INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Clavardage

Compte-rendu de Projet

Conception et Programmation Orientées Objet

Année 2019 - 2020

Lien du dépot GitHub : https://github.com/diarraas/Projet_Coo_Poo.git

DIARRA Assa
MOURET Quentin
4IR - A1

Sommaire

Introduction

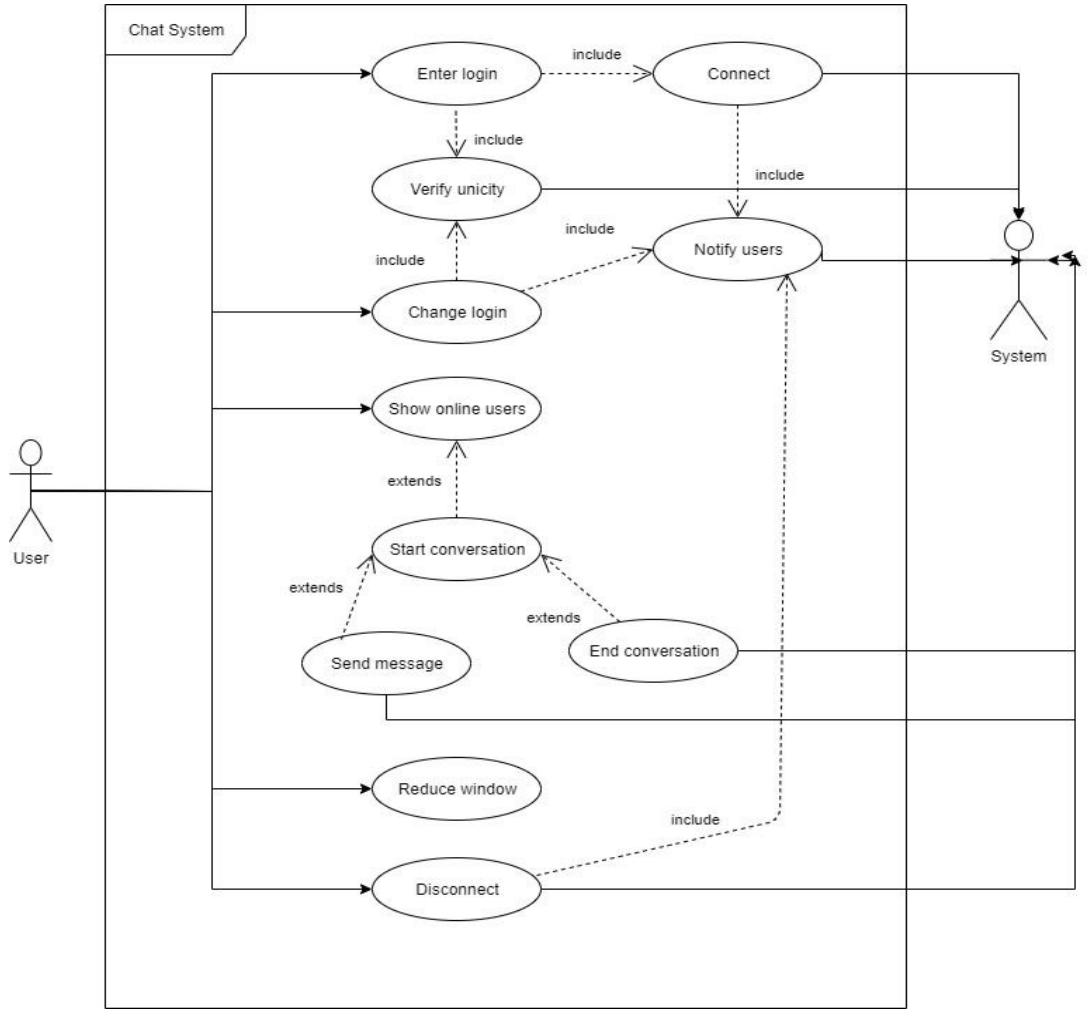
Au sein d'une entreprise, il est aujourd'hui indispensable de pouvoir communiquer rapidement avec d'autres membres de la firme. Ainsi, nous proposons un système de clavardage déployé sur un réseau local. Le cahier des charges que nous avons suivi stipule différentes exigences, dont voici les principales.

L'utilisateur doit pouvoir choisir un pseudonyme, avec lequel il sera reconnu dans ses interactions avec l'agent. Le système se doit d'assurer l'unicité du pseudonyme. Ce dernier doit être modifiable, et ne doit pas causer de problème avec une éventuelle session de clavardage ouverte. L'utilisateur connecté doit pouvoir identifier tous les membres en ligne sur le système, et commencer une session avec l'un d'entre eux. Les messages doivent être mémorisés et horodatés, afin d'être ré-atteignables si l'utilisateur relance une conversation avec un autre membre.
L'application doit être déployable sur divers systèmes d'exploitations, tels que Windows, Linux, Mac OS ou Androïd. L'agent doit être réductible.

Nous traiterons dans ce document, au travers de divers diagrammes, la mise en place de notre projet. Pour plus de lisibilité, nous n'y mettrons que les principaux, et laisserons les autres accessibles sur le Git.

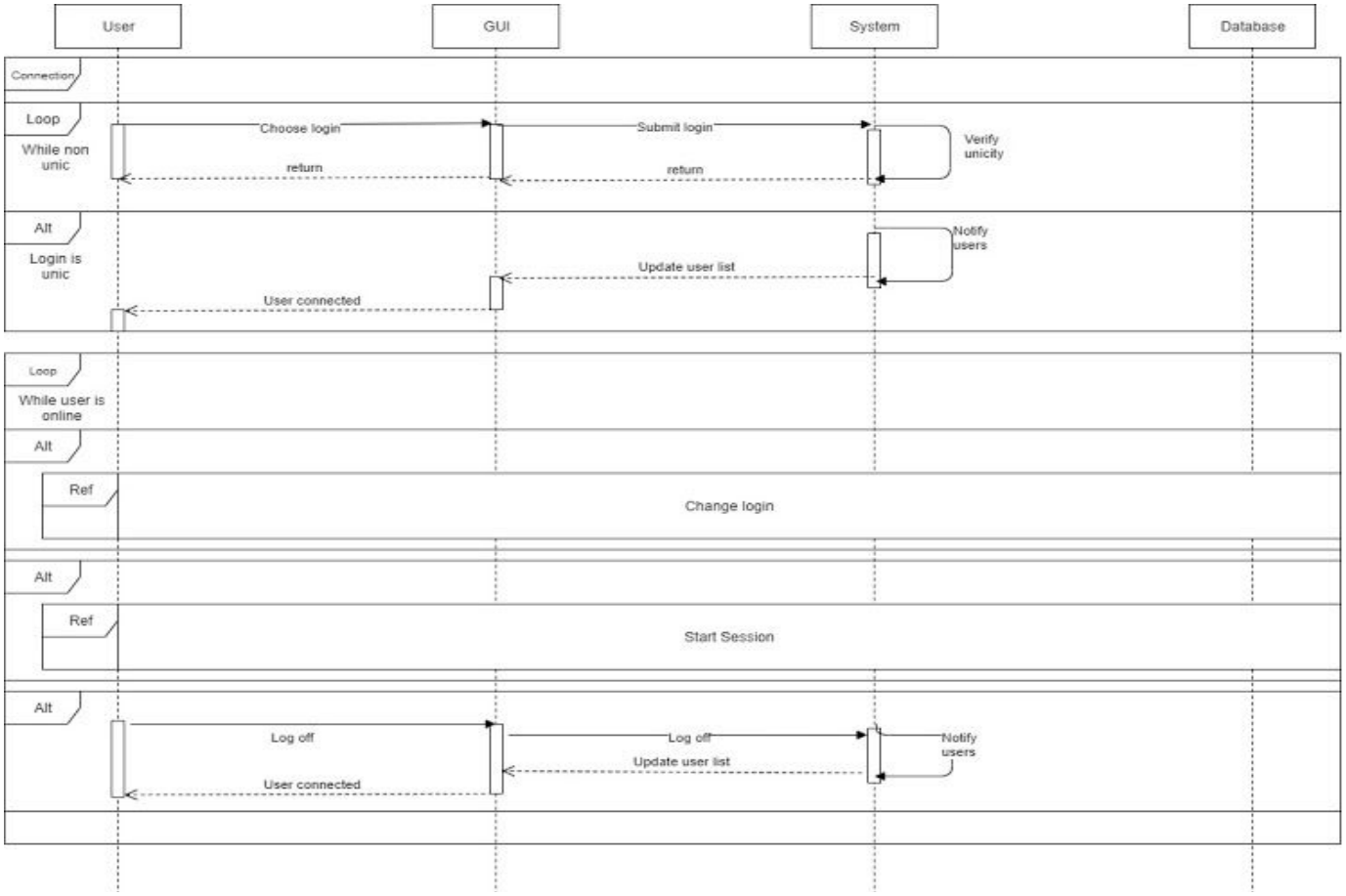
I. Diagramme des cas d'utilisation

Le diagramme de cas d'utilisation illustre les différentes actions réalisables par l'utilisateur sur le logiciel. Ces cas ont été identifiées après analyse du cahier des charges fourni et les différentes spécifications. Le diagramme ci-dessous permet notamment de mieux exposer les différentes interactions entre l'utilisateur et le système.



II. Diagrammes de séquence

Le diagramme de séquence illustre le mieux les différentes interactions entre l'utilisateur de le système. Nos diagrammes de séquences nous permettent également de mieux expliquer certaines décisions de conception et d'implémentation.

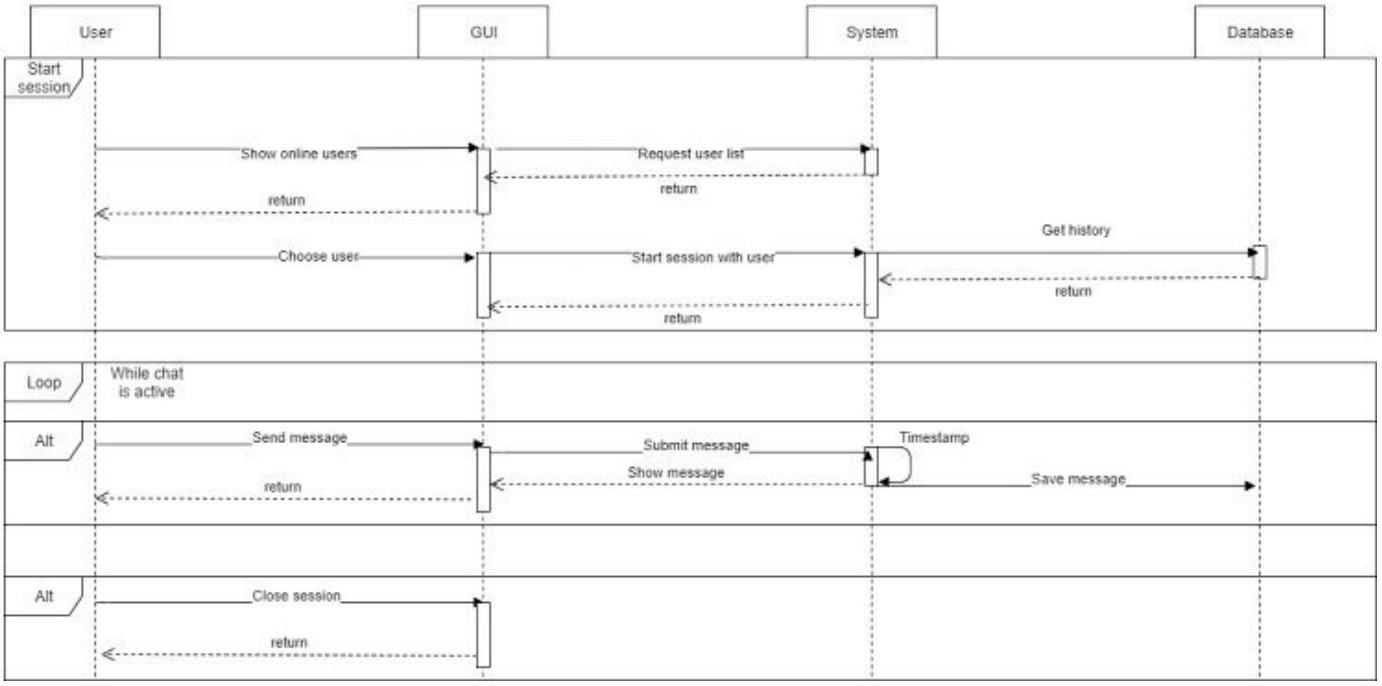


1. La connexion et la déconnexion

L'utilisateur se connecte au système à l'aide d'un pseudonyme de son choix qu'il passe au système. Le système doit vérifier l'unicité du pseudonyme avant d'authentifier l'utilisateur(cf.[CdC-Bs-10]). Pour cela, il contacte les autres utilisateurs connectés pour s'assurer que l'un d'entre eux n'utilisent pas déjà le pseudonyme proposé. L'utilisateur est notifié en quelques secondes(cf. [CdC-Bs-24]). Dans le cas où le pseudonyme est déjà utilisé, l'utilisateur doit en soumettre un autre. Sinon, l'utilisateur est authentifié.

- **L'authentification :** L'utilisateur est identifié à l'aide du pseudonyme choisi et de son adresse IP auprès des autres utilisateurs(cf. [CdC-Bs-7]).
- **La notification :** Le système envoie un message aux connectés contenant les identifiants de l'utilisateur, les informant qu'il est connecté(cf.[CdC-Bs-8]). Ces utilisateurs en retour, envoient leurs identifiants au nouvel utilisateur pour permettre une communication bidirectionnelle.
- **La déconnexion :** Le processus est similaire au processus de connexion. Depuis l'interface graphique, l'utilisateur notifie le système de sa déconnexion, qui en informe ensuite les autres utilisateurs. Ces utilisateurs sont donc notifiés que cet utilisateur n'est plus joignable et peuvent mettre à jour leur liste de contacts.

2. Le clavardage

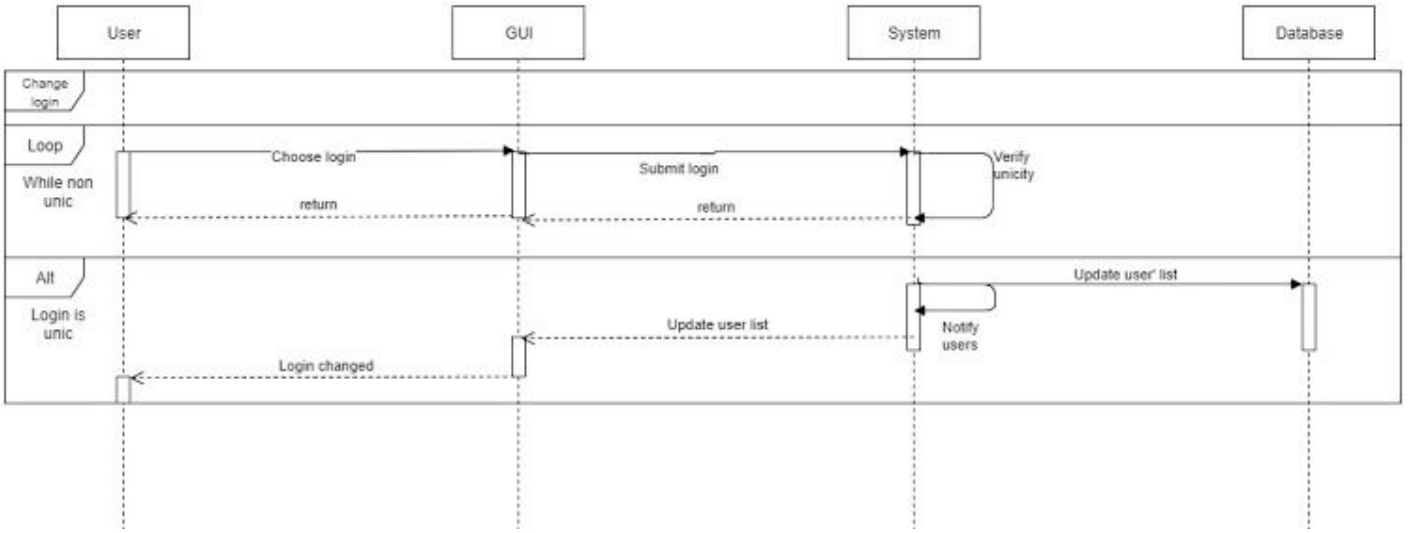


La communication se fait selon les étapes suivantes : le choix d'un interlocuteur, le choix du type de message à envoyer (texte ou fichier) et l'envoi/réception. Pour répondre au mieux au cahier des charges, la communication se fait en TCP (cf. [CdC-Bs-30]).
Un utilisateur possède donc un socket serveur pour la réception des messages et un socket client pour l'envoi. Ceci permet une communication complètement décentralisée entre utilisateurs.

- **Commencer une session de clavardage :** Une fois connecté, l'utilisateur a à disposition une liste d'utilisateurs en ligne. Lorsqu'il sélectionne un pseudonyme dans la liste, une connexion TCP est établie entre son client et le serveur de l'utilisateur distant. L'historique des messages envoyés est alors affiché sur l'interface graphique(cf [CdC-Bs-9]).
- **Envoyer un message :** Une fois la connexion établie, l'utilisateur peut envoyer un message textuel ou un fichier. Le contenu est saisi dans l'interface graphique, récupéré par le système et envoyé. Dans le cas d'un envoi de fichier, l'interface graphique permet à l'utilisateur de sélectionner le fichier à envoyer.
- **Sauvegarder le message :** A chaque échange de message, les deux parties enregistrent dans la base de données distante. Le message est enregistré à la fois à l'envoi et à la réception. Ceci pour permettre d'actualiser l'historique de la session (cf.[CdC-Bs-14] / [CdC-Bs-11]).

- **<u>Fermer une session de clavardage :</u>** Lorsqu'une session est en cours, un utilisateur peut y mettre fin unilatéralement (cf.[CdC-Bs-13]). La connexion précédemment établie est donc fermée. Aucun message ne peut être envoyé à moins de recommencer une nouvelle session. Et l'utilisateur en est notifié sur l'interface graphique.

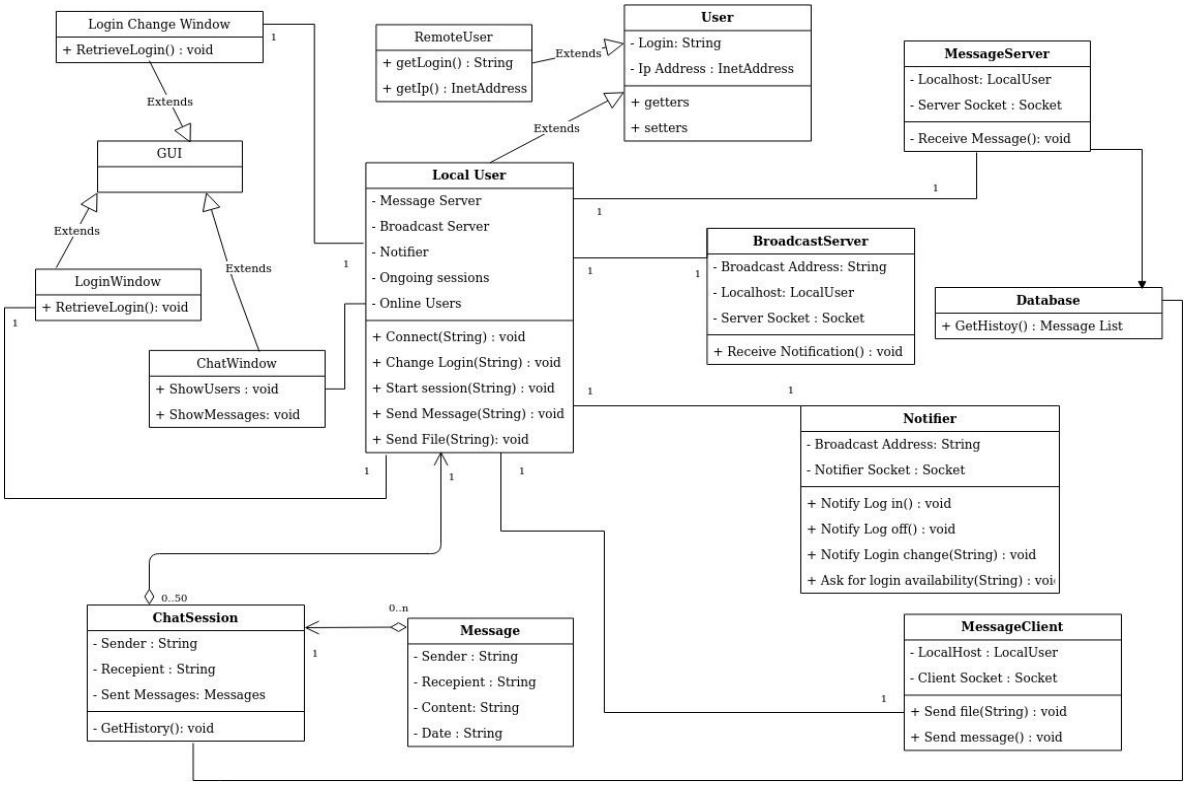
3. Le changement de pseudonyme



L'utilisateur peut changer de pseudonyme autant de fois qu'il le souhaite (cf. [CdC-Bs-16]). Le pseudonyme est entré dans l'interface graphique et soumis au système qui vérifie son unicité. Dans le cas où le pseudonyme est validé, le changement est effectué et les utilisateurs en sont notifiés par le système(cf.[CdC-Bs-17]), sans qu'une session en cours ne soit fermée (cf.[CdC-Bs-18]). Ils peuvent alors mettre à jour leur liste de contacts. La base de données est également mise à jour pour assurer une extraction future précise de l'historique.

III. Diagramme de classe

Le diagramme de classes illustre plus en détail l'architecture du projet qui est composé de trois packages : Données, Réseau et Interface Graphique.
L'utilisateur local (Données) est l'acteur principal du projet. Il gère l'accès aux données. Ses méthodes interagissent directement avec l'interface graphique et le réseau. L'utilisateur pouvant changer de pseudonyme à l'infini, il est nécessaire de l'identifier à l'aide d'un identifiant unique et invariant. La classe LocalUser a donc pour clé privée une adresse IP unique qui l'identifie sur tout le réseau.



- Les notifications du système se font à travers les classes ***Notifier*** et ***BroadcastServer***. Le premier envoie des messages de notifications en broadcast sur le réseau et enregistre les réponses des utilisateurs connectés. Le dernier est un serveur d'écoute qui enregistre toutes les informations diffusées en broadcast et informe l'utilisateur local des changements sur le réseau (nouvelle connexion, déconnexion ou changement de login).
- L'envoi et la réception de messages se font à travers les classes ***MessageSender*** et ***MessageListener***. Le premier (client) envoie un message ou un fichier en TCP et le serveur se charge de l'extraction et de l'enregistrement dans la base de données distante. Pour assurer l'horodatage, la recherche et l'affichage de l'historique, le contenu est "encapsulé" dans un paquet ***Message***, sérialisé (conversion en bytes) en

émission et désérialisé en réception. La différence entre les deux entités possibles pour l'envoi se fait en réception (notamment pour la sauvegarde des fichiers). Pour éviter toute confusion lors de la recherche de l'historique, l'enregistrement dans la base de données ne tient compte que des adresses IP émettrices et destinatrices.

- La classe ***ChatSession*** se charge principalement de l'enregistrement d'une série de messages entre deux utilisateurs localement lors d'une session de clavardage et sur le serveur distant en fin de session. Ceci dans le but de faciliter la recherche de l'historique.

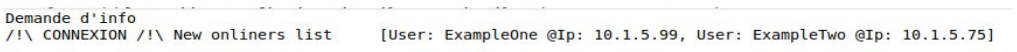
IV. Tests de Validation

Tests sur la communication locale:

Ces tests ont été réalisés localement sur des postes de travail différents. Ils portent sur l'échanges de paquets entre deux adresses IP (pour les notifications et les messages).

- **Tests sur la notification :** Depuis un poste de travail, un utilisateur se connecte et envoie un paquet en broadcast sur le réseau. Le test de notification fonctionne si le deuxième utilisateur reçoit la notification et que le premier reçoit une réponse en échange avec ses informations. Le principe est le même pour une déconnexion et un changement d'identifiant.
  Après une notification, la liste de membres actifs des utilisateurs doivent être identiques.

Terminal du premier utilisateur



Terminal du deuxième utilisateur

- **Tests sur l'échange de message :** Ces tests ont le même principe que les précédents. Les situations testées sont principalement : l'envoi d'un message sans démarrage d'une session au préalable, la fermeture d'une session, l'envoi de fichiers, la communication après un changement de pseudonyme (du destinateur ou de l'expéditeur), et la communication simultanée avec plusieurs utilisateurs.
  Tous ces tests ont été réalisés à l'aide d'affichages sur le terminal.

Tests sur la base de données:

Ces tests ont été réalisés à travers l'api Junit. Ils portent principalement sur la mise à jour des pseudonyme, l'ajout d'une adresse IP à la base de données, et des méthodes de recherche. Ils sont disponibles sur le dépôt, dans le package Tests.

Conclusion

Au vu de l'initial cahier des charges, nous pensons avoir un rendu fonctionnel. Le déploiement de notre application, quelque soit la plateforme, est fonctionnel. La gestion des pseudonymes et de leur unicité est implémentée, ainsi que l'historique des messages. C'est également le cas pour le démarrage d'une session de clavardage, ainsi que l'envoi de messages et fichiers. Enfin, une interface graphique fonctionnelle est mise en place.
Nous n'avons cependant, par manque de temps, pas implémenté le serveur de présence.

Ce projet, bien que long et ardu, est un bon exemple de ce qui nous attend dans le monde professionnel. Le fait d'avoir d'abord une partie Conception, puis une partie Réalisation est intéressant, même si nos conceptions ont complètement changé entre le début et la fin du projet.