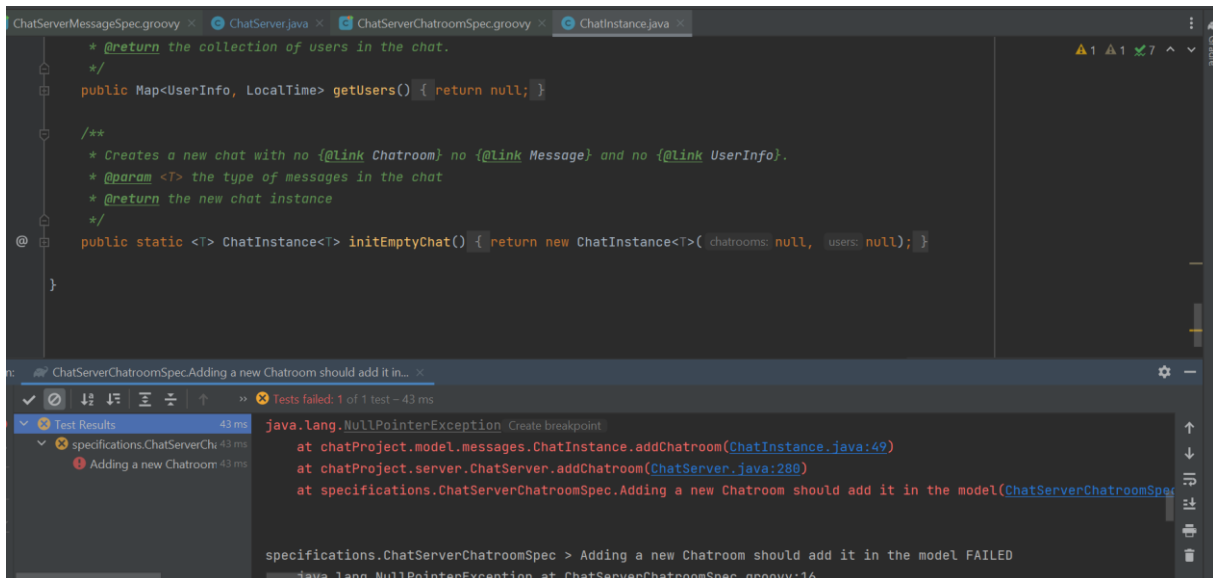
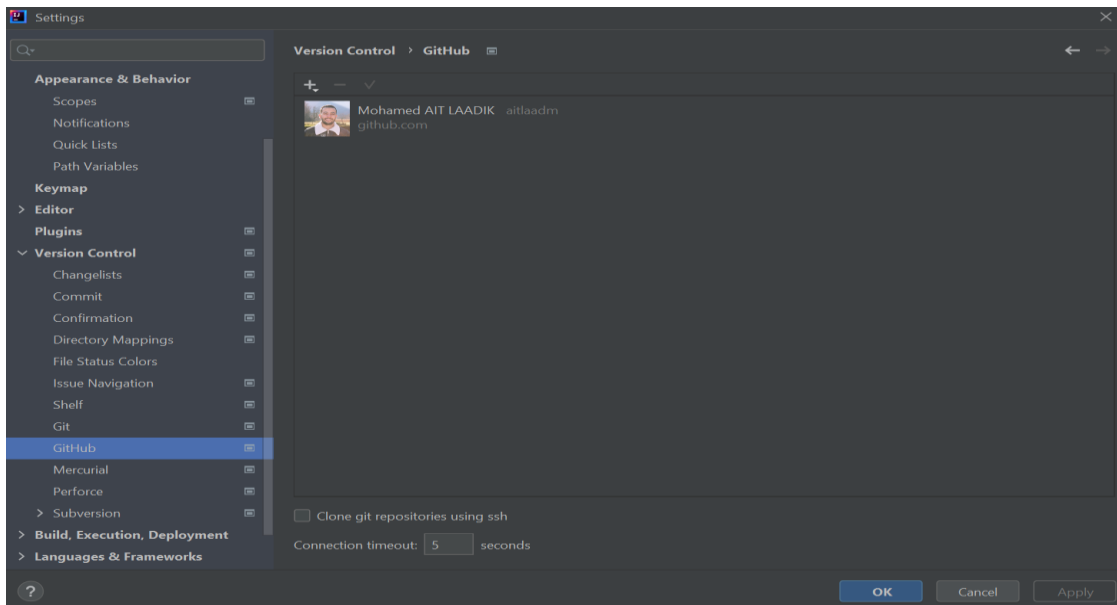


Mohamed AIT LAADIK & Mouhcine CHAOUSARY

Rapport du TP de CI (Continuous integration) intégration continue : l'Application SimpleChat

I- améliorations de la qualité de code

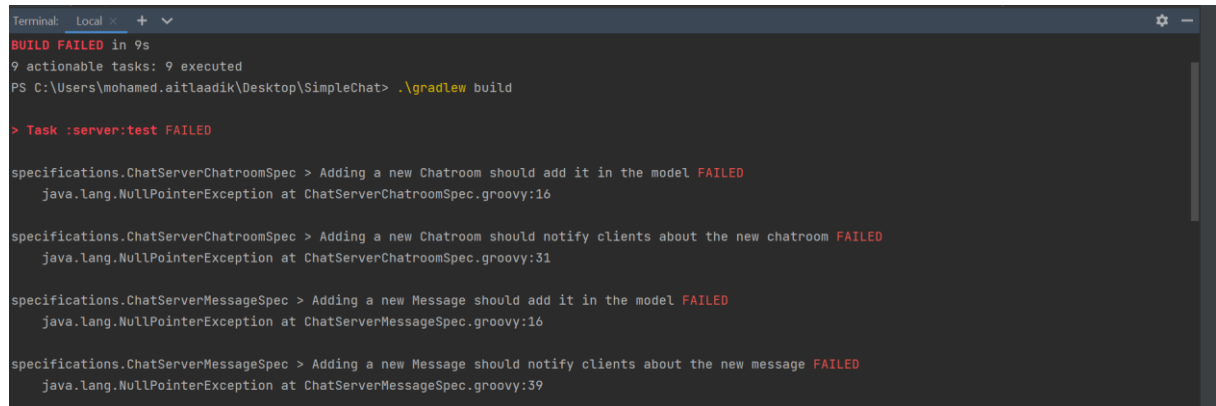
Après la récupération du code et l'installation de l'IDE IntelliJ. On a pu lier l'ide avec notre compte Github ainsi d'exécuter un premier build du projet Gradle, mais le build a échoué au premier. (Les captures ci-dessous). Des améliorations de la qualité de code ont été mis en place.



Ces erreurs étaient majoritairement liées au mauvais fonctionnement de certaines méthodes du programme. Après la modification de ces méthodes le build se termine avec succès. Prenons exemple la fonction `initEmptyChat`, à la base les paramètres de l'instanciation de la `chatInstance`

étaient « null » d'où vient l'erreur. Donc on a ajouté les bons paramètres à l'instance qui prend une List de chatroom et un HashMap d'utilisateurs (figure ci-dessus).

Après ces modifications, les tests ne marchaient toujours pas (capture ci-joint) on a dû modifier quelques fonctions afin de tourner correctement le programme.



```
Terminal: Local + -
BUILD FAILED in 9s
9 actionable tasks: 9 executed
PS C:\Users\mohamed.aitlaadik\Desktop\SimpleChat> .\gradlew build

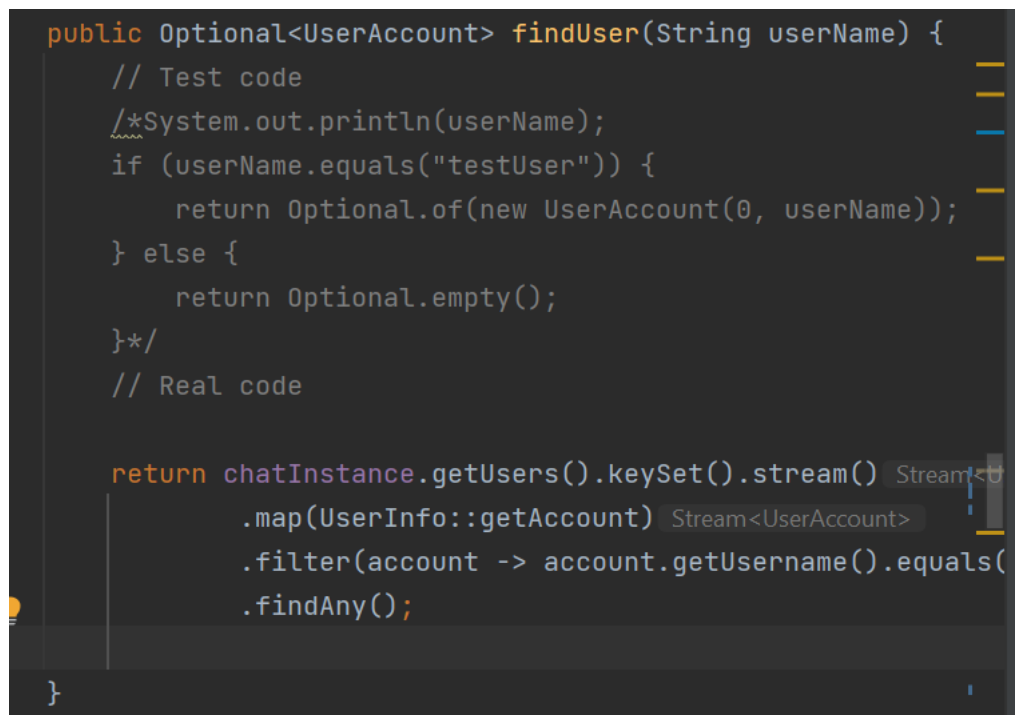
> Task :server:test FAILED

specifications.ChatServerChatroomSpec > Adding a new Chatroom should add it in the model FAILED
    java.lang.NullPointerException at ChatServerChatroomSpec.groovy:16

specifications.ChatServerChatroomSpec > Adding a new Chatroom should notify clients about the new chatroom FAILED
    java.lang.NullPointerException at ChatServerChatroomSpec.groovy:31

specifications.ChatServerMessageSpec > Adding a new Message should add it in the model FAILED
    java.lang.NullPointerException at ChatServerMessageSpec.groovy:16

specifications.ChatServerMessageSpec > Adding a new Message should notify clients about the new message FAILED
    java.lang.NullPointerException at ChatServerMessageSpec.groovy:39
```



```
public Optional<UserAccount> findUser(String userName) {
    // Test code
    /*System.out.println(userName);
    if (userName.equals("testUser")) {
        return Optional.of(new UserAccount(0, userName));
    } else {
        return Optional.empty();
    }*/
    // Real code

    return chatInstance.getUsers().keySet().stream()
        .map(UserInfo::getAccount)
        .filter(account -> account.getUsername().equals(userName))
        .findAny();
}
```

(Décommenter le « real code » et commenter le « fake code »).

```

// open a dedicated thread to manage the socket for noti
final Thread socketThread = new Thread(new Runnable() {
    @Override
    public void run() {
        try {
            server.openSocket(socketPort);
        } catch (IOException e) {
            throw new RuntimeException("Unable to open")
        }
    }
});
server.socketThread = socketThread;

//TODO: I should start the socket thread here
server.socketThread.start();
server.checkIdleClients();
return server;
}

```

Le fait de ne pas lancer le socket thread à entrainer le crash de l'application à chaque lancement du client. Donc fallait ajouter la ligne `server.socketThread.start()` .

Aussi la fonction « getUsers » de la classe ChatServer retournait un « null » ce qui empêchait la bonne exécution de l'application.

```

/**
 * Gets the list of all registered users.
 * @return the collection of users in the chat.
 */
public Map<UserInfo, LocalTime> getUsers() {
    return users;
}

```

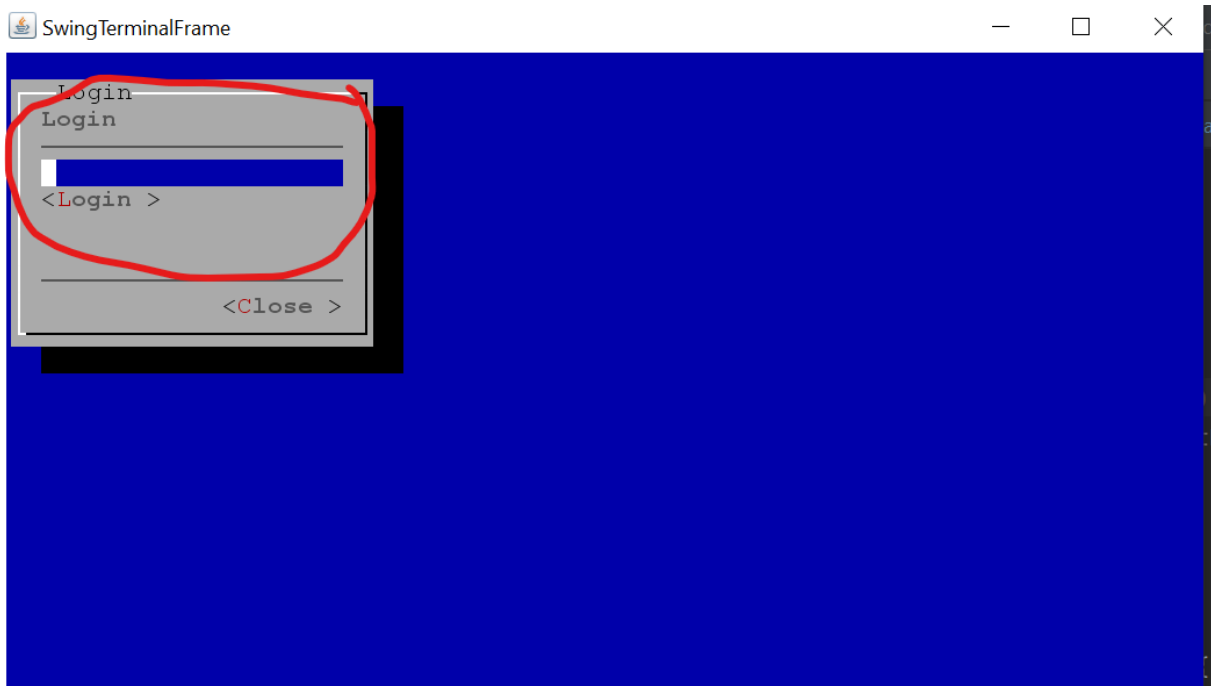
Pour cela fallait retourner les « utilisateurs » au lieu du « null » qui est un Map (API collection). Et finalement le build s'est terminé avec succès.

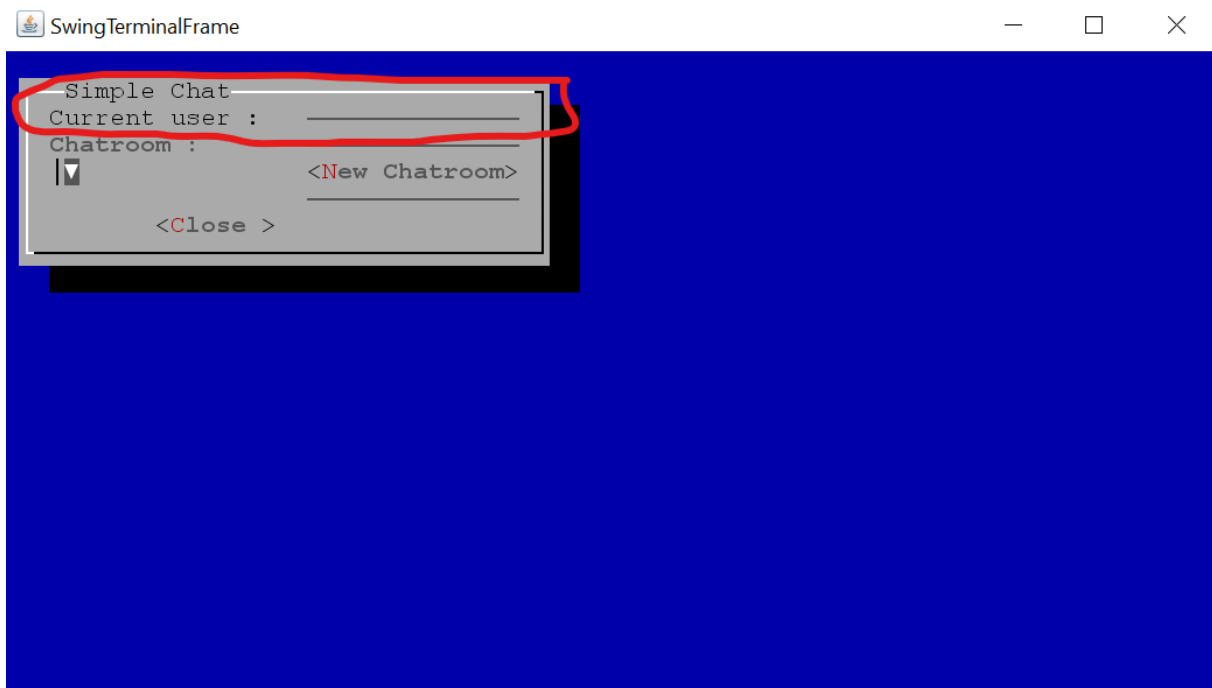
```
PS C:\Users\mohamed.aitlaadik\Desktop\SimpleChat> .\gradlew build

BUILD SUCCESSFUL in 3s
18 actionable tasks: 11 executed, 7 up-to-date
PS C:\Users\mohamed.aitlaadik\Desktop\SimpleChat> |
```

II-Les scénarios de tests réalisés à la main

La première chose qu'on a testé, c'est de se logger sans psuedo (cliquer directement sur login sans écrire son psuedo dans la barre ». ça a marché mais y'avait pas de psuedo pour cet utilisateur ! (figure ci-dessous) :



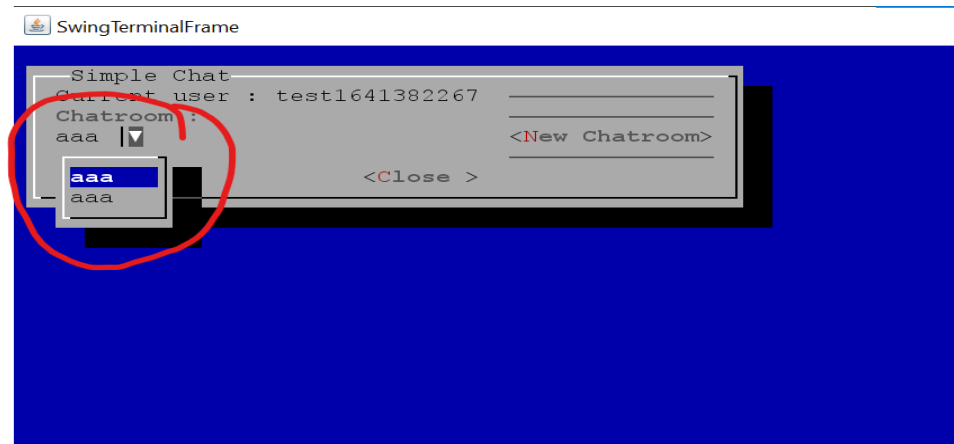


Ce bug a été résolu en modifiant la fonction getUsername de la classe UserAccount comme suit :

```
public String getUsername() {
    if(this.username.equals("")){
        this.username="test"+bootNum;
    }
    return username;
}
```

Avec bootNum un numéro générer aléatoirement grâce à la classe ThreadLocalRandom.

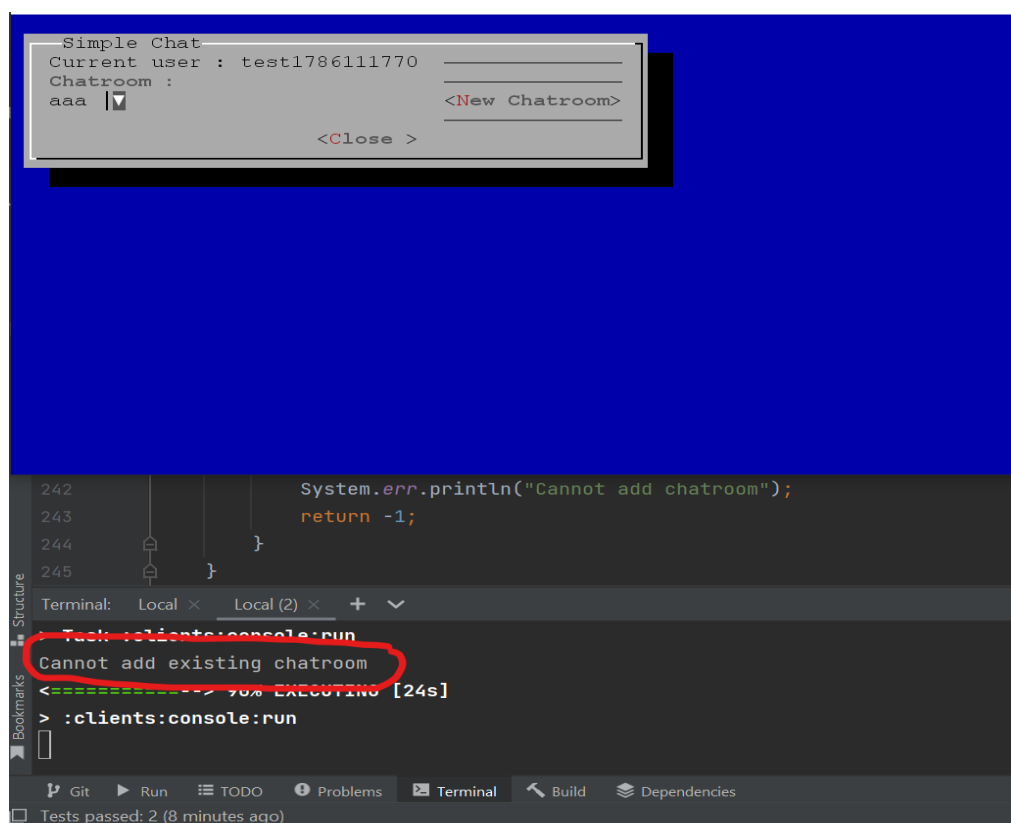
Ensuite, on a testé si on peut créer une chatroom avec le même nom d'une chatroom existante et effectivement le programme ne gérait pas cette erreur (capture ci-dessous) :



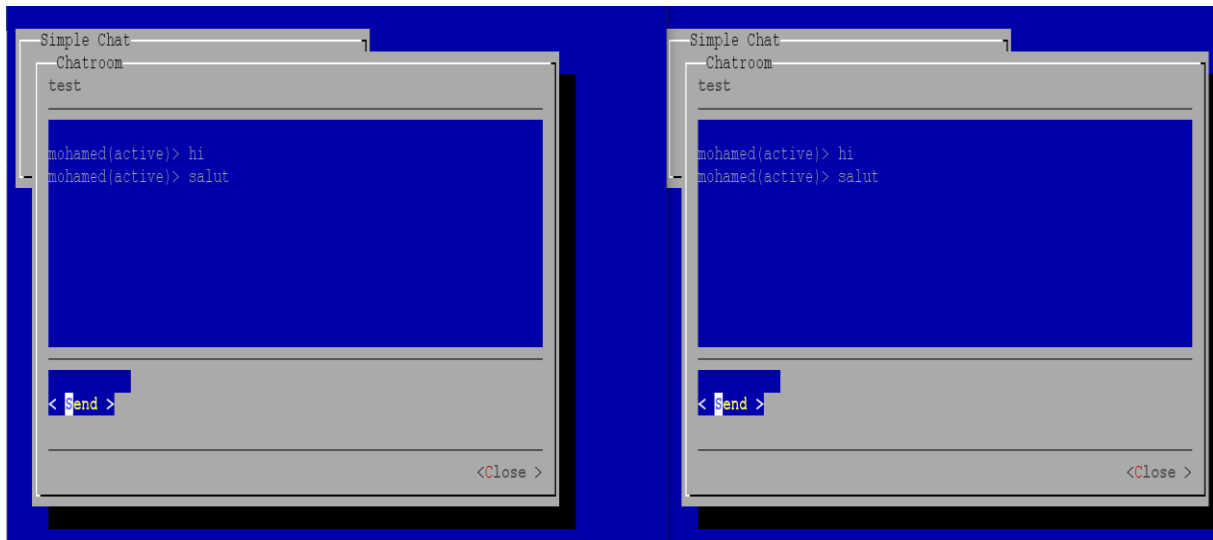
Pour cela la fonction à été modifié afin de gérer cette erreur. En ajoutant une condition sur le chatroomName passé en paramètre afin de le comparer avec les noms de chatroom déjà existant et

envoyer un message pour notifier l'utilisateur comme quoi le chatroomName qu'il a choisi est déjà utilisé si son chatroomName est effectivement déjà prit.

```
@Override
public int addChatroom(String chatroomName, UserInfo owner) {
    try {
        if(getCurrentChatroomNames().contains(chatroomName)){
            System.err.println("Cannot add existing chatroom");
            return -1;
        }else{
            final String response = Request.Put(serverUrl + "/chatroom/" + chatroomName).bodyString(
                json.toJson(owner),
                ContentType.APPLICATION_JSON
            ).execute().returnContent().asString();
            return json.fromJson(response, Integer.class);
        }
    } catch (IOException e) {
        System.err.println("Cannot add chatroom");
        return -1;
    }
}
```



Dernièrement, on a tester si plusieurs clients peuvent se connecter simultanément sur le réseau et de créer, rejoindre des chatroom, on a vérifié la bonne exécution de ces taches qui fonctionnent bien finalement sauf l'envoi des messages dans une chatroom ne fonctionnais pas comme il le faut puisque le programme affiche un seul nom d'utilisateur pour tous les messages envoyés (figure ci-dessous)



Le problème a été trouvé au niveau de la fonction login, cette fonction donne un seul et unique id « 0 » pour tout les utilisateurs.

```
/**
 * {@inheritDoc}
 */
@Override
public UserInfo login(String userName) {
    //idIncr+=1;
    final UserInfo user = new UserInfo(
        findUser(userName).orElse(new UserAccount( id: 0, userName)),
        Status.ACTIVE // user just logged in - status is active
    );
    notifyUserChange(user);

    return user;
}
```

Après le remplacement de l'id par une variable initialisé à 0 et qu'on incrémente à chaque login d'utilisateur afin de lui affecter un id différent la fonctionnalité s'exécute correctement et chaque message envoyé est référencé par son émetteur.

```
@Override
public UserInfo login(String userName) {
    idIncr++;
    final UserInfo user = new UserInfo(
        findUser(userName).orElse(new UserAccount(idIncr, userName)),
        Status.ACTIVE // user just logged in - status is active
    );
    notifyUserChange(user);

    return user;
}
```

