UML Report

Jules-Ian BARNAVON

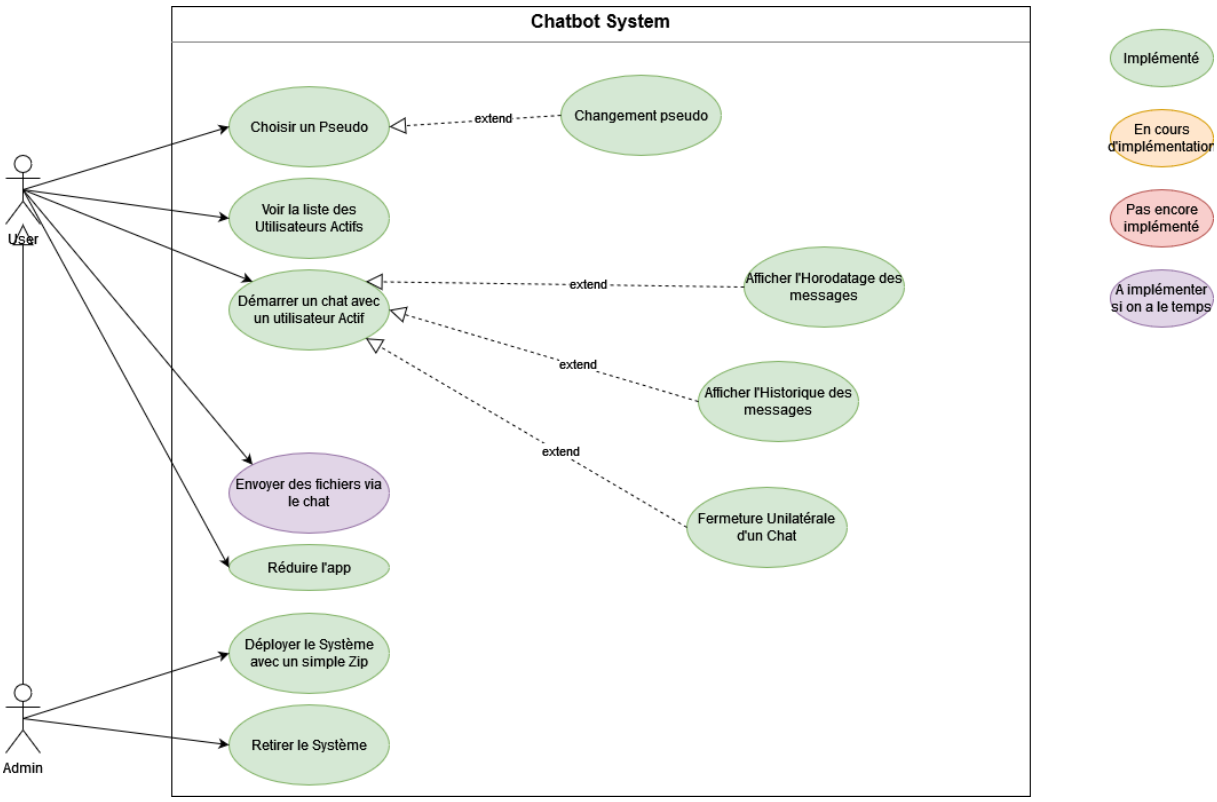
Matis RAMARA

Chatsystem UML Part

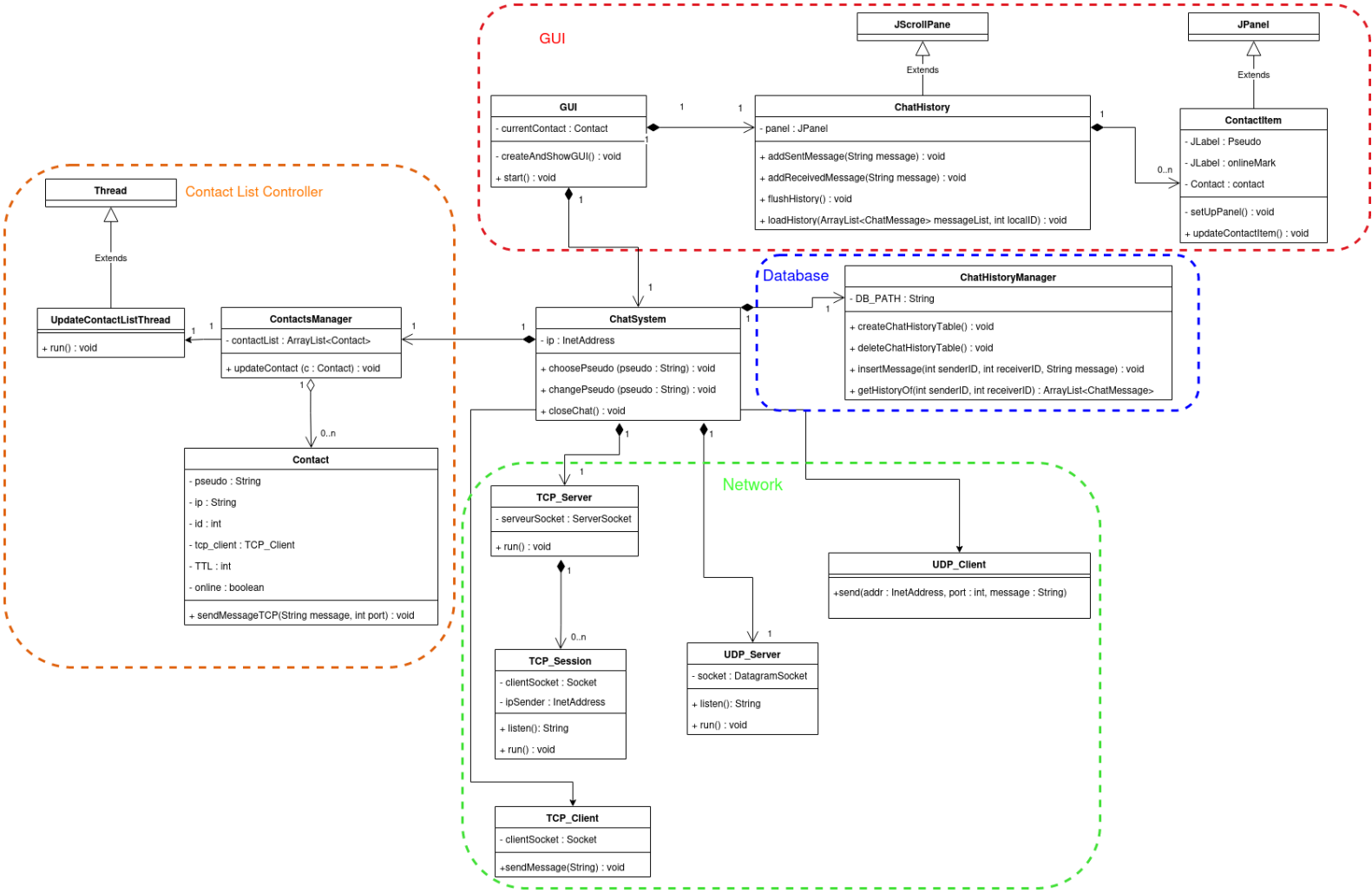
I.Actors and assumptions

Les acteurs sont un admin pour déployer le système sur des machines linux et des utilisateurs sur un même réseau local

II. Use case diagram

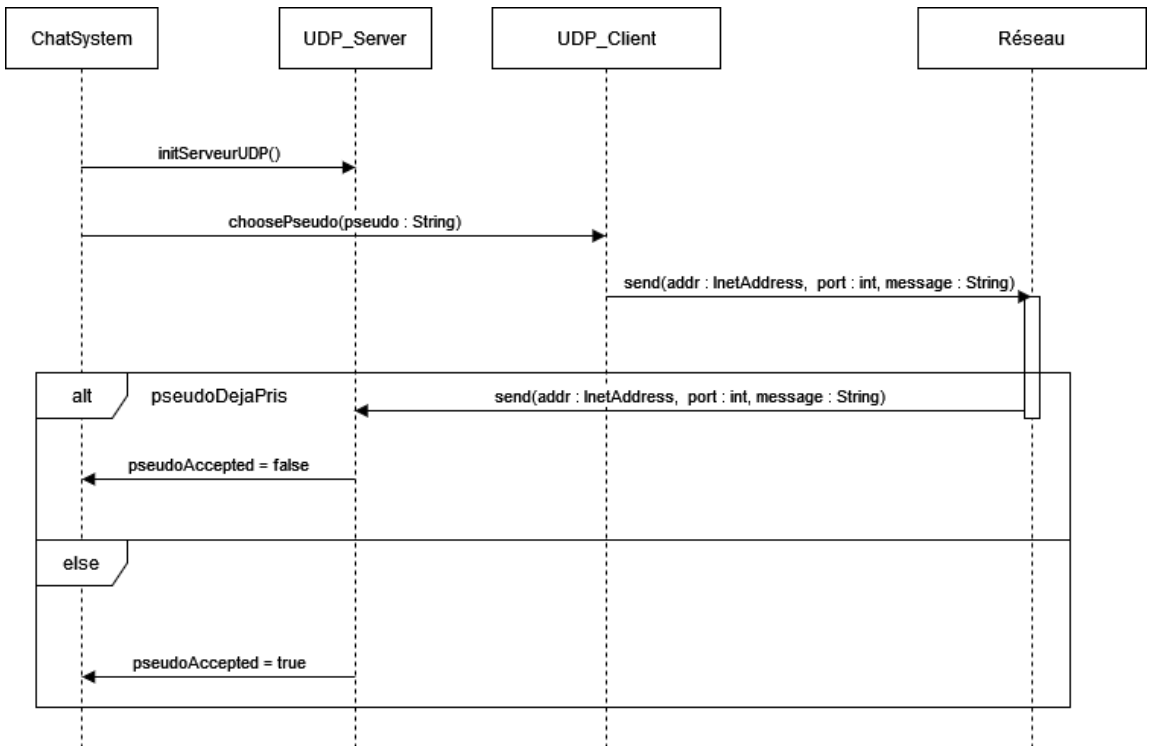


III. Class diagram

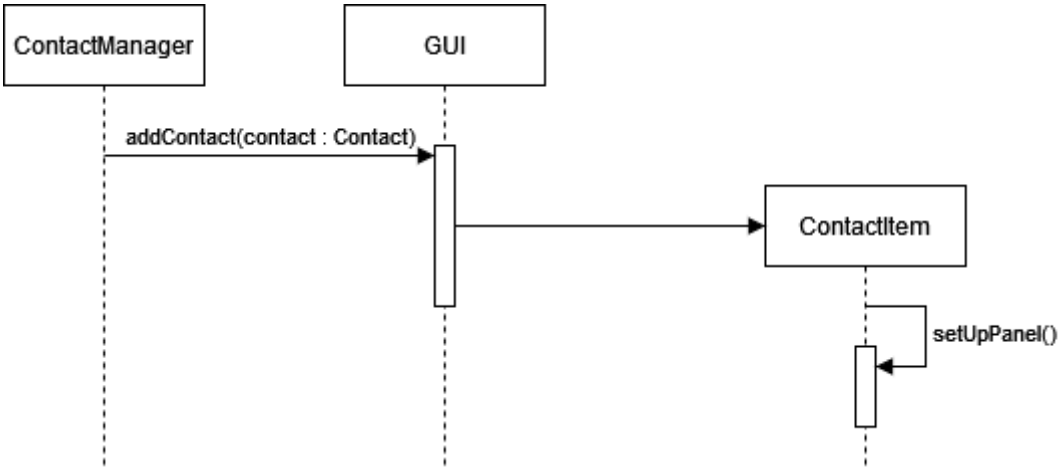


IV. Sequence diagrams

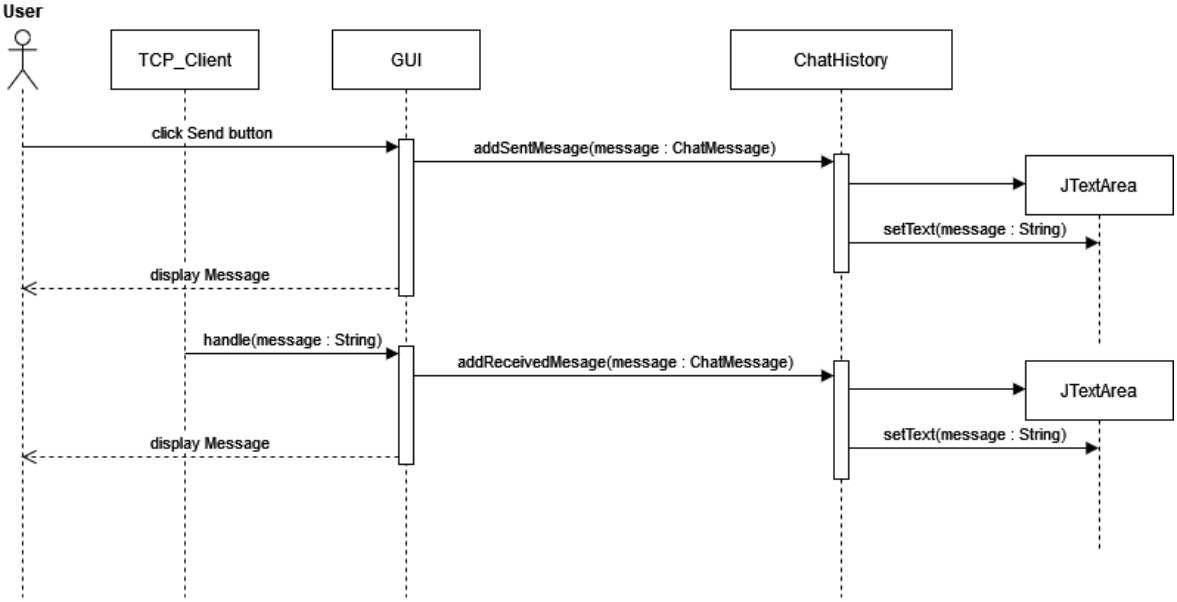
Sequence diagram for choosing a pseudo :



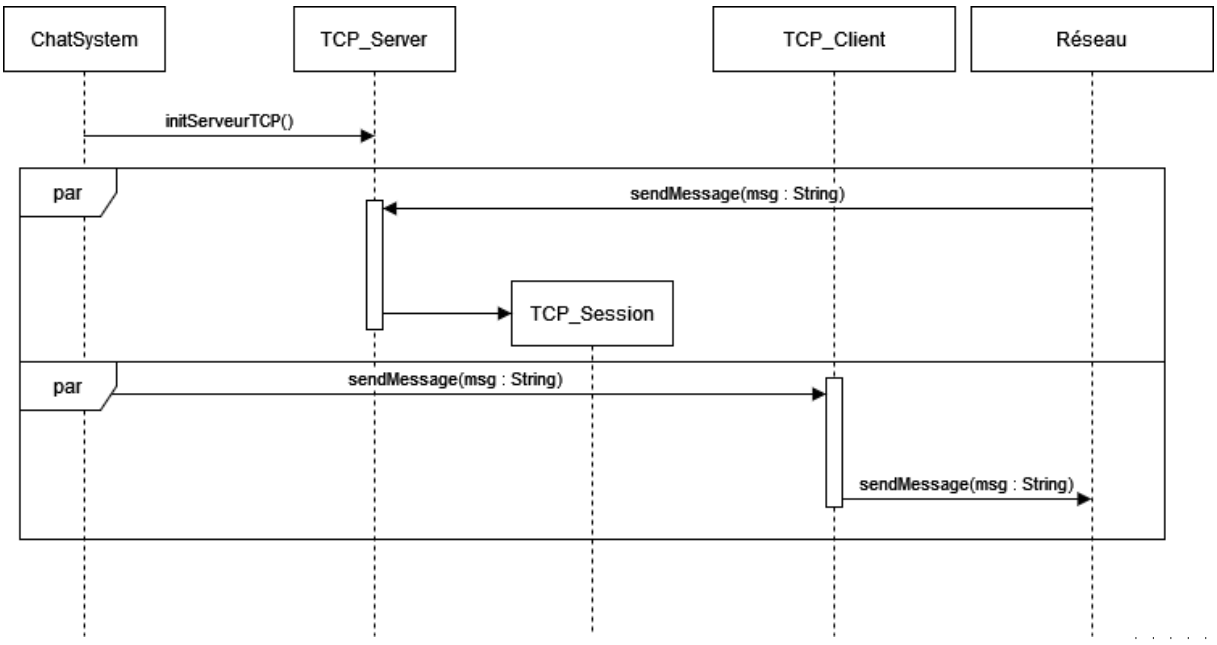
Sequence diagram for creating a Contact in the contact list :



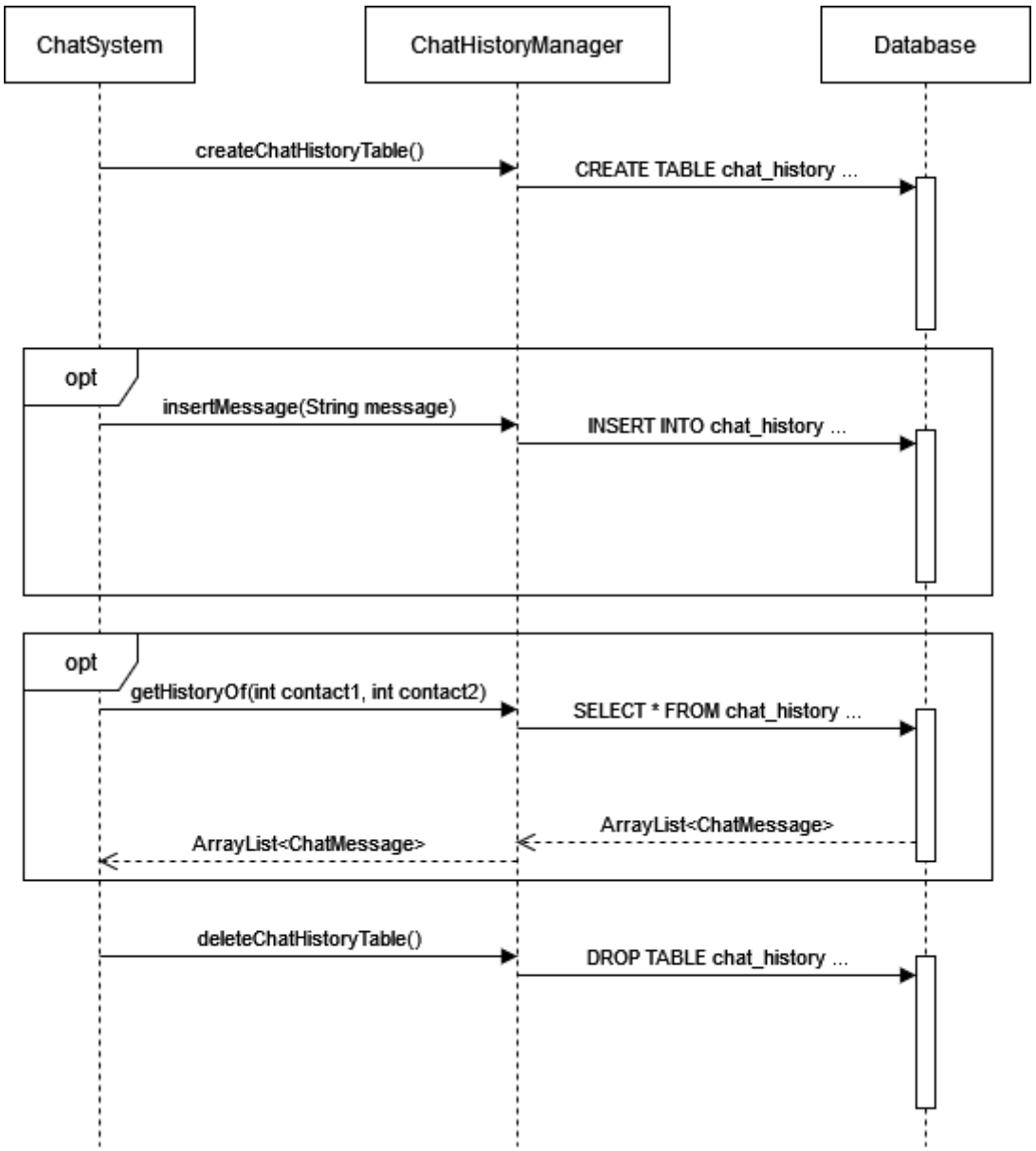
Sequence diagram for adding a message in the chat history :



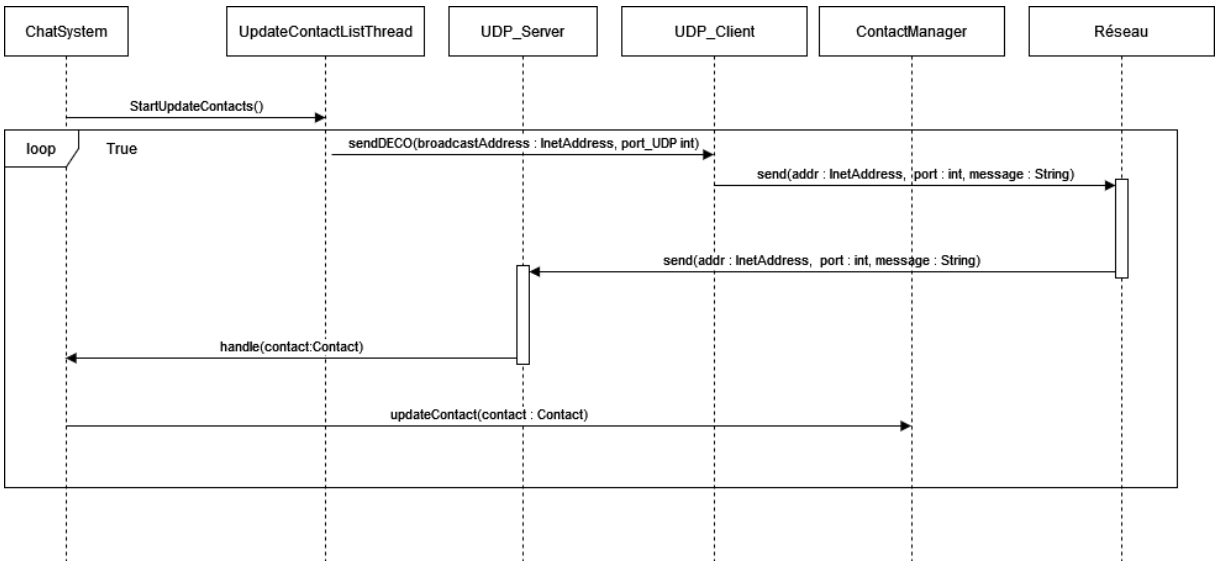
Sequence diagram of exchanged messages with TCP :



Sequence diagram of interactions with the database :



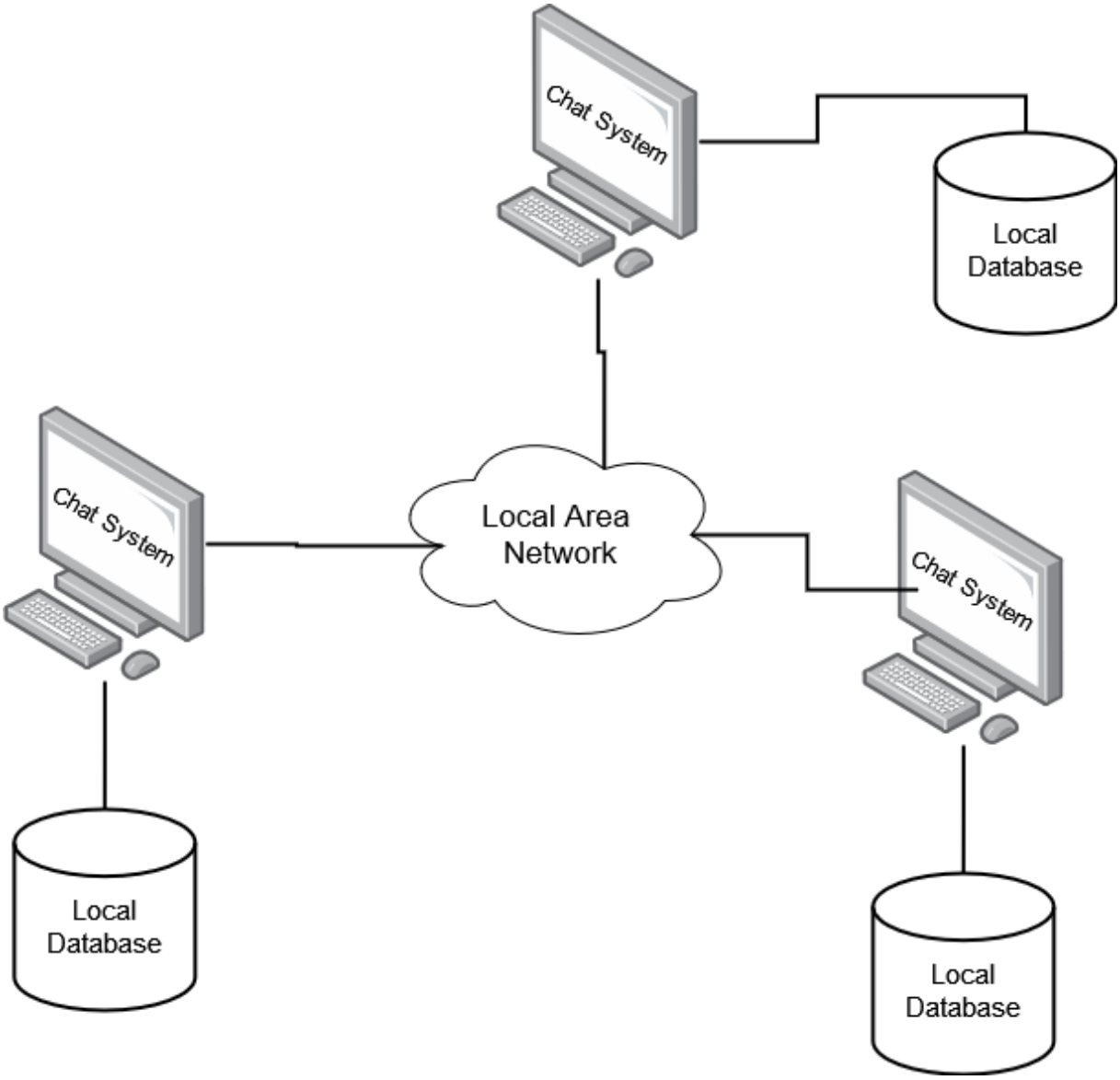
Sequence diagram for updating the contact list :



V. Database schema

chat_history	
PK	id INT
	senderId INT
	receiverId INT
	message TEXT
	timestamp DATETIME

VI. Architecture overview



PDLA

Nous avons commencé le projet en faisant un “fork” depuis le dépôt template-project puis en clonant le dépôt dans un dossier local à nos ordinateurs. Nous avons créé un projet Jira sur le site web Atlassian afin d’appliquer la méthode agile avec le scrum.

Pour compiler le projet nous avons utilisé le logiciel Maven, la première étape est de créer un fichier pom.xml où l’on pourra tout au long du projet ajouter les dépendances et les plugins.

Pour compiler le projet il faut exécuter la commande : mvn compile package

Pour le lancer après compilation : java -jar target/ChatSystem-1.0-jar-with-dependencies.jar

Il est possible d’automatiser le processus de compilation et de test à chaque push sur le dépôt avec la fonctionnalité Github Actions. Pour cela il faut remplir un fichier YAML dans le répertoire .github/workflows avec la configuration que l’on veut (version du JDK, commande Maven à exécuter…). C’est ce que nous avons fait assez rapidement dans le développement du projet car cela permet de gagner une quantité non négligeable de temps sur les tests.

Conduite de projet

Pour organiser ce projet, nous avons appliqué la méthode agile en utilisant Jira. Jira a permis une planification efficace de 5 sprints, une définition claire des objectifs et la création de tableaux de bord visuels pour suivre la progression du travail. Cette application nous a aussi permis de pouvoir s’adapter au cours du projet, notamment lorsque nous avons constaté qu’il allait falloir mettre notre projet en réseau et pas uniquement simuler plusieurs machines sur notre adresse locale. C’était une précision apportée par notre professeur mais pour nos projets futurs cela aurait pu être un client qui aurait demandé une spécificité en plus pendant le développement.

Concernant notre manière de travailler en collaboration, nous avons toujours travaillé sur une seule branche git, la branche principale. Pour travailler en simultané nous avons utilisé la fonctionnalité “Share with me” d’IntelliJ qui nous permet de travailler en même temps sur le même projet en ayant les modifications apportées par l’autre instantanément. Quand nous ne travaillons pas en même temps, un simple “push“ après avoir apporté des modifications suffisait.