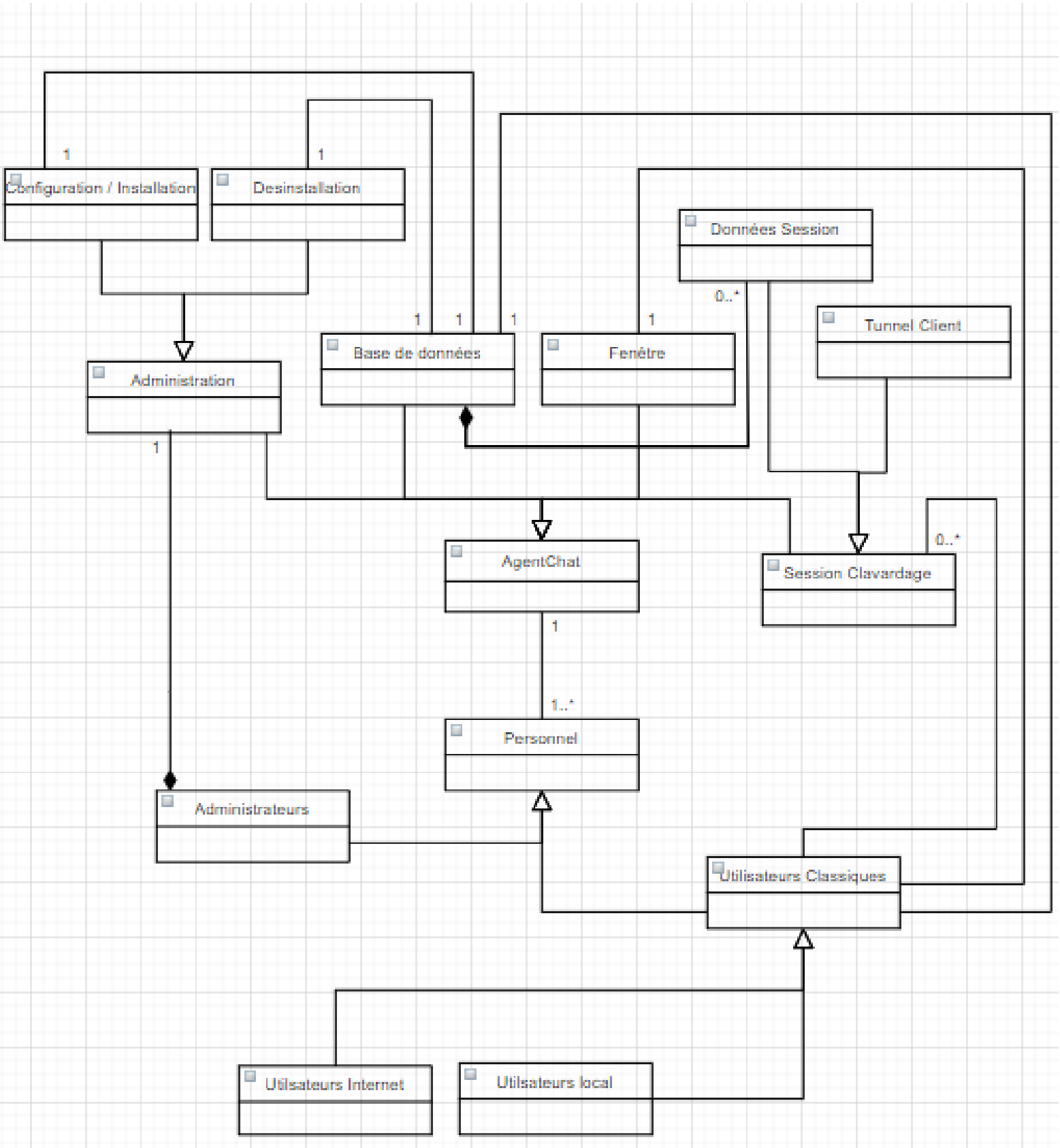
Gestion des sessions de chat :



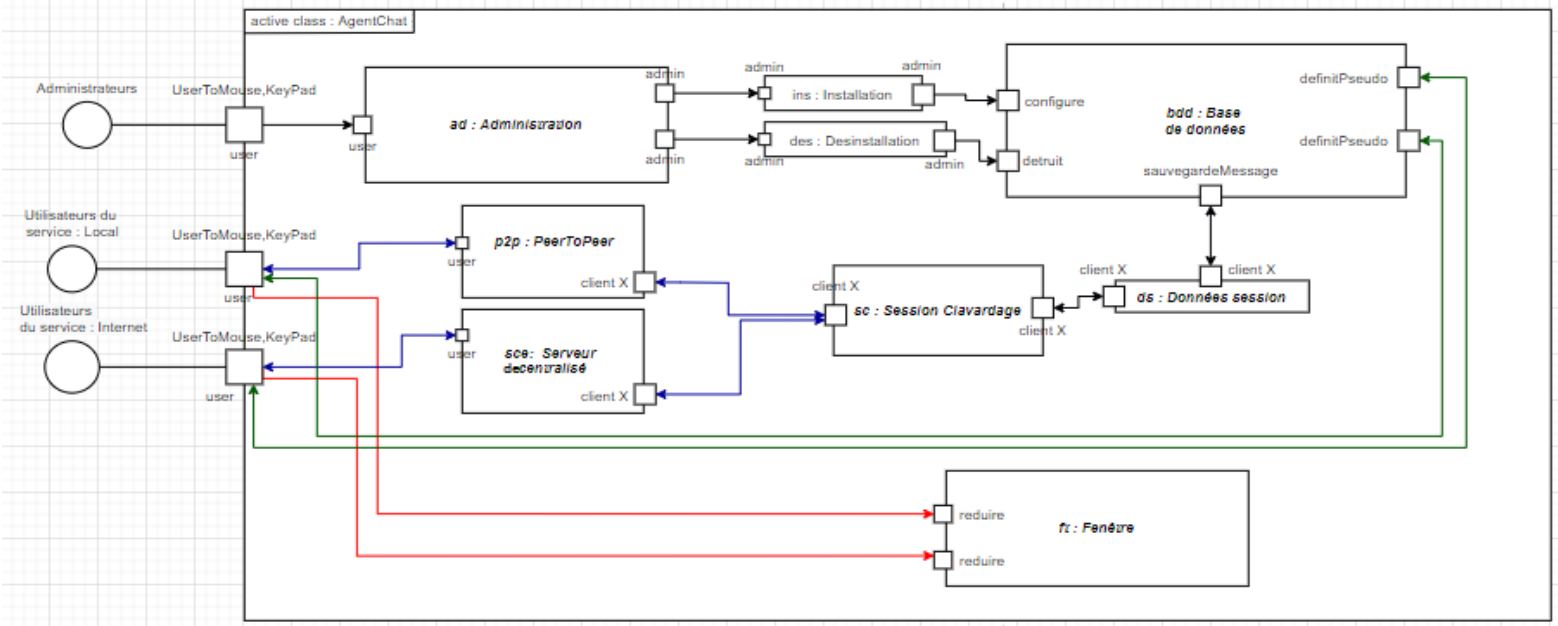
Gestion des requêtes SERVLET :



C) Class diagram


Configuration / Installation
Desinstallation
Données Session
Base de données
Fenêtre
Tunnel Client
Administration
AgentChat
Session Clavardage
Personnel
Administrateurs
Utilisateurs Classiques
Utilsateurs Internet
Utilsateurs local
1
0..*
1..*

D) Composite structure diagrams



E) States diagram