Projet Java - Sockets

Connect !

M1 STRI 2015/2016

Second Rapport

Table des matières

Introduction :....3
I) Conception....4
a. Modélisation et mise à jour de la base de données....4
i. MCD....4
ii. MLD....4
b. Modélisation ULM....5
i. Diagramme de Classe....5
ii. Diagramme d'état....6
II) Serveur....7
a. Gestion du serveur....7
b. Codes de retour du serveur....7
c. Service Serveur Messagerie....8
III) Rôles du Client TCP....9
a. Gestion du client....9
b. Utilisation du client....9
c. Codes d'envoi du client vers le serveur....9
i. ECRIRE_MAIL|idDest, Message....10
ii. PASSER_EN_ECOUTE|0|20....11
iii. LIST_USER_ECOUTE :....11
iv. PARLER|idUser|num_port....11
v. Schéma pour la messagerie Instantanée....12
vi. RELEVER_MESSAGES|2....13
vii. LIRE_MESSAGE|2|idMail....14
IV) Conclusion....15

Introduction :

Dans le cadre de notre formation Système Télécommunication Réseaux Informatiques nous devons réaliser un projet java.
Le but est mettre en œuvre un annuaire partagé permettant aux utilisateurs du système de connaître ses coordonnées (électroniques et téléphoniques), son année de diplomation s'il y a lieu ainsi que ses compétences. On appellera « compétence », un thème qui peut se réduire à un mot clé ("Java" ;)) ou à un ensemble de mots ("routage ISIS" ;)). Ce système suppose que chaque étudiant soit équipé d'une application lourde JAVA.

Ce projet se fera en 3 versions. Et 3 rapports seront à rendre.

Pour cette version nous avons gardé la même architecture que pour la première version avec un seul serveur. Et la même interface (interface texte).

Version 2 : Messagerie Instantanée

Dans cette étape, un utilisateur doit pouvoir dialoguer en privé avec un autre utilisateur du système. Deux modes de messagerie sont à prévoir :

- Conversations en direct :

Un utilisateur peut ouvrir une fenêtre de dialogue pour discuter avec un autre contact. Ces conversations privées ne devront pas transiter par le serveur. On utilise donc une communication "peer to peer".

- Messagerie

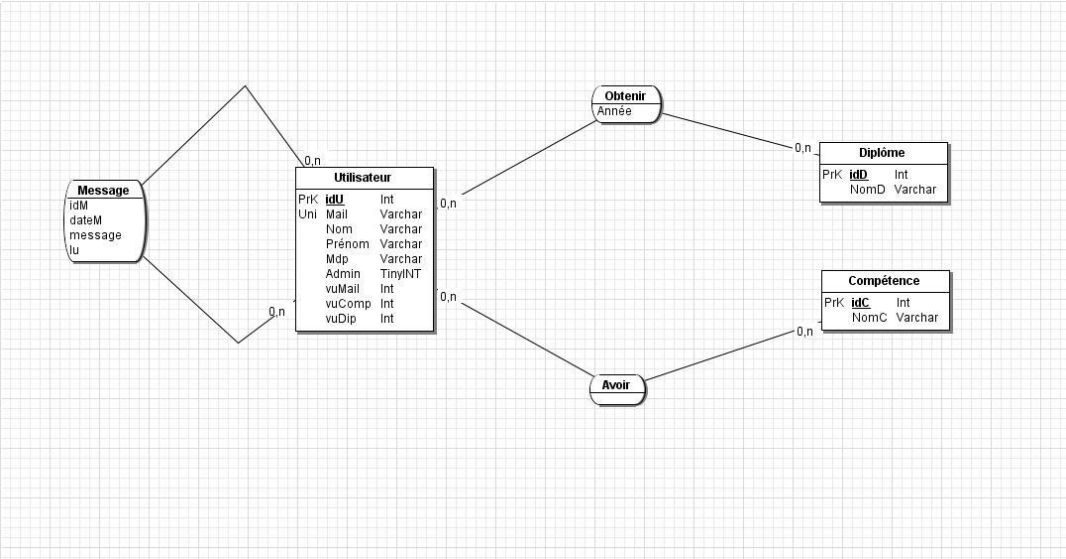
Dans ce cas de figure, un utilisateur peut laisser un message à l'attention d'un autre utilisateur. Cela peut être un cas de figure si un étudiant ne laisse pas accessible ses coordonnées. Ces conversations privées devront être hébergées sur une plateforme spécifique.

Cette première version sera à rendre le 2 mars 2016.

I) Conception :

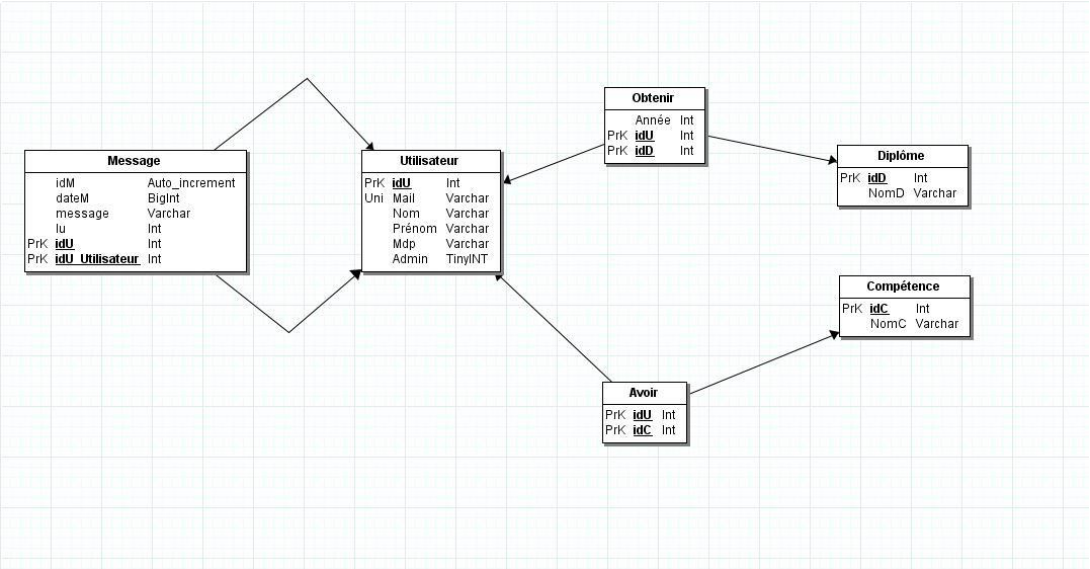
a. Modélisation et mise à jour de la base de données :

i. MCD :



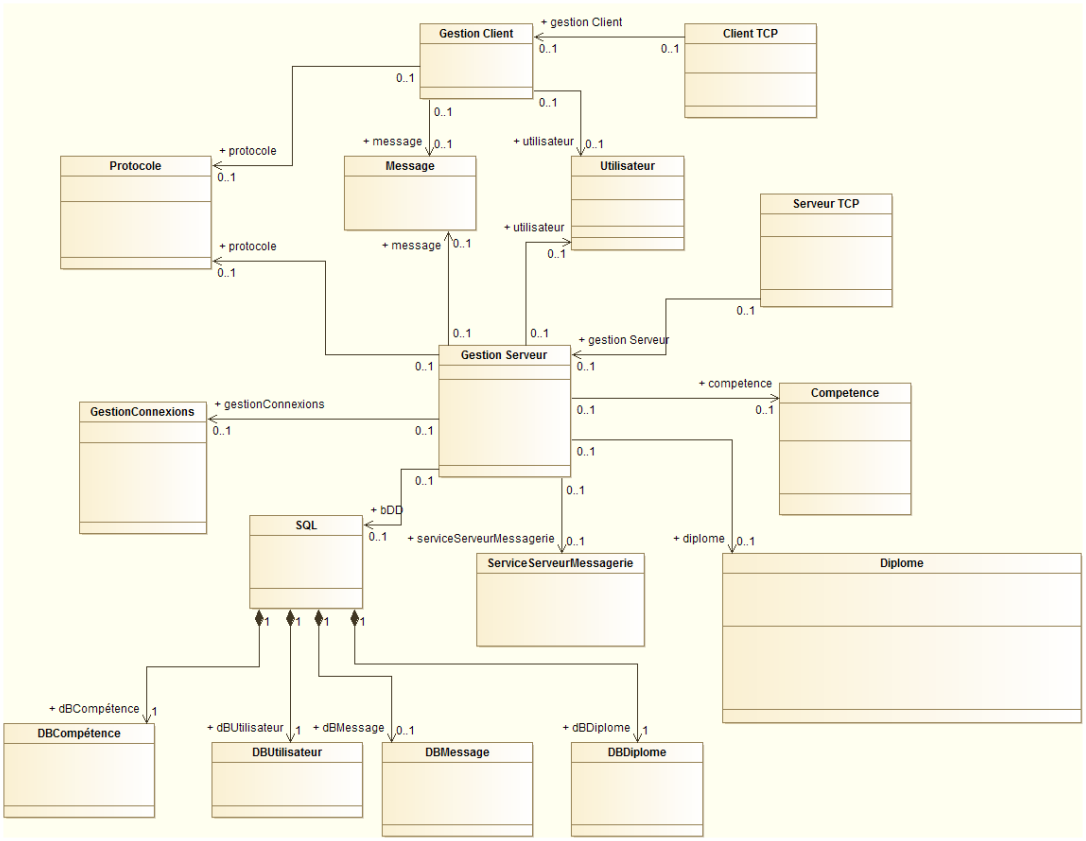
Ici pour la seconde version de la base de donnée nous avons rajouté une relation réflexive « message » qui a pour clé primaire idM et récupère les clés étrangère des Utilisateurs qui communiquent par mail (envoyeur et receveur). ça permet à un utilisateur de pouvoir échanger par messagerie différée avec un autre utilisateur.
Nous avons aussi rajouté 3 attribues pour la l'entité Utilisateurs.

ii. MLD :



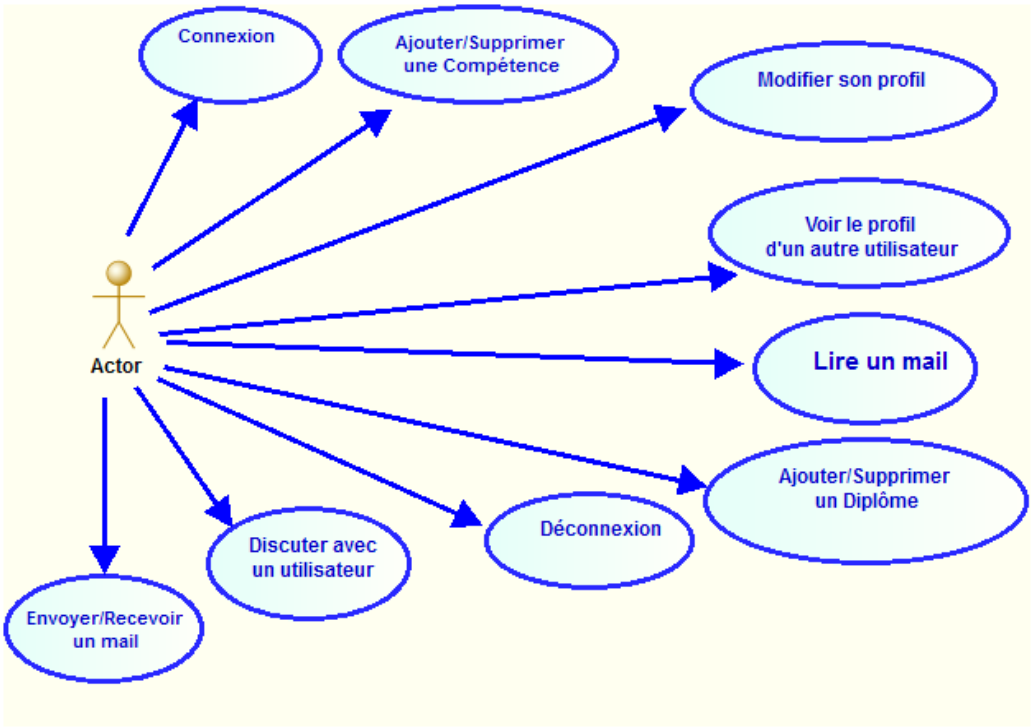
b. Modélisation ULM :

i. Mise à jour du Diagramme de Classe :



Dans la classe gestion client nous avons rajouté les différentes opérations pour que le client puisse gérer l'écrire la lecture et la réception de messages. Puisse voir aussi les utilisateurs connectés et parler ou se mettre en écoute. Nous avons fait de même côté serveur. Nous avons implémenté une classe message qui aura besoin d'une nouvelle classe DBMessage qui découle de la classe MySQL.
Nous avons aussi implémenté une nouvelle classe qui est un service qui gère la messagerie côté serveur.

ii. Diagramme d'état :



II) Serveur :

Pour cette seconde version nous avons choisi d'implémenter un nouveau serveur de messagerie. Le serveur pour la messagerie instantanée ne gère pas les transmissions de messages. Il dit à utilisateur si un tel ou un tel est en discussion ou pas. Si l'utilisateur est disponible alors on peut se connecter avec lui sur son port d'écoute. Ainsi le serveur notifiera aux autres utilisateurs que ce dernier n'est pas disponible.

a. Gestion du serveur :

Le serveur reçoit une requête d'un client pour écrire un mail à un autre utilisateur, lire un mail qu'il a reçu d'un autre utilisateur et relever ses messages. Le serveur reçoit le socket de l'utilisateur et va se connecter à la base de données utilisateur.

Le serveur reçoit une requête pour qu'un client puisse discuter avec un autre utilisateur il va regarder les utilisateurs qui sont en écoute et les lister.

b. Codes de retour du serveur :

Le serveur possède 2 requêtes qu'il envoi au client.
Les requêtes sont faites de la façon suivante :
- 200|OK :
signifie que la requête envoyer est bonne.
Le client lui de son côté ne verra que OK.
- 400|Erreur Protocole
signifie que le client s'est trompé de requêtes, que la syntaxe n'existe pas ou alors a fait une erreur de syntaxe ou que la requête DB s'est mal passé
- 200|ip|port :
signifie que la requête s'est bien passé et que le serveur renvoi à un utilisateur le port pour pouvoir discuter avec un autre utilisateur.

C. Service Serveur Messagerie :

Cette nouvelle classe récupère un socket limité à une instanciation pour ne pas avoir plusieurs utilisateurs connectés sur un même utilisateur et le livre à un client qui sera en écoute prêt à être demander en conversation directe.
Un utilisateur a son socket d'écoute fermé lorsqu'il en train de parler avec un autre utilisateur. Ainsi lorsqu'on demande la liste des utilisateurs connectés ceux qui sont en train de parler ont leur socket d'écoute fermé et donc le serveur ne notifie pas aux autres utilisateurs leur socket.
C'est une version mis à jour du ServiceServeur de la première qui livre le socket au client.

III) Rôles du Client TCP :

a. Gestion du client :

La gestion du client va prendre en compte les nouveaux protocoles mis en place pour la messagerie instantanée et différée.
Dans la messagerie différée, le client va écrire et envoyer un mail à un autre utilisateur. Pour ça il lui faudra seulement l'id de l'utilisateur et le corps du message. Cette requête sera envoyée au serveur. Qui mettra à jour la base de données.
Un identifiant mail sera créer ce qui permettra à l'utilisateur qui a reçu le message de retrouver le mail et le lire.
Pour la messagerie instantanée, le client peut voir qui est connecté. Lorsque le client est connecté avec un autre utilisateur, les autres utilisateurs ne peuvent entrer en communication directe avec eux. Les sockets de ces clients passent en écoute et ils peuvent échanger. Pour quitter la conversation les utilisateurs devront faire « q » (quit), les ports des utilisateurs sont alors supprimés.

b. Utilisation du client :

Le client une fois connecter avec un autre utilisateur en ouvrant son socket d'écoute il sera en constante discussion tant qu'il n'a pas fait « q » pour quitter la conversation. Il pourra envoyer et recevoir les messages avec l'utilisateur qui est connecté avec lui.

c. Codes d'envoi du client vers le serveur :

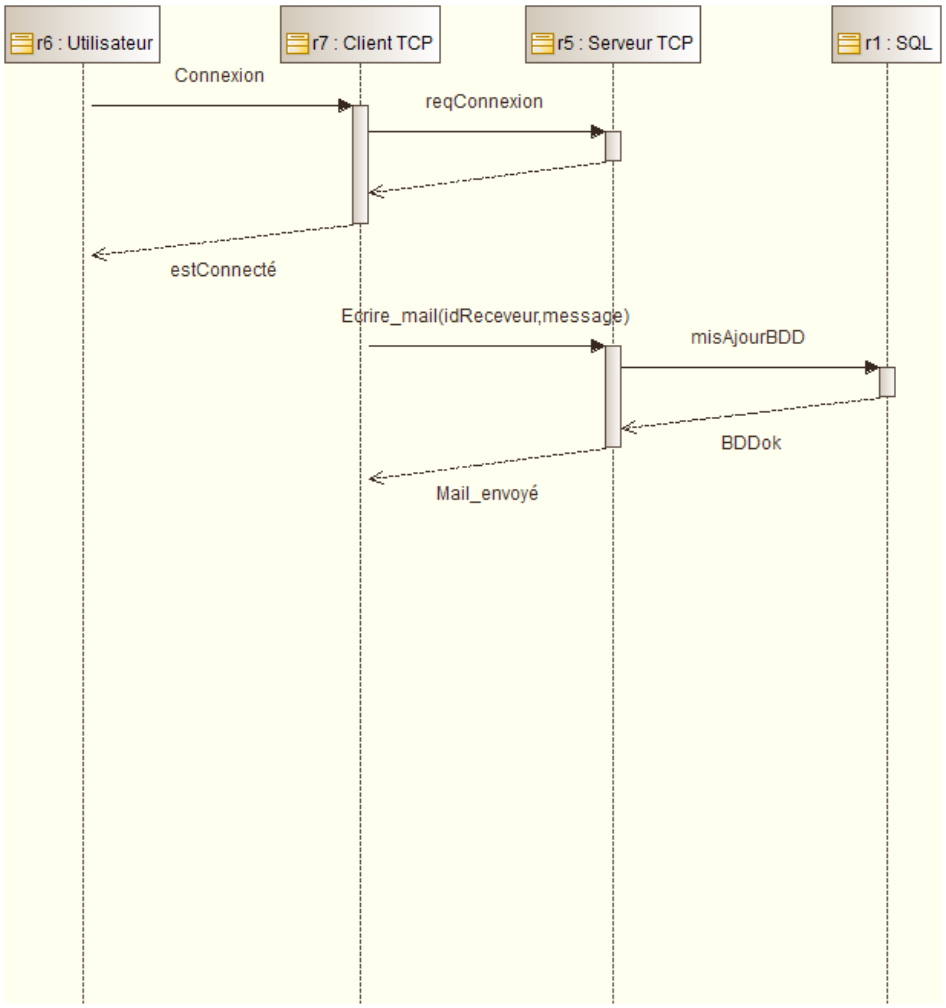
Le client possède des requêtes qu'il enverra vers le serveur TCP pour se connecter avec un autre utilisateur ou pour lui envoyer un mail.
Toutes les requêtes seront disposées de la façon suivante :
COMMANDE | LISTE_PARAM

Il y a donc minimum 2 paramètres le premier étant la commande que va effectuer le client et la seconde est le paramètre dont il a besoin pour effectuer la requête. Chaque paramètre est séparé par un pipe. Si un de ces paramètres est incorrect ou n'existe pas le serveur renverra alors un message d'erreur. Voici la liste des commandes utilisé par le client avec ses paramètres :

Pour connaître la liste des commande on tape : « ? ».

i. ECRIRE_MAIL|idDest, Message :

Le serveur va recevoir cette requête pour qu'un utilisateur puisse envoyer un mail à un autre utilisateur. Le serveur mettra à jour la base de données pour que l'utilisateur qui a reçu le mail puisse le voir et le lire.



ii. PASSER_EN_ECOUTE|0|20 :

Cette requête permet à un utilisateur de notifier au serveur qu'il a activé son socket d'écoute.

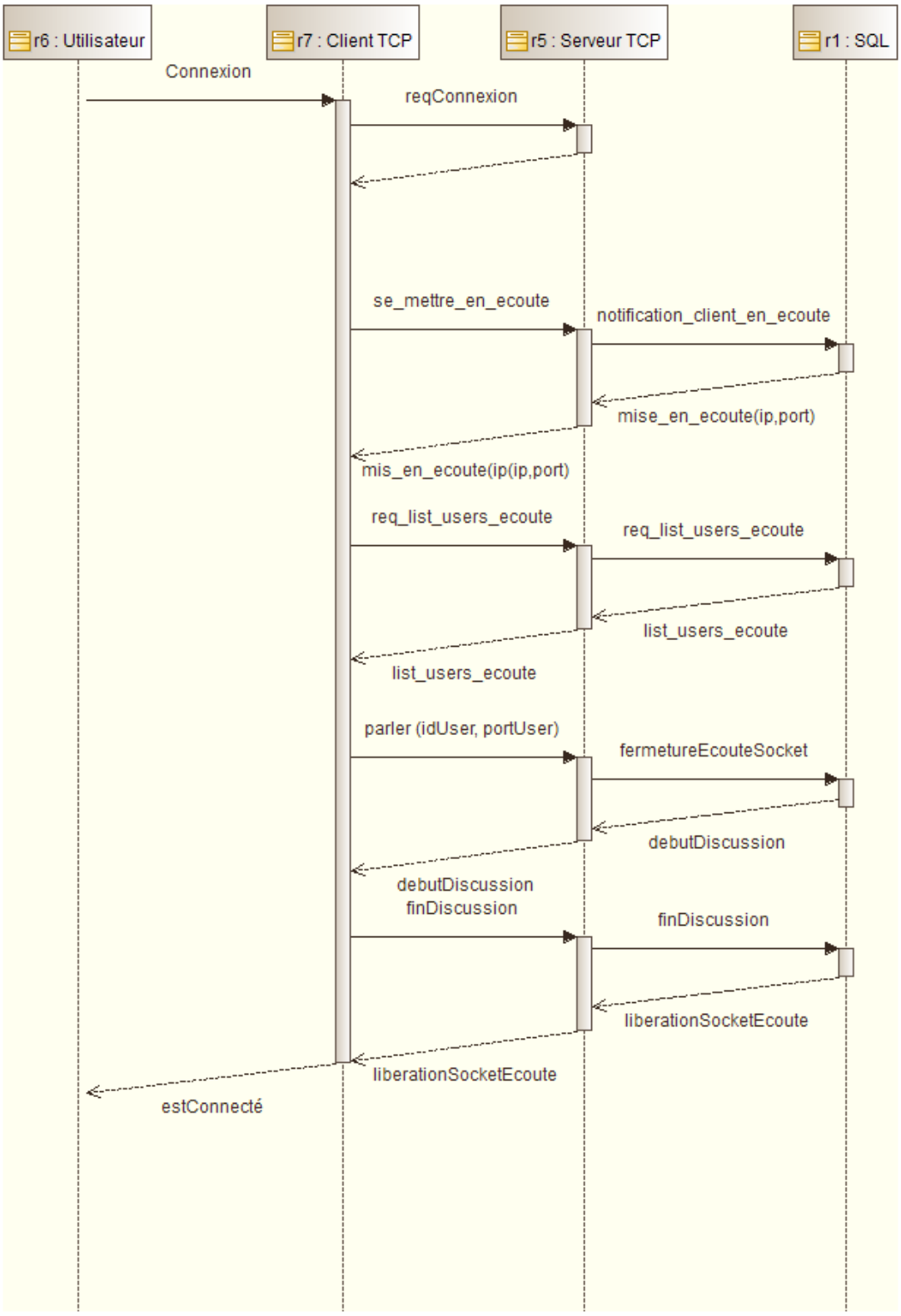
iii. LIST_USER_ECOUTE :

Cette requête envoyée au serveur permet de savoir quels sont les utilisateurs connectés qui sont en écoute

iv. PARLER|idUser|num_port :

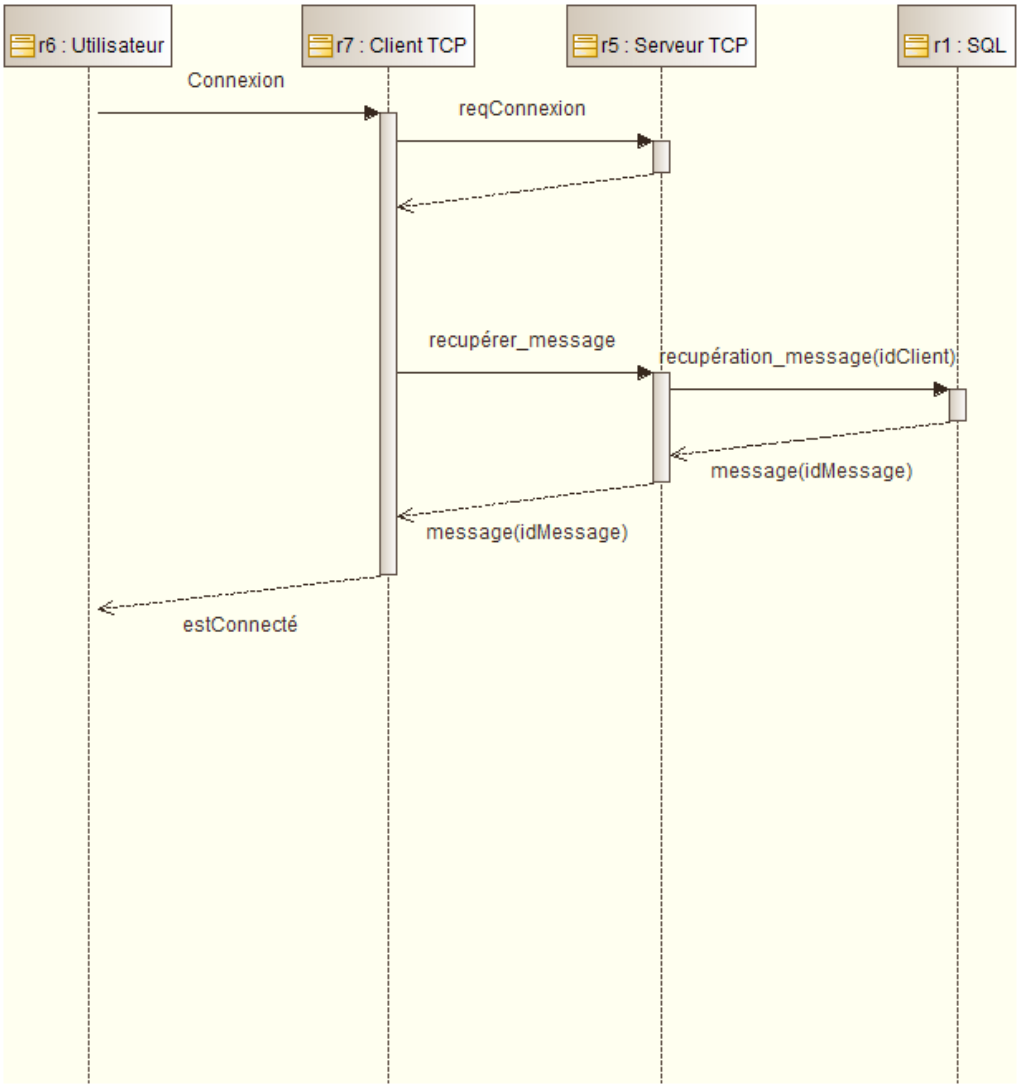
Une fois qu'un utilisateur à notifier à l'utilisateur qu'il est en écoute et qu'il a eu la liste des users en écoute alors l'utilisateur enverra cette requête et sera en discussion avec l'utilisateur choisi.

V. Schéma pour la messagerie Instantanée :



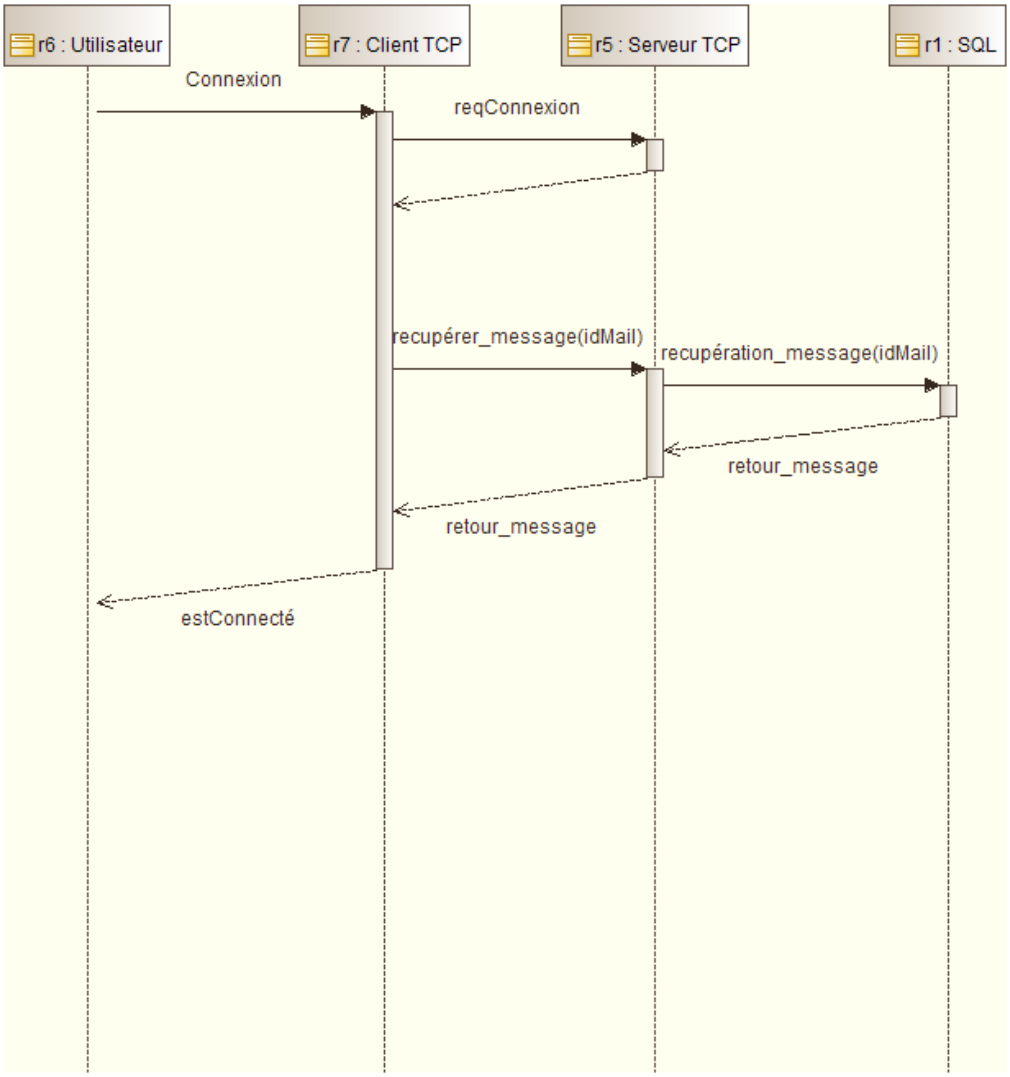
vi. RELEVER_MESSAGES|2 :

Cette requête envoyée au serveur permet de voir les mails qu'un utilisateur a pu recevoir.



vii. LIRE_MESSAGE|2|idMail :

Cette requête envoyer au serveur permet de lire les mails que l'utilisateur a reçu. Le client envoi une requête au serveur avec l'identifiant du mail. Le serveur va donc relever l'identifiant du mail et envoyer le mail en question que l'utilisateur a reçu.



IV) Conclusion :

Pour cette seconde version nous avons continué à nous partager le travail comme pour la première version. Nous sommes mis d'accord sur la conception de la version 2. Nous avons cette fois-ci tenu le cahier des charges sans rajouté d'éléments en plus, nous avons donc pu bien prendre en compte les besoins de cette version, ce qui nous a permis de pas être en retard sur l'avancement de la version.