MURAT Félix
OMRE Erik
4 IR B

Rapport de projet de Conception et Programmation Orientées Objet

Liens utiles: -Git: https://github.com/Tsjerfell/ChatSystem
-Jira: https://sembomre.atlassian.net/jira/software/projects/PROJ/boards/3

I-Description des différentes parties de notre projet

Ports utilisés

Le multicast se base sur trois ports différents.

- Le premier est le port 12345 qui écoute sur l'adresse multicast “225.6.7.8”. Ce port, ou cette adresse multicast, est utilisé quand quelqu'un vient de se connecter ou de se déconnecter, ou quand quelqu’un change son pseudonyme.
- Le port 12346 est utilisé lorsqu’on se connecte, pour recevoir des paquets nommés “AckConnexion” venant de tous les autres utilisateurs déjà connectés. Ces paquets contiennent les pseudonymes et les adresses IP des autres utilisateurs.
- Le troisième est le port 12347 qui est en permanence à l’écoute d’ utilisateurs qui voudraient initialiser une conversation avec l’utilisateur local.

Connection

La connection se fait en trois étapes:
Premièrement, un thread est lancé qui crée un socket qui se connecte sur le port 12346. Ce socket va servir à recevoir des paquets de type “AckConnexion”. On expliquera plus tard l’utilité de ce socket.
Deuxièmement, un second thread est créé qui se connecte sur l’adresse IP “225.6.7.8” avec le port 12345 et écoute en permanence.
Troisièment , on se connecte sur l’adresse multicast “225.6.7.8” et on y envoie un paquet de type “Connexion”.
Tous les autres utilisateurs, qui sont déjà connectés à l'adresse multicast, vont recevoir ce paquet. Ils vont stocker cet utilisateur sous forme d’un type nommé “ otherUser” comprenant son adresse IP et son pseudonyme. Suite à cela, ils vont envoyer un paquet de type “AckConnexion” vers l’adresse de l’utilisateur qui vient de se connecter, vers le port 12346. C’est pour cela que l’on a commencé par créer un socket sur le port 12346. Ensuite, l’utilisateur qui vient de se connecter va recevoir les paquets des autres utilisateurs qui sont déjà connectés. Comme cela, il reçoit le pseudo et l’adresse IP des autres utilisateurs.

Déconnection

En suivant le même principe que pour la connexion, on envoie sur l'adresse multicast un paquet de type "Déconnexion".
En recevant un tel paquet, notre programme enlève simplement l'utilisateur expéditeur de la liste des utilisateurs connectés.

Changement de Pseudonyme

Comme on est toujours au courant de tous les pseudonymes de tous les utilisateurs connectés, il suffit de chercher dans la liste nommée " ListOtherConnectedUsers" et de voir si le pseudo n'est pas déjà utilisé. Si ce n'est pas le cas, on le change et on envoie un paquet de "Changement de pseudo" sur l'adresse multicast. Si le pseudo est déjà utilisé, on ne fait rien, sauf dire à l'utilisateur que le pseudo est déjà utilisé.
Pour récupérer le pseudo que l'utilisateur souhaite utiliser, on utilise un JTextField, dans lequel l'utilisateur peut écrire, et un JButton qui, quand on clique dessus, lit ce qu'il y a dans le TextField pour après continuer avec ce qui est écrit dans la partie Connection.
En recevant un paquet "Changement de pseudo", on modifie simplement le pseudonyme dans la liste "listOtherConnectedUsers".

Serialization

Pour envoyer des paquets en UDP, il faut que le contenu de ces derniers soit une chaîne de caractères . C'est pour cela que nous utilisons la classe SerializationUtils. Son rôle est de changer le format de notre classe paquet en char[ ] et inversement.

Interface

Comme nous l'avons vu dans les cours et les TDs, nous avons décidé d'utiliser JavaSwing comme library pour faire l'interface.
L'interface est découpé en cinq espaces: Changement de pseudo, Connected User, User you are talking to, Sending message et Deconnection. Nous allons maintenant détailler le fonctionnement des différentes parties.

Liste des utilisateurs connectés

Le champ dans lequel les utilisateurs distants apparaissent est un JTextArea. Pour y écrire, il nous faut un thread. C'est pour cela que nous avons créé la class ThreadTextAreaConnectedUsers. De plus, il nous fallait un mutex. En effet, lorsqu'on se connecte, on reçoit plein des paquets de type "AckConnection" qui impliquent l'ajout de nouveaux utilisateurs dans la liste de Connected Users. Beaucoup de threads veulent donc écrire dans le même JTextArea en même temps.
Nous avons aussi mis en place un bouton: "Initialize conversation with user clicked above". Pour savoir avec quel utilisateur l'utilisateur local veut parler, on utilise un CaretListener. Ce Listener trouve la ligne "i", qui est la ligne sur laquelle l'utilisateur a cliqué. Ensuite, pour

récupérer le bon utilisateur, on cherche dans la liste l'utilisateur numéro “i” (en prenant en compte que la première ligne correspond au numéro 0).

Liste des utilisateurs avec lesquels une conversation est en cours

Cette partie ressemble à la partie "Connected Users”, avec une liste des utilisateurs avec lesquels l’utilisateur est en conversation, et un thread qui, basé sur cette liste, met à jour le JTextArea.
Il y a également un bouton pour arrêter la conversation avec un autre utilisateur. Ici on utilise le même principe que celui vu plus haut, où une CaretListener détecte le numéro de la ligne pour trouver l’utilisateur correspondant. En cliquant sur le bouton “end conversation”, le logiciel envoie un message commençant par 0 pour indiquer que la conversation est terminée. Tous les messages commencent soit par “1”, soit par “0”, pour indiquer si on continue la conversation ou si on l’arrête.

Base de données

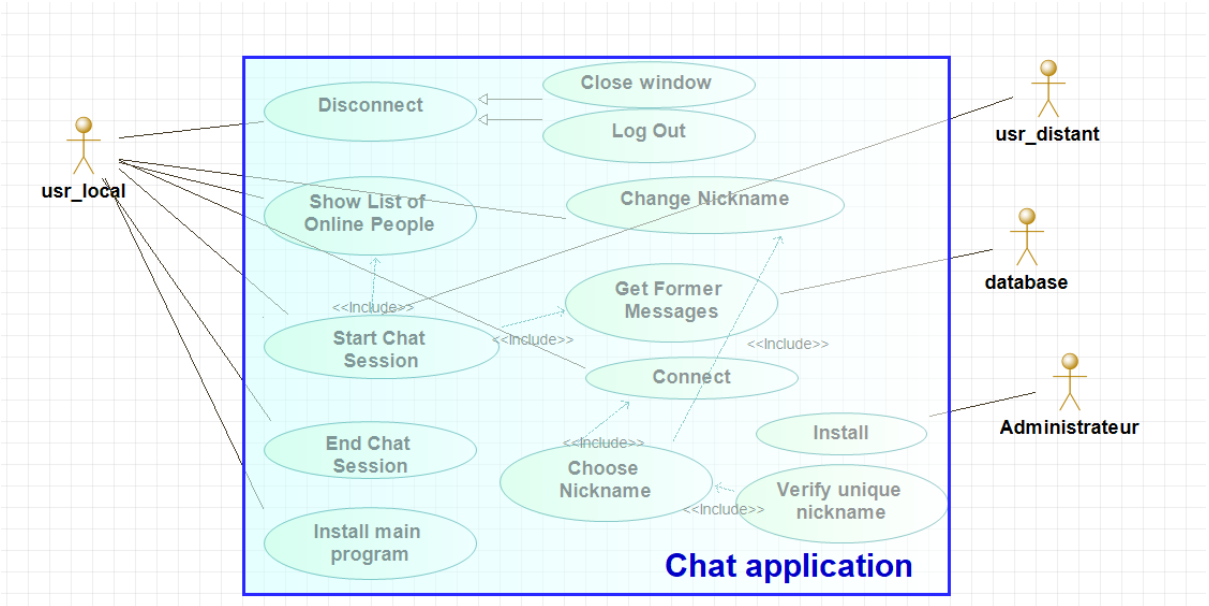
Afin de stocker l’historique des messages échangés entre tous les utilisateurs de l’application, on utilise une database externe aux machines utilisatrices, située sur un serveur du GEI. Dans cette database se trouve la table “Conversation”. Chaque ligne de cette table contient un message, sa date, et les adresses IP de son expéditeur et de son destinataire. Il est possible d’ajouter des entrées à cette table, ainsi que d’obtenir une conversation entre deux utilisateurs grâce au type conversation, composé de quatre ArrayList de strings. Les listes correspondent aux adresses IP (expéditeur et destinataire), aux dates et aux messages. Les éléments provenant des différentes listes mais pris au même indice dans chaque liste, correspondent entre eux, ce qui permet de reconstituer l’historique des conversations entre deux utilisateurs.

Envoyer un message

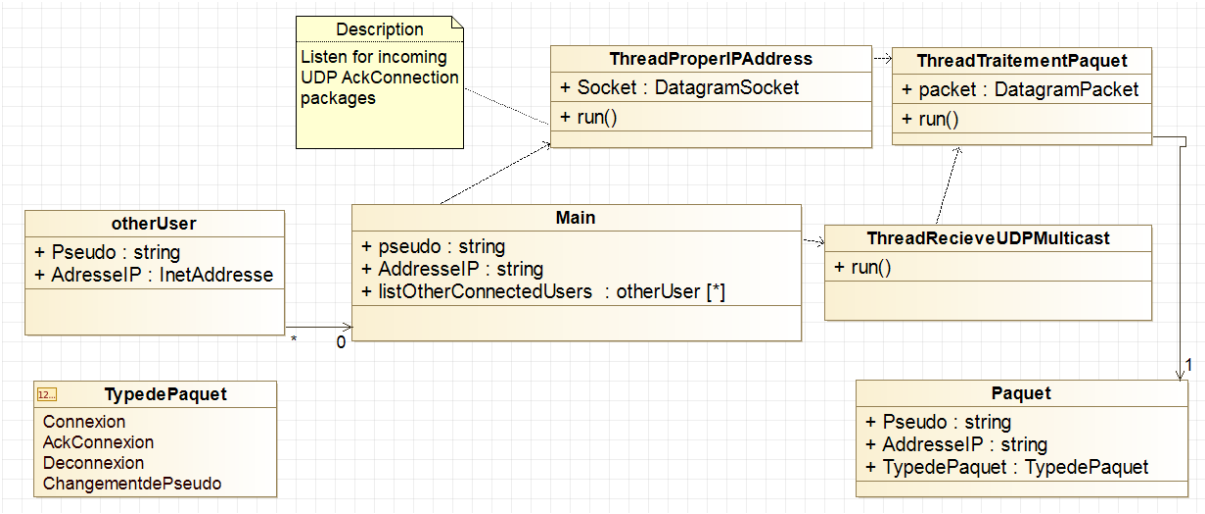
Lorsqu’on initialise une conversation avec un autre utilisateur, l’historique entre ces deux utilisateurs s’affiche. Il s’agit également d’un thread qui écrit dans le JTextArea. Pour afficher l’historique, on contacte la database qui nous renvoie l’historique entre les adresses IP locale et distante.
Après avoir envoyé un message, on ajoute ce dernier à notre database et on actualise l’affichage de l’historique entre les deux utilisateurs. Aussi, on alerte l’autre utilisateur qu’on a envoyé un message, afin qu’il puisse actualiser son affichage d’historique grâce à la database.

II-Diagrammes:

Diagramme de cas d'utilisation:

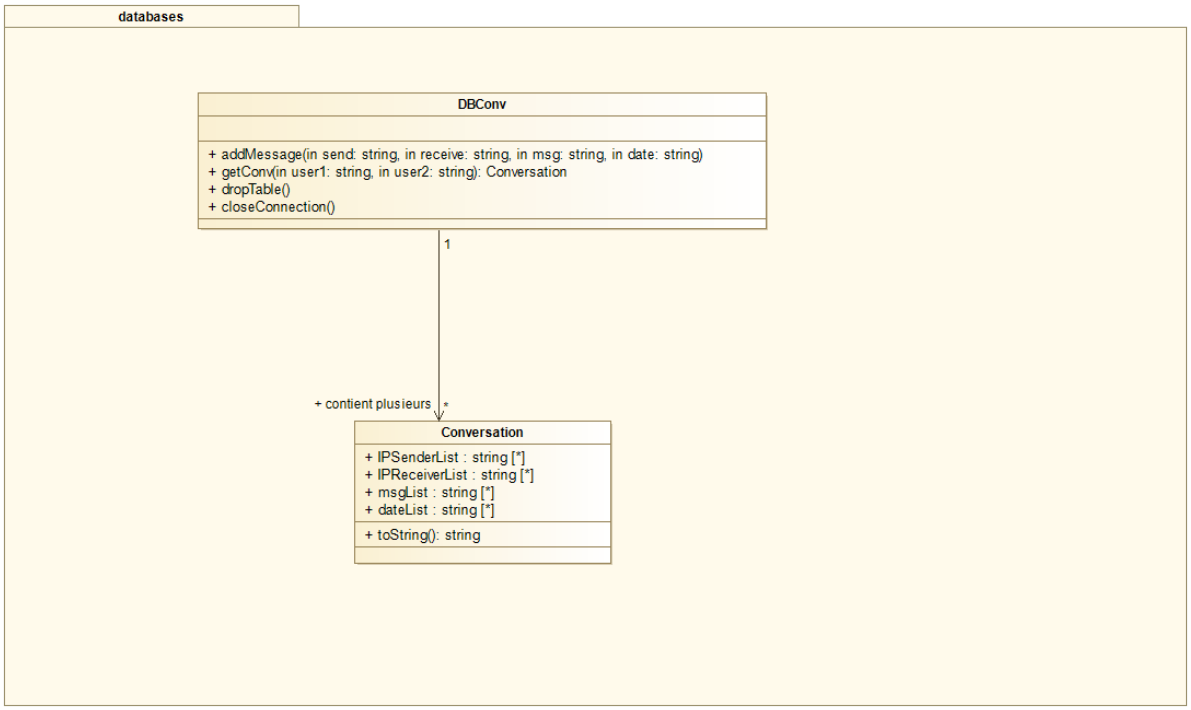


Diagrammes de classes:

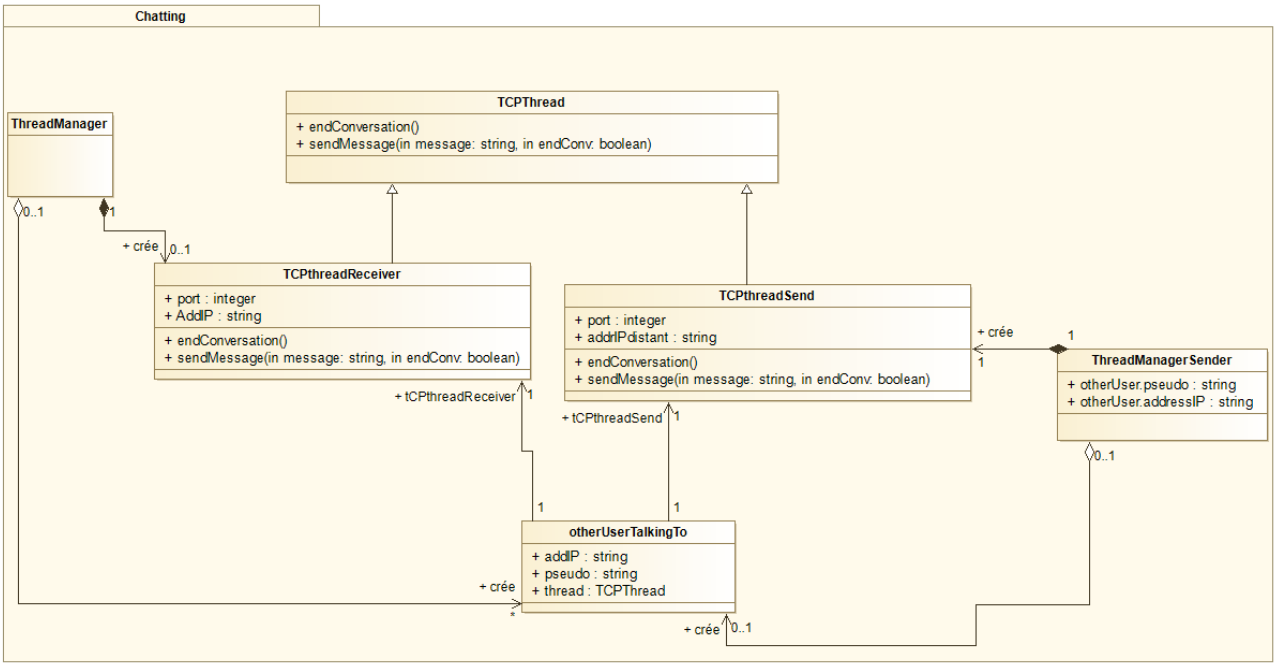


Package Connect

MURAT Félix
OMRE Erik



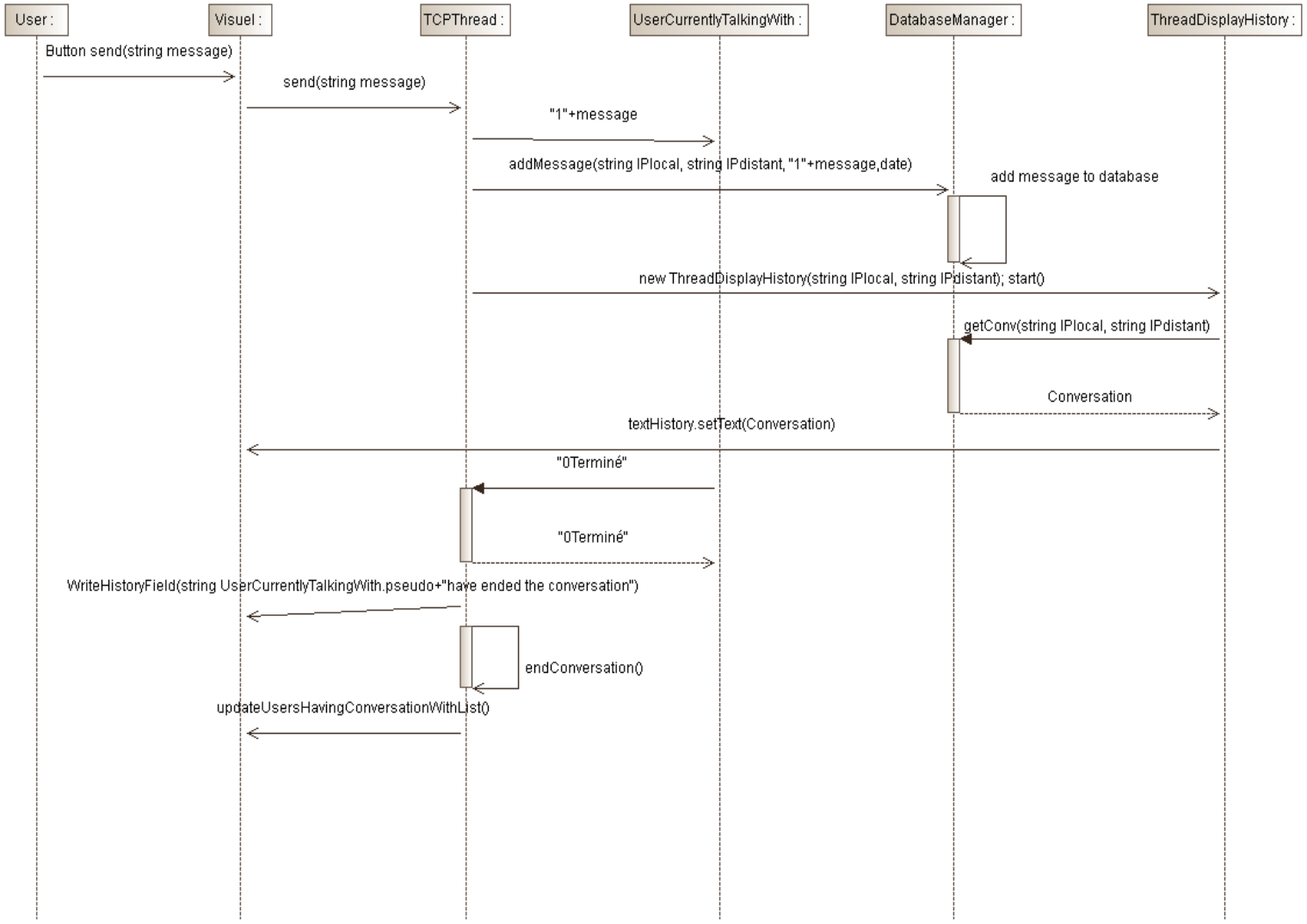
Package Databases



Package Chatting

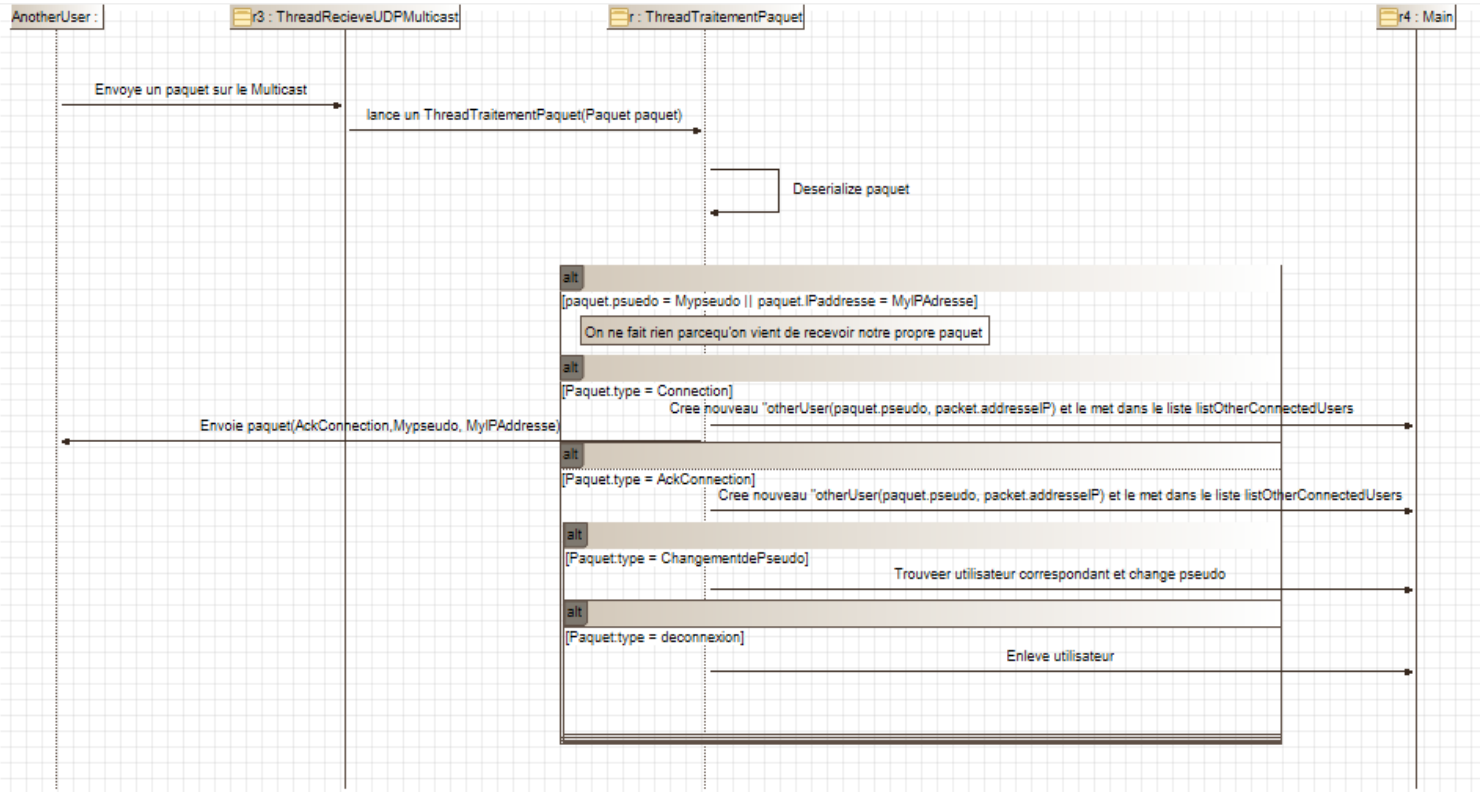
MURAT Félix
OMRE Erik

Diagrammes de séquence:

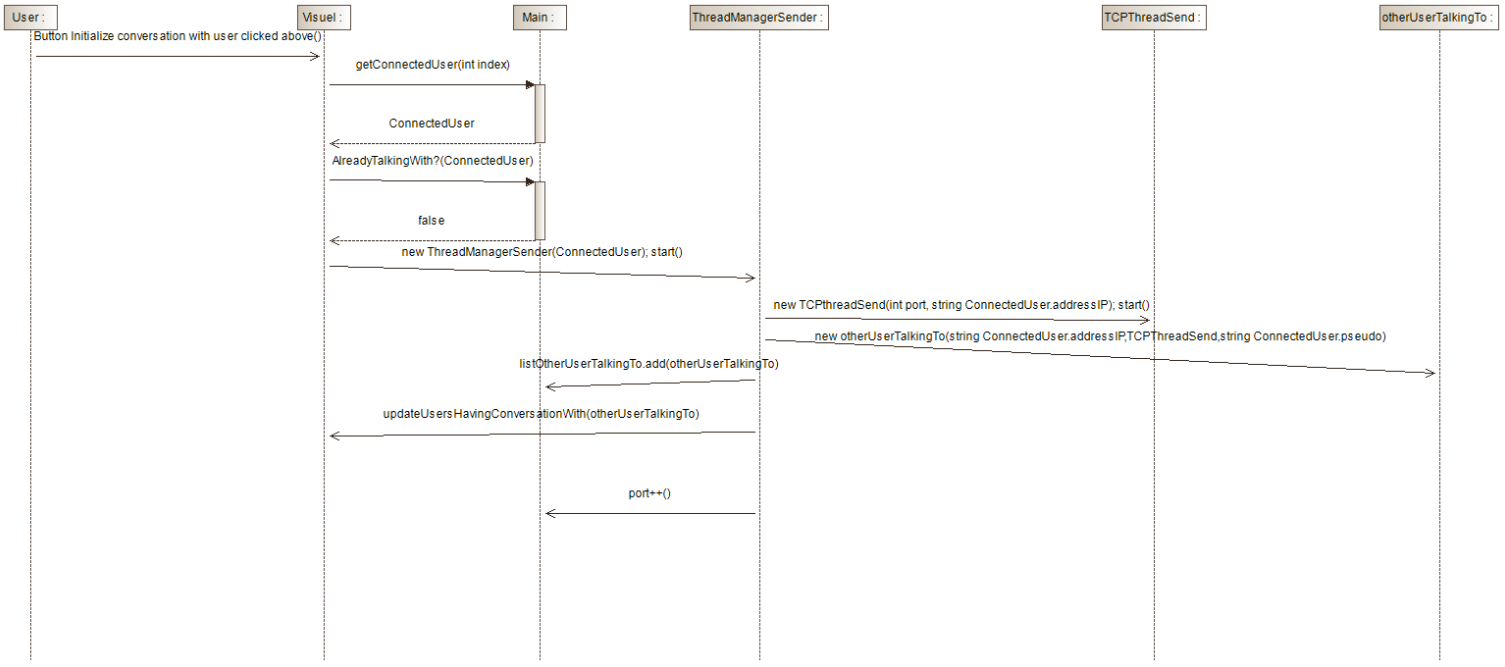


Envoie d'un message et réception d'une demande de fin de conversation

MURAT Félix
OMRE Erik



Réception et traitement d'un paquet UDP



Initialisation d'une conversation

III-Nos méthodes de travail:

Git

Pour le git, on a décidé de faire cinq packages: Connect (pour la partie multicast), Chatting (pour la partie communication TCP), Visuel (pour la partie interface), Database (pour la database), et Main (pour tous les variables global et la fonction main).

Jira

Pour ce qui concerne Jira, nous ne l'avons pas trop utilisé. Pendant les séances de TD de PDLA, nous avons bien défini ce qu'on pensait faire pendant les séances de TP, et nous avons commencé le premier sprint. Mais finalement, nous nous sommes rendu compte qu'il y avait une grande différence entre ce que l'on avait prévu et la réalité, de plus la division des tâches initiale pouvait se trouver non pertinente au fur et à mesure que nous découvrions la manière de les réaliser. Nous avons donc décidé de ne plus utiliser Jira qui ne nous servait pas. Nous avons préféré nous répartir les tâches au fur et à mesure de nos besoins, en même temps que nous découvrions les outils et méthodes nécessaires à la réalisation de notre projet.

Intégration continue

Ayant eu beaucoup de difficultés avec Jenkins, nous n'avons finalement pas utilisé ce dernier. Seul le créateur du dépôt Git (Erik) avait le droit de modifier le répertoire sur Github. Ainsi, à chaque modifications de Félix, ce dernier envoyait le code rajouté à Erik avec l'emplacement souhaité, afin qu'Erik puisse inscrire dans Github ces modifications sans que celles-ci entrent en collision avec son propre travail. Cette méthode ne nous a pas spécialement handicapé car nous communiquions régulièrement à propos de quels fichiers nous devions modifier.

Le partage de travaille

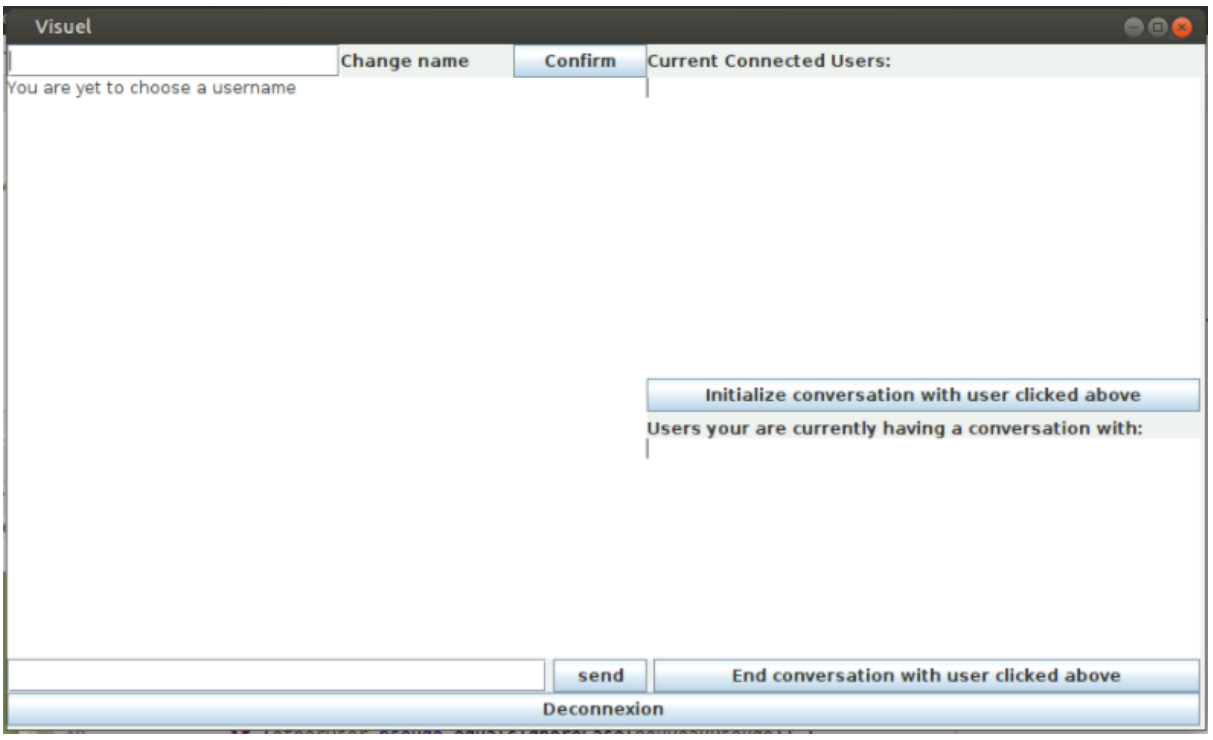
Erik a commencé à coder la partie "Connect" et "Chatting" pendant que Félix mettait en place l'interface. Finalement, Félix a programmé la partie "database" et l'a lié aux autres programmes afin de pouvoir gérer l'historique, pendant qu'Eric reliait l'interface avec les parties Connect et Chatting. Ensuite, nous avons travaillé ensemble pour tester notre programme .

Test de notre travail

Afin de tester nos programmes, nous avons créé de nombreuses classes destinées uniquement à tester nos différentes fonctionnalités. Ces tests ne sont pas forcément compréhensibles des utilisateurs mais nous ont permis de trouver les défauts de nos programmes. Nous avons également effectué de nombreux tests en situation réelle, en se connectant sur plusieurs machines d'une même salle afin de vérifier que la communication était correcte et que l'utilisation de l'interface était correcte et ne soulevait pas d'exceptions.

IV-Manuel d'utilisation

Lorsque vous lancez le programme, cette interface se lance :



Votre pseudo initial sera votre adresse IP . Pour le changer, écrivez le pseudo de votre choix dans le champ en haut à gauche et cliquez sur "Confirm".
Si d'autres utilisateurs lancent l'application, leurs pseudos vont apparaître dans le champ en dessous du texte "Current Connected Users". Pour initialiser une conversation avec un de vos collègues, cliquez sur son pseudo et cliquez sur le bouton "Initialize conversation with user clicked above". Vos anciens messages seront affichés, s'il en existe. Puis, pour envoyer un message, écrivez le dans le champ à côté de bouton "Send" et cliquez sur ce dernier pour envoyer.
Si vous lancez plusieurs conversations, ou que plusieurs utilisateurs distants lancent une conversation avec vous, les pseudos de toutes les personnes avec qui une conversation est en cours seront affichées dans le champ en dessous du texte "Users you are currently having a conversation with". Pour mettre fin à une conversation, cliquez sur le pseudo correspondant à la conversation que vous voulez terminer, puis cliquez sur le bouton "End conversation with user clicked above".
La croix rouge en haut à droite ne marche pas. Pour se déconnecter, il faut obligatoirement cliquer sur le bouton "Déconnexion".