UML report

Achraf Bensebaa, Sara Bakri

January 2024

1 Introduction

2 Use Case Diagram

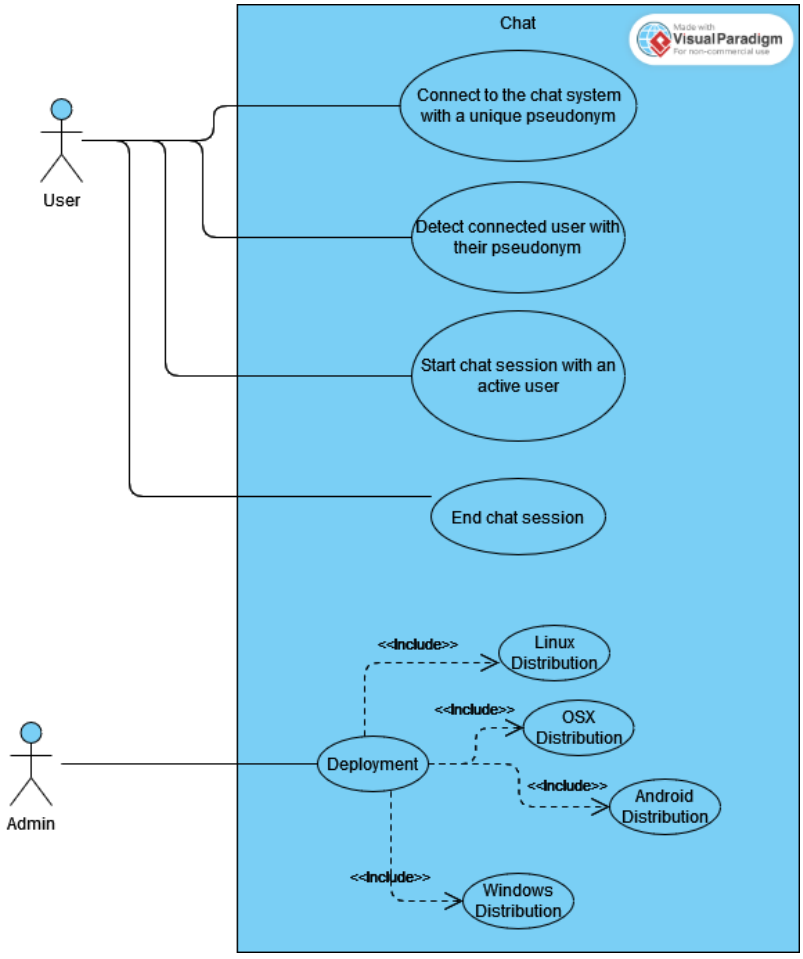

Figure 1: Use case diagram for ChatSystem

3 Database schema

- 'id' is the primary key and an auto increment integer
- 'envoye' is a boolean precising if I'm the sender of the message or not
- 'ip_contact' represents the identity of the contact with which I'm having a conversation
- the content of the message

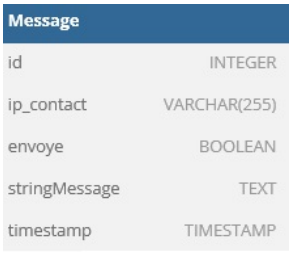

Figure 2: Database Schema

To retrieve a specific conversation with a contact, I only have to select messages in which the 'ip_contact' matches that of the contact and order by id. Ordering by id allows me to retrieve messages in the correct order.

4 Sequence Diagrams

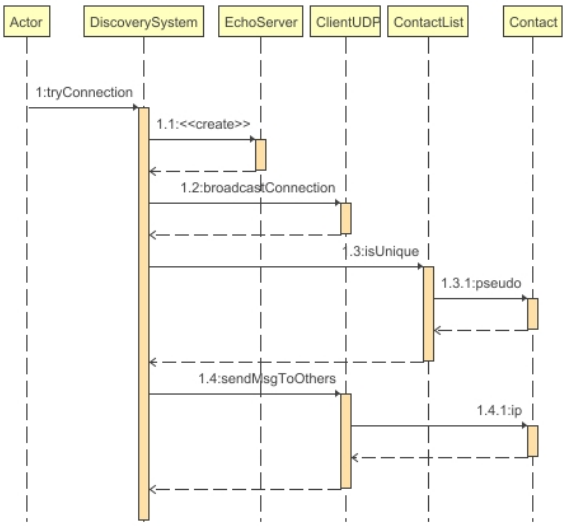

Figure 3: Sequence diagram : Log in

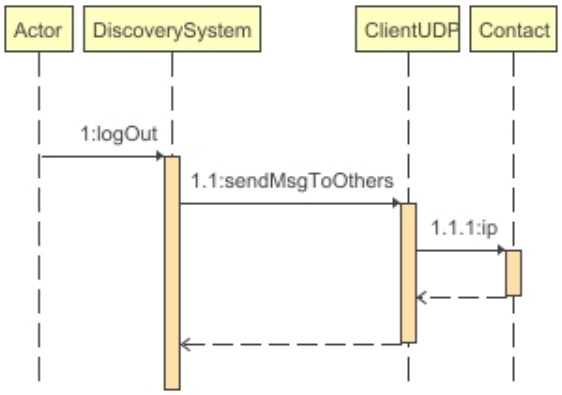

Figure 4: Log out

The sequence diagram below covers both "Start Conversation" and "get History Messages" as the former includes the latter.

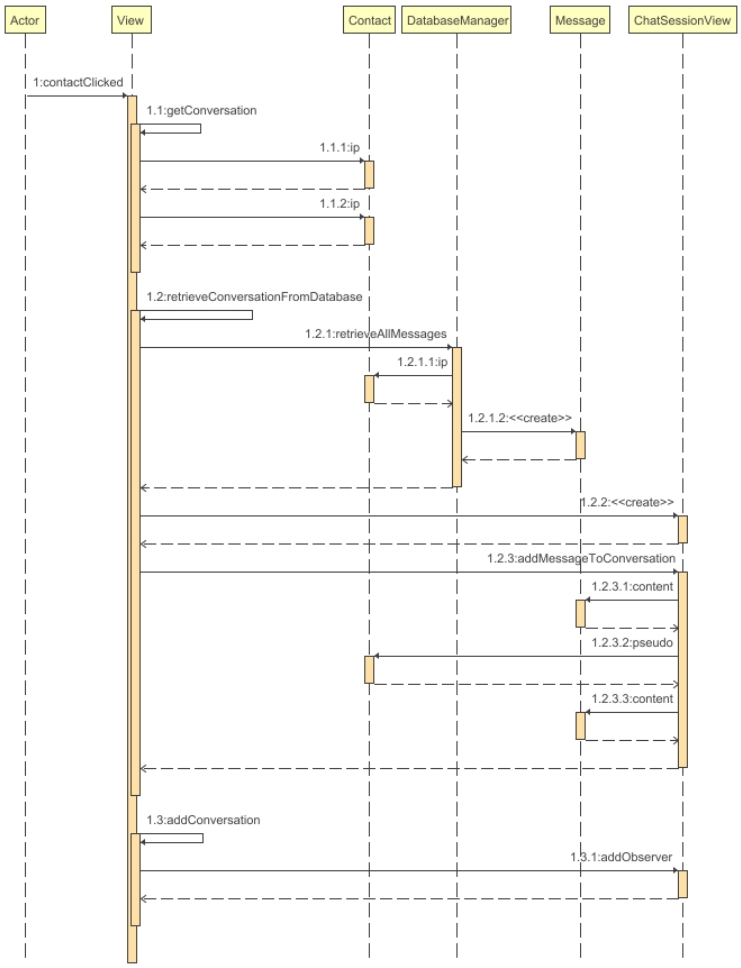

Figure 5: Start conversation

5 Class Diagram

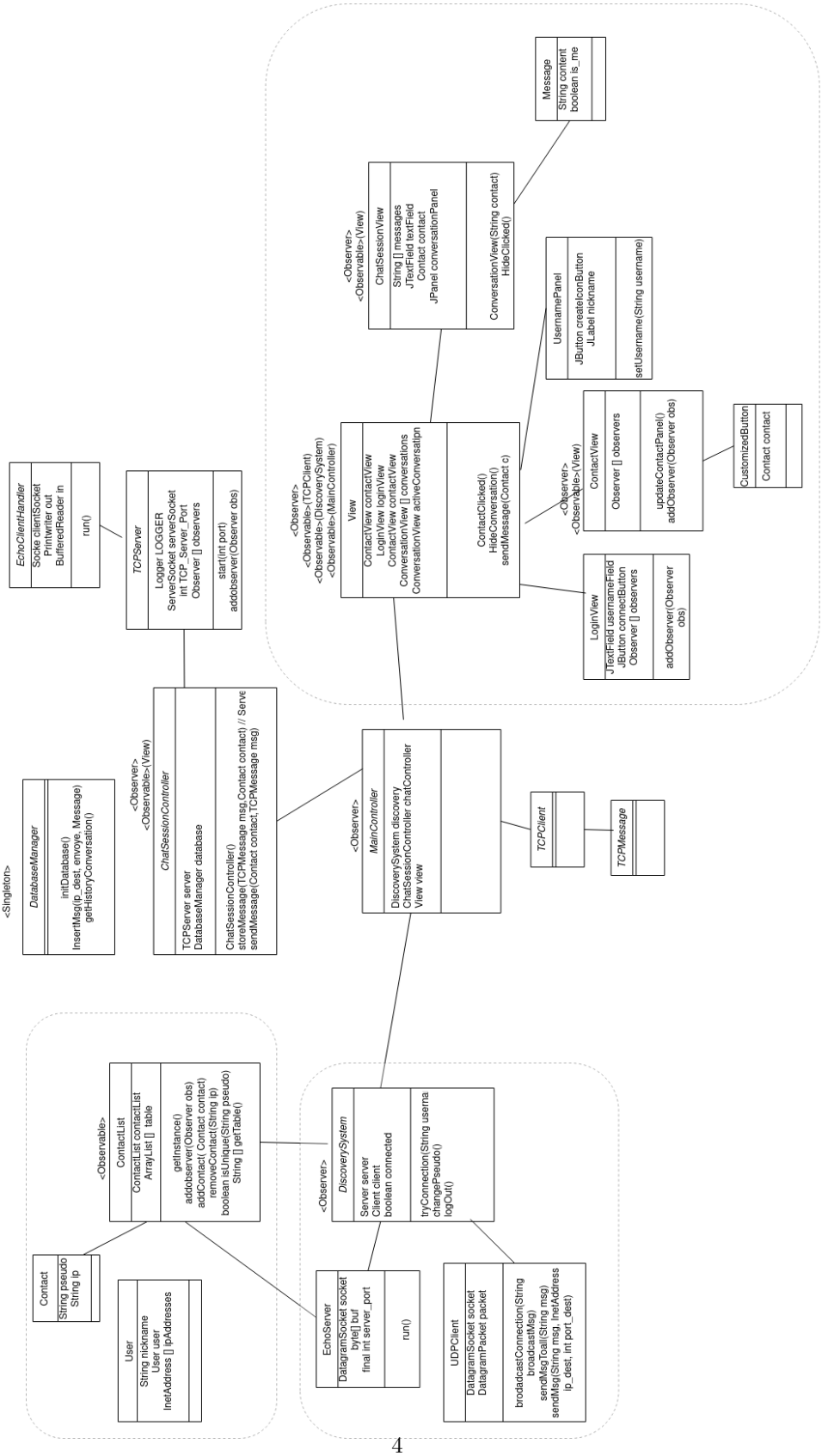

Figure 6: Class Diagram

6 Architecture overview

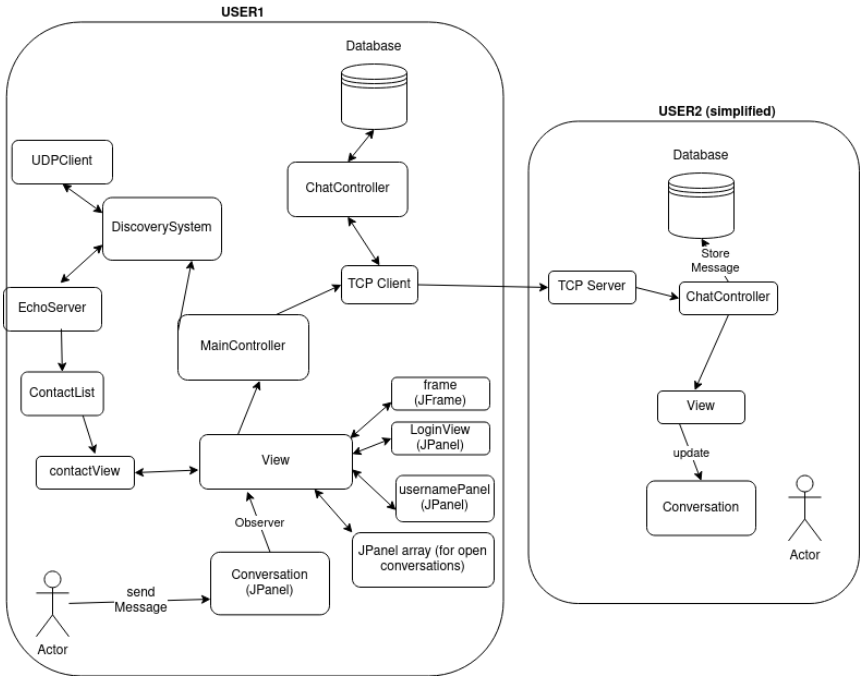

Figure 7: Architecture overview

The schema above shows what happens inside the ChatSystem app when sending a message. The message goes from ConversationView to the View (with an observer) which is then passed to TCPClient. TCPClient will then send the message which will be received from the server, passed to ChatController, then to the View which will find the right Conversation and update it.

7 Project Management

We conducted the the management of this project using the agile method. At first, we identified higher priority user stories and we identified each of those user stories. We relied pretty much on our teachers that explained us the features of the project.

Moving to the individual tasks, we tried to divide independent tasks between us (when it was possible) using Jira and then we launched sprints of one to two weeks to have enough time to complete the tasks. For example, I (Achraf) implemented the TCP section of the ChatSystem while Sara handled the Database. These two parts are completely independent allowing each one of us to work at his own pace.

At last, we reviewed our progression after each sprint during TP sessions.

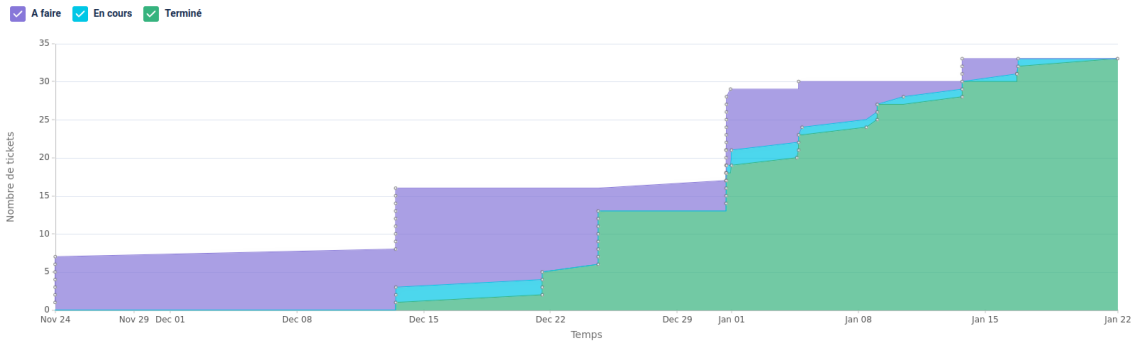

Figure 8: Flow diagram of Jira tickets state

8 PDLA

To make sure our code was working, we implemented a continuous integration. Thus, each time we pushed code to the remote repository, we were able to check if the code was working or not. Also, to be efficient and make progress all along the project, we wrote code for our classes and in parallel tests to make sure our code worked fine. For example, right after implementing TCP, I wrote tests that sent five packets to my localhost, received them and then asserted that payloads packets were intact. During the project, we frequently pushed code to our remote repository. We made sure our commits were clear, specific and of reasonable size. Thus, if an issue were to happen, we were confident we would easily flag it.

Most of the time, we used only one branch to advance on the project. To avoid, conflicts, we frequently pulled code from the repository before pushing our changes. However, to implement the view, I used another branch to test my code as I was never sure of the result produced by my code. Indeed, the JUnit tests were useless when it came to implement the view.

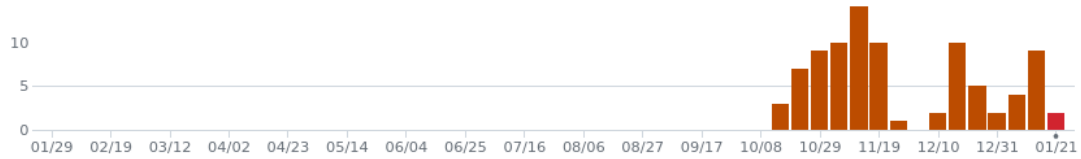

Figure 9: Weekly Commits