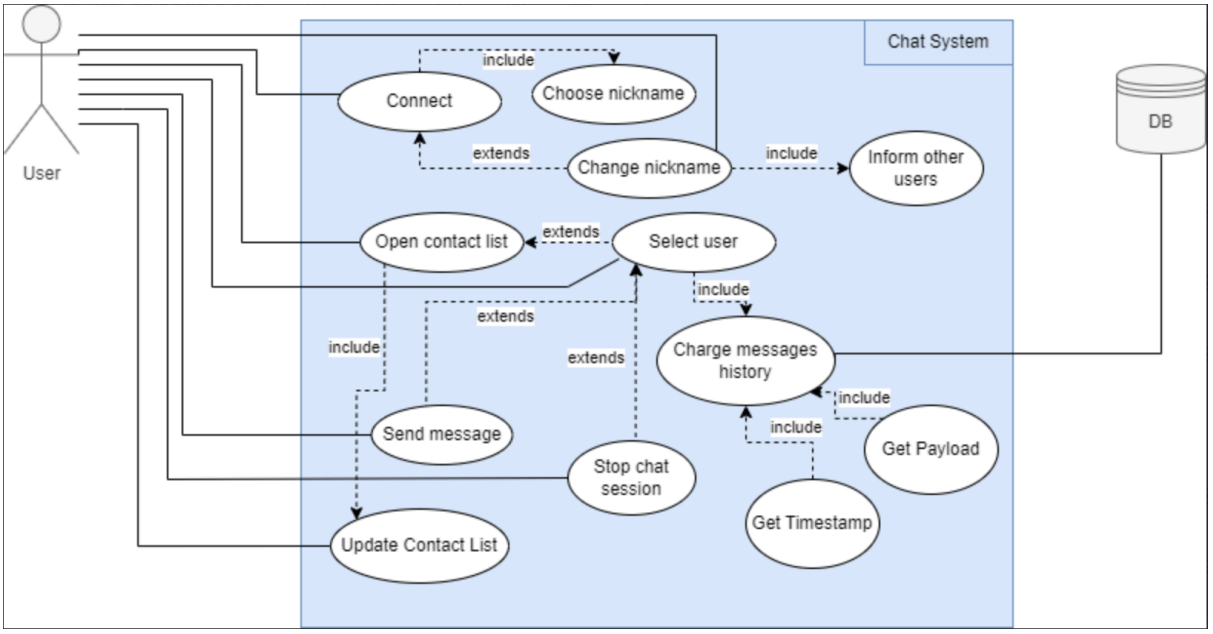
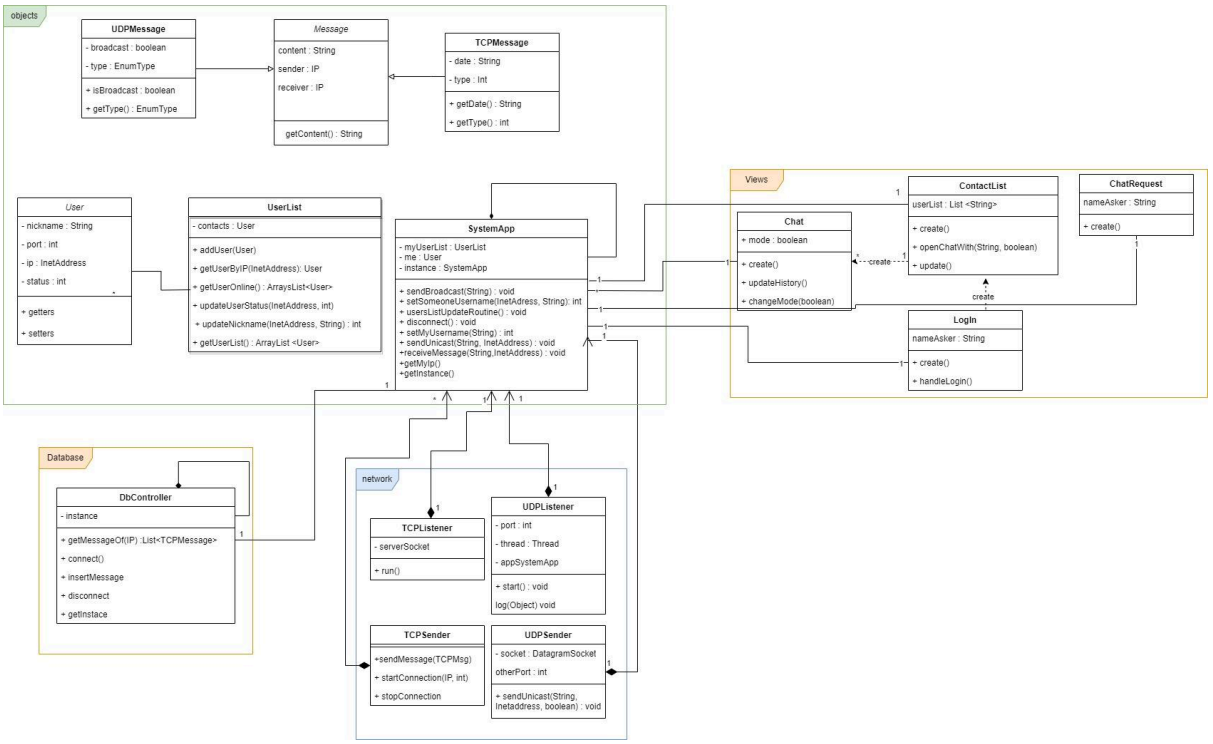
Use case diagram :



The user that uses the application will be able to connect in the first place and this will include choosing a nickname, different or not from his previous one. Once logged in, he will be able to open the contact list to see who he can open a chat with. Once a chat is opened, he will be able to close the chat. At any time, the user can refresh the contact list manually to see if somebody he suspects has had a crash or an unwanted interruption of the app.

Class diagram

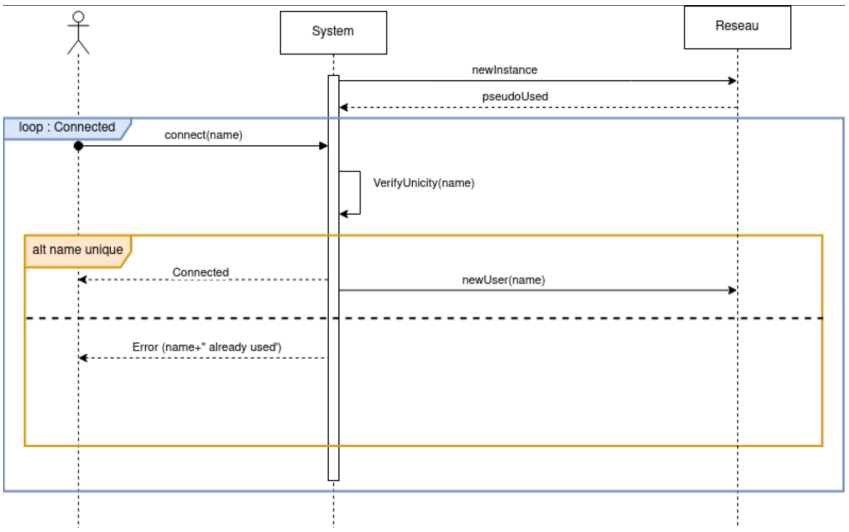
:



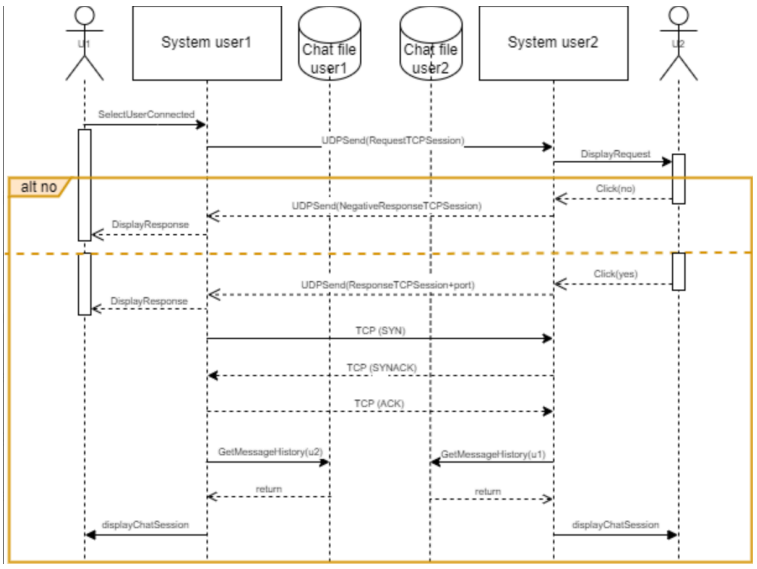
The class diagram implements the main classes and with their main methode of our ChatSystemApp. We haven't added them all to keep our diagram clear.

Sequence diagrams :

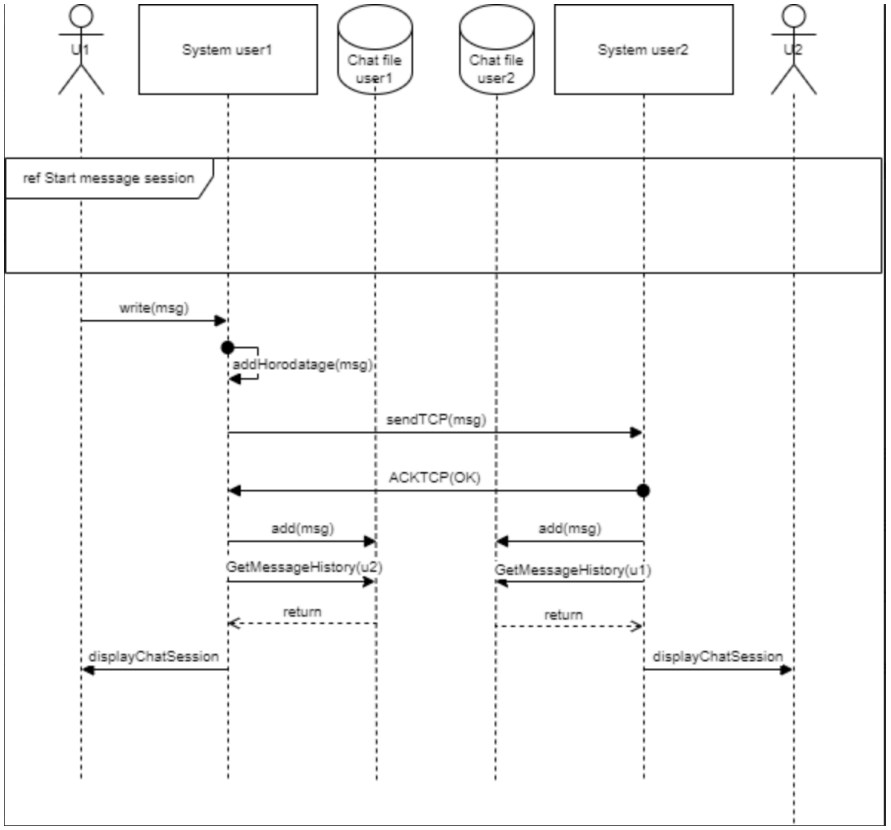
Connecting on the app :



Starting a chat :



Sending a message once the connection is opened :



We do not need a database diagram because we use SQLite and decided to put all the messages together in one table. When we need to access some messages, we just need to use a request to load messages sent to and received from this user.

Overall, we tried to implement as much as possible the design patterns that we saw in class in order to make it easy to implement new functionalities or to modify existing ones. In terms of directory, we managed to group the files together so that we can find it easily e.g. “TCPSender” is a socket so it is used to communicate so we can find it in the “network” directory. We also used the singleton mechanism to avoid having two instances of a class that needs to be unique.