Berquez Léa
Dedieu Rémi
TDA2

Projet S5 : Interface de gestion des ressources universitaires

Tuteur : Mathieu RAYNAL (mathieu.raynal@irit.fr)

Berquez Léa : lea.berquez@univ-tlse3.fr

Dedieu Rémi : remi.dedieu@univ.tlse3.fr

Table des matières

I. Introduction

“Ce projet a pour but de mettre en relation les étudiants et enseignants (appelés dans la suite du projet « utilisateurs du campus ») avec les différents services administratifs ou techniques de l’université. L’objectif est de pouvoir faire remonter les problèmes relevés sur le campus et d’en suivre l’évolution.”

Nous allons donc créer une application en java permettant d’envoyer des messages d’un service à un autre.
L’interface utilisateur aura pour but de permettre l’envoi de messages à un autre groupe. Une personne pourra choisir de créer un ticket, une discussion en sélectionnant un groupe destinataire, en écrivant un titre et en envoyant un premier message. Tous les membres des groupes (destinataire(s) et émetteur) pourront participer à cette conversation.
L’interface serveur permettra de récupérer les données de la base de données SQL et de modifier cette dernière : créer un nouvel utilisateur, groupe, etc…

Nous allons utiliser plusieurs classes et différentes ressources. Deux principales : l’utilisateur et le serveur.

La base de données utilisée sera au format SQL.

Pour réaliser la liaison serveur-client, nous allons utiliser les éléments fournis au préalable dans la section Projet de Moodle et sur les sockets. Nous allons utiliser des notions de json pour envoyer les données et les messages.

Nous vous présenterons tous ces éléments dans la suite de ce rapport.

II. Le protocole de communication permettant de transmettre les différentes informations entre l'interface utilisateur et le serveur

On va utiliser des sockets. Cela va nous permettre de faire la communication entre le serveur et le client. Ce sera une communication connectée à double sens.

Les différentes communications seront :

- La connexion : lorsque le client tente de se connecter il envoie une requête au serveur pour que ce dernier renvoie toutes les données de cet utilisateur.
  Utilisateur : sendMessage(message); puis enverra de la même manière l'identifiant et le mot de passe sous forme de json.
  Serveur : renverra les données du client si cela a été un succès, sinon il notifiera l'échec.
- Pour envoyer un message : l'interface utilisateur envoie une requête au serveur lui indiquant le ticket choisi et le message à envoyer. Si l'envoi s'est correctement réalisé, le serveur renvoie un succès sinon il notifie l'échec.
  Utilisateur : sendMessage(message); puis enverra les éléments nécessaire pour créer le message (contenu du message, …) sous forme json.
  Serveur : créer le nouveau message, envoie ce dernier à l'utilisateur et à tous les autres utilisateurs connectés, renvoie aussi si la création c'est bien passé.
- Pour créer un nouveau ticket : l'interface utilisateur envoie une requête au serveur lui indiquant le titre du ticket, le groupe choisi, et le message. Si la création s'est correctement effectuée, le serveur renvoie un succès sinon il notifie l'échec.
  Utilisateur : sendMessage(message); puis enverra les éléments nécessaire pour créer le ticket (titre, groupe émétteur, …) sous forme json.
  Serveur : créer le nouveau ticket, envoie ce dernier à l'utilisateur et à tous les autres utilisateurs connectés. Renvoie aussi si la création s'est bien passé.

Toutes les données seront envoyées sous forme de json que l'on traitera à partir de fonctions spécifiques.

Le « message » sera sous la forme :

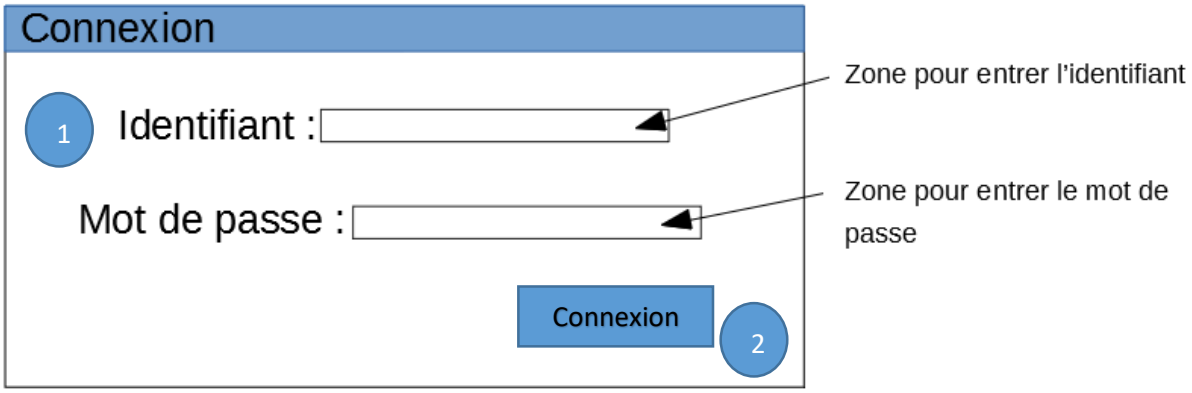
{
    « action » : « nom de l'action (Demande de connexion, Demande envoie message, Demande creation ticket) »,
    « donnees » : { donnee correspondantes sous forme json }
}

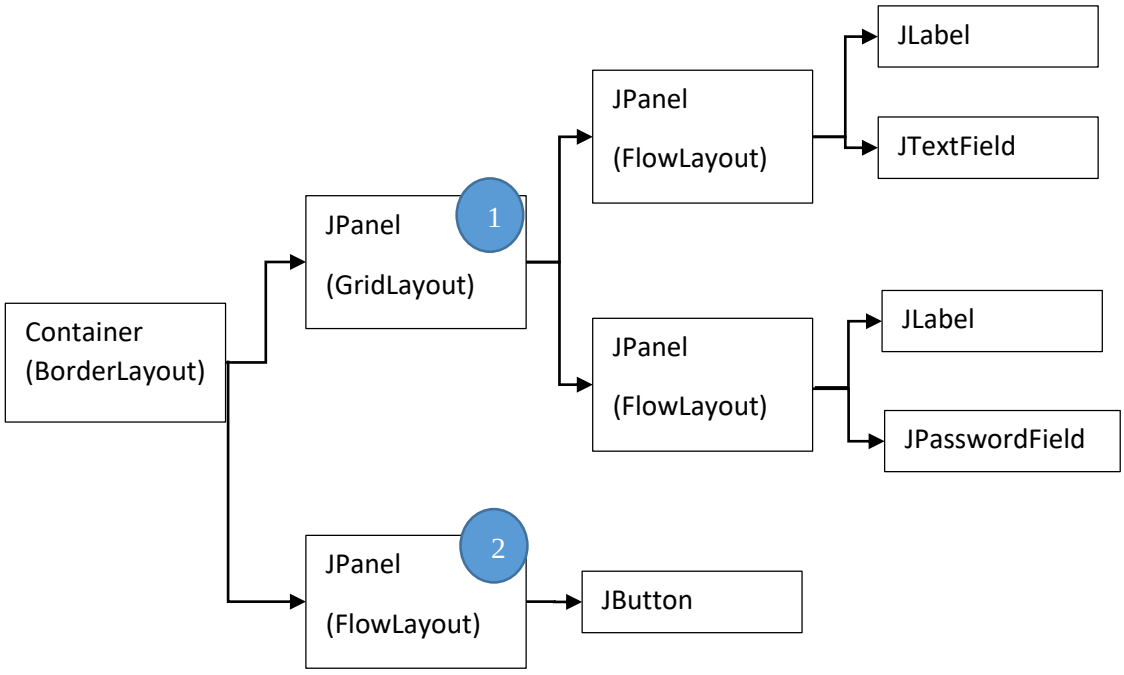
III. Les différentes interfaces constituant à la fois la partie utilisateur et la partie serveur :

1) Interface côté utilisateur

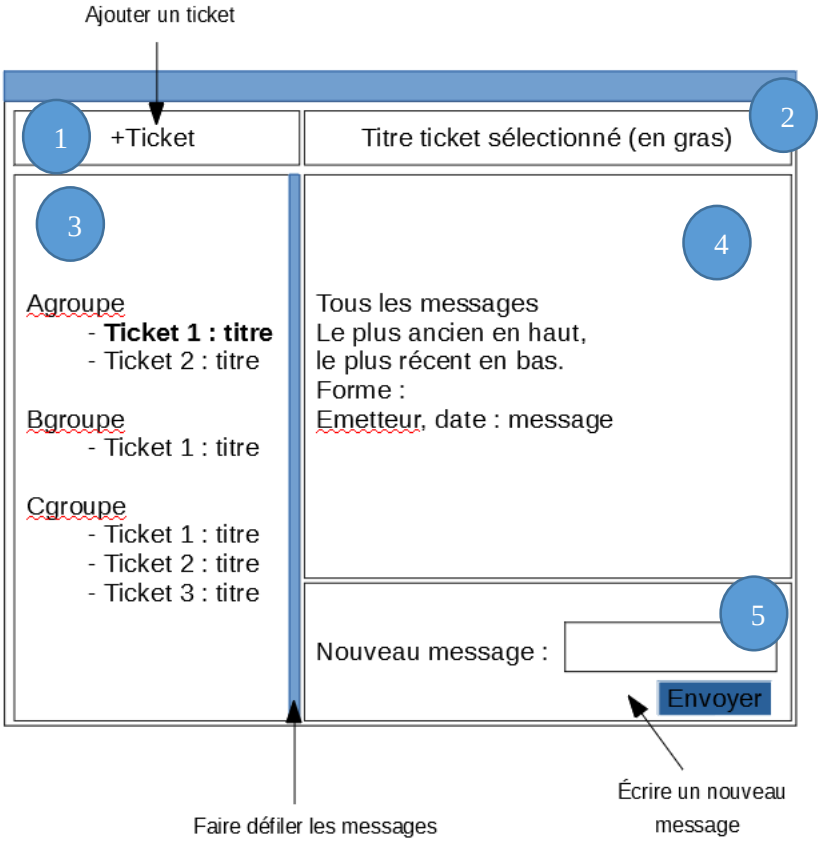
Il faut une fenêtre principale avec une JFrame.

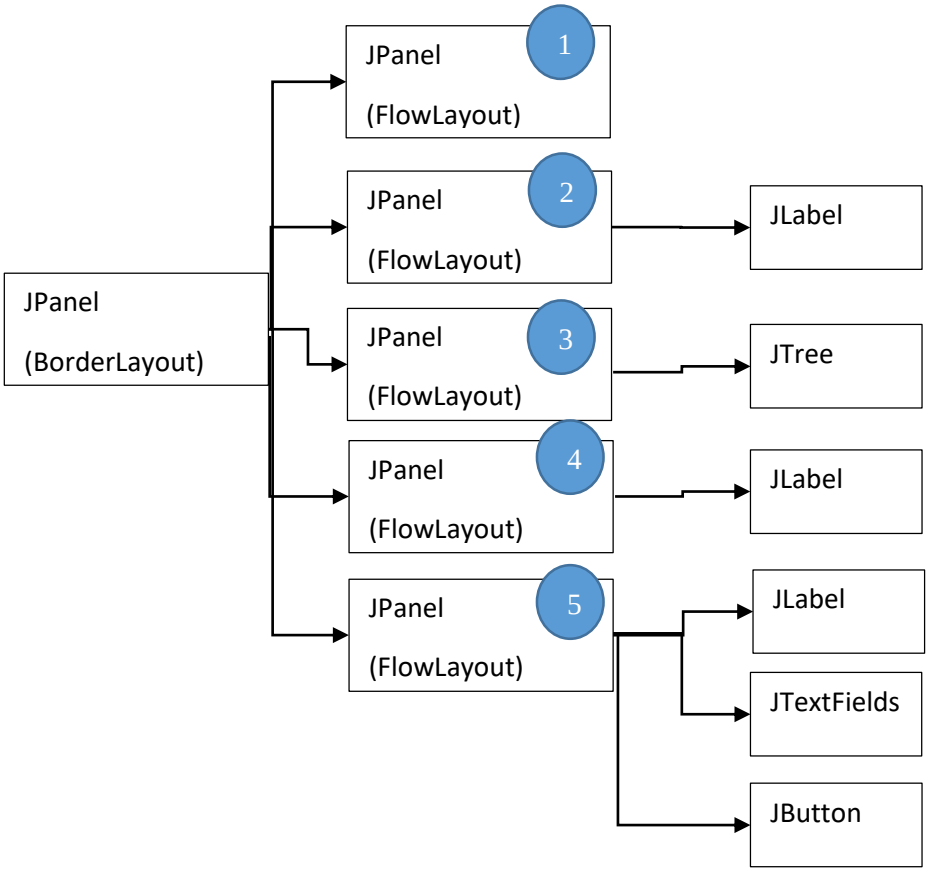
- **Interface de connexion :** interface qui permet la connexion de l'utilisateur. Le mot de passe ne doit pas apparaître lorsqu'on le tape. Elle restera ouverte si l'identifiant et/ou le mot de passe ne sont pas corrects (n'existent pas, ne sont pas reliés, …).



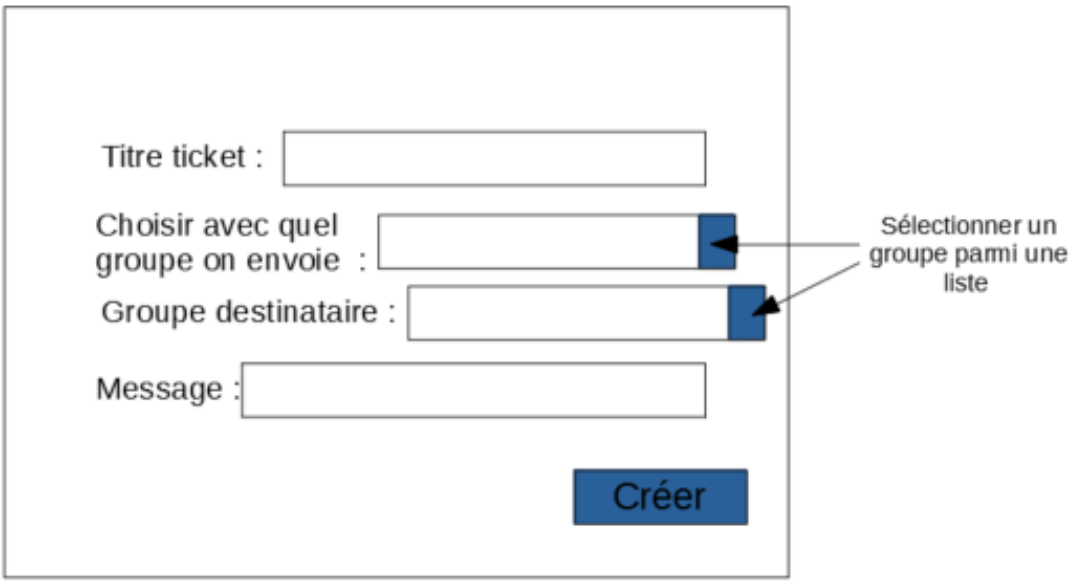


- **Interface de l'utilisateur :**



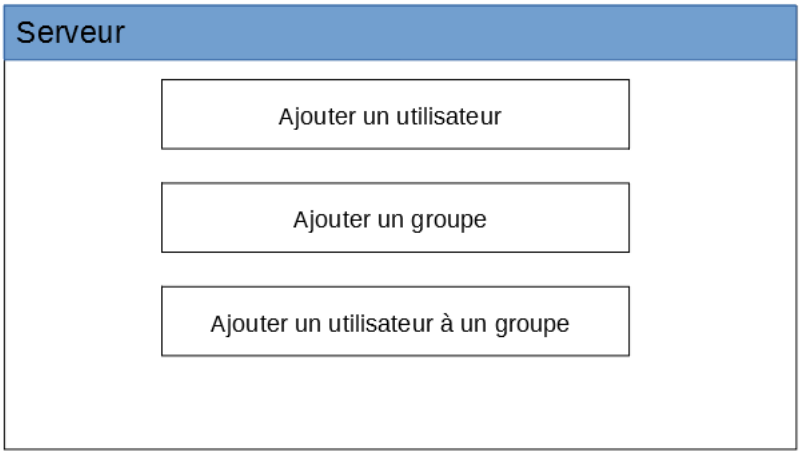


- **Interface d'ajout d'un ticket** : permet d'ajouter un ticket, on renseignera le titre, le groupe à qui l'on souhaite envoyer le message, le groupe destinataire et le message.

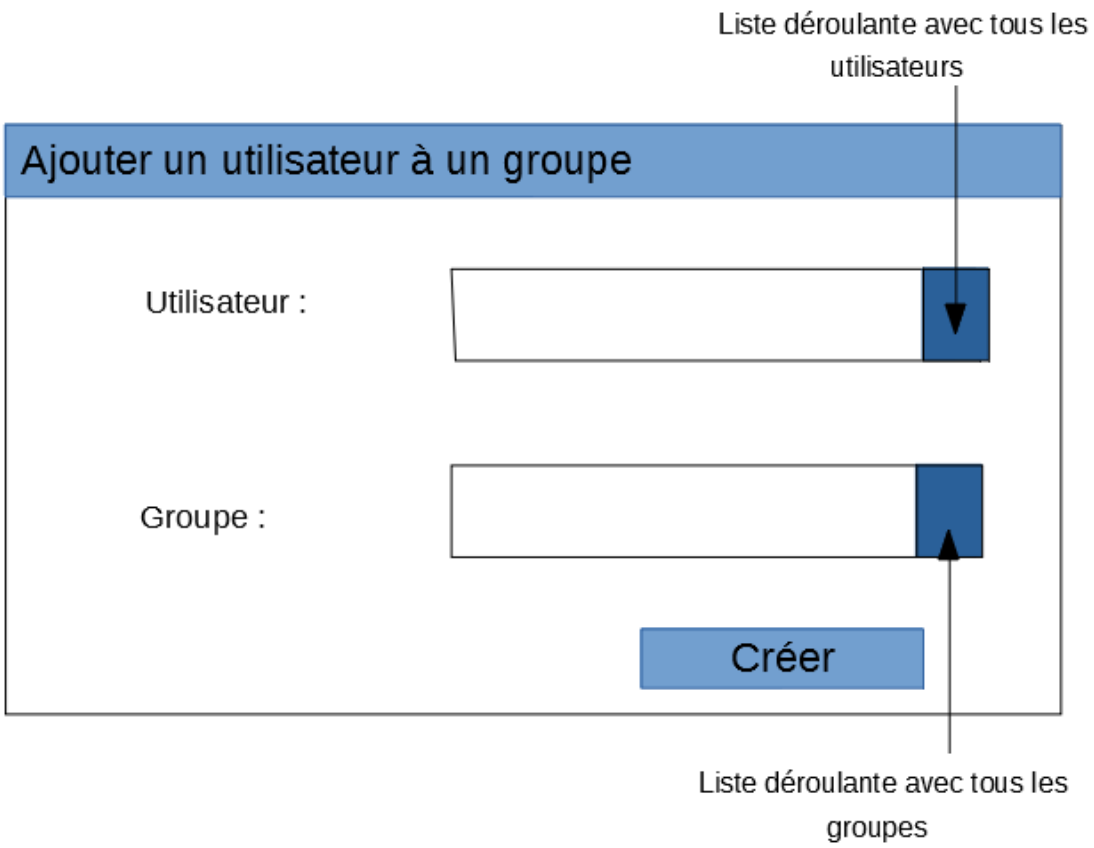


2) Interface côté serveur

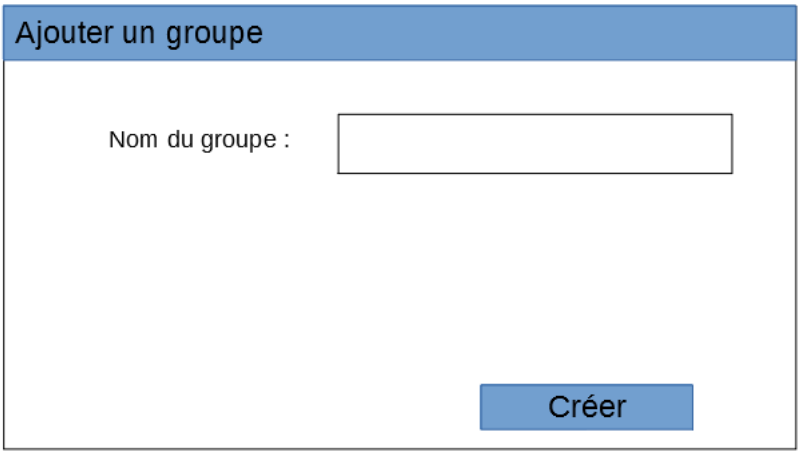
- **Interface première** : permet de choisir l'opération à effectuer (on en a mis seulement trois juste pour l'exemple)



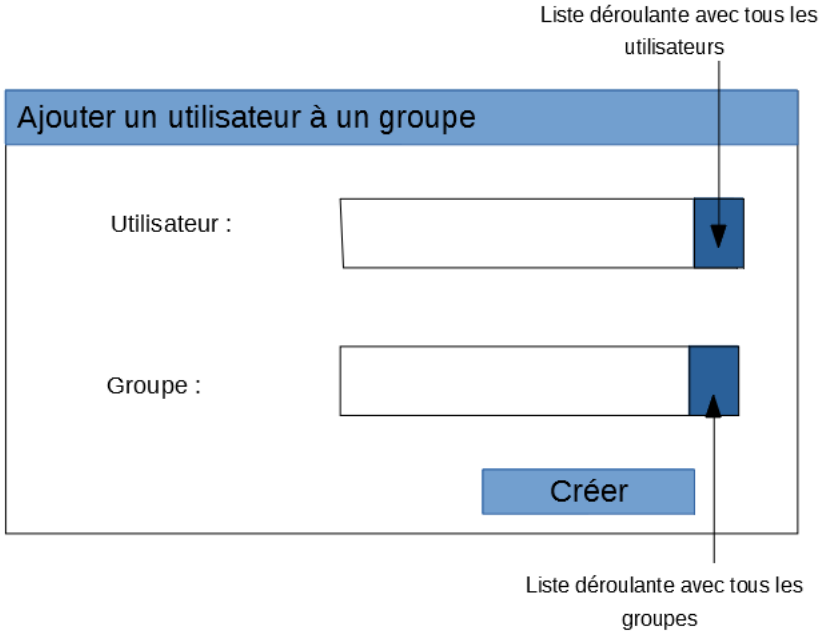
- **Interface ajouter un utilisateur**



- **Interface ajouter un groupe**

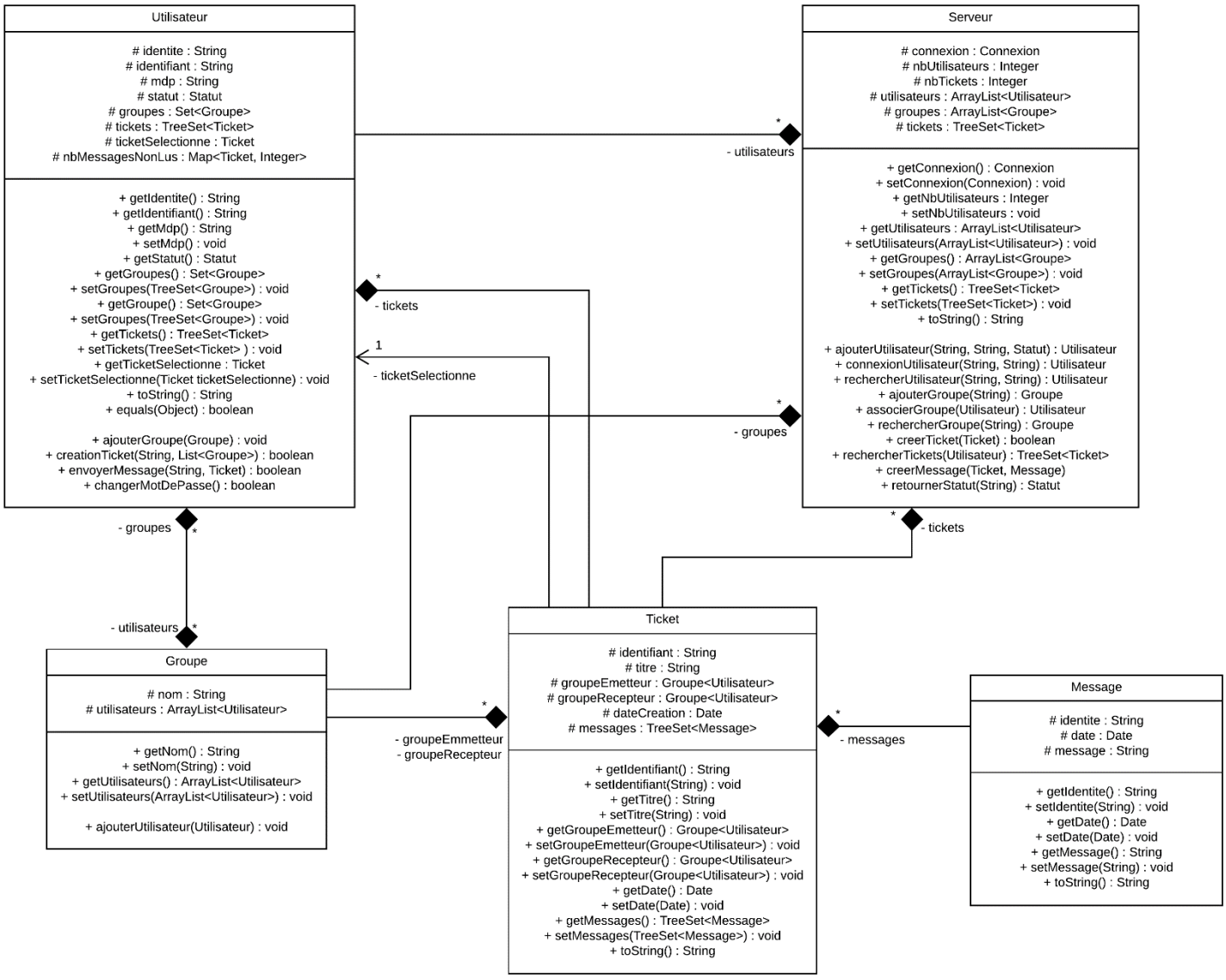


- **Interface ajouter un utilisateur à un groupe**



IV. Diagramme de classe

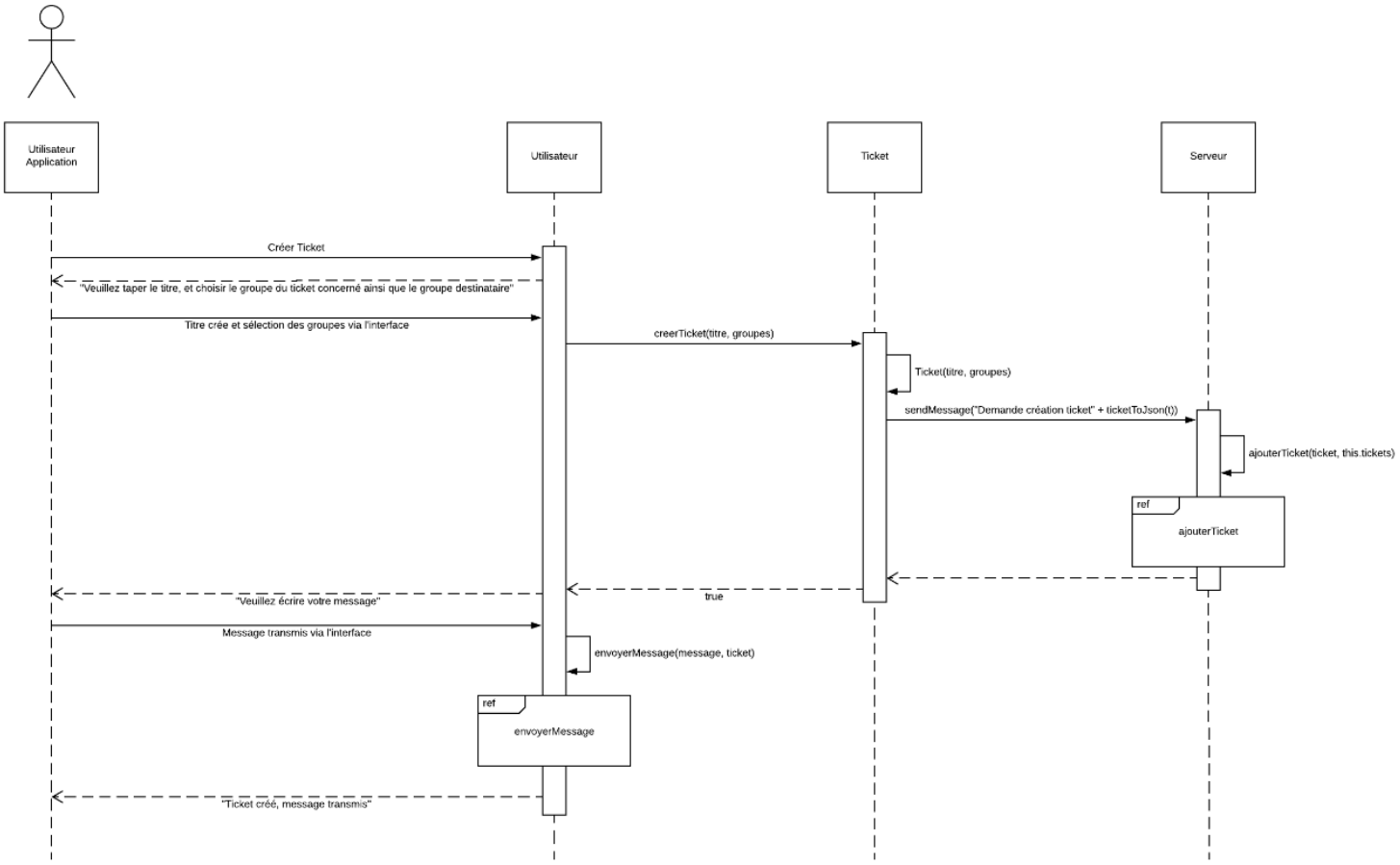
Voici nos réflexions autour des classes dont nous avons besoin. Ce premier diagramme sera amené à évoluer au fur et à mesure de l'avancée du projet.



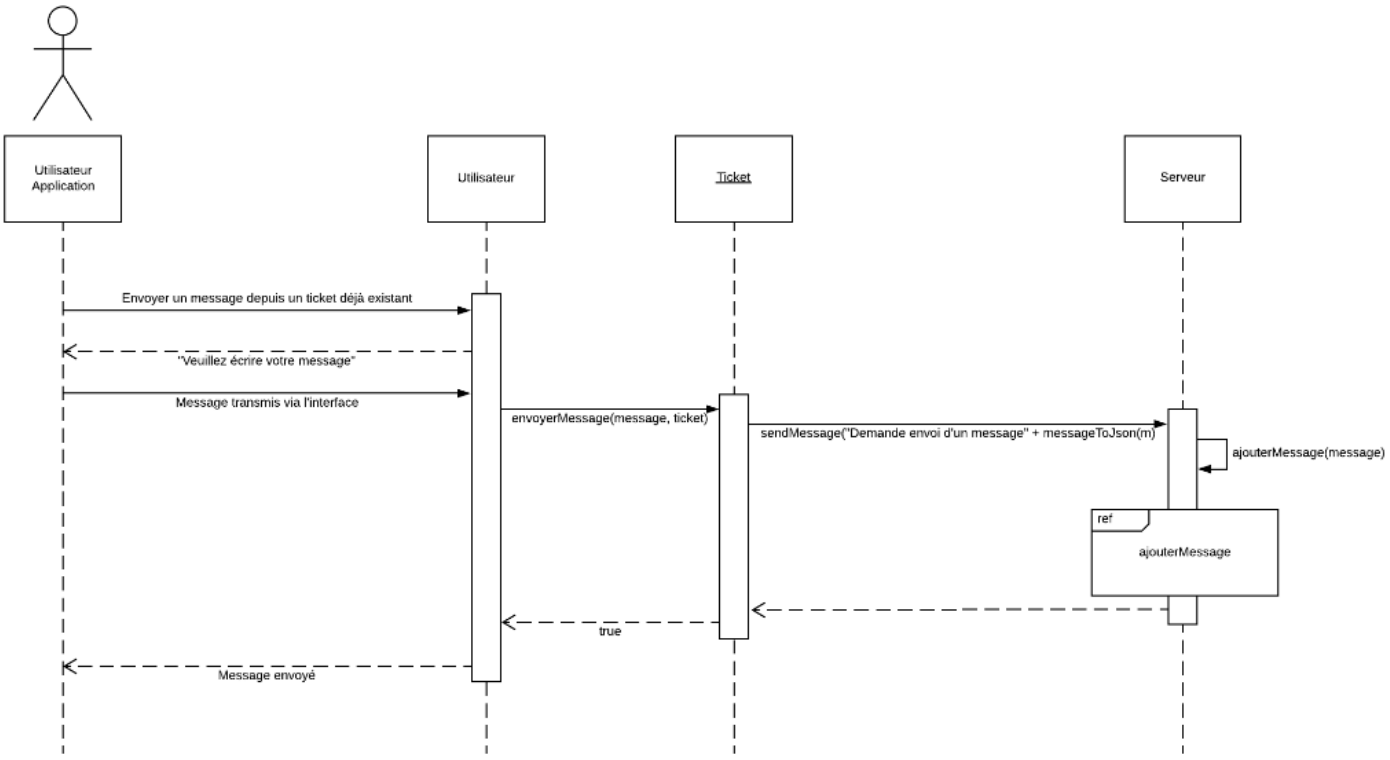
V. Diagramme de séquence

Liste de méthodes côté utilisateurs :

- Créer un ticket : public boolean creationTicket(String titre, List<Groupe> groupes)



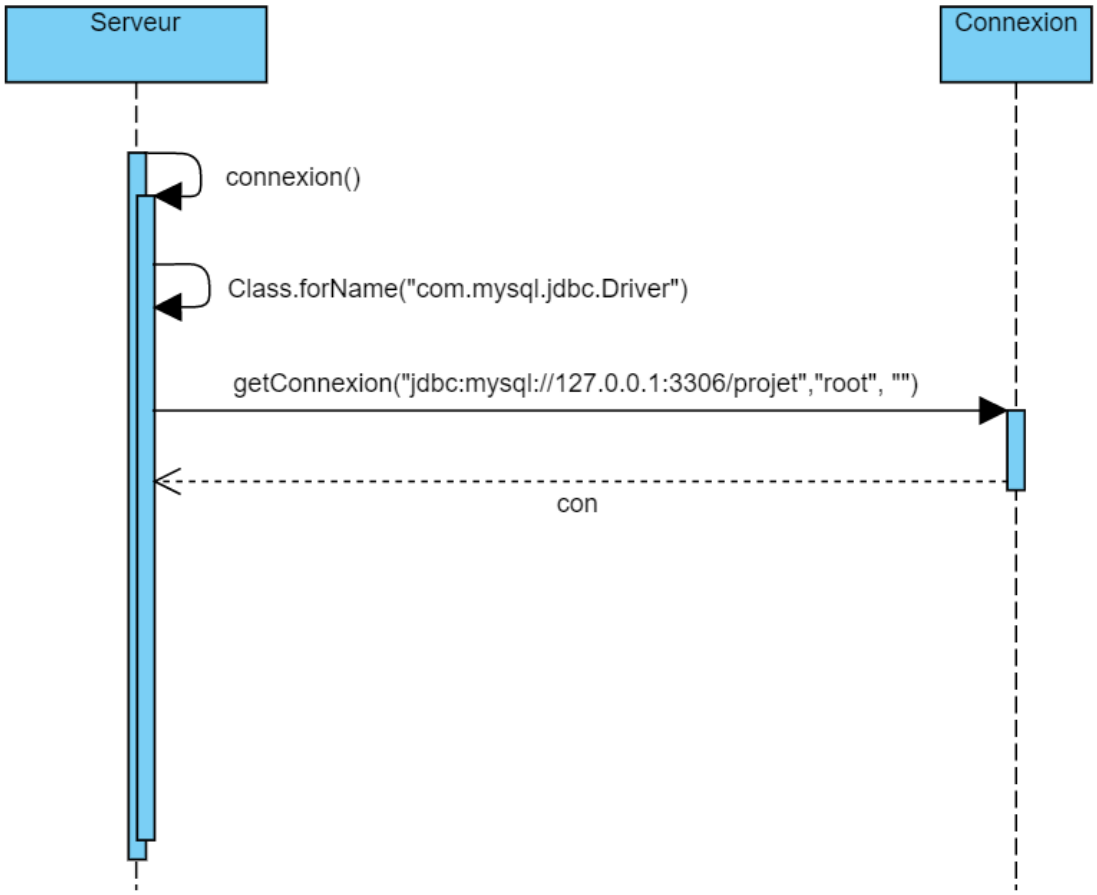
- Envoyer un message : public boolean envoyerMessage(String message, Ticket ticket)



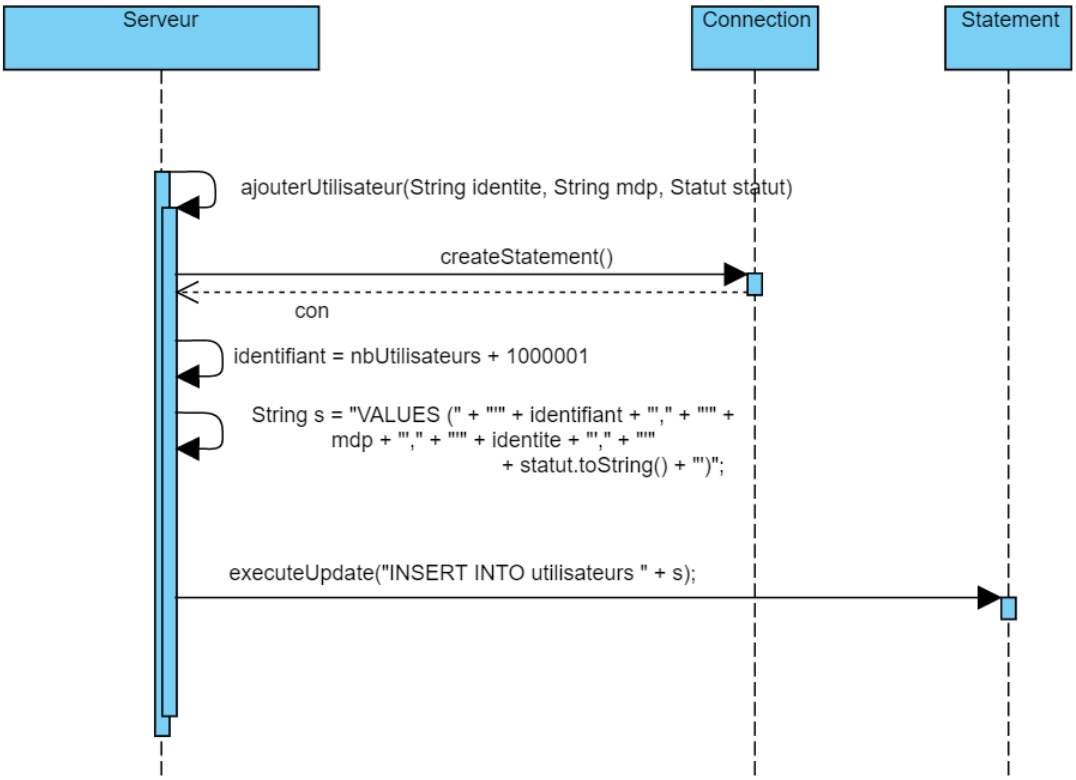
- Changer le mot de passe : changerMotDePasse()

Liste des méthodes côté serveur :

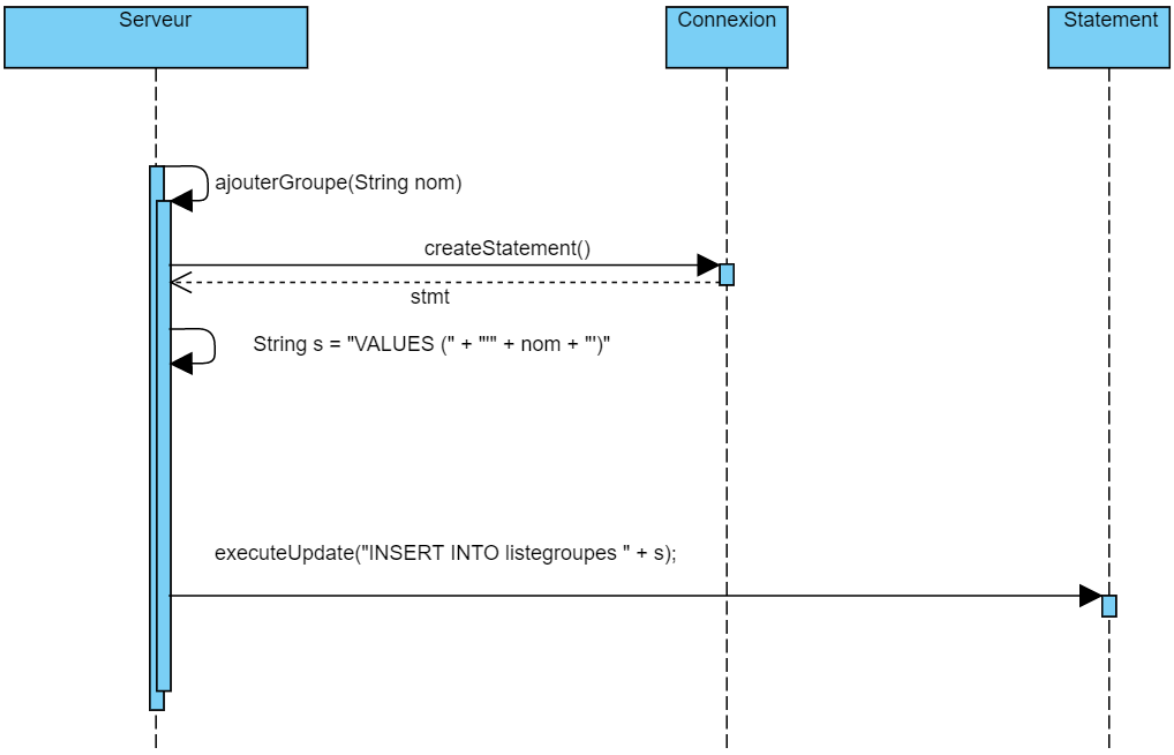
- Connexion : connexion()



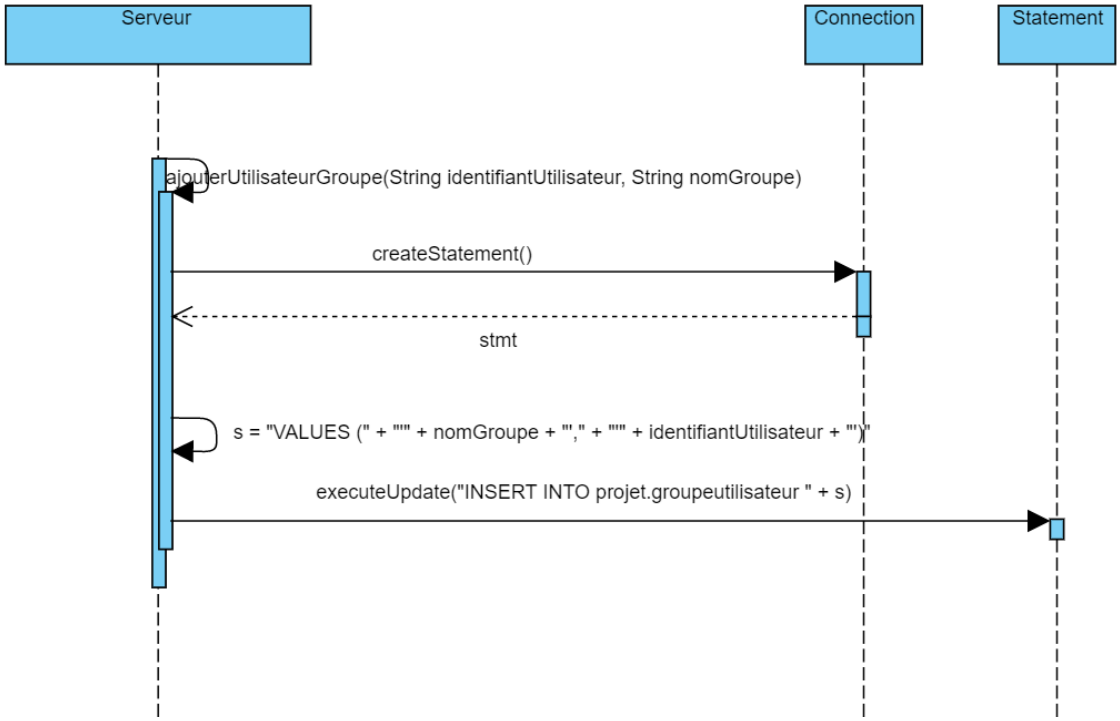
- Ajouter un utilisateur : ajouterUtilisateur(String identité, String mdp, Statut statut)



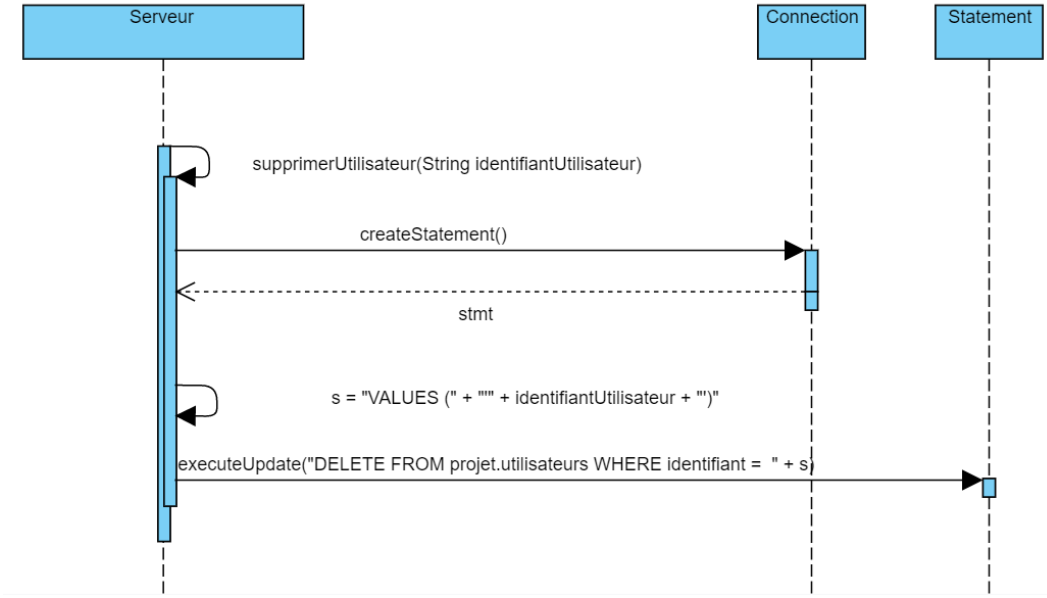
- Ajouter un groupe : ajouterGroupe(String nom)



- Ajouter un utilisateur à un groupe : ajouterUtilisateurGroupe(String identifiantUtilisateur, String nomGroupe);



- Supprimer un utilisateur : supprimerUtilisateur(String identité)



Toutes les autres fonctions sont basées sur le même principe.

- Modifier un utilisateur : modifierUtilisateur(String identité)
- Supprimer un groupe : supprimerGroupe(String nom)
- Modifier un groupe : modifierGroupe(String nom)
- Créer un ticket : creerTicket(String identifiantTicket, String titre, String groupeEmetteur, String groupeRecepteur);
- créer message : creerMessage(String identifiantTicket, String identite, String message);

VI. Modèle relationnel de notre base de données

Nous aurons une base de données « projet » contenant différentes tables.

- Une table “utilisateurs” sous la forme :

identifiant	mdp	identite	statut

Elle contient tous les utilisateurs, leur identifiant, leur mot de passe, leur identité (NomPrenom) et leur statut (ENSEIGNANT, ETUDIANT, …).

- Une table « messages » sous la forme :

identifiantTicket	date	identite	message

Elle contient tous les messages sous la forme identifiant du ticket, date du message, identité de l’émetteur et le message associé.

- Une table “groupeutilisateur” sous la forme :

nomGroupe	identifiantUtilisateur

Elle permet de faire l’association entre un groupe et un utilisateur. Elle prend le nom du groupe et l’identifiant de l’utilisateur qui est dans ce groupe.

- Une table “tickets” sous la forme :

identifiantTicket	titre	date	groupeEmetteur	groupeRecepteur

Elle contient tous les tickets. Un ticket est défini par l’identifiant du ticket sous la forme (T+nombreunique), le titre de ce dernier, sa date de création, le groupe qui envoie ce ticket et le groupe qui le reçoit.

- Une table “listegroupe” sous la forme :

nom du groupe

Cette base contient tous les noms de groupe. Il faudra veiller à ce que les groupes soit tous unique.

VII. Conclusion

Voici donc une grosse partie de notre réflexion sur le projet. Cette structure initiale est amenée à évoluer selon les problèmes rencontrés ou les conditions requises durant l’avancée du projet.

Une partie que nous n’avons pas traitée est celle de la réception des messages : le traitement des messages suivants, s’ils ont été reçus par le serveur, par tous les utilisateurs ou non…