INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

SYSTÈME DE CLAVARDAGE DISTRIBUE INTERACTIF MULTI-UTILISATEUR TEMPS RÉEL

Programmation Orientée Objet

4IR – B2

Thomas Ballotin

Terani Luque

Table des matières

Introduction

Le but de ce projet est d'implémenter un système de clavardage distribué interactif multi-utilisateur temps réel qui sera disponible sur tous les systèmes d'exploitation. En effet, nous allons mettre en place un logiciel permettant aux employés d'une entreprise de communiquer entre eux. L'installation et l'administration du logiciel sera faite par les experts et techniciens de cette même entreprise.

Ce projet est conçu en langage JAVA implémentant les technologies SWIFT, pour l'interface ainsi que JCDB pour la mise en œuvre de base de données. Ce logiciel mettra en place une communication TCP et UDP.

I. Architecture du système

Notre architecture de système contient tout d'abord une architecture réseau, les différentes communications qui peuvent intervenir dans notre projet sont :

- Le protocole sans connexion de la couche transport qu'est UDP est utilisé durant la phase de connexion afin de connaître les autres utilisateurs présents et de les notifier d'une nouvelle connexion. Chaque nouvel utilisateur envoie son pseudo, son adresse IP et son port d'écoute à chaque utilisateur déjà présent sur le réseau. Ces derniers lui envoient leurs informations à leur tour. Enfin, en possession des pseudos de tous les utilisateurs, le nouvel utilisateur les ajoute dans une liste et les compare avec le sien afin de vérifier qu'il soit unique. Si ce n'est pas le cas, il choisit un autre pseudo et réitère la communication UDP précédente. En résumé, le protocole UDP sert à gérer la liste des utilisateurs connectés.

- Le protocole TCP, lui, possède une QoS (quality of service) plus évoluée que UDP. En effet, il permet d'avoir une fiabilité d'ordre et de reprise des pertes. Il nous permettra de gérer l'envoi et la réception de messages entre deux utilisateurs.
  En effet, lorsqu'un utilisateur a réussi sa connexion, il lance un serveur TCP en arrière-plan, ce serveur attend qu'une connexion lui soit demandée, si une connexion est demandée il devra donc l'accepter.
  Quand la connexion est acceptée, le serveur créer un client TCP pour que celui-ci communique avec l'initiateur de connexion.

Dans le cas où l'utilisateur est lui-même l'initiateur de connexion le serveur ne fait rien, un client est directement créé et il demande à son tour une connexion au serveur de l'utilisateur voulu.

Quand les deux clients sont créés et opérationnels, un système de clavardage est donc possible et les messages sont transmis entre eux. Ces messages sont également horodatés et mis en base de données.

Ce système comporte donc également une partie comportant une base de données locale, chaque ordinateur implémentant notre système possède un système de gestion de base de données (logiciel MySQL). Lors de la connexion, le logiciel vérifie qu'une base de données existe pour celui-ci ainsi qu'une table 'history' qui permet de conserver l'historique entre deux utilisateurs dans le cas où elle n'existe pas, il la crée. Cette table se compose des messages envoyés, du destinataire, de l'expéditeur ainsi que de la date et l'heure quand le message à été envoyé.

Lors de l'ouverture d'une conversation, après avoir initié les clients TCP, l'historique des deux utilisateurs est demandée grâce à une requête SQL puis affichée sur le moniteur. Les deux utilisateurs peuvent donc visualiser les anciens échanges avec leurs collègues de bureau.

Ensuite à chaque envoi de message une autre requête est formulée afin d'ajouter ce nouvel échange en base. Les schémas suivants décrivent notre architecture de manière simplifiée.

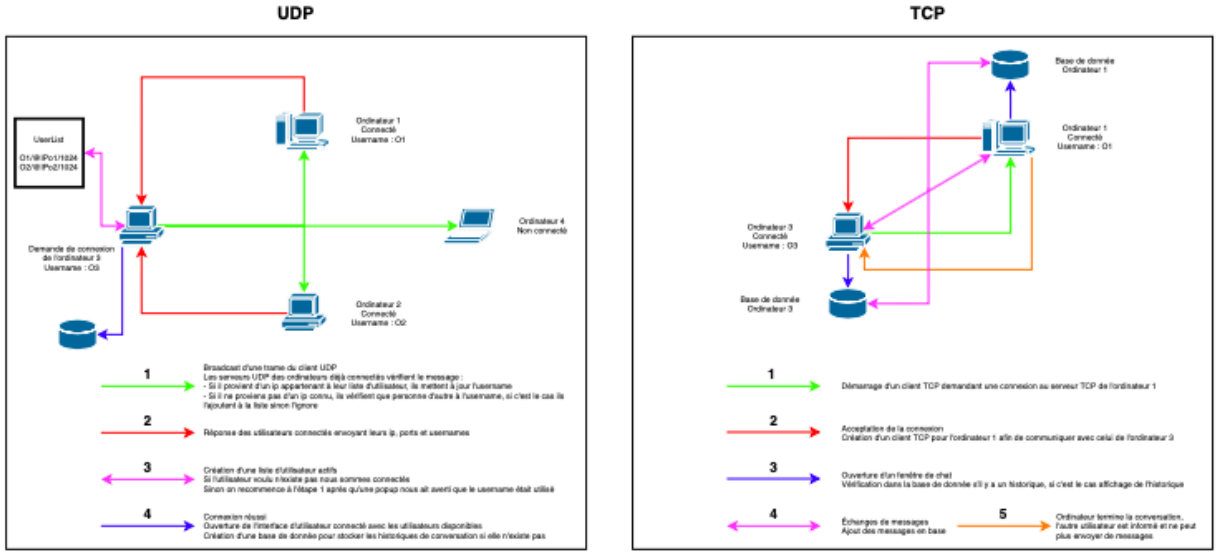


Figure 1 - Schémas simplifiés de l'architecture réseau du système

Grâce à ces 2 types de transport réseau et de ces bases de données locales notre système de clavardage peut donc être mis en place et fonctionner parfaitement sur un réseau local d'une petite comme une grande entreprise.

II. Choix implementation

Nous avons séparé ce projet JAVA en plusieurs packages : *bdd* (base de données)*, graphicallInterface* (interface graphique)*, networkTCP* (réseau TCP)*, networkUDP* (réseau UDP) et *other* (autres types de classes)*.*

A. bdd

Pour stocker les messages entre deux utilisateurs, le système utilise une base de données qui identifie un utilisateur par son adresse IP. Ceci implique qu'un employé, une fois connecté via un ordinateur, se doit de toujours se connecter avec ce dernier, sinon il n'aura pas accès à l'historique des messages échangés avec d'autres utilisateurs. De plus, si le DHCP (Dynamic Host Configuration Protocol) change l'IP de son ordinateur, il perdra son historique.

La première idée était de gérer les historiques avec une base de données centralisée, qui aurait pu nous permettre d'avoir l'historique de toutes les conversations au même endroit. Cependant, dû aux conditions de travail en distanciel nous n'avons pu l'implémenter étant donné que nous avions des soucis de VPN.

L'idée pour laquelle nous avons opté est la création de bases de données locales afin de ne pas pouvoir déployer notre système sur n'importe quel réseau sans passer par un VPN. En effet, chaque ordinateur possède sa propre base de données avec tous les messages que son utilisateur à envoyer ou reçu.

Pour mettre en place ces base de données nous avons tout d'abord installé sur nos ordinateurs MySQL qui nous permet de créer et de gérer des bases de données sur l'ordinateur. Pour interagir et créer nos bases ainsi que notre table stockant l'historique des messages nous avons implémenté une classe JAVA "Bdd.java" qui utilise le driver JCBD.
Ce driver permet de mettre en place des interfaces de déclaration permettant de mettre à jour une base ou de l'interroger/la gérer.

B. GraphicallInterface

Ce package contient les classes instanciant les interfaces graphiques de notre système.
Nous avons opté pour la bibliothèque graphique Swing car c'est celle qui nous a été présentée en cours.
La classe *DiscutionWindow* possède deux constructeurs afin de différencier le cas où l'utilisateur est l'initiateur de la discussion et le cas où il ne l'est pas.

Lorsqu'un utilisateur initie une communication, il clique sur le pseudo d'un autre utilisateur qui apparaît sur la fenêtre *Connected*. Cela entraîne la création d'une instance de *DiscutionWindow* qui crée un client TCP. Puisque le client est créé par DiscutionWindow, son constructeur n'a besoin que de l'utilisateur auquel on veut parler.

Or, lorsqu'un utilisateur est le destinataire d'une tentative de communication c'est le serveur TCP qui crée un client (dans la classe TCPS). Ainsi pour ouvrir une fenêtre de discussion, il a besoin de l'utilisateur qui a initié la communication mais aussi du client TCP créé en amont par le serveur.

Nous avons fait le choix de séparer TCP et UDP en deux interface différentes sachant qu'elles n'interviennent pas au même moment dans notre système (cf. Architecture du système).

C. networkTCP

Ce package contient les classes TCPC et TCPS qui permettent de gérer les flux de messages entre utilisateurs en utilisant des serveurs et clients TCP.

Dès qu'un utilisateur a réussi à se connecter, un serveur TCP est créé et se met en attente de demande de connexion. Lorsqu'une connexion entrante est acceptée, le serveur crée un client TCP. La communication se fera alors entre deux clients TCP.

Nous avons choisi d'utiliser des threads pour implémenter les classes TCPS et TCPC. En effet, la création mais surtout l'activité des serveurs et clients TCP doit s'effectuer en parallèle des autres tâches.

D. networkUDP

A l'instar de *NetworkTCP,* ce package contient les classes ClientUDP et ServerUDP qui permettent de gérer la connexion et la liste des utilisateurs connectés au réseau (cf architecture du système).

Pour gérer la liste des utilisateurs actifs nous avons choisi d'utiliser une JList. Cela nous permet de facilement mettre à jour la liste affichée sur l'interface graphique lorsque de nouveaux utilisateurs se connectent ou au contraire lorsque certains se déconnectent.
Le serveur gère aussi tout changement de pseudo des autres utilisateurs. En effet, lorsqu'un utilisateur actif modifie son pseudo, les autres utilisateurs présents sur le réseau doivent prendre en compte cette modification.

E. other

Le dernier package est "other" en effet il comporte la classe "User.java" cette classe permet de créer des utilisateurs. Ces utilisateurs sont composés d'un nom, d'un ip et d'un port.

Ces utilisateurs sont créés à partir de la réponse des utilisateurs actifs lors d'une demande de connexion et stockés dans une liste, ils sont également créés ou supprimés lors d'une déconnexion ou d'une connexion d'un autre agent.

Des getters ont été mis en place afin de pouvoir accéder de partout aux informations d'un utilisateur, seulement un setter a été mis en place dans le cas où l'utilisateur change de username. Les setter d'ip et de port n'étaient pas nécessaire sachant qu'un utilisateur est reconnu sur le système et dans la base de données par son adresse ip, il ne peut donc pas le changer.

III. Manuel d'utilisation

Cette partie va vous permettre de savoir comment utiliser et mettre en place notre système. Certains bugs peuvent intervenir, nous avons expliqué la raison dans la partie suivante (cf. Améliorations possibles et contraintes rencontrées).

A. Premier lancement de l'application

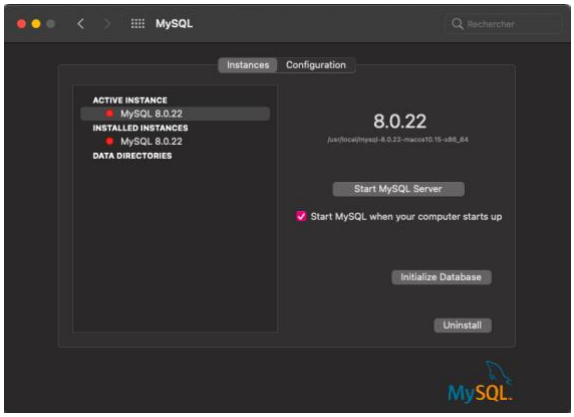
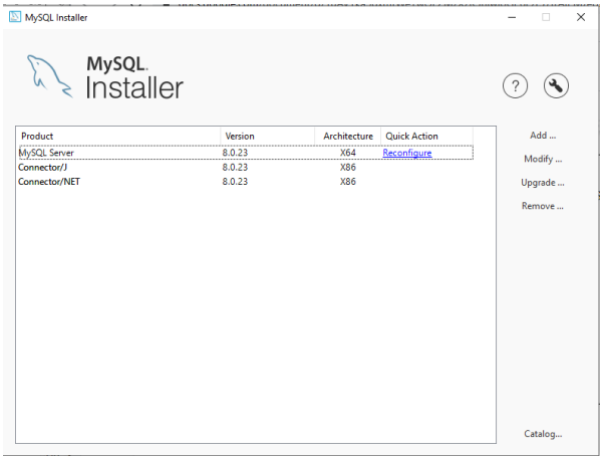
1. *Installation et lancement de MySQL*

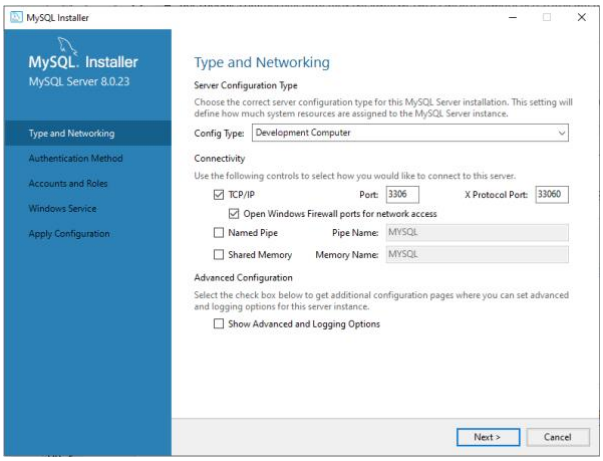
Afin de pouvoir faire fonctionner notre base de données nécessaire au bon fonctionnement de notre système, MySQL est nécessaire.

Lien du fichier exécutable à installer (disponible pour toutes les plateformes) : https://dev.mysql.com/downloads/mysql/

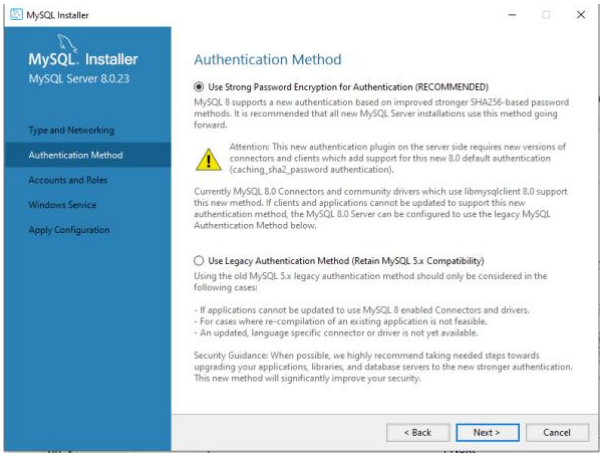
Lors de l'installation ou du premier lancement de serveur MySQL (dépend de l'OS) un mot de passe va vous être demandé, il est nécessaire de s'en souvenir.

Comment lancer le serveur MySQL ?

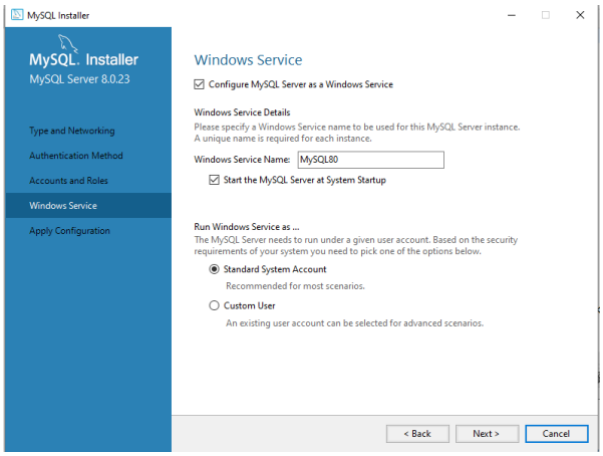
Mac OS :	**Windows 10 :**
Préférences système -> MySQL -> *Start MySQL server* 	Démarrer -> MySQL -> MySQL Installer  Ensuite cliquer sur *Reconfigure*



Next



Next, ensuite le mot de passe vous est demandé



Ensuite *next*, puis *execute*.

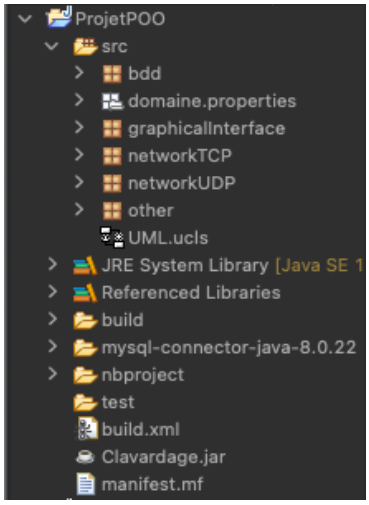
2. *Téléchargement du projet*

Après avoir installé MySQL, il faut télécharger le système à partir du lien suivant : https://github.com/Thxmasb/ProjetPOO_Ballotin_Luque

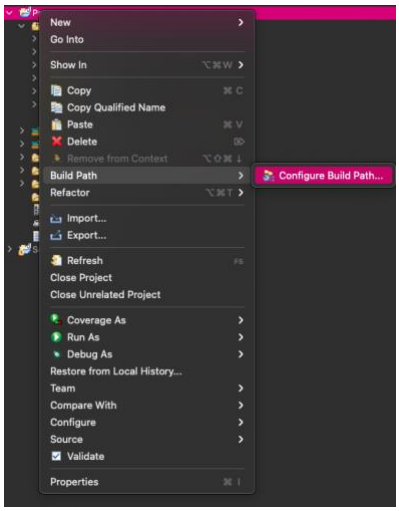
3. *Premier lancement sous git*

Ensuite, il faut l'ouvrir sur Éclipse.

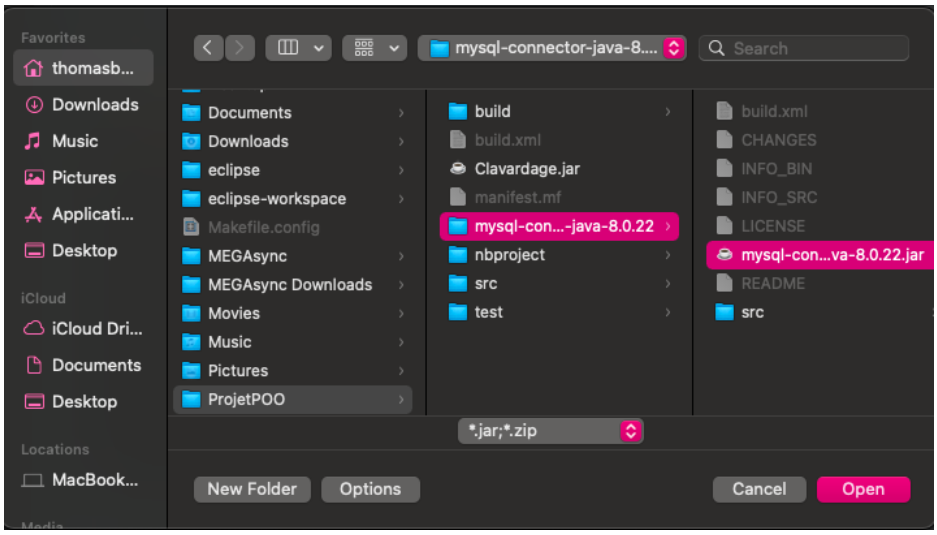
Par suite de l'ouverture vous devriez avoir un Explorateur de projet comme ceci :



À la première ouverture une erreur apparaîtra sur le nom du Projet "Projet POO", il faudra cliquer droit sur le nom du projet puis cliquer sur "*Configure build path*" :



Ensuite, il faut cliquer sur “*Librairies*” puis sélectionner “mysql-connector-java-...” et cliquer sur "*Edit*" et choisir le fichier du même nom dans le répertoire du projet comme suit :
(Cette étape est nécessaire car éclipse ne permettait pas qu’on choisisse le lieu de stockage en choisissant nous-même l’emplacement du fichier. Nous aurions pu mettre un lien « ./ » qui permet de dire que le fichier est dans le même que le projet)



Après avoir choisi nous appuyons sur ouvrir puis « *Apply and close* », ensuite il faut cliquer droit sur le nom du projet et appuyer sur « *Refresh* ».

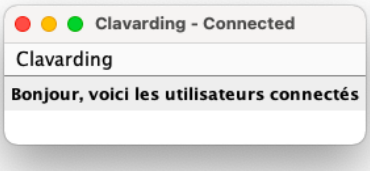
La dernière étape pour rendre le projet fonctionnel est d’éditer le fichier « config.properties » qui se trouve dans « src/domaine.proerties/ ».
Ce fichier permet de rentrer le login et le mot de passe que nous avons choisi lors de la configuration de MySQL. Si vous n’avez pas choisi de login lors de l’installation veuillez laisser « root » en login.

```
#######################CONFIG BDD#######################
#Please do not change the values before the equal sign#
sgdb.login=root
sgdb.password=VEUILLEZ RENTRER VOTRE MOT DE PASS
```

Après cela, pour la première ouverture du projet il faut essayer le Run as Java Application afin de pouvoir visualiser dans la console si la base de données a bien été configurée.

Pour cela, allez dans l’explorateur de projet -> src -> graphicalInterface, clic droit sur connexion puis Run as -> Java Application, l’interface de connexion s’ouvre, il faut choisir un username et valider le choix.

Si l'interface d'utilisateur connecté s'ouvre sans aucun souci, ensuite c'est que tout est correctement installé et que vous avez bien suivi le tutoriel de lancement du projet.



Cependant, si une erreur "Time Zone non reconnue" s'affiche sur la console il est nécessaire de lancer le programme de ligne de commande MySQL et de rentrer ceci :
SET GLOBAL time_zone = '+3:00';

Après avoir suivi ce tutoriel le système est prêt à être utilisé, veuillez donc maintenant vous référer à la suite.

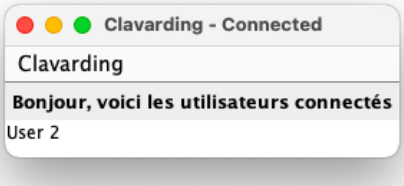
B. Utilisation du système après la première installation.

Pour un lancement plus simple, il est possible d'exécuter seulement le fichier Clavardage.jar, ce fichier ouvre donc l'interface de connexion du système.

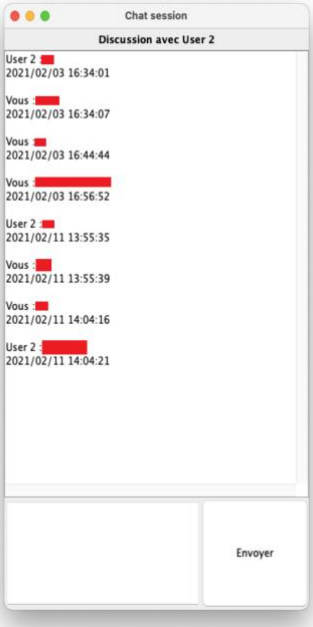
Comme pour l'installation, il faut choisir un pseudo, si le nom d'utilisateur est déjà utilisé vous devriez voir afficher un pop-up comme suit :

Il faudra donc choisir un autre nom d'utilisateur.

Après avoir choisi un nom d'utilisateur valide vous serez redirigé vers l'interface d'utilisateurs connectés. Cette interface permet de changer de nom d'utilisateur en cliquant sur « *Clavarding* » puis « *Change Username* ».



Pour démarrer une session de chat avec une personne il faut cliquer sur le nom de celle-ci.



L'historique s'affiche s'il y en a un, ensuite l'échange se fait simplement en écrivant et cliquant sur le bouton « *Envoyer* ».

Si un utilisateur quitte la conversation un pop-up s'affiche :

En effet, 2 choix s'ouvrent à l'utilisateur, soit il quitte lui aussi la conversation soit redemande une reconnexion avec l'utilisateur.

Pour se déconnecter, la solution est de quitter l'interface d'utilisateur connectés.

NB : certains problèmes peuvent survenir avec la connexion (liste d'utilisateur vide ou la non-disparition de l'utilisateur se connectant sur l'autre ordinateur) car étant donné que c'est de l'UDP si la connexion est instable il peut avoir de la perte de paquet, il suffit de relancer le système.

NB 2 : Vous trouverez une vidéo montrant le fonctionnement du projet à l'URL suivant :

IV. Améliorations possibles et contraintes rencontrées

Dû aux conditions extrêmement particulières de cette année nous nous sommes confrontés à de nombreux soucis :

- Tout d'abord, nous n'avions pas sur la majorité des séances deux ordinateurs étant donné que nous ne pouvions pas nous voir. Les tests réseaux étaient donc quasi-impossible à mettre en place ce qui nous a fortement retardé.
- Nous avons également, comme cité plus haut, rencontré de gros problèmes avec le VPN INSA qui nous a contraint à passer de nombreuses séances pour tenter de le régler. Finalement nous avons dû choisir des bases de données locales.

- Ce manque de temps nous a donc mis en impossibilité de :
    - Faire une partie outdoor du système pour communiquer avec des personnes qui ne sont pas sur le réseau local.
    - Faire des tests unitaires afin de rendre le déploiement plus facile (nous aurions pu notamment créer un test qui vérifie directement que la base de données est bien mise en place au lieu de vérifier s'il y a des erreurs sur la console).

Quant aux améliorations possibles nous aurions pu :

- Faire une interface de meilleure qualité avec notamment JavaFx.
- Permettre l'envoi de fichiers.
- Ajouter des photos de profil.
- Reconnaître un utilisateur autrement que par son IP afin qu'il puisse changer d'ordinateur. Cela entraînerait aussi la mise en place d'une base de données centralisée.

Conclusion

Pour conclure, ce projet nous a permis d'acquérir de solides compétences en programmation JAVA en dépit des conditions de réalisation en distanciel.
Nous avons pu expérimenter la création d'un logiciel de A-Z en utilisant de nombreuses librairies telles que JDBC ou Swing ainsi que de gérer les threads, etc.
Nous nous sommes confrontés à un réel projet tel que nous pouvons avoir en entreprise.
Bien que le sujet et le développement étaient intéressants, nous avons eu un manque de temps pour le finir. En effet, bien que le deadline ait été rallongée, avec l'INSA à coté il était difficile d'avancer tous les jours.

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

Université Fédérale Toulouse Midi-Pyrénées

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

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