Rapport de projet

Système de clavardage distribué interactif multi-utilisateur en temps réel

Brice Moutanin
Benjamin Vendeville
4IR_A1
Promotion 54 / Année 2019/2020

Introduction

Dans le cadre de l'évaluation des enseignements de conception et de programmation orientées objet, un projet de développement nous a été soumis. Ce projet consiste à mettre en place un système de communication au sein d'une entreprise, à travers le développement d'un logiciel destiné à cet effet.
Dans ce rapport, nous verrons plus en détails les modalités de ce projet, en explicitant le cahier des charges fournis. Puis, nous expliciterons notre démarche dans la conception du système, en passant en revu les différents diagrammes de conception que nous avions établi. Nous expliquerons également les choix que nous avons fait, et pourquoi ils ont été nécessaires. Ensuite, nous établirons un manuel d'utilisation, afin d'expliquer comment fonctionne le système. Enfin, nous parlerons brièvement des tests que nous avons mené pour prouver le bon fonctionnement de notre application.

1] Modalités du projet

A) Description générale du rendu attendu

Commençons tout d'abord par donner une explication plus détaillée du projet demandé :

Il s'agit d'un système de communication qui sert de support aux équipes et groupes de l'entreprise afin de leur permettre d'accroitre leur efficacité naturelle.
Le système utilise un(des) agent(s) afin d'accomplir sa mission.
En tant que relais de communication, le système permet aux intervenants de se coordonner par l'échange de messages textuels entre eux mais également des illustrations sous la forme d'images, des schémas en plus de permettre l'échange de fichiers de travail.
Le système supporte la dynamicité du contexte. Il détecte et prend en compte les (de) connexions des utilisateurs subites au fil de l'eau (en cours d'exécution). Selon la situation, l'architecture du système change. Le système prévoit des comportements et des procédures de communication différentes et adaptés. En entreprise, lorsque tous les utilisateurs sont sur le réseau propriétaire, le système est dépourvu d'une architecture centralisée. Toutefois, lorsque certains utilisateurs se connectent de l'extérieur de l'entreprise, le système repose sur une architecture 3-tiers.

B) Cahier des charges

Fonctionnalité relatives à l'agent Fonctionnalités d'administration de l'agent

[CdC-Bs-1] Le système doit pouvoir être déployé sur un poste de travail fonctionnant sur le système d'exploitation Windows

[CdC-Bs-2] Le système doit pouvoir être déployé sur un poste de travail fonctionnant sur le système d'exploitation Linux

[CdC-Bs-3] Le système doit pouvoir être déployé sur un poste de travail fonctionnant sur le système d'exploitation OS X

[CdC-Bs-4] Le système doit pouvoir être déployé sur un poste de travail fonctionnant sur le système d'exploitation Android

[CdC-Bs-5] Le déploiement du système devra se limiter à la copie sur le poste de travail d'une série de fichiers et la création d'un raccourci pour l'utilisateur

[CdC-Bs-6] La taille globale de l'ensemble des ressources composant le système ne devra pas excéder 50 Méga-Octets sous une forme non compressée quel que soit le système d'exploitation sur lequel il est déployé

Fonctionnalités d'utilisation de l'agent

[CdC-Bs-7] Le système doit permettre à l'utilisateur de choisir un pseudonyme avec lequel il sera reconnu dans ses interactions avec le système

[CdC-Bs-8] Le système doit permettre à l'utilisateur d'identifier simplement l'ensemble des utilisateurs pour lesquels l'agent est actif sur le réseau

[CdC-Bs-9] Le système doit permettre à l'utilisateur de démarrer une session de clavardage avec un utilisateur du système qu'il choisira dans la liste des utilisateurs pour lesquels l'agent est actif sur le réseau

[CdC-Bs-10] Le système doit garantir l'unicité du pseudonyme des utilisateurs pour lesquels l'agent est actif sur le réseau

[CdC-Bs-11] Tous les messages échangés au sein d'une session de clavardage seront horodatés

[CdC-Bs-12] L'horodatage de chacun des messages reçus par un utilisateur sera accessible à celui-ci de façon simple

[CdC-Bs-13] Un utilisateur peut mettre unilatéralement fin à une session de clavardage

[CdC-Bs-14] Lorsqu'un utilisateur démarre un nouvelle session de clavardage avec un utilisateur avec lequel il a préalablement échangé des données par l'intermédiaire du système, l'historique des messages s'affiche

[CdC-Bs-15] L'utilisateur peut réduire l'agent, dans ce cas, celui-ci se place discrètement dans la barre des tâches sous la forme d'une icône lorsque le système d'exploitation permet cette fonctionnalité

[CdC-Bs-16] Le système doit permettre à l'utilisateur de changer le pseudonyme qu'il utilise au sein du système de clavardage à tout moment

[CdC-Bs-17] Lorsqu'un utilisateur change de pseudonyme, l'ensemble des autres utilisateurs du système en sont informés

[CdC-Bs-18] Le changement de pseudonyme par un utilisateur ne doit pas entraîner la fin des sessions de clavardage en cours au moment du changement de pseudonyme

Exigences de performances

[CdC-Bs-19] Le déploiement du système doit être réalisable en 2 heures à partir de la prise de décision de déploiement.

[CdC-Bs-20]Le changement de pseudonyme d'un utilisateur doit être visible de l'ensemble des autres utilisateurs dans un temps inférieur à 20 secondes

[CdC-Bs-21] Le temps écoulé entre l'envoi d'un message par un utilisateur et la réception par un autre utilisateur ne doit pas excéder 1 seconde

[CdC-Bs-22] Le système doit permettre la mise en place de 1000 sessions de clavardage simultanées au sein de celui-ci

[CdC-Bs-23] L'agent doit permettre la mise en place de 50 sessions de clavardage simultanées

[CdC-Bs-24] Lorsque la vérification de l'unicité du pseudonyme de l'utilisateur échoue, l'utilisateur doit en être informé dans une période ne dépassant pas 3 secondes

[CdC-Bs-25] Le temps d'apparition des utilisateurs au sein de la liste des utilisateurs pour lesquels l'agent est actif ne doit pas excéder 5 secondes

[CdC-Bs-26] Le système doit permettre un usage simultané par au minimum 100 000 utilisateurs

Exigences de ressources

[CdC-Bs-27] Le système doit avoir une empreinte mémoire inférieure à 100Mo **[CdC-Bs-28]** Lors de son exécution, le système ne doit pas solliciter le processeur plus de 1% du temps lorsque la mesure est réalisée sur un intervalle de 5 secondes

[CdC-Bs-29] Le système doit présenter une réactivité normale pour une application de clavardage

Exigences de sûreté de fonctionnement

[CdC-Bs-30] Le système doit garantir une intégrité des messages supérieure à 99,999%
[CdC-Bs-31] Une utilisation normale du système ne doit pas avoir d'impact sur le reste du système

C) Choix de conception

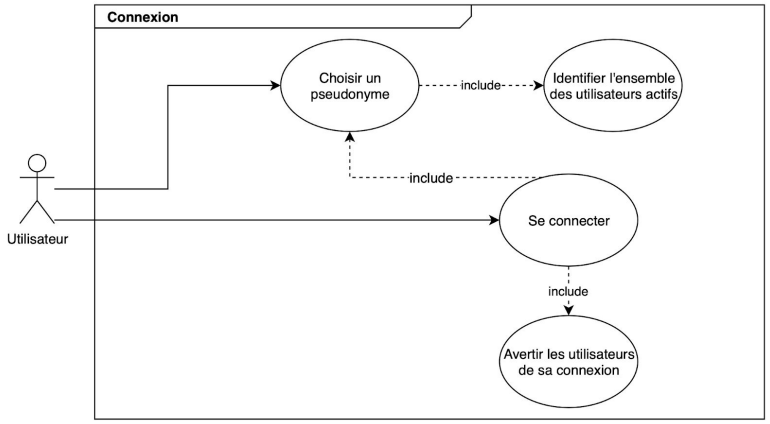
Pour mener à bien ce projet, nous avons dû faire certains choix.
Tout d'abord, puisque le système ne sera déployé que sur un réseau local dans un premier temps, avec des machines reliées entre elles par des câbles Ethernet, nous avons choisi de permettre l'envoi des messages par UDP. Dans ce cas, les envois de messages restent relativement fiables, puisque l'on est dans un cadre de connexion filaire.
Ensuite, nous sommes partis du principe que chaque personne se connecte toujours sur la même machine. Ainsi, nous avons décidé de stocker l'historique des messages avec chaque utilisateur localement, sur la machine concernée.
Pour les raisons évoquées ci-dessus, nous avons également choisi d'identifier une machine par son adresse IP : en effet, sur notre réseau local, chaque machine conservera toujours la même adresse IP. Nous avions, dans un premier temps, essayé d'identifier chaque machine par son adresse physique, mais il a finit par apparaître que les méthodes utilisées en Java pour obtenir cette adresse ne renvoyait pas toujours les bonnes valeurs, nous avons donc décidé d'abandonner cette idée.

Les messages internes, c'est à dire non visibles par l'utilisateur mais qui servent au bon fonctionnement du système, seront également transmis en UDP. Nous verrons plus tard quelle est la forme de ces messages internes et ce qu'ils signifient.

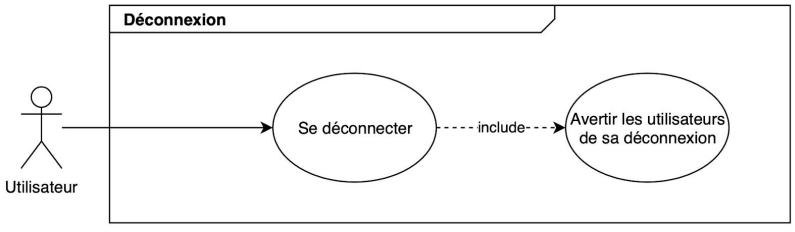
2] Conception du projet

Pour nous aider à concevoir notre projet, nous nous sommes appuyé sur notre cours de conception orientée objet. Nous avons donc suivi plusieurs étapes, en définissant des cas d'utilisations pour cerner les principales fonctions du logiciel, puis des diagrammes de séquences pour comprendre comment chaque élément de notre système fonctionnent ensemble, et enfin un diagramme de classe pour identifier les différents composants finaux de ce système.

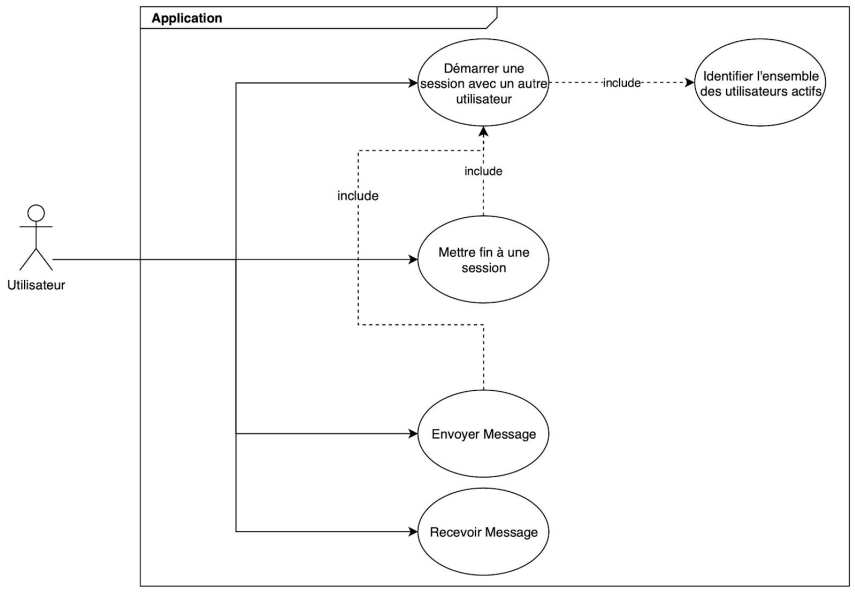
A) Cas d'utilisation



Pour qu'une connexion se fasse, l'utilisateur doit choisir un pseudonyme. Cependant, il est nécessaire de s'assurer que le pseudonyme qu'il choisira ne soit pas déjà choisi par un autre des utilisateurs actifs. C'est pourquoi avant de choisir un pseudonyme, le système aura déjà dû identifier l'ensemble des utilisateurs actifs, avec leurs pseudonymes et leurs adresses IP.
Une fois qu'un pseudonyme sera choisi et validé par le système, l'utilisateur peut se connecter. Une fois connecté, il doit informer les autres utilisateurs qu'il est lui même connecté, pour qu'ils puissent l'enregistrer dans leurs listes d'utilisateurs actifs.



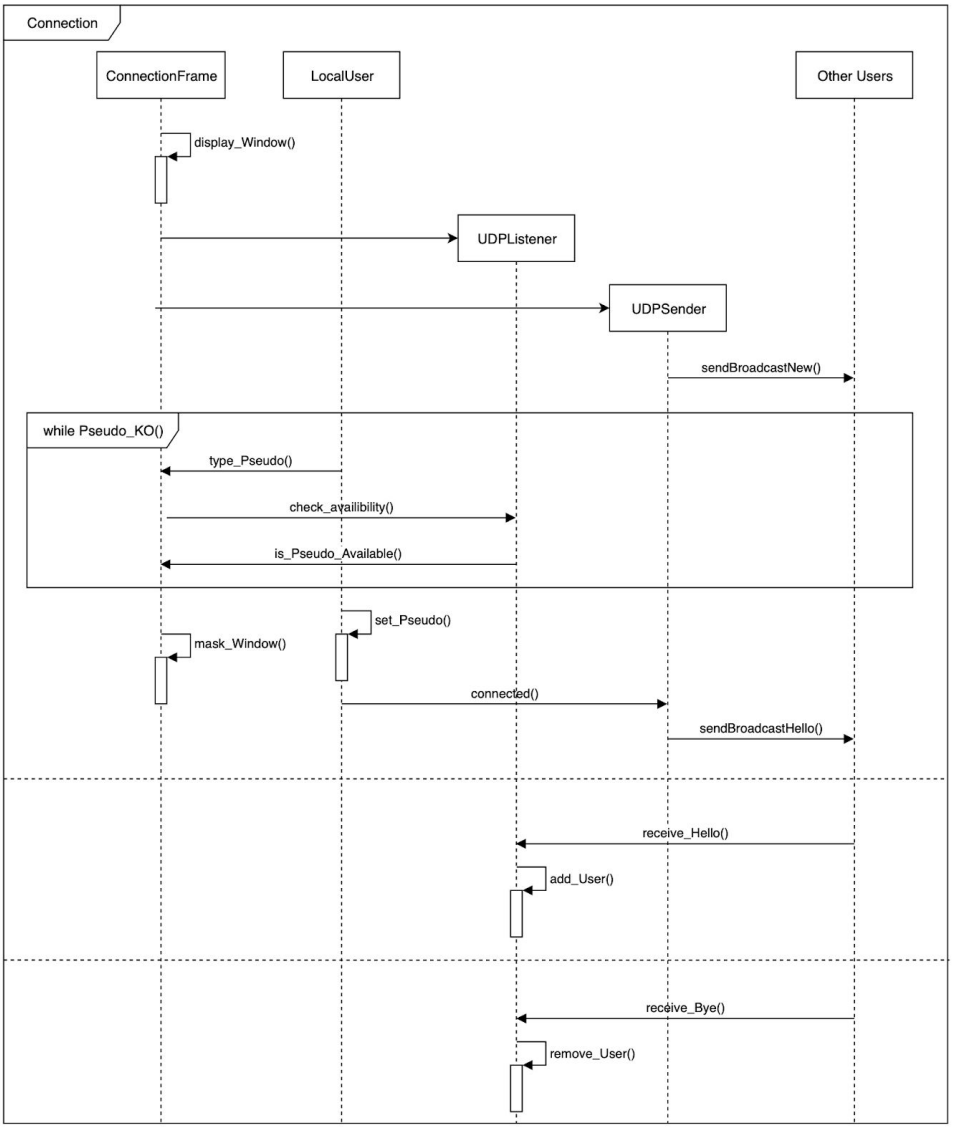
Pour ce qui est de la déconnexion, les choses sont un peu plus simples. En effet, il suffit simplement de faire en sorte que, lorsque l'utilisateur procède à la fermeture de l'application, le système informe les autres utilisateurs actifs que cet utilisateur se déconnecte et n'est plus lui-même un utilisateur actif.



Enfin, la communication est la fonctionnalité principale du logiciel. Un utilisateur doit être capable de démarrer une session de discussion avec un autre utilisateur actif. Pour cela, il faudra bien sûr que le système ait identifié tous les utilisateurs actifs sur le réseau.
Une fois que l'utilisateur aura choisi un autre utilisateur avec qui il voudra démarrer une session, il pourra au choix envoyer un message, ou mettre fin à cette session.
Parallèlement, il pourra toujours recevoir des messages de n'importe quel utilisateur actif.
De plus, un utilisateur sera également capable d'ouvrir simultanément plusieurs sessions de discussions avec différents utilisateurs.

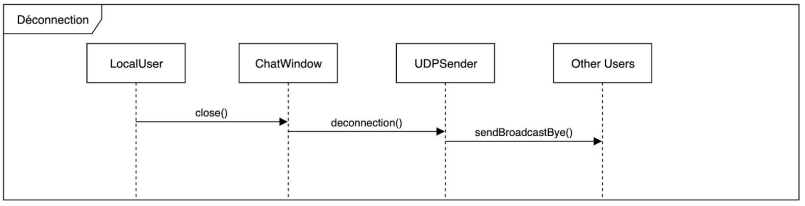
B) Diagrammes de séquences

Pour décrire les différentes actions qui s'exécutent lors du lancement du programme, voici quelques diagrammes de séquences correspondant aux interactions principales qu'auront les utilisateurs avec le système.

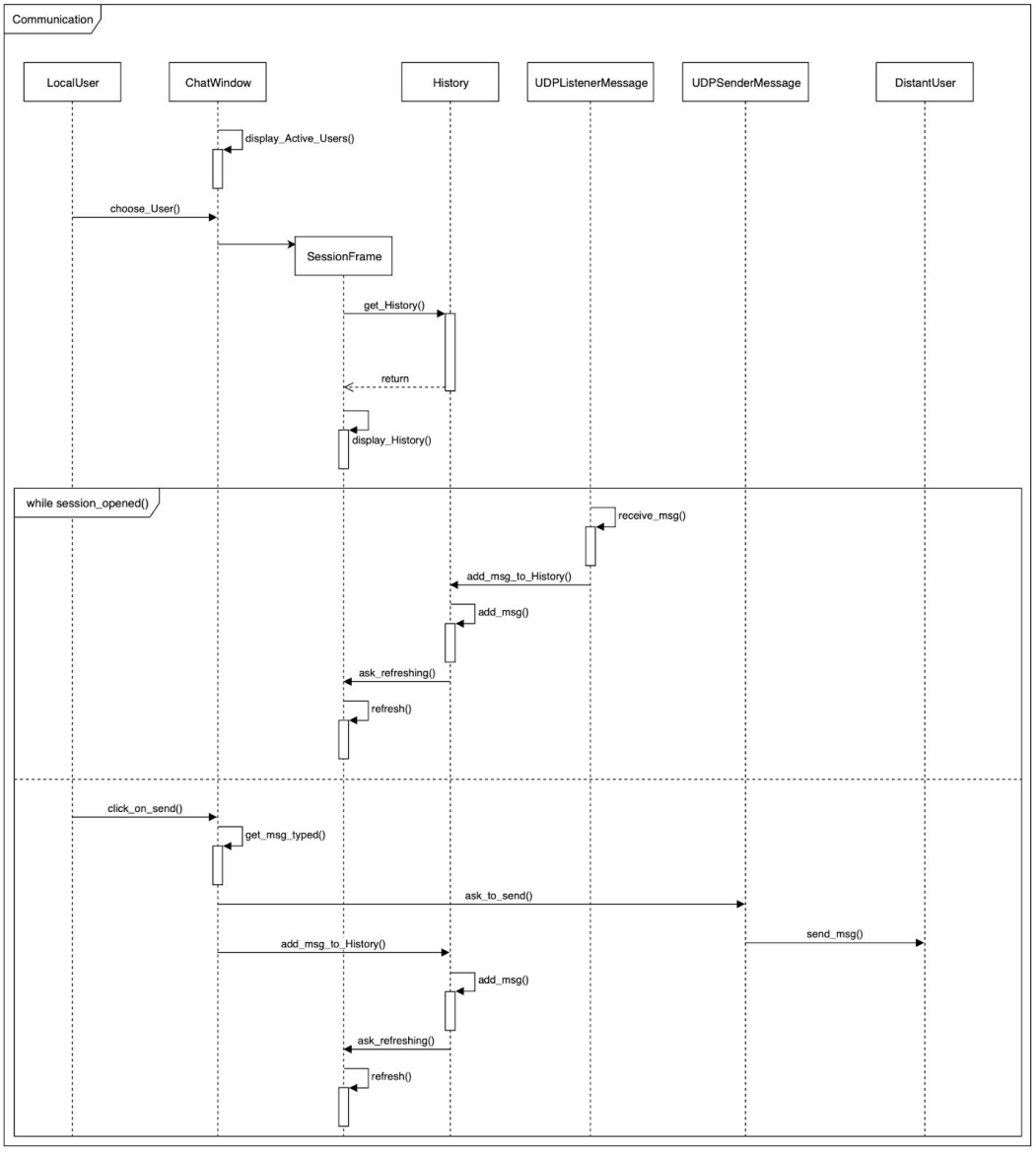


Pour se connecter au système, l'utilisateur devra choisir un pseudo. En même temps que va s'afficher la fenêtre, le système envoie en broadcast un message interne à toutes les autres machines. Les messages internes sont de la forme suivante :
Signal Pseudo adresseIP
Le signal peut être soit New, Hello ou Bye. Le signal New signifie qu'une machine veut se connecter, mais qu'elle n'est pas encore considérée comme active, elle veut simplement recueillir les informations sur les autres utilisateurs actifs (Pour un New, le pseudo n'est pas important puisqu'il n'a pas encore été choisi). Le signal Hello sert à informer que la machine qui l'envoie est maintenant connectée et active. Enfin, le signal Bye sert à informer qu'une machine se déconnecte du système.
On a donc lors de l'apparition de la fenêtre de connexion un envoi du signal New en broadcast à tous les utilisateurs actifs. Lorsqu'une machine reçoit un message New de la part d'une nouvelle machine, elle doit transmettre un message Hello à cette nouvelle machine, pour qu'elle puisse enregistrer tous les utilisateurs actifs.
Une fois que la machine connaît tous les utilisateurs actifs, elle peut décider si le pseudo choisi par l'utilisateur local est disponible ou pas. Tant que le pseudo proposé n'est pas

disponible, il faut que l'utilisateur local en propose un autre. Une fois que le pseudo est disponible, la fenêtre de connexion disparaît, et la fenêtre de chat apparaît. De plus, la machine envoie un message Hello en broadcast à tous les utilisateurs actifs pour prévenir qu'à présent elle est connectée et active.
En parallèle à tout ce processus, chaque machine sait aussi que si elle reçoit un message Bye, elle doit retirer la machine émettrice de ce message de sa liste d'utilisateurs actifs.



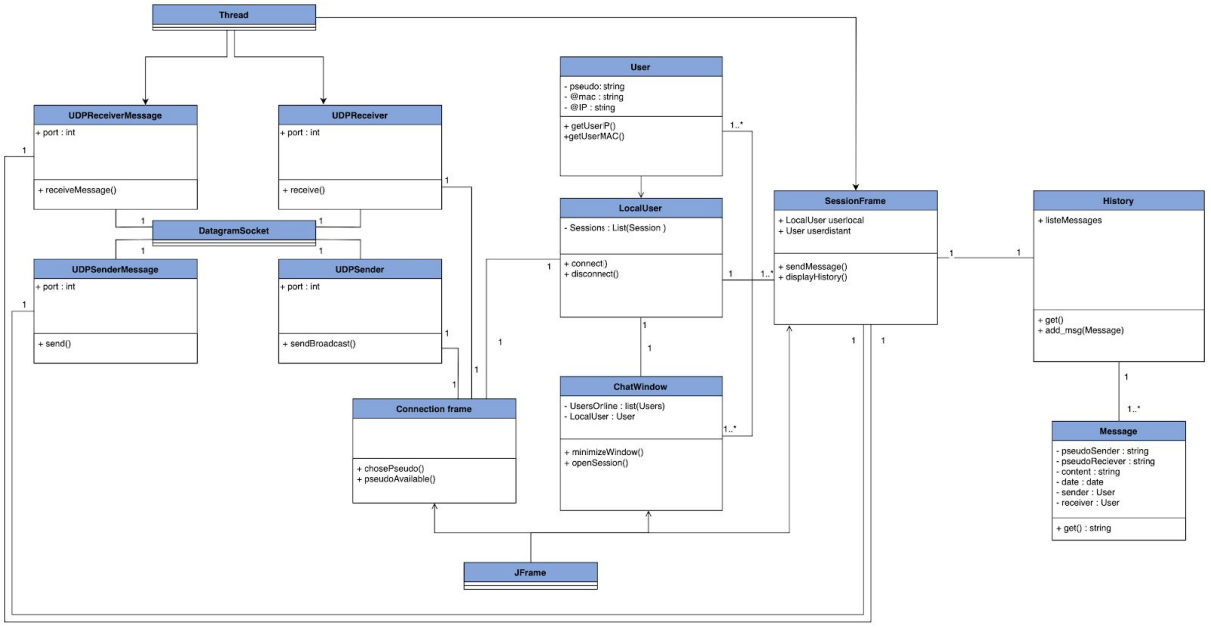
La déconnexion est un processus simple, il suffit simplement à la machine de l'utilisateur qui se déconnecte d'envoyer, à la fermeture de l'application, un message Bye en broadcast.



Lorsqu'un utilisateur veut communiquer avec un autre, il doit choisir cet autre utilisateur dans la liste affichée par la fenêtre de chat. Une fois qu'il l'a choisi, une nouvelle fenêtre de discussion s'ouvre, propre à la discussion avec l'utilisateur choisi.
Lors de l'ouverture de cette fenêtre, le système va chercher l'historique de cette conversation, stockée localement, et afficher cet historique dans la fenêtre.
A partir de là, deux processus s'exécutent en parallèle : l'envoi et la réception des messages.
Pour envoyer un message, l'utilisateur local tape un message puis clique sur envoyer. Le message est transformé afin de pouvoir être envoyé à l'utilisateur visé via UDP. De plus, lorsque le message s'envoie, il est ajouté à l'historique de discussion, et la fenêtre de discussion rafraîchit l'historique afin de faire apparaître ce message.
La réception d'un message est automatique : si un message est reçu, il est directement ajouté à l'historique de la conversation associée. Si la fenêtre de discussion avec cet utilisateur est déjà ouverte, elle est rafraîchie pour pouvoir afficher ce dernier message reçu.

C) Diagramme de classes

Grâce à nos différents choix de conception, et la manière dont fonctionne notre système, nous avons pu dresser le diagramme de classes suivant :



3] Manuel d’utilisation

Pour exécuter l’application, il suffit d'exécuter le fichier .jar qui se trouve ainsi :
Projet-COO-POO/POO/chatsystem.jar
Pour exécuter ce fichier, il faut taper la ligne de commande suivante :
java -jar chatsystem.jar

Lorsque l’application se lance, la fenêtre suivante apparaît :

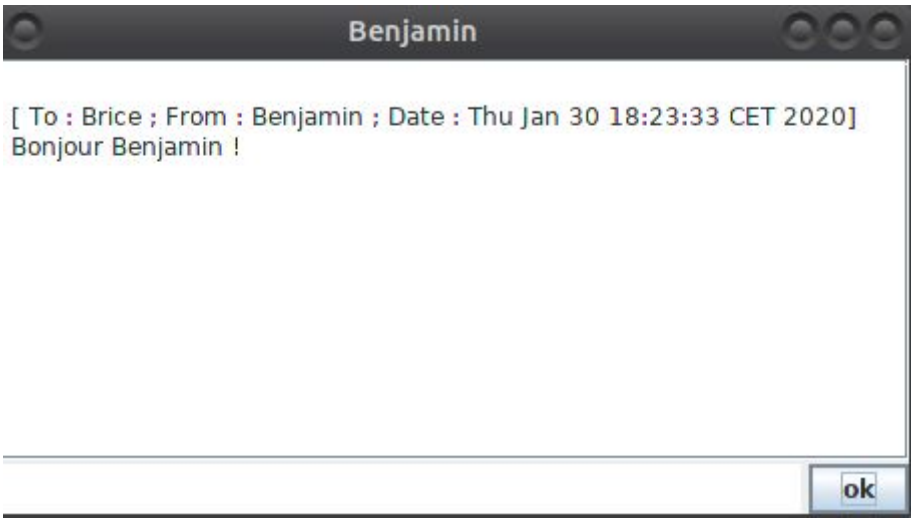
La fenêtre vous invite à choisir un pseudo. Pour rentrer votre pseudo, tapez simplement le pseudo de votre choix, puis cliquer sur “ok”, ou appuyez sur la touche Entrée.
Si le pseudo que vous avez choisi est déjà utilisé par un autre utilisateur, la zone de texte se vide et le message “Pseudo déjà utilisé s’affiche”.

Si le pseudo est disponible, la fenêtre de connexion disparaît, et la page de Chat apparaît :

Sur la page de Chat, une liste des utilisateurs actifs est affichée et mise à jour en permanence. C’est parmi cette liste que vous pourrez choisir un utilisateur avec lequel démarrer ou poursuivre une discussion.

Il est également normal de retrouver votre propre pseudo dans cette liste : en effet, communiquer avec nous-même nous a permis de poursuivre nos tests lorsque nous travaillions sur nos machines personnelles.
Edit : Sur la nouvelle version, un bouton "Changer de pseudo" est maintenant disponible. Cependant, il y a quelques soucis avec son utilisation, puisque s'il permet de changer son pseudo, les autres utilisateurs gardent également dans leurs listes l'ancien pseudo, ce qui crée des doublons. Nous n'avons malheureusement pas eu le temps de corriger cette erreur.

Sur cette fenêtre de Chat, il faudra double-cliquer sur la personne de votre choix pour ouvrir une discussion avec elle. Une fenêtre de discussion s'ouvrira alors :



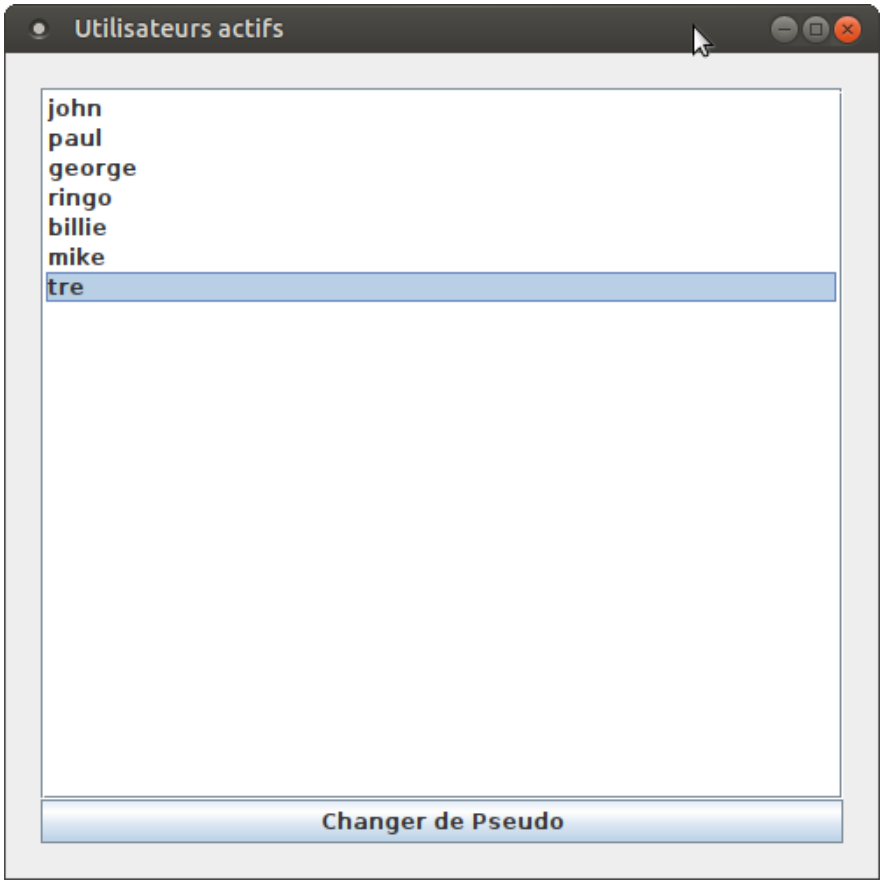
Dans cet exemple, nous sommes Brice et nous ouvrons une fenêtre de discussion avec Benjamin. De plus, un message a été envoyé par Brice : "Bonjour Benjamin !"
A chaque discussion ouverte, l'historique de la conversation est affiché.
Pour envoyer un message, il suffit de taper le message à envoyer dans la zone de texte en bas, puis de cliquer sur "ok".
Les messages envoyés et reçus sont automatiquement ajoutés à l'historique, et affichés à l'écran.

IMPORTANT : - Il est important de noter que, dans notre application, pour pouvoir communiquer, deux utilisateurs doivent avoir au préalable ouvert chacun la fenêtre de discussion avec l'autre utilisateur.
- Il existe également un bouton send file qui ouvre une fenêtre contextuelle pour le choix d'un fichier à envoyer. Cependant, l'envoi du fichier n'a pas été implémenté, par manque de temps.

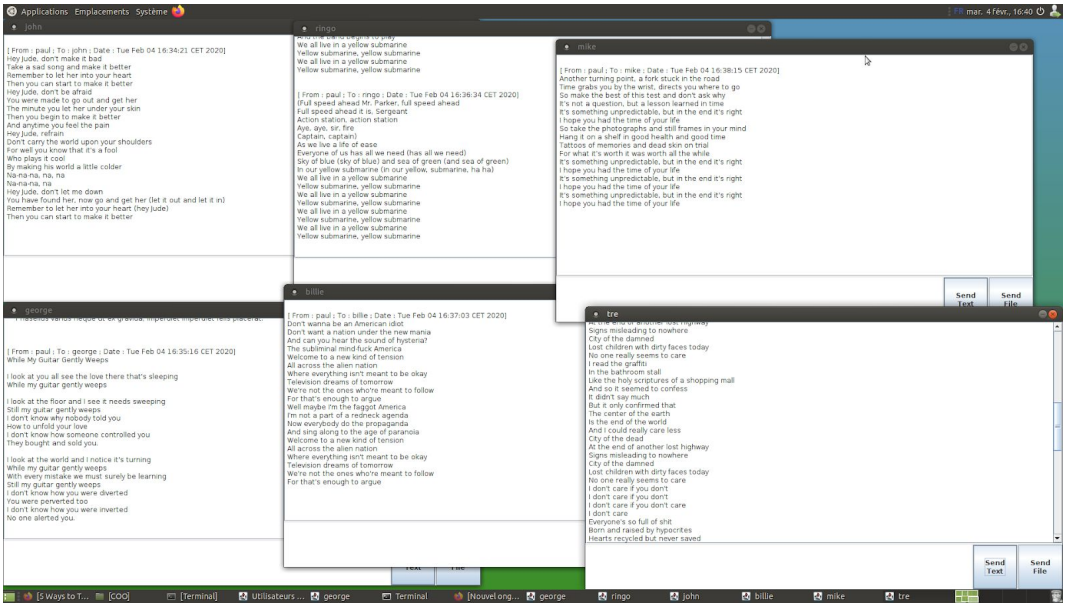
Remarque : Dans l'en-tête du message affiché, on remarque que l'émetteur et le récepteur du message ont été interchangés. Cette erreur a été corrigée.

4] Tests d’utilisation

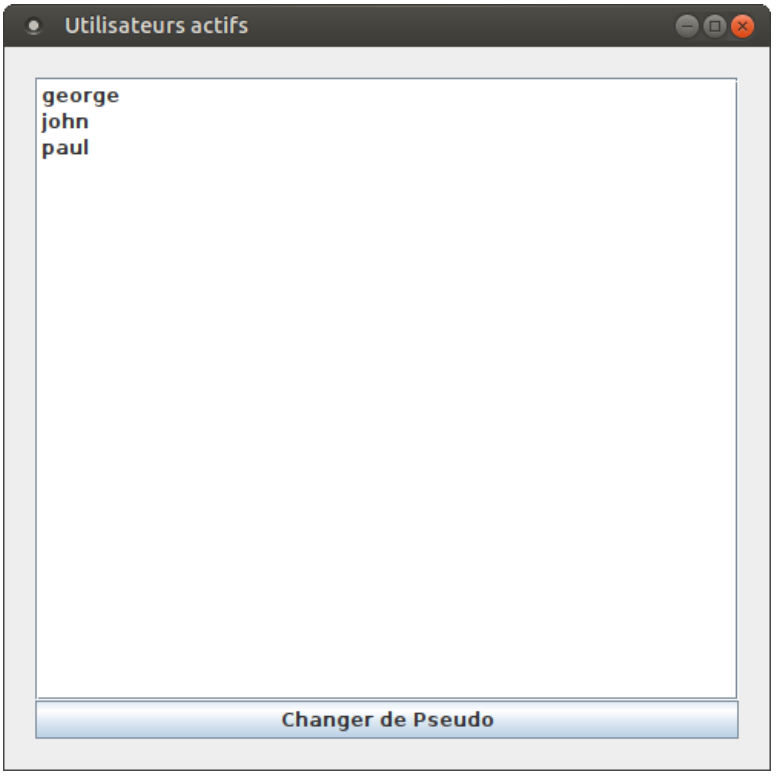
Pour les tests, nous avons effectué plusieurs scénarios : tout d’abord nous avons connecté 6 utilisateurs (à défaut de pouvoir faire plus à cause de la quantité limitée de matériel à notre disposition) puis vérifié que chaque utilisateur voyait les autres comme connectés.



Ensuite nous avons vérifié que, même avec autant d’utilisateurs, nous pouvions envoyer des messages et qu’ils les recevraient.



Et enfin nous avons testé que les utilisateurs disparaissent bien de la liste lorsqu'ils se déconnectent :



5] Conclusion

Ce travail nous aura permis dans un premier temps, de nous familiariser avec Java, mais surtout d'apprendre à gérer un projet. Nous avons d'abord dû concevoir le projet et réfléchir à l'avance à toutes les parties nécessaires, ce qui nous a non seulement permis de mieux nous organiser par la suite, mais aussi de construire un projet le plus efficace possible. Au fur et à mesure de l'avancement du projet, de la découverte de problèmes inattendus, nous avons dû faire évoluer la conception, ce qui a été très formateur. Enfin, nous n'avons pas pu implémenter toutes les fonctionnalités (comme l'envoie de fichiers) pour cause de temps, mais celles-ci restent parfaitement compatibles avec notre conception.