

Yuheng CHEN - Louis GREINER - Benjamin MISSAOUI

## SR03 **Devoir 2**

Introduction aux WebSockets

# DÉVELOPPEMENT D'UNE APPLICATION WEB DE CHAT



***P21***

## TABLE DES MATIÈRES

TABLE DES MATIÈRES	1
1. INTRODUCTION	2
2. FONCTIONNEMENT DE L'APPLICATION	3
2.1 FONCTIONNEMENT INTERNE	3
2.2 RÉSUMÉ DE L'ARCHITECTURE : DIAGRAMME DE CLASSE	13
3. ENVIRONNEMENT DE L'APPLICATION	14
3.1 PRÉ-REQUIS	14
3.2 TESTS ET SCÉNARIOS	15

## 1. INTRODUCTION

L'objectif de ce projet est de concevoir une application permettant à différents utilisateurs d'interagir dans de multiples salons de discussions (chats).

Les utilisateurs sont enregistrés dans une base de données, et peuvent, une fois connectés, gérer (créer, modifier, supprimer, se connecter à) des chats.

Un chat a un unique propriétaire, son créateur, qui peut ensuite inviter d'autres utilisateurs à rejoindre son chat (sans invitation, les utilisateurs ne peuvent pas rejoindre un chat dont ils ne sont pas propriétaires). Les chats et invitations sont aussi stockés dans une base de données.

En se connectant à un chat, l'utilisateur peut choisir un pseudo, et peut voir les utilisateurs connectés en même temps que lui au chat. Les utilisateurs connectés à un même chat peuvent converser entre eux, voir la liste des utilisateurs connectés (leurs pseudos précédemment choisis), jusqu'à déconnection. La déconnexion d'un utilisateur averti les autres utilisateurs connectés au chat en question, et renvoie l'utilisateur à sa page d'accueil utilisateur. Les messages ne seront pas enregistrés, donc une personne se connectant après une discussion, n'y aura pas accès.

## 2. FONCTIONNEMENT DE L'APPLICATION

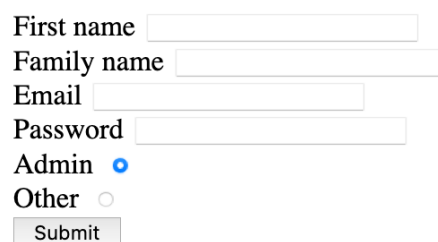
Dans cette première partie, intéressons-nous au fonctionnement interne de l'application. Nous allons d'abord expliquer en détail ce qu'il se passe lorsque l'utilisateur utilise chacune des fonctions proposées, ainsi que les choix que ces fonctionnalités nous ont amenés à faire. Puis nous établirons un diagramme de classe général pour résumer notre architecture, avec les différentes classes Java et les associations qui les relient.

### 2.1 FONCTIONNEMENT INTERNE

L'application doit pouvoir gérer de nombreuses fonctionnalités : connexion et inscription des utilisateurs, création de chats, invitation d'autres utilisateurs et accès aux salles de discussion. Pour chacune de ces fonctionnalités, nous avons dû adapter notre architecture. Dans cette partie, nous allons donc voir ce qu'il se passe lorsque l'utilisateur se sert de chacune de ces fonctions.

#### 1. Inscription d'un utilisateur

L'application doit d'abord pouvoir gérer l'ajout de nouveaux utilisateurs. Nous avons supposé que seuls les admins peuvent inscrire de nouvelles personnes. Pour cela, il leur suffit de se connecter (nous détaillerons le processus de connexion plus bas), et de cliquer sur "Créer un nouveau utilisateur". Ils se retrouvent alors devant ce formulaire :



First name

Family name

Email

Password

Admin ☒

Other ☐

*Fig 1 : interface d'ajout d'un utilisateur (formSQL.html)*

Une fois les informations remplies, ce formulaire est envoyé à `SignUpSQL.java` pour traitement. Le sujet impose d'abord de vérifier qu'un utilisateur de même nom et prénom n'existe pas déjà. Nous devons donc garder une trace des utilisateurs de l'application, avec leurs informations et leur identifiants de connexion. Nous utilisons pour cela une table `userdb` dont la structure est la suivante :

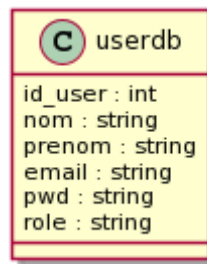


Fig 2 : structure de la table userdb dans la BDD

Pour utiliser les informations contenues dans cette table, nous utilisons le Design Pattern *Active Record*. Lorsqu'un utilisateur est inscrit, nous commençons par appeler la méthode `findByNames` de `User`. Si celle-ci nous indique qu'un utilisateur homonyme existe déjà, nous demandons confirmation à l'admin, sinon nous ajoutons le nouvel utilisateur à la table `userdb` via la méthode `save()`.

L'ajout d'un utilisateur peut être résumé avec le diagramme de séquence suivant :

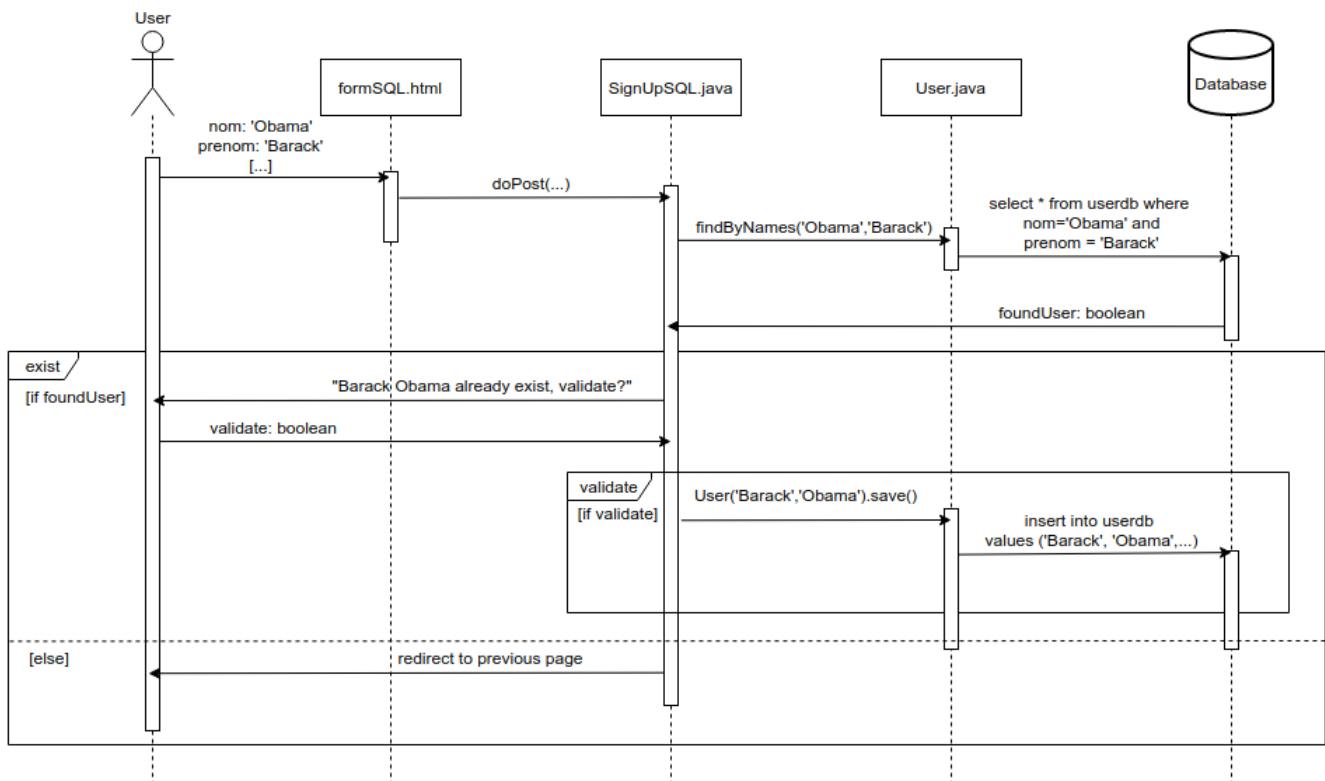


Fig 3 : Diagramme de séquence : ajout d'un utilisateur

## 2. Connexion d'un utilisateur

Une personne inscrite peut se connecter avec le formulaire `connexion.html` :

Email

Password

Fig 4 : interface de connexion

Les identifiants saisis sont envoyés à *userconnect.jsp*, qui commence par appeler la méthode `findAll()` de la classe *User*. Cette méthode envoie une requête SQL à *userdb* pour récupérer l'ensemble des utilisateurs inscrits, puis retourne une liste `ArrayList<User>`. Nous n'avons alors plus qu'à comparer les identifiants de la personne souhaitant se connecter avec tous les utilisateurs de la base. Si l'utilisateur est reconnu, il est redirigé vers la page d'accueil, sinon il reçoit un message d'erreur :

## Bienvenue Angela !

### Vos Chats:

Aucun utilisateur n'a été trouvé avec ce login ou mot de passe

Vous n'avez créé aucun chat...

### Vos invitations :

Vous n'avez aucune invitation...

Fig 5 : connexion réussie (à gauche) et échouée (à droite)

La connexion peut être résumée avec le diagramme de séquence suivant :

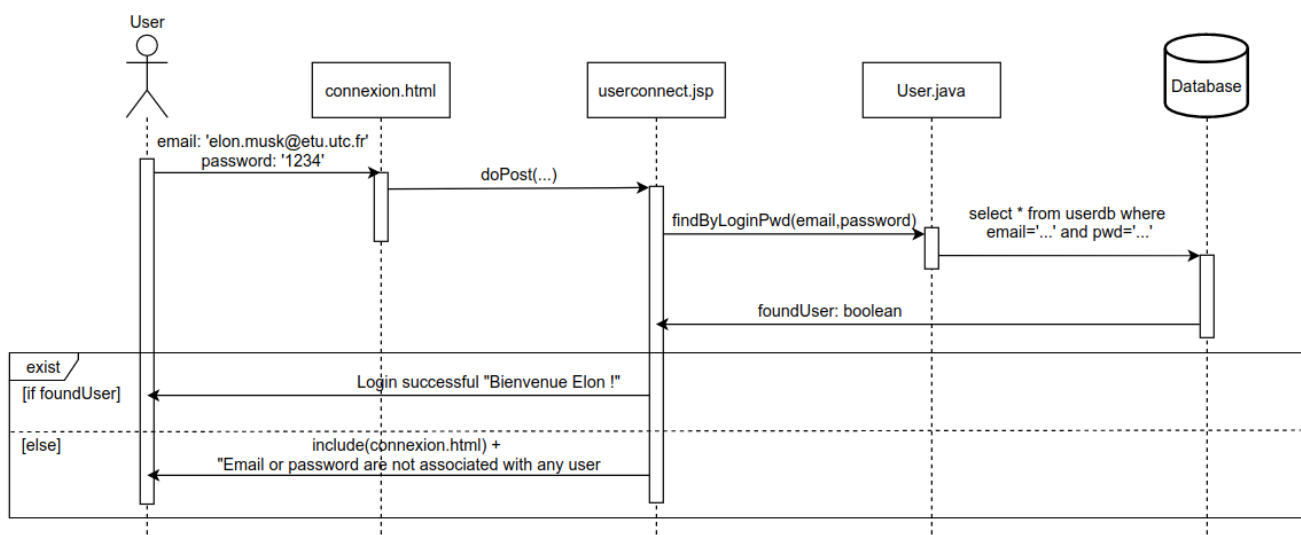


Fig 6 : Diagramme de séquence : connexion d'un utilisateur

## 3. Création d'un chat

L'ajout d'un chat est similaire à l'inscription d'un utilisateur et se fait par un admin. Nous devons d'abord vérifier que le nom du chat n'est pas déjà pris. Pour garder en mémoire

les chats créés par les différents utilisateurs, nous avons donc ajouté une table `chat` à notre BDD :

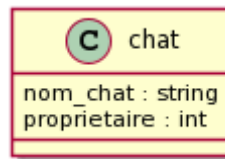


Fig 6 : structure de la table chat dans la BDD

L'attribut `proprietaire` est ici une clé étrangère vers la table `userdb`. En pratique, lorsque l'utilisateur crée un chat, le propriétaire est retrouvé automatiquement grâce à la `session`, il lui suffit donc de saisir le nom du chat, qui est un identifiant unique.

Nous disposons à nouveau d'une classe `Chat.java` qui implémente le design pattern `Active Record`. Lorsque l'utilisateur crée un chat, nous appelons `findByName()` de la classe `Chat.java` qui nous indique si un chat de même existe ou non dans la BDD. Si ce n'est pas le cas, nous pouvons créer ce nouveau chat avec la méthode `save()`.

Lors de la création du chat, l'utilisateur peut aussi saisir les utilisateurs qu'il souhaite inviter. Lorsqu'ils se connectent, ce nouveau chat apparaîtra alors dans leurs invitations :

Nom du chat

Description du chat

Users to invite:

Date de fin de validité

Horaire de fin de validité

### Bienvenue Angela !

**Vos Chats:**

Vous n'avez créé aucun chat...

**Vos invitations :**

ID	Lien
secret-meeting	<a href="localhost:8080/SR03-Devoir2/client.html?chat=secret-meeting">localhost:8080/SR03-Devoir2/client.html?chat=secret-meeting</a>

Dernière visite le 29/04/2021\_22:59:47

Fig 7 : interface de création d'un chat (à gauche), page d'accueil d'un invité (à droite)

Cette fonctionnalité nécessite aussi une nouvelle table : `invitation` et la classe `Active Record` associée : `Invitation.java`

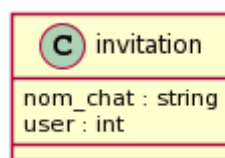


Fig 7 : structure de la table invitation dans la BDD

Pour chacun des invités, on vérifie si la personne était déjà invitée, auquel cas on prévient seulement l'utilisateur. Si elle ne l'était pas, on l'ajoute avec la méthode `save()`.

Le diagramme de séquence lié à l'ajout d'un chat est le suivant :

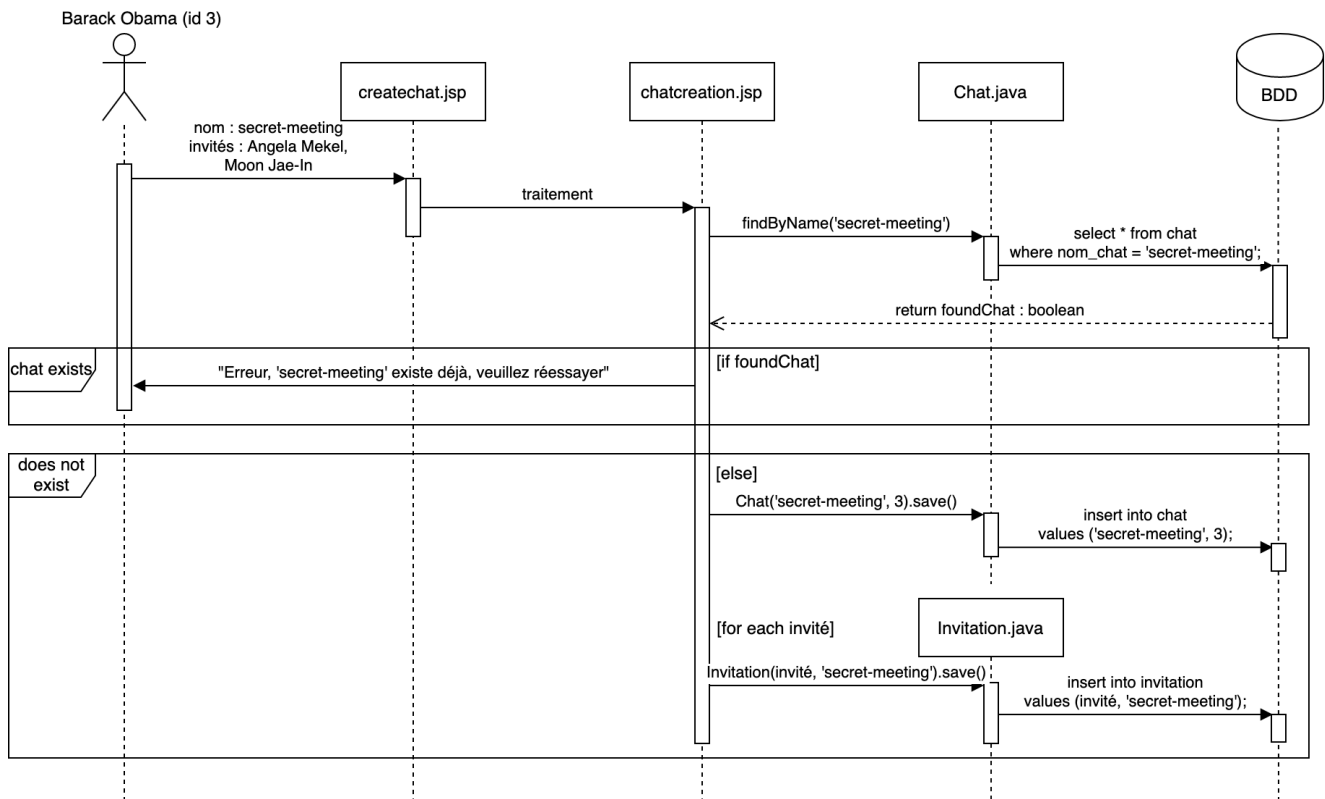


Fig 8 : Diagramme de séquence : création d'un chat

#### 4. Invitation d'autres utilisateurs

Un admin connecté peut aussi ajouter des invités à un chat existant en cliquant sur "Inviter des utilisateurs à un chat". Il doit alors indiquer le chat et les utilisateurs qu'il souhaite inviter :

secret-meeting

Chat :

Bill Clinton  
 Georges Washington  
 Angela Merkel  
 Moon Jae-In

Users to invite:

Submit

Retour à l'accueil

Fig 9 : interface d'invitation d'utilisateurs à un chat existant

La procédure est la même que précédemment, on invite les utilisateurs qui ne l'étaient pas encore, et on indique à l'utilisateur ceux qui étaient déjà invités :



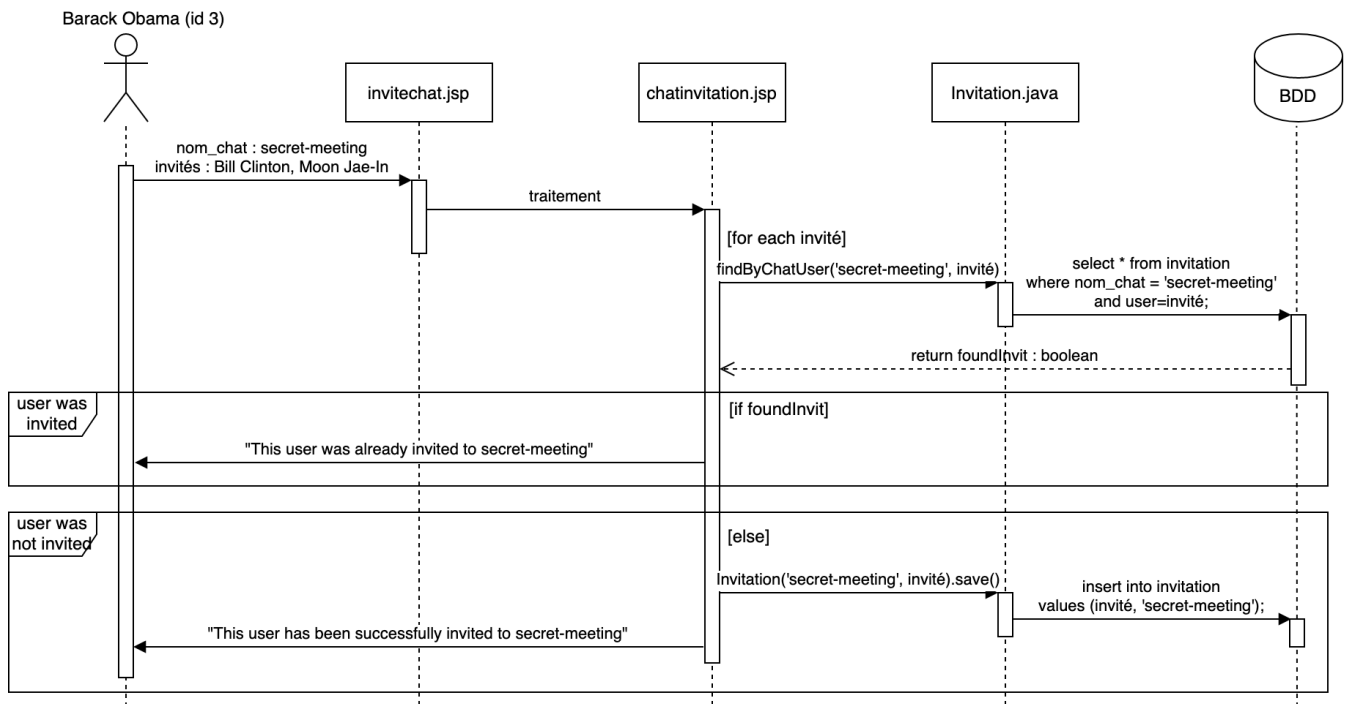


Fig 10 : Diagramme de séquence : invitation d'utilisateurs à un chat existant

## 5. Supprimer des invitations d'un chat

A tout moment, un admin peut retirer des invités d'un chat en cliquant sur "Retirer des utilisateurs d'un chat". Il se retrouve alors devant le formulaire suivant :

secret-meeting

Chat :

Users to delete from this chat:

Barack Obama  
Bill Clinton  
Georges Washington  
Angela Merkel

Submit

Retour à l'accueil

Fig 11 : interface de suppression d'utilisateurs d'un chat existant

A nouveau, le traitement est similaire : on supprime les utilisateurs qui étaient effectivement invités, et on indique à l'utilisateur ceux qui ne l'étaient pas :

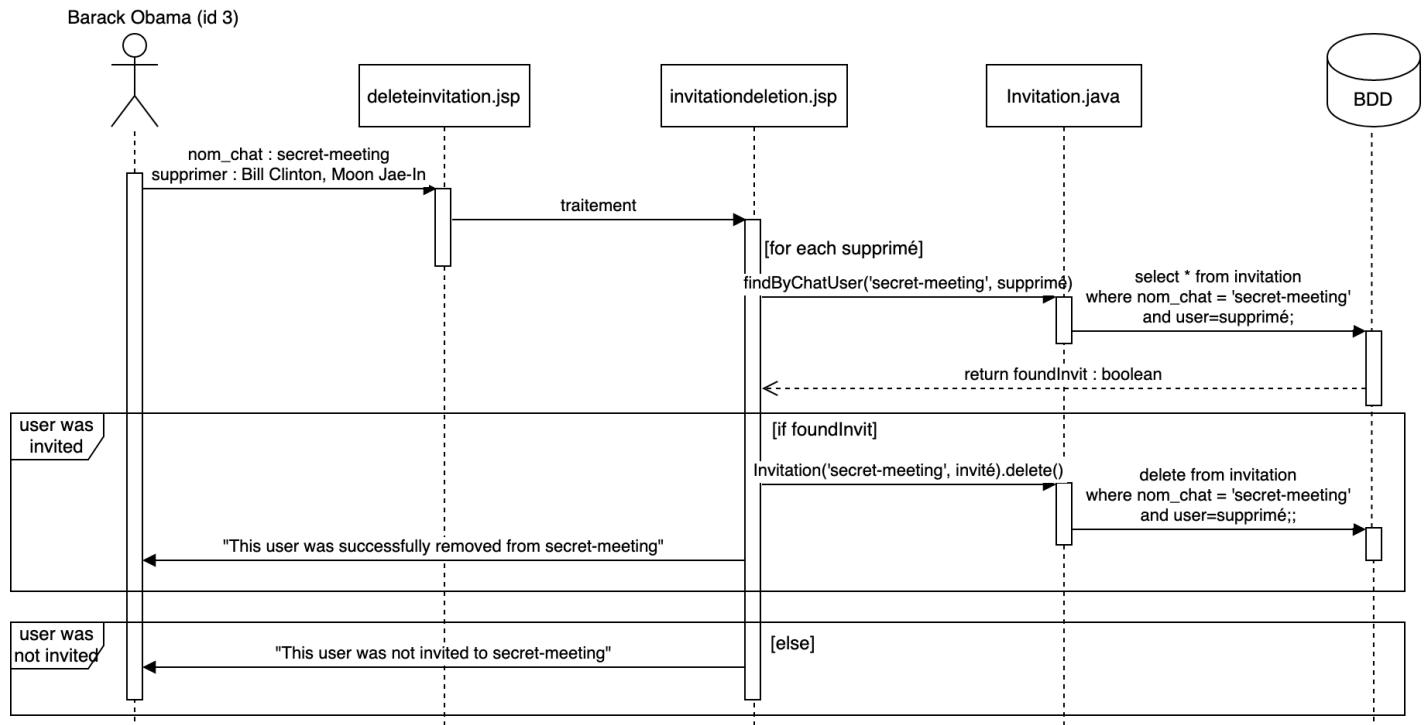


Fig 12 : Diagramme de séquence : suppression d'invitations

## 6. Accès aux salles de discussion

La fonctionnalité principale de l'application, qui ne nécessite pas d'être admin, est l'accès aux salles de chat pour discuter avec d'autres usagers, jusqu'à ce que l'utilisateur se déconnecte de ce chat, retournant ainsi à la page d'accueil de son compte utilisateur. Tout utilisateur peut, depuis sa page d'accueil, consulter ses invitations à des chats, et s'il est admin, les chats qu'il a créés. En cliquant sur le lien relatif à un chat bien précis, l'utilisateur peut entrer un pseudo temporaire, initialisant ainsi une nouvelle instance de WebSocket qui simulera sa connexion:

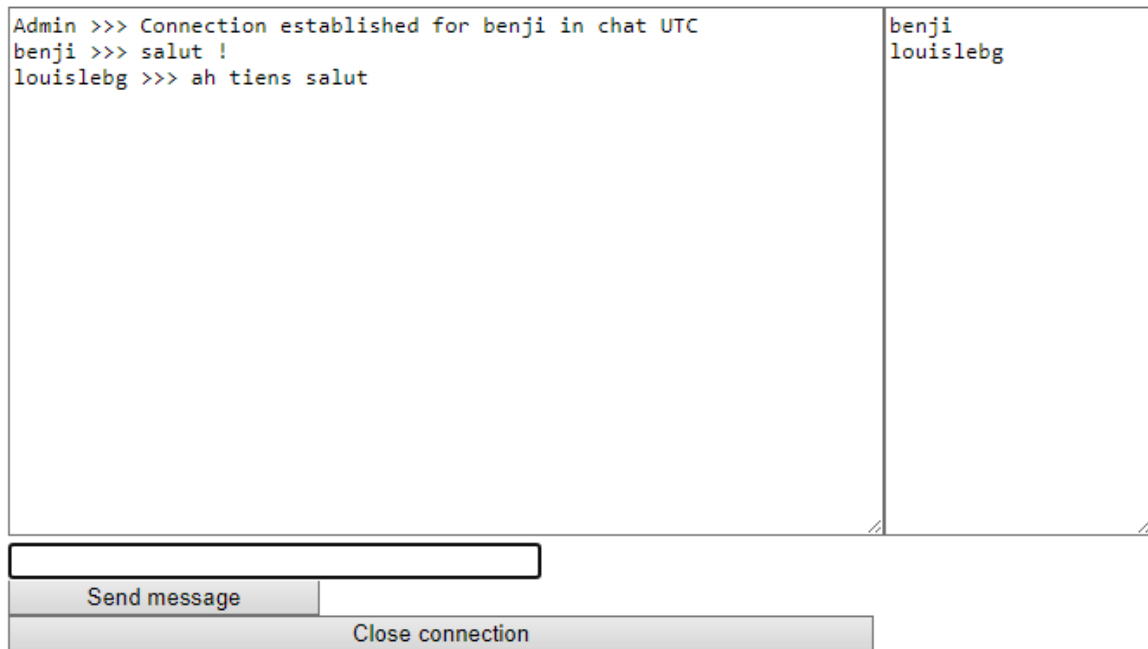
```
let ws = new WebSocket( "ws://localhost:8080/SR03-Devoir2/" +
  queryDict["chat"] + "/" + pseudo );
```

La page de gestion du chat intègre du code javascript, qui permet à l'utilisateur de discuter en temps réel avec les autres utilisateurs connectés, jusqu'à ce que cet utilisateur se déconnecte. Les messages sont envoyés à chaque instance de WebSocket connectées à un même chat.

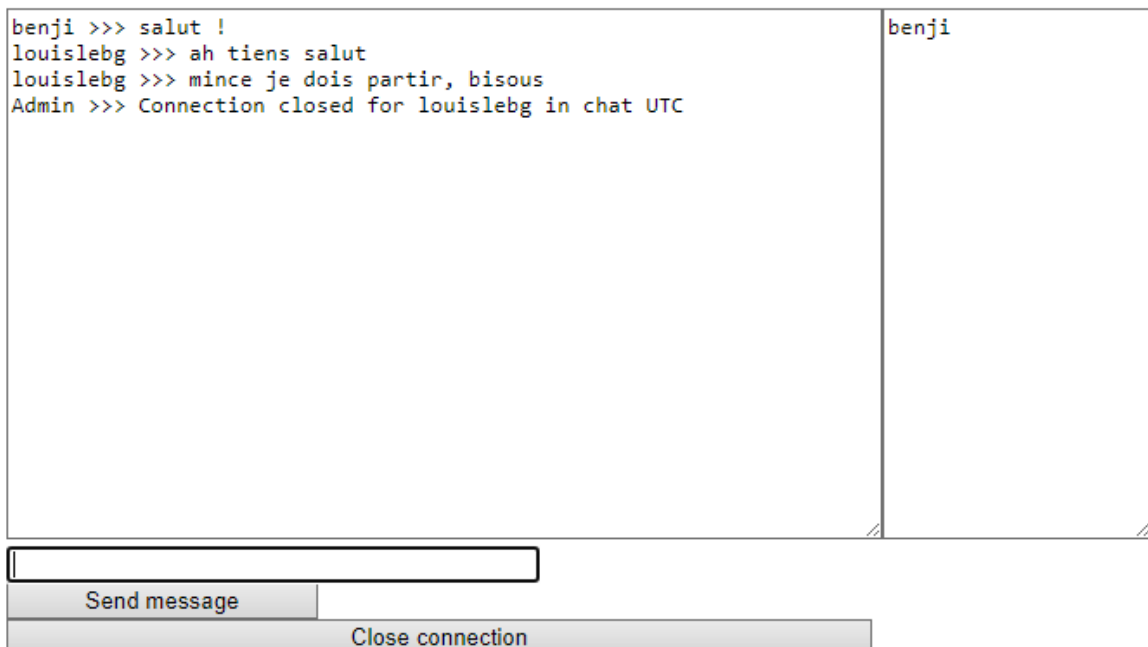
La liste des utilisateurs connectés est actualisée en temps réel. D'abord on liste tous les utilisateurs dans une table:

```
Hashtable<String, Session> sessions = new Hashtable<>();
```

La méthode `CreateUserJsonData` de `ChatServer` parcourt à tout moment la liste d'utilisateurs stockés dans `sessions` pour mettre à jour l'objet JSON qu'elle manipule: la liste d'utilisateur, qui fonctionne de façon similaire à une Hashtable. Ainsi la liste d'utilisateurs connectés est modifiée en temps réel, et les utilisateurs sont prévenus en cas de déconnexion de l'un d'entre eux.



*Fig 13 : Fenêtre de chat avant déconnexion*



*Fig 14: Fenêtre de chat après déconnexion*

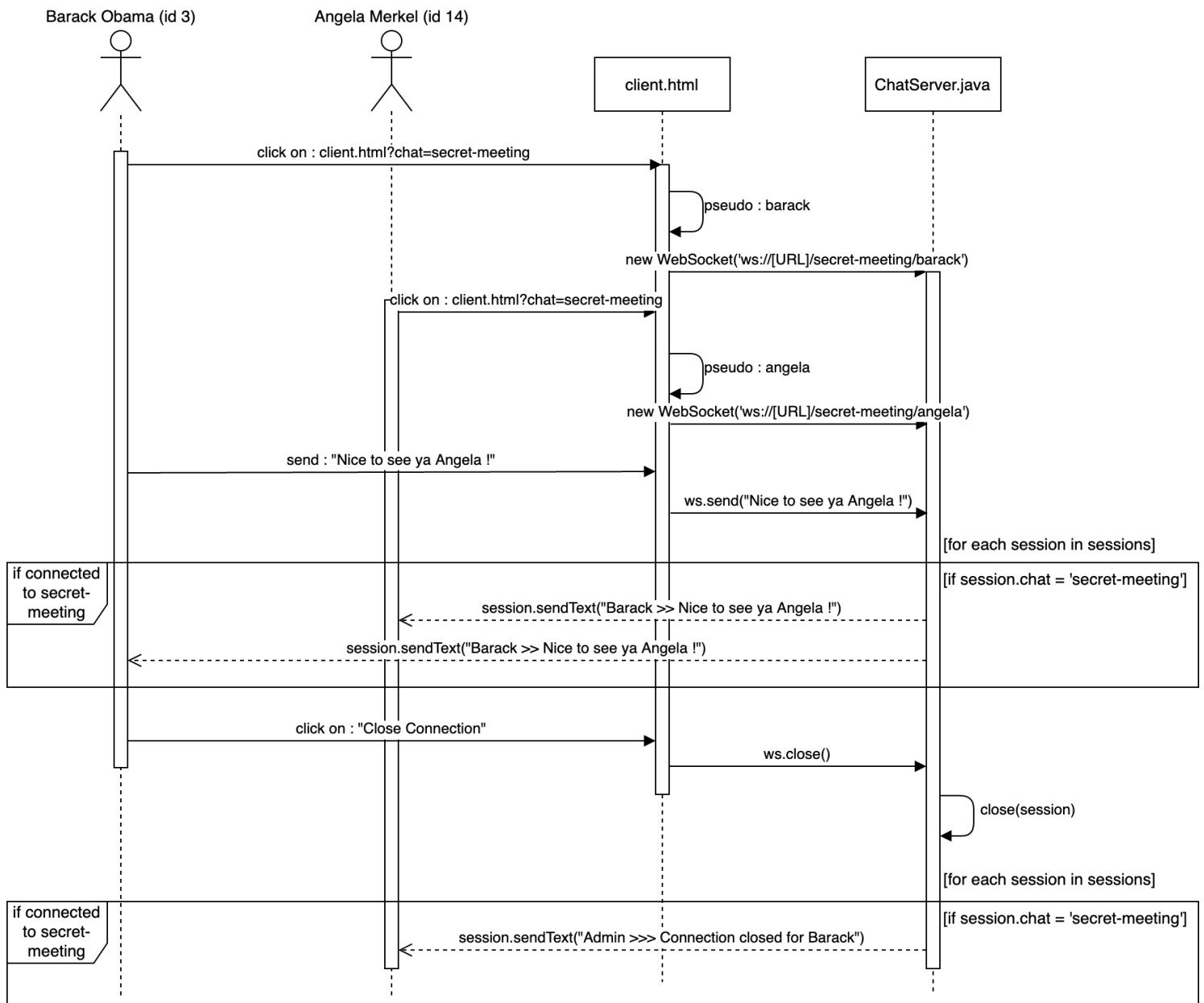


Fig 15: Diagramme de séquence: Fonctionnement d'un chat

## 7. Récapitulatif de toutes les fonctionnalités depuis l'écran d'accueil utilisateur

Un utilisateur admin aura à sa disposition des fonctionnalités supplémentaires par rapport à un utilisateur non-admin. Cependant, le corps de l'application reste le même pour les deux types d'utilisateurs. Sur le diagramme de séquence suivant, la connexion à la base données est sous-entendue pour chaque fonctionnalité, l'utilisateur est en permanence connecté à la base de données.

## Bienvenue Benjamin !

Que souhaitez-vous faire? [ADMIN]

- [Créer un nouveau utilisateur](#)
- [Afficher la liste des utilisateurs et des chats](#)
- [Créer un nouveau chat](#)
- [Inviter des utilisateurs à un chat](#)
- [Retirer des utilisateurs d'un chat](#)

Vos Chats:

ID / Nom Lien	Description
SR03 <a href="localhost:8080/SR03-Devoir2/client.html?chat=SR03">localhost:8080/SR03-Devoir2/client.html?chat=SR03</a>	Groupe de soutien pour Louis Greiner (il est a la ramasse)

Vos invitations :

ID / Nom	Lien	Description
Secret meeting	<a href="localhost:8080/SR03-Devoir2/client.html?chat=Secret meeting">localhost:8080/SR03-Devoir2/client.html?chat=Secret meeting</a>	Qu'est-ce que vous ne comprenez pas dans le terme secret??
UTC =)	<a href="localhost:8080/SR03-Devoir2/client.html?chat=UTC">localhost:8080/SR03-Devoir2/client.html?chat=UTC</a>	Groupe Facebook étudiant de l'UTC

Dernière visite le 02/05/2021\_14:44:01

Déconnexion

Fig 16 : Interface de la page d'accueil utilisateur d'un admin

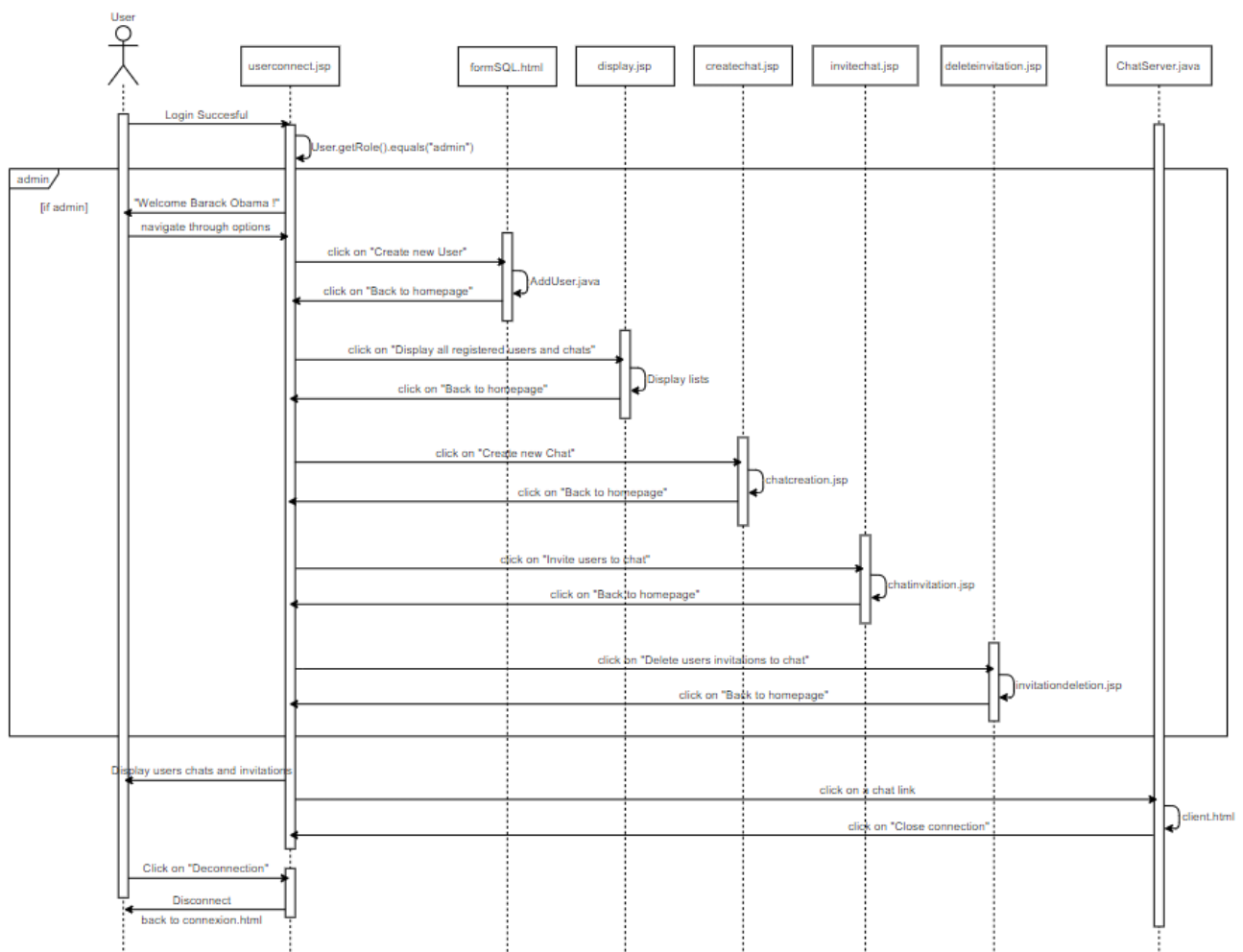


Fig 17 : Diagramme de séquence : Vue d'ensemble de l'application

## 2.2 RÉSUMÉ DE L'ARCHITECTURE : DIAGRAMME DE CLASSE

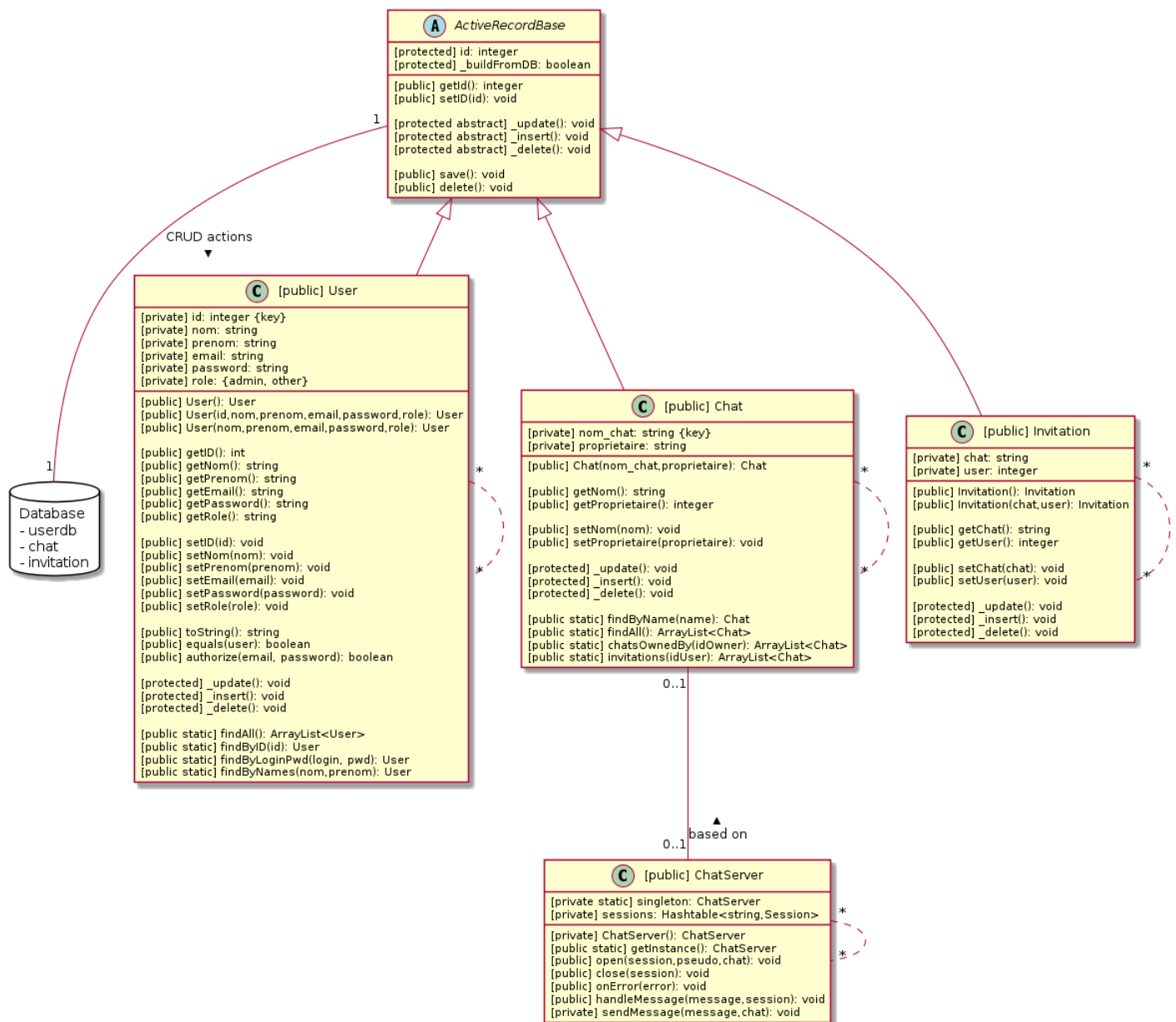


Fig 18 : Diagramme UML des objets manipulés dans l'application

Voici donc un résumé de notre architecture. Son but est donc de pouvoir interagir facilement avec la BDD. Comme expliqué plus haut, nous avons choisi d'implémenter le Design Pattern Active Record. Les objets des classes `Invitation`, `User` et `Chat` sont donc destinés à contenir les informations d'une ligne que l'on souhaite stocker ou récupérer dans la table correspondante de la BDD. Ces classes possèdent aussi des méthodes qui nous permettent d'exécuter simplement des requêtes répétitives, comme `findByName(name)` (dans la classe `Chat`) qui nous permet de récupérer l'ensemble des chats contenus dans la BDD dont le nom est égal au `name` passé en paramètre. Cette méthode est naturellement très pratique pour s'assurer, avant de créer un chat, que le nom n'est pas déjà pris.

## 3. ENVIRONNEMENT DE L'APPLICATION

### 3.1 PRÉ-REQUIS

Pour pouvoir tester l'application sur votre machine, veuillez vérifier que vous remplissez toutes les conditions pré-requises ci-dessous, et que vous avez bien tous les fichiers (notamment les librairies):

- IDE: Eclipse IDE for Java EE Developers (2021-03)
- JDK: JavaSE-15
- Serveur HTTP, gestion des servlets et JSP: Apache Tomcat v8.0 (at localhost)
- Libraires ajoutées: "json-20210307.jar" et "mysql-connector-java-8.0.23.jar"
- Gestion de la base de données: PhpMyAdmin, importez simplement le fichier "sr03-devoir2.sql" pour pouvoir tester l'application avec une base de données pré-remplie
- PHP: version 8.0.3, extension "mysqli"
- Mise en place d'un serveur web local: XAMPP, MAMP ou WampServer (utilisation de MySQL, port 3306 sous XAMPP)
- Modifier le fichier "fichierProp.txt" (sr03\_devoir 2v2/build/classes/projet2/fichierProp.txt) pour respecter votre configuration (e.g. changer le numéro de port s'il est différent de 3306, ou éventuellement le login/mdp pour accéder à la BDD si vous les avez précédemment changés).
- **Modifier manuellement la ligne 4 du fichier "Properties.java"**  
(/SR03-Devoir2/src/main/java/projet2/Properties.java), en remplaçant le chemin d'accès au fichier "fichierProp.txt", par celui de votre machine.

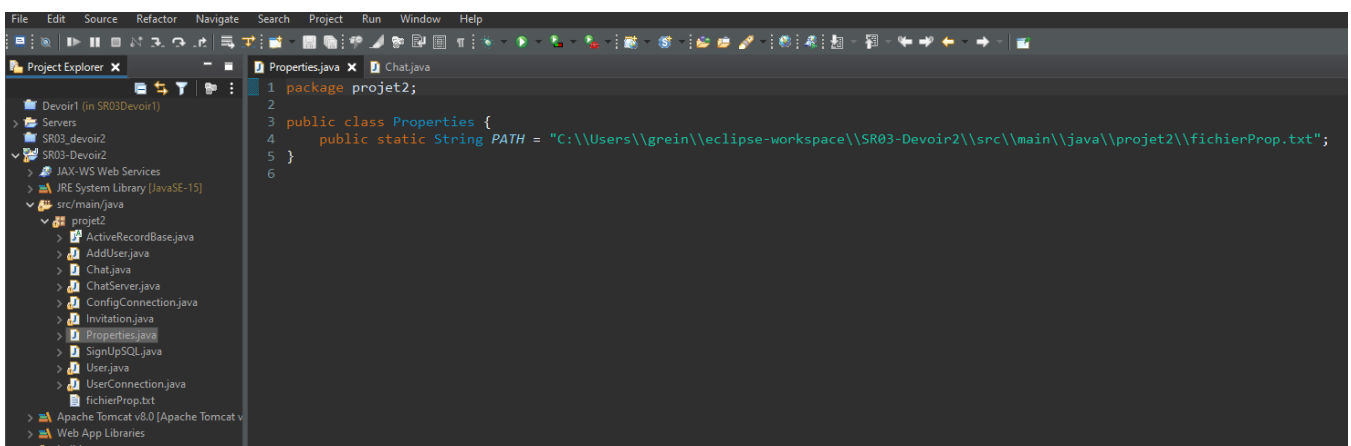
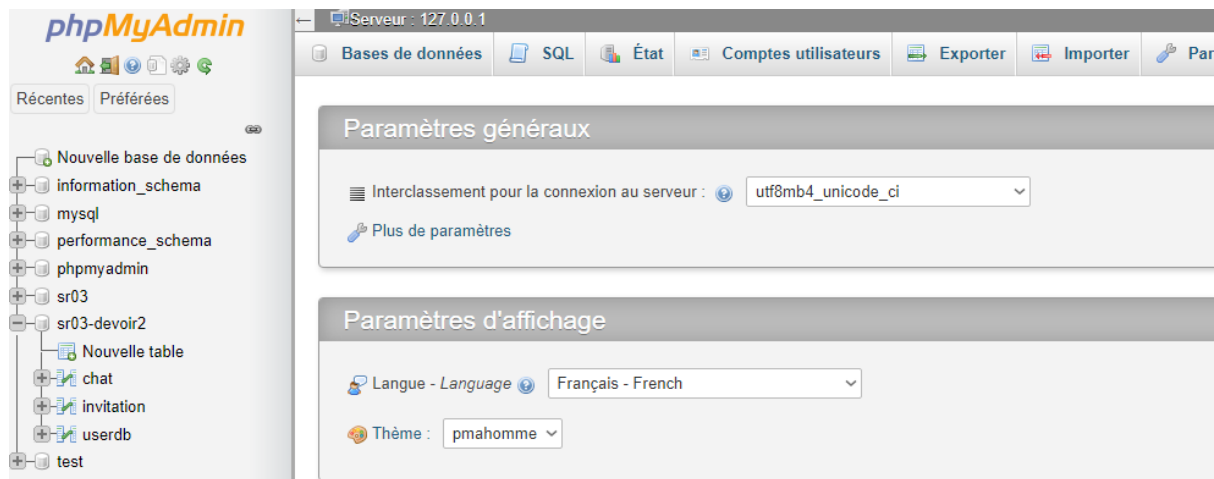


Fig 19 : Chemin de fichierProp.txt à changer

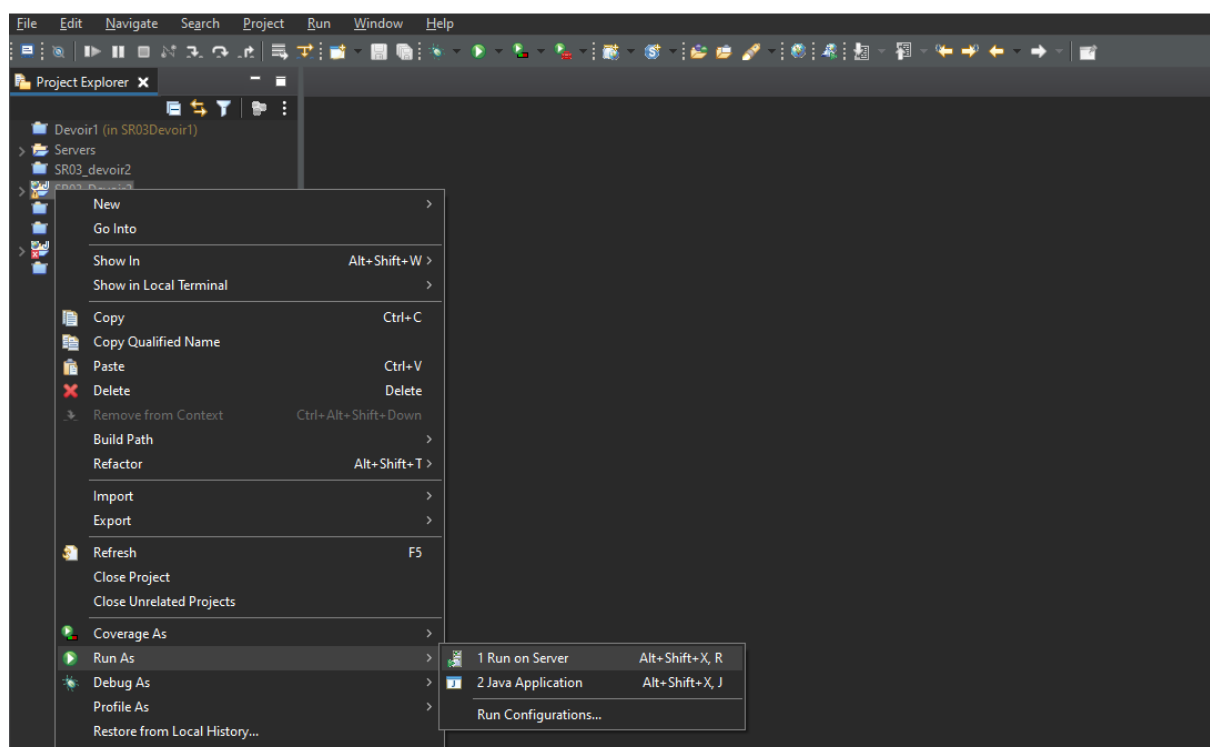
## 3.2 TESTS ET SCÉNARIOS

Pour tester l'application, tout d'abord vérifiez les versions de votre environnement de travail (cf. II.A.).

- 1) Lancez XAMPP ou autre logiciel similaire (gardez le à proximité, on risque d'en avoir besoin pour le test de l'application) et démarrez MySQL
- 2) Dans un navigateur web, entrez "localhost/phpmyadmin/index.php", cliquez sur "Importer" en haut à droite, et importez le fichier sr03-devoir2.sql"



- 3) Lancez Eclipse, ouvrez le projet "sr03\_devoir2v2" que vous aurez téléchargé et décompressé depuis le dépôt Git (lien : [ici](#)). Clic droit sur le projet -> Run as -> Run on Server (Tomcat v8.0)





- 4) Entrez l'URL "<http://localhost:8080/SR03-Devoir2/connexion.html>" dans un navigateur web, puis connectez vous avec un compte préalablement créé par un admin dans la base de données
- 5) Prêt? Feu, testez!