GitHub : https://github.com/insa-4ir-chatsystem/chatsystem-bonnet-cazeneuve
Bonnet Ronan
Cazeneuve Anna
4 IR B2
2023/2024

Architecture du Système de Clavardage: Diagrammes UML

1. Introduction

Ce document a pour but de rendre compte des choix architecturaux et de conception de notre application dans le cadre du projet chatsystem. L'objectif de ce projet était de développer une première version d' une application de clavardage dans un réseau d'entreprise. Il s'agit de permettre à des personnes d'un même réseau local de communiquer un à un par le biais de messages textuels.

Ce projet a été découpé en deux parties : la première dans laquelle il était question de la partie "Découverte des contacts", et la deuxième partie au cours de laquelle les fonctionnalités de messagerie étaient rajoutées. Pour chacune des deux parties, l'architecture de l'application a été pensée et modélisée avec notamment des diagrammes UML.

Nous tâcherons de présenter dans ce dossier ces diagrammes et choix architecturaux selon le plan suivant :

- Acteurs identifiés et hypothèses
- Diagrammes de cas d'utilisation
- Diagrammes de classe
- Diagrammes de séquence
- Architecture de la base de données
- Structure générale de l'application

2. Acteurs et hypothèses

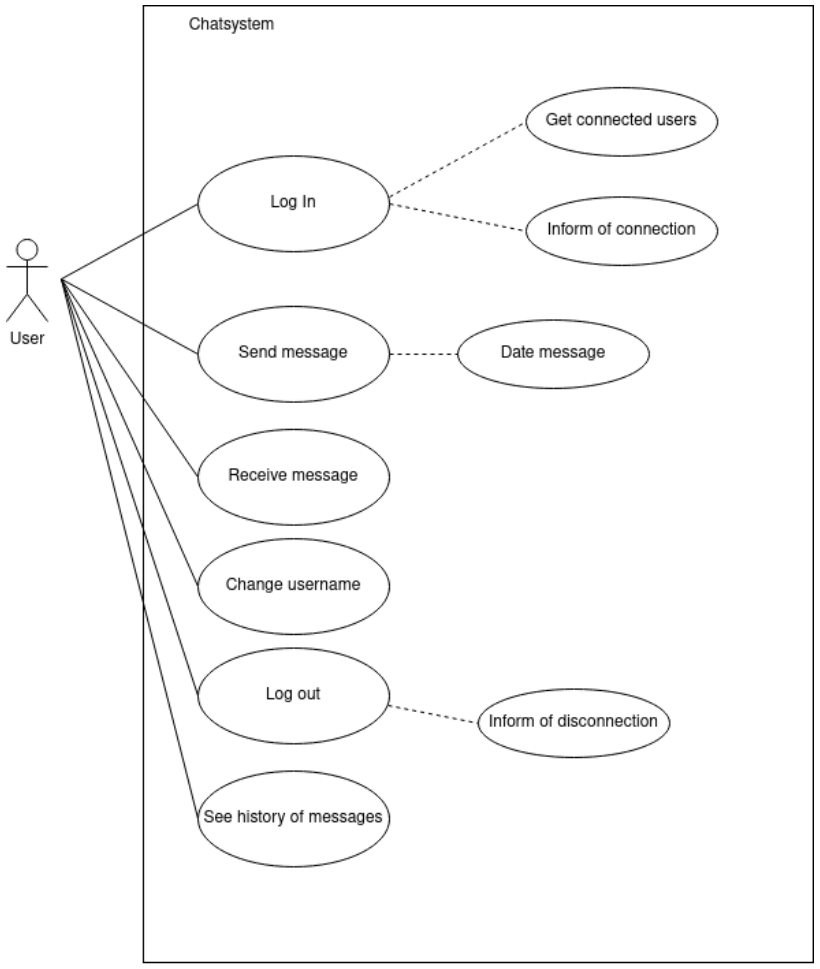
Les acteurs que nous avons identifiés sont les suivants:

- utilisateurs

Le projet étant une version préliminaire du produit final (une maquette), nous avons fait quelques hypothèses de travail pour pouvoir présenter un résultat dans le temps imparti:

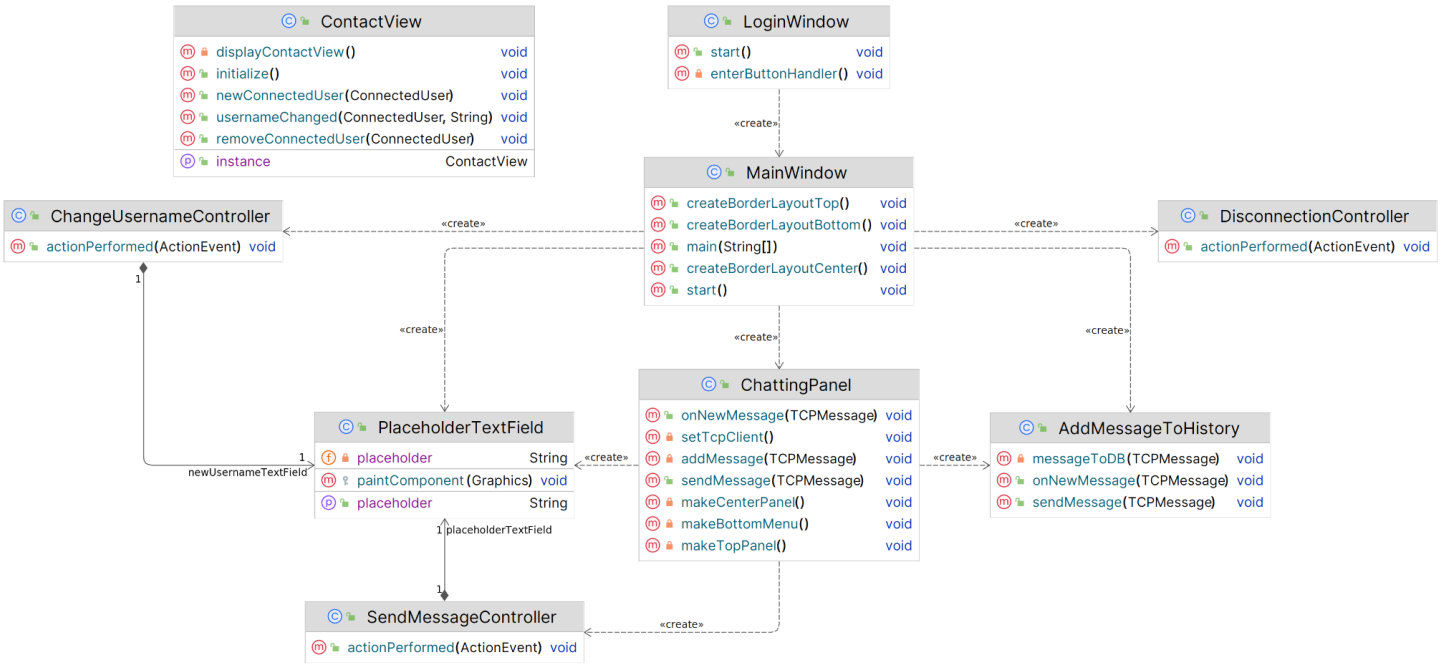
- les utilisateurs utilisent tous le système d'exploitation Linux ou Windows
- les utilisateurs n'ont pas à s'identifier avec un mot de passe (pas de confidentialité)
- les utilisateurs ne s'envoient (pour notre implémentation à ce jour) que des messages textuels
- les utilisateurs sont tous sur un même réseau d'entreprise
- une machine est toujours utilisée par un seul utilisateur à la fois

3. Diagrammes de cas d'utilisation

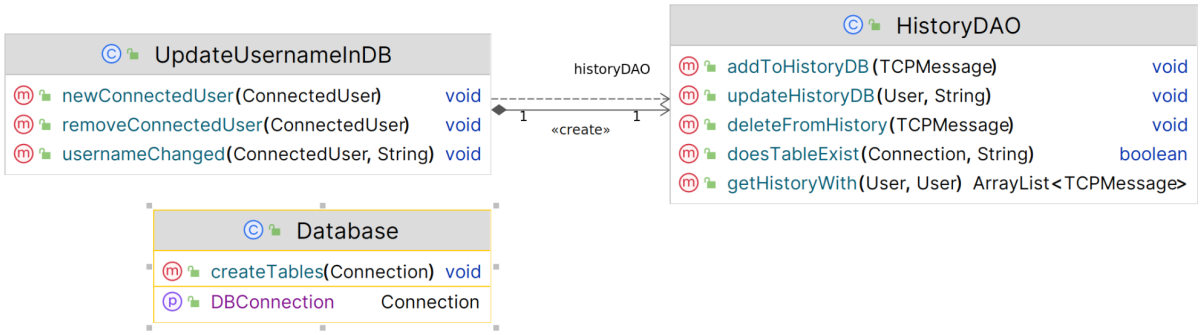


4. Diagrammes de classe

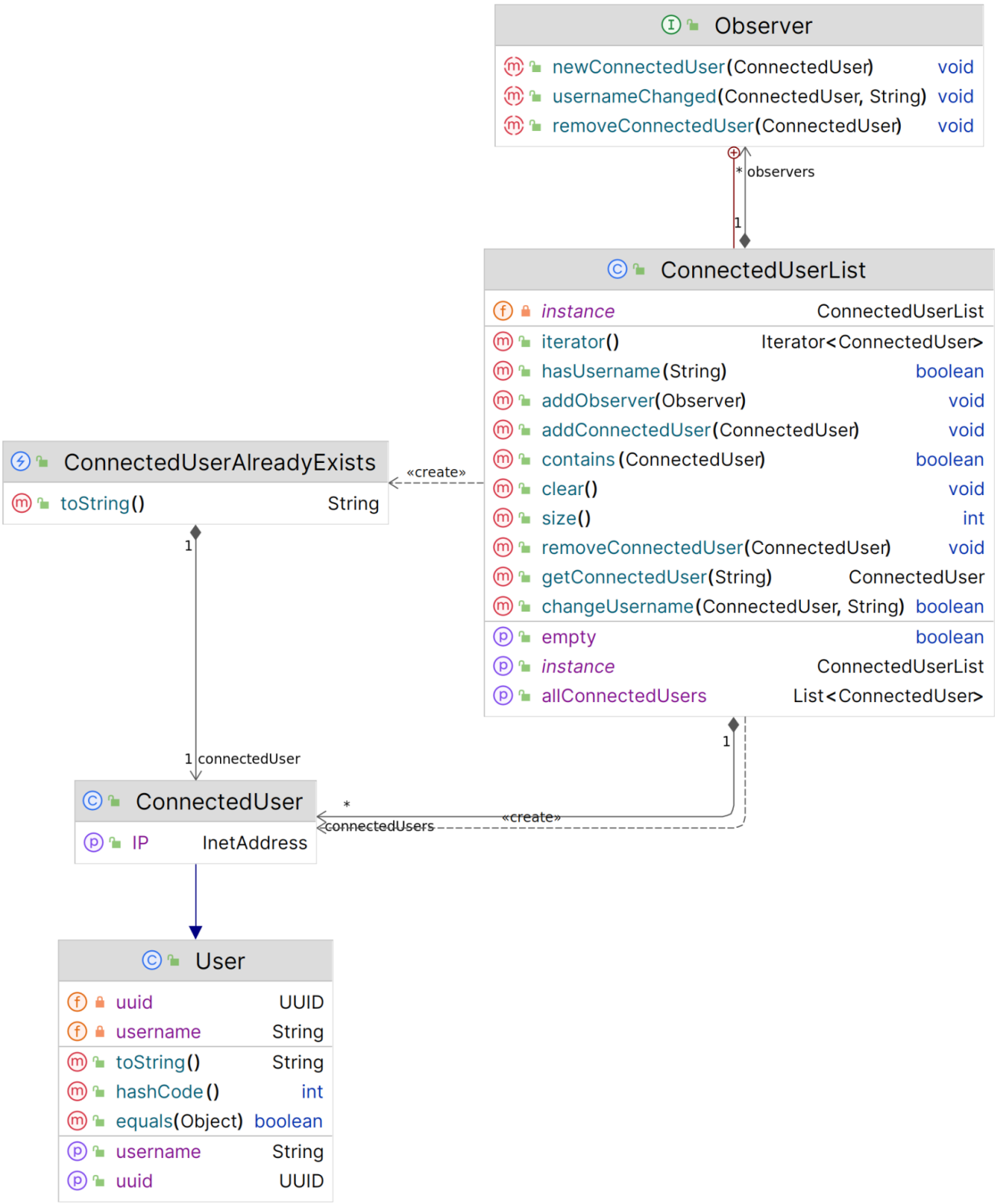
Package GUI



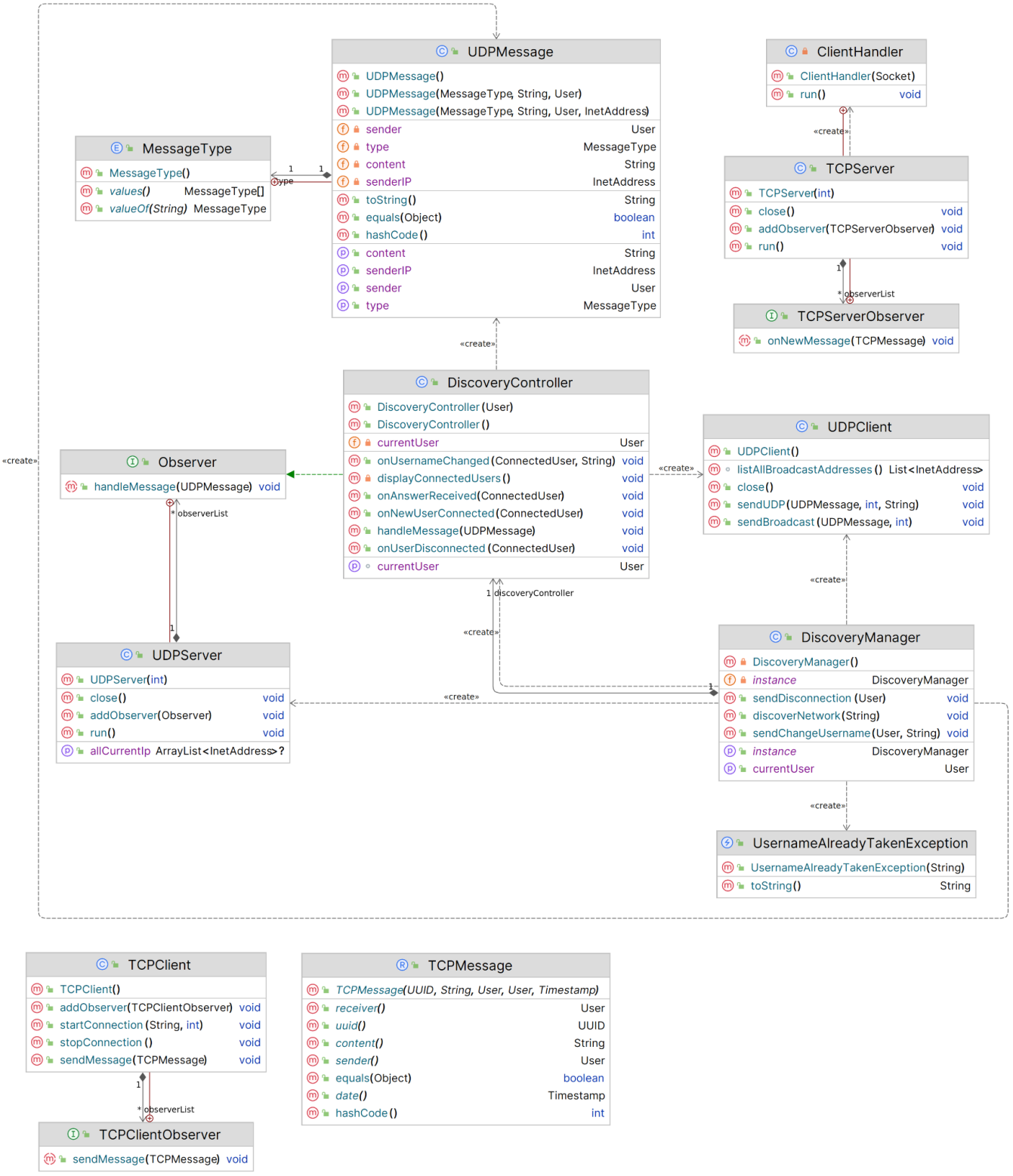
Package Database



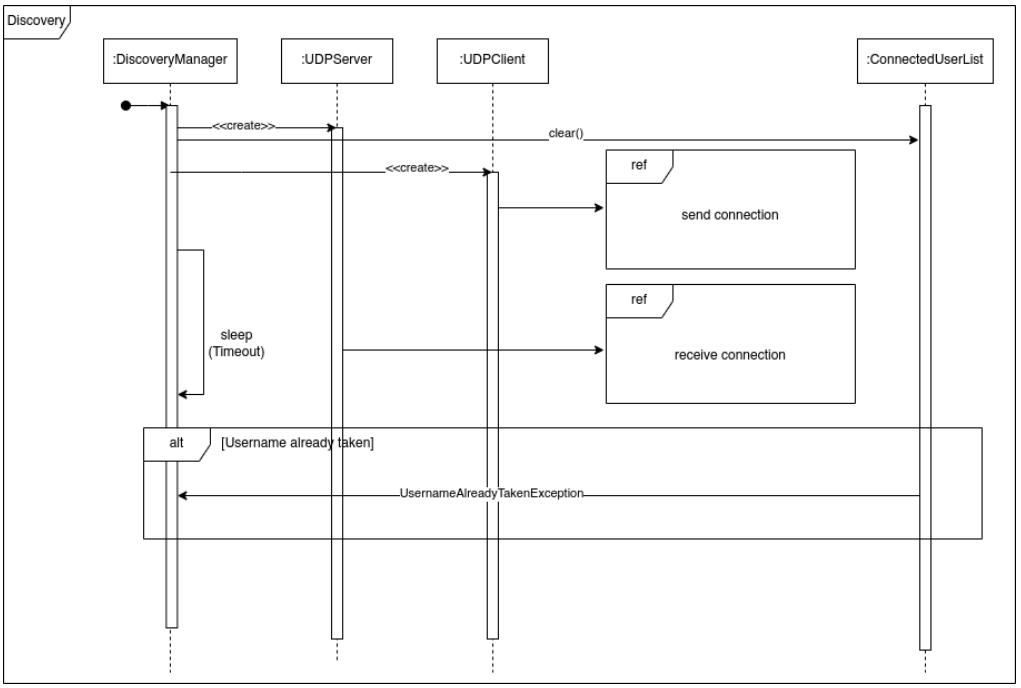
Package users

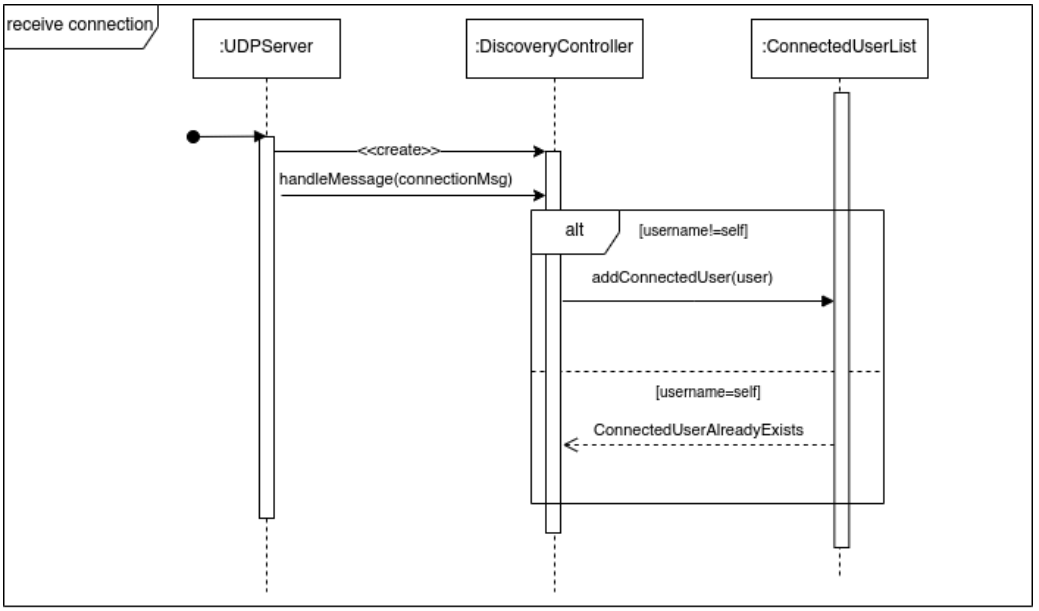


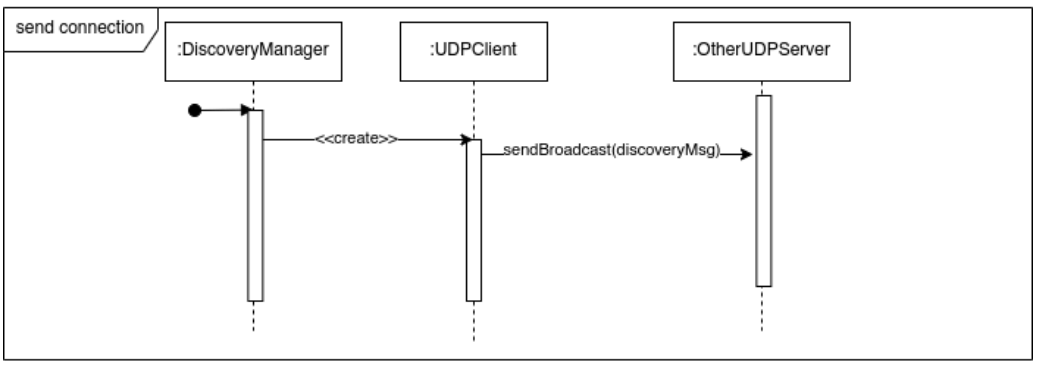
Package network

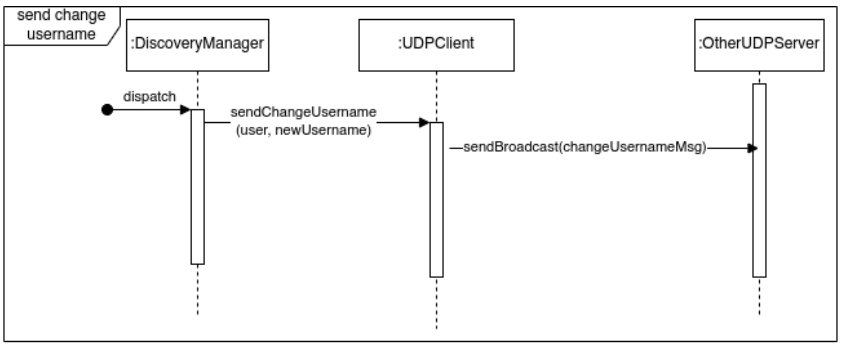


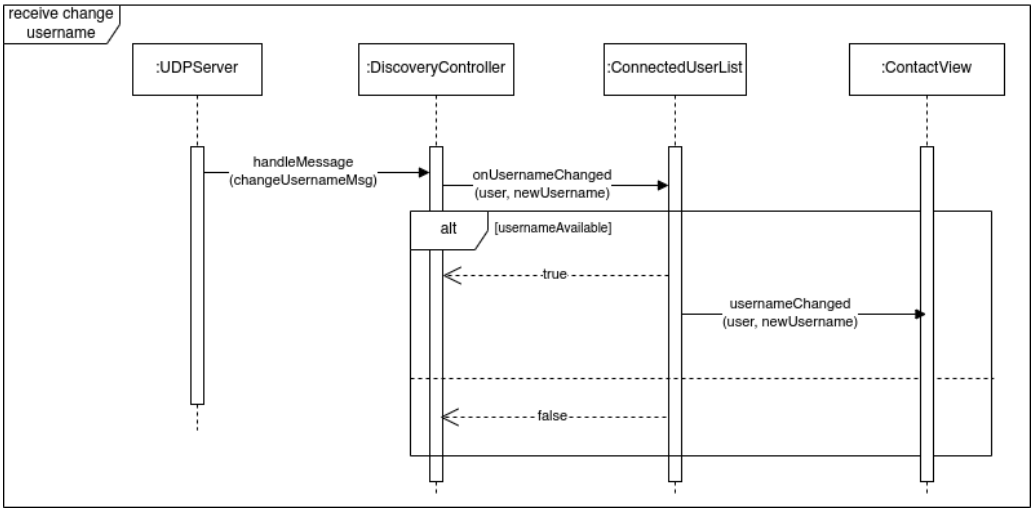
5. Diagrammes de séquence

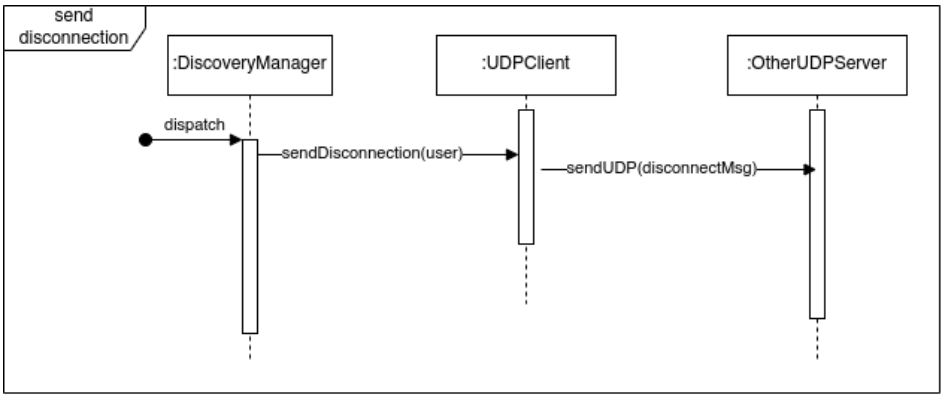


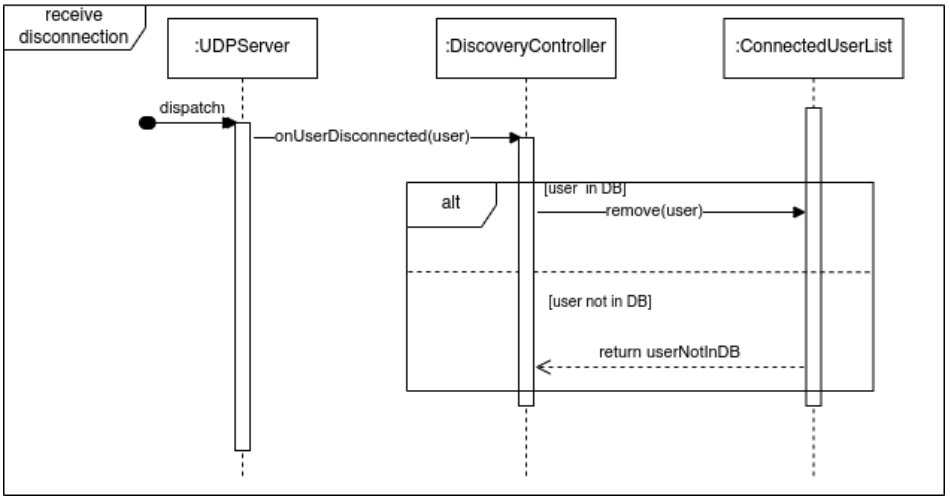




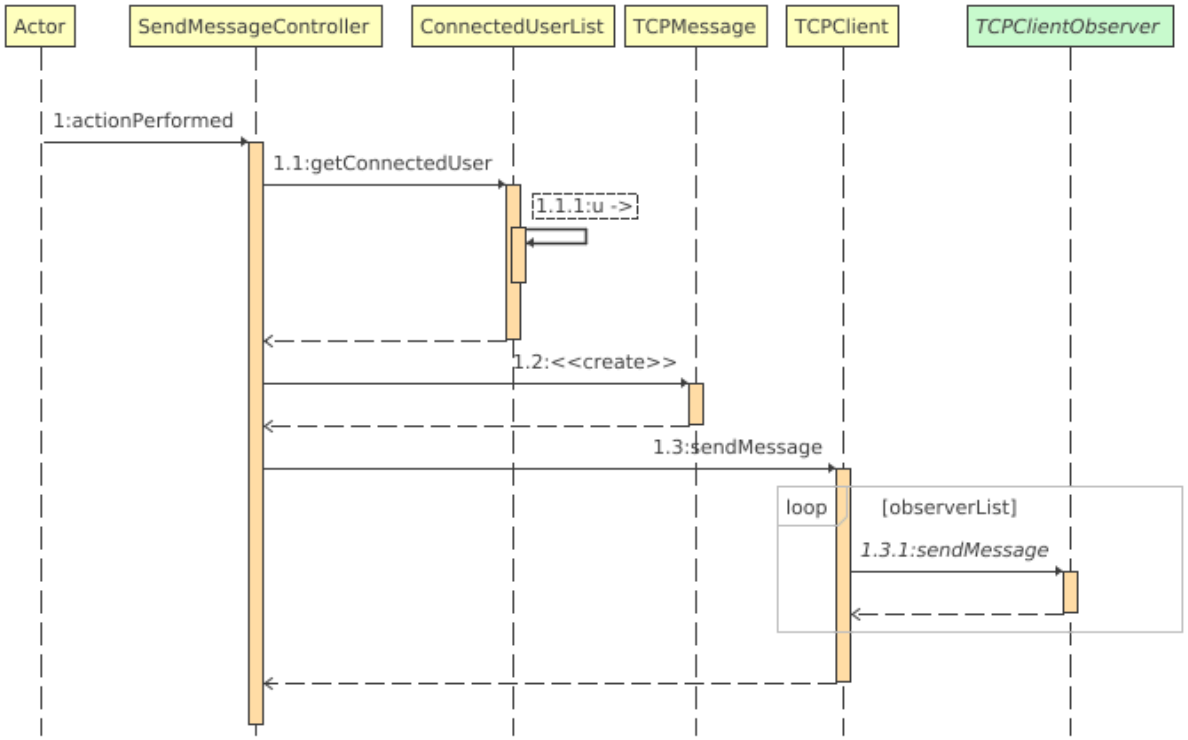

send change username
:DiscoveryManager
:UDPClient
:OtherUDPServer
dispatch
sendChangeUsername
(user, newUsername)
sendBroadcast(changeUsernameMsg)


receive change username
:UDPServer
:DiscoveryController
:ConnectedUserList
:ContactView
handleMessage
(changeUsernameMsg)
onUsernameChanged
(user, newUsername)
alt
[usernameAvailable]
true
usernameChanged
(user, newUsername)
false


send disconnection
:DiscoveryManager
:UDPClient
:OtherUDPServer
dispatch
sendDisconnection(user)
sendUDP(disconnectMsg)


receive disconnection
:UDPServer
:DiscoveryController
:ConnectedUserList
dispatch
onUserDisconnected(user)
alt
[user in DB]
remove(user)
[user not in DB]
return userNotInDB

Send message to other user



6. Architecture de la base de données

Nous avons utilisé une base de données SQLite pour garder en mémoire les conversations précédentes ; il s'agit d'un historique des messages.
Notre base de données se présente ainsi:

uuid	content	date	sender_username	receiver_username
VARCHAR(36) not null	TEXT not null	DATETIME	VARCHAR(200) not null	VARCHAR(200) not null

Les messages y sont enregistrés dans l'ordre (par *date* croissant) et avec un identifiant unique (*uuid*). On utilise le type TEXT pour le contenu du message pour avoir la possibilité, dans un second temps, d'ajouter des fonctionnalités nécessitant l'envoi de plus de données ; par exemple, l'envoi de fichiers.

7. Structure générale de l’application