Projet Java - Sockets

Connect !

M1 STRI 2015/2016

Table des matières

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.

Version 1 : Base

Ajout de contacts :

Chaque étudiant peut créer son compte et gérer ses informations attachées. Seules ses informations sont modifiables. On souhaite aussi permettre aux étudiants qui ont créé un compte de limiter la visibilité de toute ou partie de leurs informations pour toute ou partie des utilisateurs (admin, utilisateur, anonyme).

Utilisation courante :

Tous les utilisateurs du système peuvent récupérer la liste de tous les étudiants et interroger le système pour connaître les détails concernant tel ou tel étudiants.

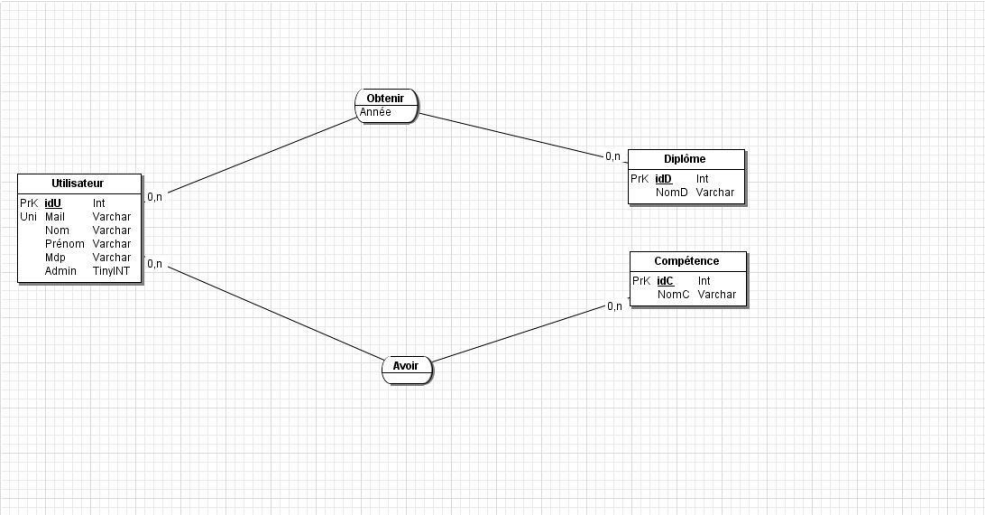
Notre application sera orienté client/serveur. Le client et le serveur auront la connaissance de protocoles d'émission et réception. Le serveur sera connecté à la base de données MySQL où la base sera actualisée à chaque requête du client vers le serveur. Notre application sera faite sous Eclipse et les diagrammes UML seront faits sur Modélio. Nous avons choisi de ne pas développer une interface graphique, l'application sera utilisé en mode commande sur un terminal.

Cette première version sera à rendre le 10 février 2016.

Conception :

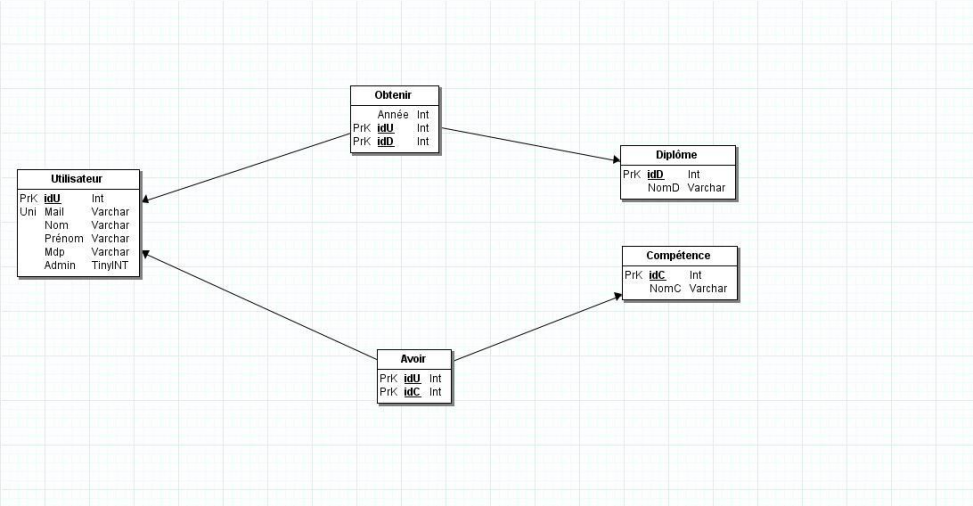
a. Modélisation de la base de données :

i. MCD :



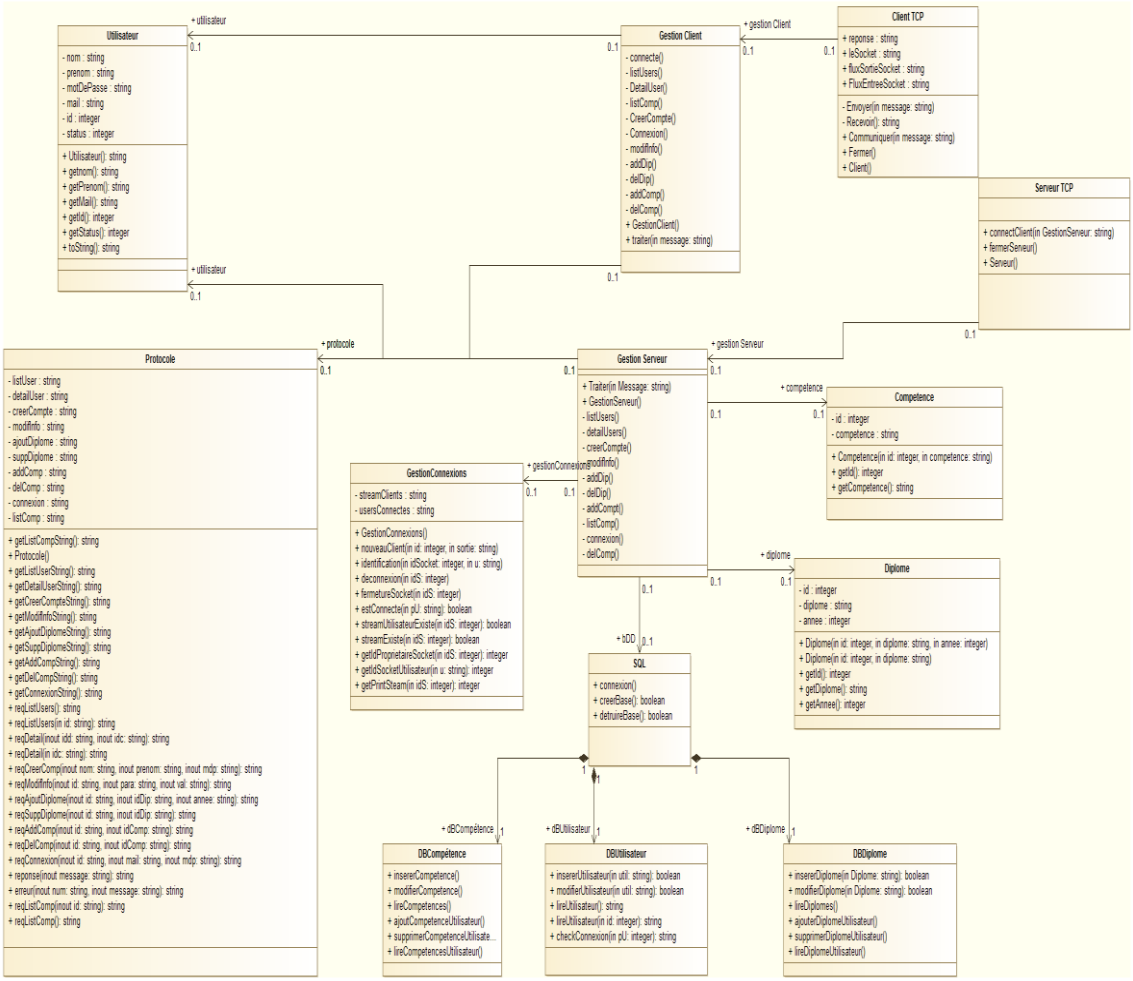
Ici pour la première version nous n'avons besoin dans notre MCD que 3 entités (Utilisateur, Diplôme et Compétences).

ii. MLD :

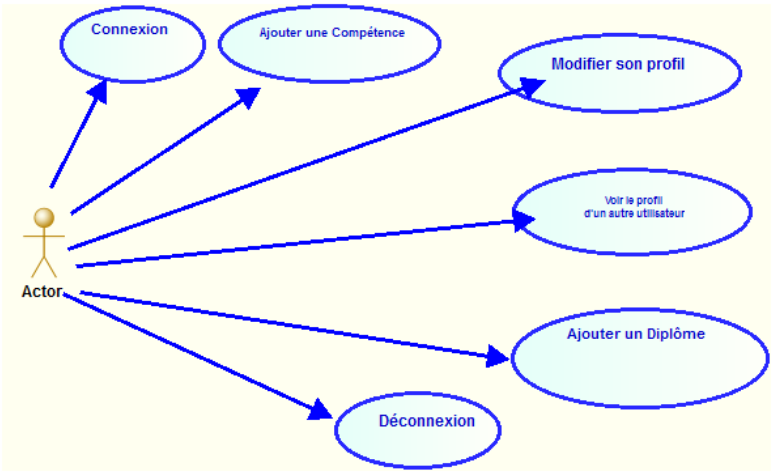


b. Modélisation ULM :

i. Diagramme de Classe :



ii. Diagramme d'état :



II) Rôles du serveur TCP :

Nous avons choisi un seul serveur qui gèrera à la fois les connections et les requêtes des clients.
Nous avons aussi utilisé une base de données pour avoir une gestion des données utilisateurs.
Nous avons choisi d'utiliser TCP car nous avions déjà travaillé dessus sur les TPs précédents et donc savons travailler sur TCP contrairement à UDP que nous n'avons pas pu trop manipuler. Le choix de TCP par rapport s'est par facilité pour nous. Peut-être que UDP aurait été mieux adapté.
Nous avons choisi d'utiliser des Thread pour faire du multi-client.
Nous avons choisi d'utiliser des sockets pour pouvoir communiquer entre les clients et le serveur à l'aide d'une adresse IP et d'un port, ces sockets permettront de gérer des flux entrant et sortant afin d'assurer une communication entre les deux (le client et le serveur), soit de manière fiable à l'aide du protocole TCP/IP.

a. Gestion du serveur :

Le serveur TCP interagie directement avec le client par des messages de retour lorsque le client envoi une requête.
La gestion du serveur exécutera tous les messages que le client va envoyer et les analysera et à renvoyer un code de retour en fonction du message envoyé.
Le serveur va d'abord analyser la requête que lui demande le client, s'il n'y en a pas il renvoi un message d'erreur.
Le serveur va mettre à jour la base de donnée lorsqu'il reçoit une nouvelle requête utilisateur. Ou alors il ira chercher l'information.

b. Utilisation du serveur :

Le serveur TCP a trois phases une partie où il va se connecter et créer un socket. Le serveur sera alors en attente de connexion avec un socket d'écoute. Ensuite une autre où il va recevoir les connexions de clients en utilisant sa partie de gestion pour analyser les requêtes entrantes avec un socket de service jusqu'à qu'il soit fermé.

c. 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.
- 400|Erreur Procotole
signifie que le client s'est trompé de requêtes, que la syntaxe n'existe pas, que le client a fait une erreur de syntaxe, que le client est déjà connecté ailleurs ou alors qu'il n'y a pas le bon nombre de paramètres.
Le serveur ira d'abord se connecter à l'entité en question sur la base de donnée et entrera la commande MySQL en question pour savoir si l'utilisateur s'est trompé ou pas. Ainsi la base de donnée lui retournera le type d'erreur en fonction de l'entité où le serveur à fait la requête.

III) Rôles du Client TCP :

a. Gestion du client :

Le client va se connecter au serveur sur la machine sur le port 12345. Il y a création d'un socket, création d'un flux d'entrée et flux de sorti où les messages, les requêtes et les réponses seront envoyés/reçus.

b. Utilisation du client :

Le classe client TCP va servir de la classe GestionClientTCP et utilisateur pour pouvoir envoyer des requêtes. Elle aura besoin de la classe protocole pour connaitre toutes les requêtes qu'il peut envoyer au serveur.

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

Le client possède des requêtes qu'il enverra vers le serveur TCP. 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 paramètre 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 :
- LISTE_USER :
Permet d'obtenir la liste des utilisateurs présent. Pour cette commande on aura besoin comme paramètre pour cette commande de l'ID utilisateur.
- DETAIL_USER :
Permet d'obtenir le détail d'un utilisateur. Nous avons donc besoin pour cette commande d'avoir 2 paramètres, l'identifiant du client qui demande et l'identifiant du client cible.
- CREER_COMPTE :
Cette commande est faite pour créer un nouveau compte utilisateur. Cette commande a besoin de tous les paramètres obligatoire qui ne peuvent pas être nuls pour que le serveur puisse les insérer et mettre à jour la base de données. Les paramètres obligatoires sont l'id de l'utilisateur, son nom, son prénom, le mot de passe et le mail.

- MODIF_INFO : Cette commande permet à l'utilisateur de modifier ses informations. La requête envoyée n'est valable que pour un paramètre. 1 requête envoyé = 1 paramètre modifié. Il faut donc le nom du paramètre qu'on veut modifier et la nouvelle valeur que l'utilisateur souhaite mettre. Le serveur mettra alors à jour la base de données.
- AJOUT_DIPLOME : Permet à l'utilisateur d'ajouter un diplôme qu'il a obtenu en spécifiant l'année de l'obtention du diplôme. Il y aura donc 2 paramètres. Le serveur mettra alors à jour la base de données.
-SUPP_DIPLOME : Permet à l'utilisateur de supprimer un diplôme. Il n'y a qu'un seul paramètre qui est l'identifiant du diplôme. Le serveur mettra alors à jour la base de données.
- AJOUT_COMP : Permet à l'utilisateur d'ajouter une ou plusieurs compétences il y a donc minimum un paramètres. Chaque une compétence = un paramètre. Le serveur mettra alors à jour la base de données.
- DEL_COMP : Permet à l'utilisateur de supprimer une compétence. Cette requête contient donc 1 paramètre. Avec l'identifiant de la compétence. Le serveur mettra alors à jour la base de données.
- CONNEXION : L'utilisateur a besoin de 2 paramètre pour cette requête que sont son identifiant et son mot de passe.
- LIST_COMP : Permet à l'utilisateur d'afficher la liste des compétences disponibles.
- LIST_DIP : Permet à l'utilisateur d'afficher la liste des diplômes disponible.

IV) Bases de Données :

a. Connection à la base de données

La classe MySQL se connectera à la base de données en utilisant comme paramètres l'url de la base, l'Username et le mot de passe.
Une fois qu'on est connecté à la base on va envoyer les requêtes pour créer la base et donc chaque table.
Il y aura aussi des requêtes pour créer la base.
Chaque classe aura besoin de la classe MySQL pour se connecter à la base et pouvoir mettre à jour des informations concernant des utilisateurs.

b. Connexion à l'entité diplôme

Pour se connecter à l'entité diplôme, on se connectera avec MySQL. On aura besoin des classes diplôme et utilisateur pour mettre à jour la base de donnée sur chaque action qui concerne l'entité diplôme. Il y a deux méthodes pour la classe diplôme la première c'est quand l'utilisateur donne une année d'obtention de diplôme et la seconde quand il ne la donne pas dans ce cas-là on mettra 0 pour l'année, c'est-à-dire vide.
Les résultats seront renvoyés dans la classe compétence.

c. Connexion à l'entité compétence

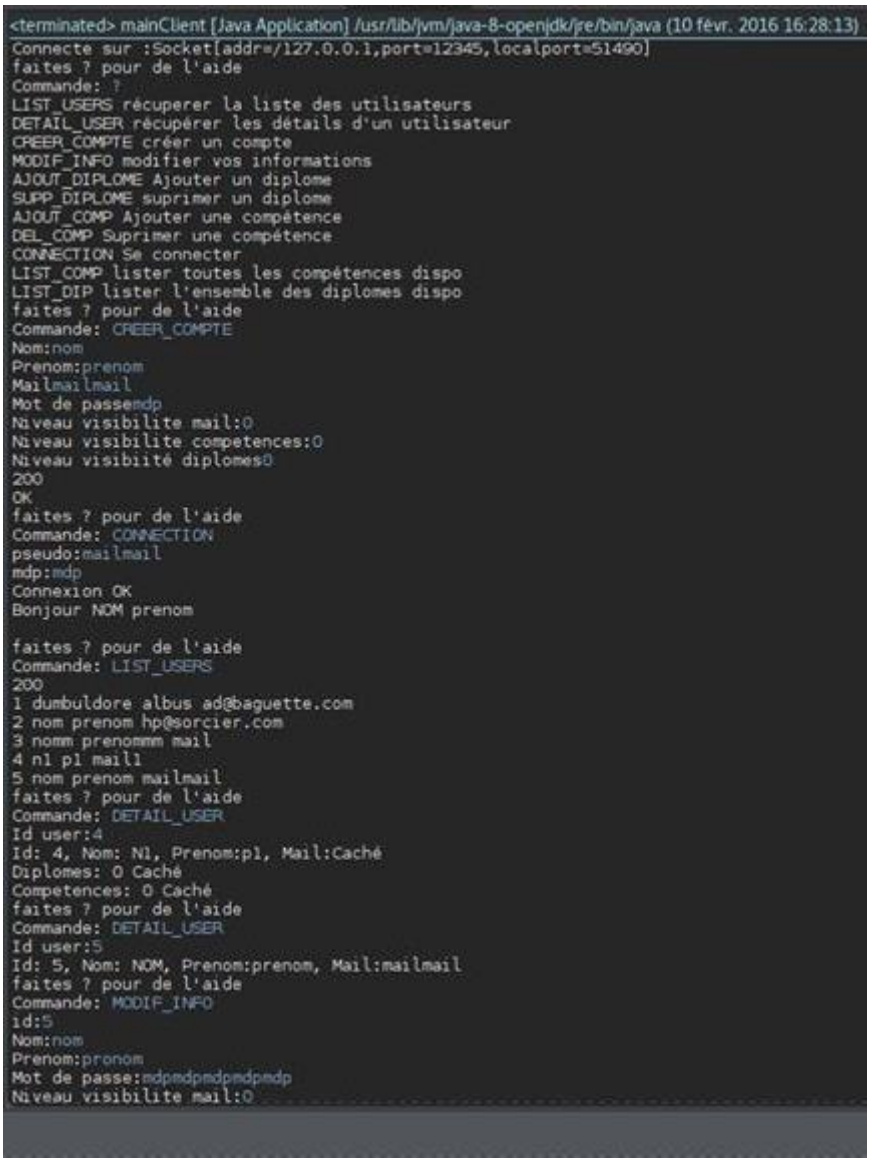
Pour se connecter à l'entité compétence on va avoir besoin des classes utilisateur, compétence et diplôme. À chaque action qu'on fera et qui concerne l'entité compétence on se connectera en MySQL à la base. On renverra les résultats dans la classe compétence.

d. Connexion à l'entité utilisateur

Dans un premier temps la classe de DBUtilisateur va se connecter à la base de données puis elle mettre à jour les informations concernant l'utilisateur. À chaque opération la classe va se connecter à la base et faire des opérations de lecture et écriture en fonction de la requête transmise.

V) Actions Possibles de l'application

Les actions possibles de l'application reprennent toutes les requêtes de la classe protocole.



Conclusion :

La première partie du projet Java nous a permis de travailler en équipe, d'utiliser nos connaissances et de se répartir les tâches. Ce projet est dans la continuité de ce que nous faisions en TP.