INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

# Rapport POO

## Système de Clavardage distribué multi-utilisateur temps réel

IR Systèmes Communiquants 2020/2021

Rédigé par:
Mathieu Dechambe, David Pereira

# Contents

# 1 Guide d'administration

## 1.1 Récupérer les sources

Avant toute chose, vous devez récupérer les sources du projet en clonant le dépôt github suivant : `https://github.com/TheoFontana/BE-COO_POO`

Les instructions qui suivent sont un duplicat du README que l'on peut trouver sur le dépôt github.

## 1.2 Dépendances

Le système a été pré-compilé avec **openjdk11** donc si la version de java sur votre machine n'est pas la même, il vous faudra peut-être recompiler le système avec le script **setup.sh** ou le **Makefile** (plus d'informations plus bas).

En cas de déploiement sur des postes macOS ou Windows. Des instructions de déploiement plus spécifiques sont précisées plus bas si la version précompilée ne fonctionne pas.

## 1.3 Instructions d'installation

Les fichiers jar précompilés peuvent être trouvés dans **target/Clavardage/clavardage.jar** et dans **target/PresenceManagementServer/presencemanagementserver.jar**

Pour les recompiler à la main, le script **setup.sh** peut être utilisé, il suffit de l'exécuter sans aucun argument.

La commande **make** sans argument invoquée depuis le répertoire où se trouve le **Makefile** peut être utilisée pour générer les fichiers jar.

Si vous voulez déployer le système sur des postes Windows, il est possible d'utiliser le script **setup.bat**. Il suffit de l'éxécuter sans argument également.

Si vous voulez déployer le système sur des postes OSX et que la version précompilée ne fonctionne pas, voici les commandes à exécuter : Créez les dossiers **target/Clavardage/** et **target/PresenceManagementServer/** puis exécutez :

```
javac -d target/Clavardage/ Clavardage/src/*.java
javac -d target/PresenceManagementServer/ PresenceManagementServer/src/*.java
```

pour compiler les fichiers sources en fichiers class. Placez vous ensuite successivement dans les dossiers target/Clavardage/ puis dans target/PresenceManagementServer/ et exécutez

```
jar cfe <nom_du_fichier_jar_desire> Main *.class
```

Les fichiers .class sont inutiles à partir d'ici donc vous pouvez les supprimer.

Pour démarrer le client de chat : se placer dans **target/Clavardage/** et exécuter la commande

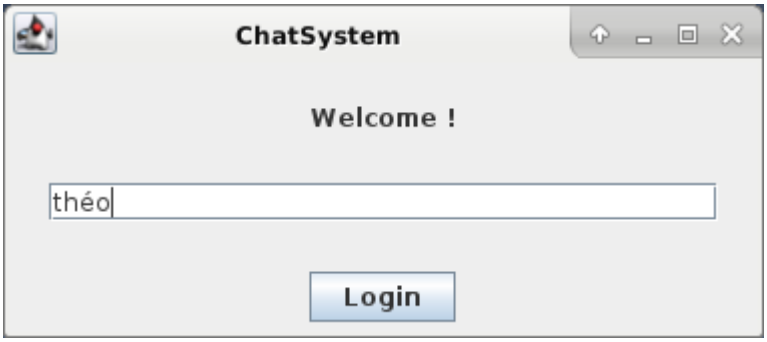
```
java -jar clavardage.jar
```

La procédure est identique pour le PresenceManagementServer : se placer dans **target/PresenceManagementServer** et exécuter

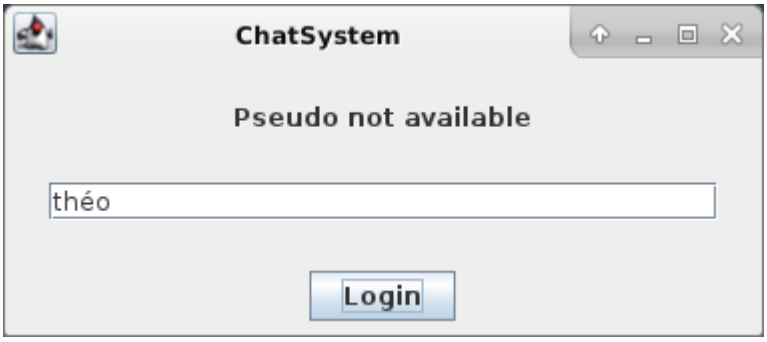
```
java -jar presencemanagementserver.jar
```

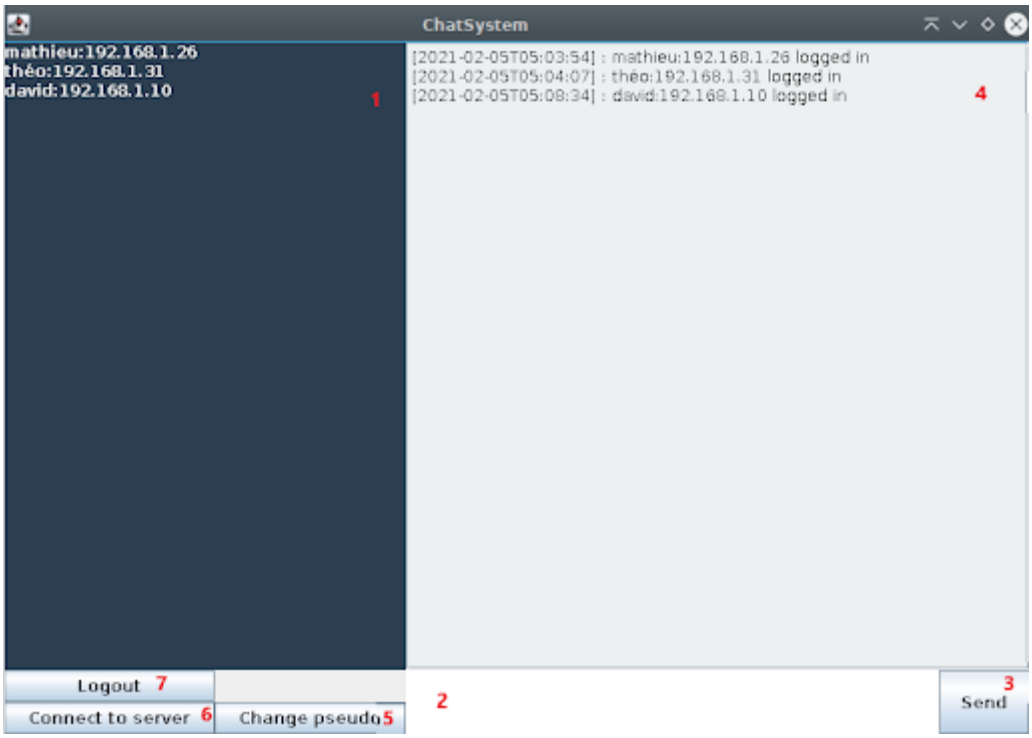
## 1.4 Utilisation du système

Pour utiliser le client de clavardage, démarrer le client comme indiqué ci-dessus. Le système demande de choisir un pseudo. Cliquez sur Login pour se connecter.

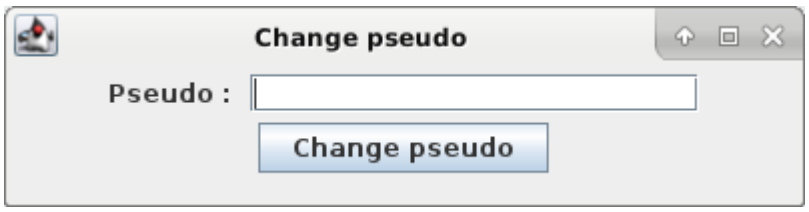



Si vous choisissez un pseudo déjà utilisé, il vous le fera savoir gentiment.

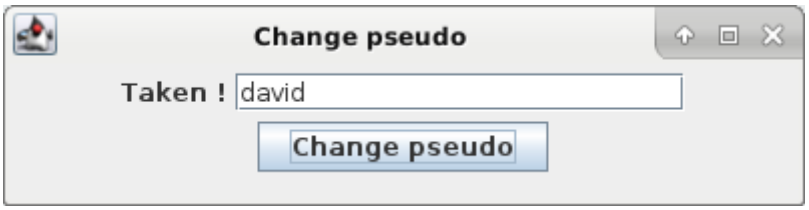





Une fois sur l'interface de chat, le système affiche sur la gauche [1] la liste des autres utilisateurs qu'il a détectés. Pour discuter avec l'un d'entre eux, sélectionner son nom avec la souris et taper le message à envoyer dans la zone de texte situé en bas de l'interface[2]. Une fois le message tapé, appuyez sur le bouton **Send** [3] pour envoyer le message. Les messages envoyés et reçus sont affichés et horodatés sur la zone de texte centrale[4].
L'utilisateur a la possibilité de changer de pseudonyme en cliquant sur le bouton **Change pseudo** [5]. Pour valider son choix, l'utilisateur doit cliquer sur **Change pseudo**.




Le changement de pseudonyme est soumis aux mêmes règles que le choix du login. Si celui-ci est déjà utilisé,le message "Taken" apparaît.



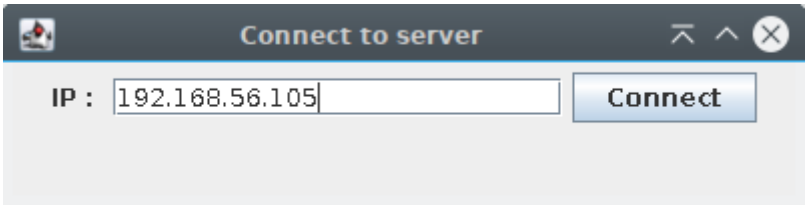

Pour commencer une discussion, l'utilisateur choisit une personne et lui envoie un message. L'envoi d'un message est signalé par "− >" suivi de l'horodatage. La réception d'un message est indiquée par "< −" suivi de l'horodatage.

```
->[2021-02-08T05:45:50@théo:192.168.1.32 (me)] : salut théo!!
<-[2021-02-08T05:46:04@théo:192.168.1.32] : salut mathieu!!
```

Lorsqu'un utilisateur veut se déconnecter, il lui suffit de cliquer sur "Logout". Une notification apparaît immédiatement sur l'interface des autres utilisateurs. Et l'utilisateur est redirigé vers la page de connexion.

```
[2021-02-05T05:56:02] : mathieu:192.168.1.26 logged out
```

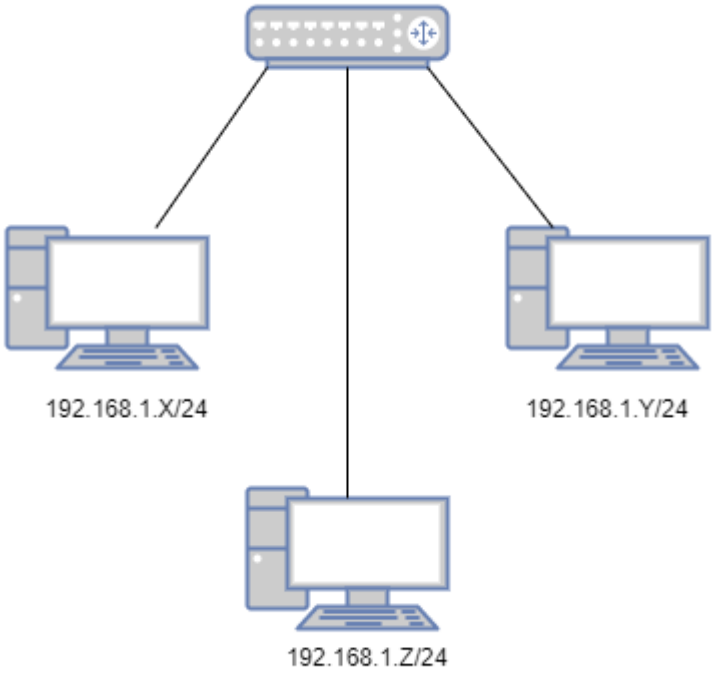
Dans le cas d'un utilisateur distant(qui ne se situe pas sur le même réseau d'entreprise), il peut néanmoins clavarder avec eux grâce au PresenceManagementServer si celui-ci fonctionne sur l'une des machines de la zone démilitarisée du réseau principal(comprendre DMZ, machine directement connectée au réseau d'entreprise et à l'Internet). L'utilisateur peut utiliser le bouton **Connect to server**[6] et rentrer l'adresse IP de cette machine.Enfin, l'utilisateur devra cliquer sur le bouton "Connect". Il apparaîtra dans la liste des utilisateurs connectés sur les autres machines et verra apparaître les autres utilisateurs connectées sur les machines du réseau d'entreprise. Il pourra donc démarrer une session de clavardage avec l'un d'entre eux.




Le PresenceManagementServer ne nécessite pas d'intervention de l'administrateur. Une fois démarré, il est automatiquement détecté par les utilisateurs situés sur le réseau d'entreprise. Les utilisateurs distants doivent s'y connecter manuellement.

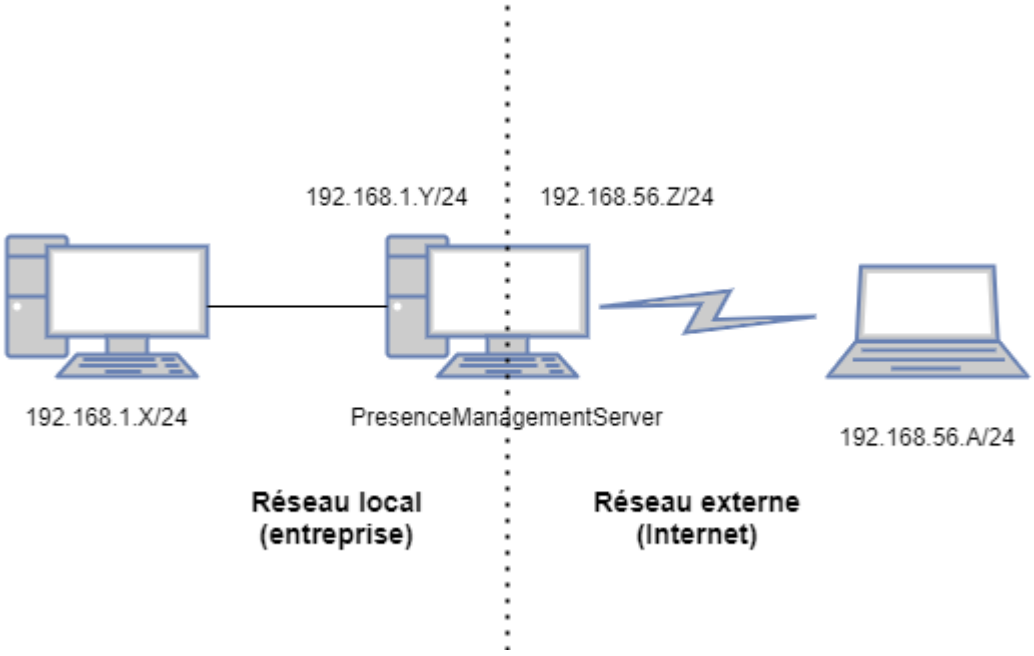
## 1.5 Procédures de tests

En ce qui concerne la procédure de tests, le choix a été fait (et en partie imposé par la situation) d'utiliser des machines virtuelles pour simuler la présence de plusieurs clients sur le même réseau local pour s'approcher le plus possible des conditions réelles d'une entreprise.




De plus, les tests ont été principalement effectué sur machines Linux (Debian 10) et Windows 10. Les tests sur OSX n'ont pas pu être fait pour la simple raison que nous n'en avions pas à disposition. Cependant, sachant que Java est un langage multi-platerforme, il paraît réaliste que l'agent fonctionne correctement sur cette plateforme.
Dans un premier temps, nous avons testé l'agent de manière isolée pour savoir si son utilisation répondait bien au cahier des charges: tests connexion/déconnexion, tests de l'interface graphique. Quand cette partie donnait satisfaction, nous avons commencé à nous pencher sur la partie communication entre utilisateurs. Nous nous sommes efforcés de vérifier si les contraintes (fonctionnement attendu du système, latence, affichage...) étaient respectées et d'apporter des solutions si cela n'était pas satisfaisant.
En ce qui concerne les tests avec le serveur de présence, nous avons utilisé une machine Linux avec deux interfaces: une interface sur le réseau local de l'entreprise et une autre sur un autre réseau. Cela a eu pour but de vérifier le fonctionnement du serveur ainsi que la détection et l'échange de messages.

Cette configuration est la plus proche d'une situation réelle en entreprise: ce type de serveur étant mis en DMZ, elle est en réalité exposée sur Internet sur l'une de ses interfaces réseaux.




Pour cette partie, les tests ont principalement été axé sur la vérification de la capacité du serveur à détecter les utilisateurs des deux côtés(interne et externe), au relais des informations d'un côté vers l'autre, de l'acheminement des messages et du délai. Dans notre topologie de test, le délai est très court. Dans la réalité, l'acheminement réseau est plus complexe et ne reflète pas la réalité.

# 2 Choix d'implémentation

## 2.1 Historique

Nous sommes partis du principe que les utilisateurs du système l'utiliseraient toujours depuis la même machine et qu'un système d'historique entièrement décentralisé était donc possible. Les messages envoyés à et par un même utilisateur sont stockés dans un fichier **CSV** sur la machine de cet utilisateur. Le choix de **CSV** a été guidé par la simplicité d'utilisation car c'est un format texte simple, facilement éditable depuis le code Java ou avec un éditeur de texte pour les procédures de test.

Les champs **CSV** sont remplis de la façon suivante :

```
idSender, idReceiver, date, content
```

Les id sont des valeurs numériques uniques associés à l'adresse ip de chaque utilisateur du système, excepté pour l'utilisateur local dont l'id est défini à la valeur 0. Le champ date comporte la date et l'heure avec une précision à la seconde de l'envoi du message.

Le désavantage de cette manière de stocker l'historique est que lors du chargement de l'historique lors d'une session de clavardage ultérieure, les correspondances $id \rightarrow utilisateur$ sont perdues car non stockées d'une session à l'autre et seuls les id sont affichés car la traduction n'est plus possible.

## 2.2 Système de commandes UDP

Nous avons décidé de faire communiquer les différents acteurs du système (les clients et le **PresenceManagementServer** via un système de commandes UDP avec arguments. La syntaxe de ces commandes est la suivante :

```
COMMANDE:argument1:argument2:...
```

En conséquence, nous n'avons pas utilisé le protocole HTTP pour établir une communication entre le **PresenceManagementServer** et les clients, toutes les communications se font avec les commandes UDP. Ce choix permet d'avoir une plus grande maîtrise sur les actions qui peuvent être effectuées par le système. En effet HTTP ne supporte qu'un nombre limité de commandes (GET, POST et quelques autres). En créant notre propre système, nous pouvons préciser exactement à quoi sert chaque commande et implémenter les comportements associés de manière claire. Cela évite également de créer une surcouche que nous avons jugée inutile des librairies java pour les servlets. Avec ce système, il est également très facile d'implémenter de nouvelles commandes si le besoin s'en fait sentir.

La logique de traitement de ces commandes est implémentée dans **PresenceManagementServer.java** pour le **PresenceManagementServer** et dans **UDPListener.java** pour le client.

## 2.3 Bogues connus

Au cours de nos tests, nous avons identifiés quelques bogues et qui nécessitent une future mise à jour. En premier lieu, cela se cractérise lorsqu'un utilisateur change son pseudonyme, il faut cliquer sur la liste des utilisateurs pour que le pseudonyme se mette à jour dans l'interface graphique au niveau de la liste des utilisateurs. De plus, en ce qui concerne l'enregistrement de l'historique de clavardage, il est obligatoire de cliquer sur le bouton logout. Fermer l'application sans se déconnecter causera la perte des derniers messages envoyés.
Si vous constatez des bogues lors de votre utilisation, je vous invite à nous contacter à l'adresse suivante: contact.clavardage@gmail.com.