C# Project : Multithreaded chat using TCP/IP connection

17/12/2017

MAXIME BOUN – DAVY TRICHANH

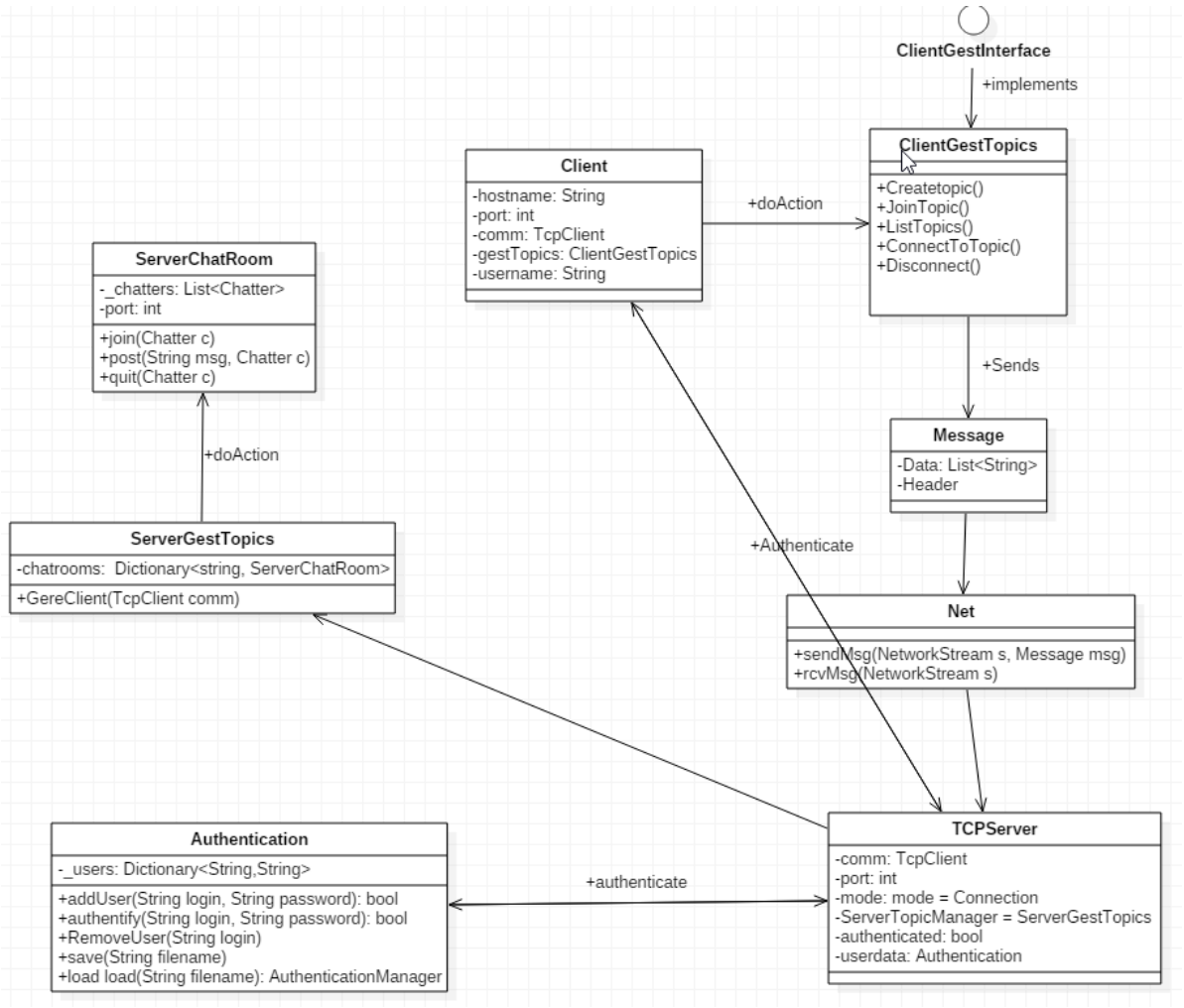
Introduction

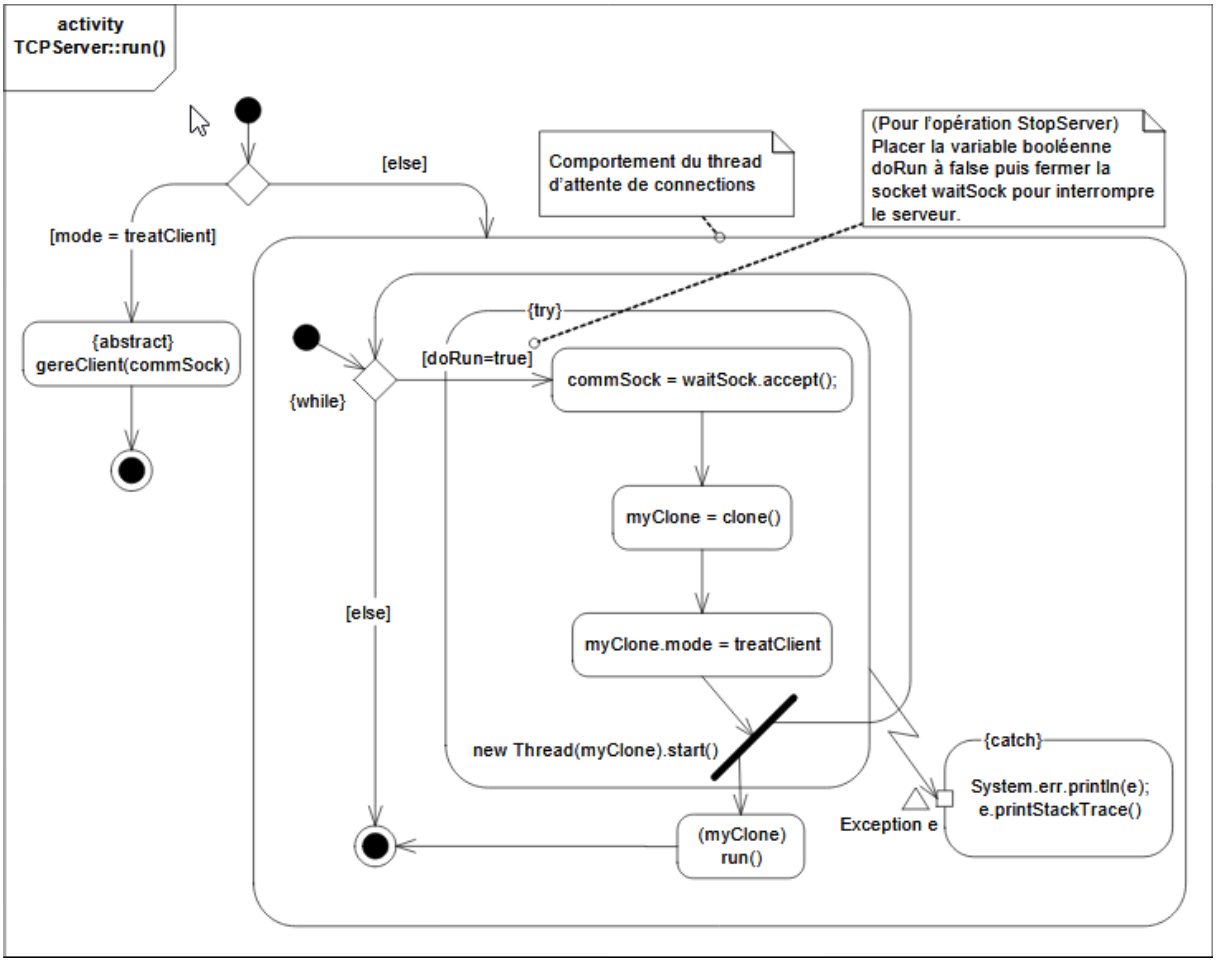
The aim of the project is to implement a chat application in C#. The solution will be divided in different projects:

- The Chat project: implements basic functionalities in console: authenticate, add, delete a user, create or delete a chatroom, send and receive a message.
- The Client project: establishes a connection with the Server project. The user interacts with the server project through this project. He is able to perform several actions, such as list all the topics, create a new topic, or join an existing topic.
- The Com project: provides an implementation of the Message class, which will be used by both the server and the client. Defines also the methods to send or receive a message.
- The server project: Accepts a connection from the client and executes his requests.
- The ChatGraphique project: implements a GUI, uses the Client methods to communicate with the server

Important! In order to run the program, you must run the ChatGraphique project and the Server project at the same time. To do so, please right-click on the solution 'Chat' and click on "Définir plusieurs projets de démarrage". Then tick "plusieurs projets de démarrage" and choose Démarrer on both ChatGraphique and Server.

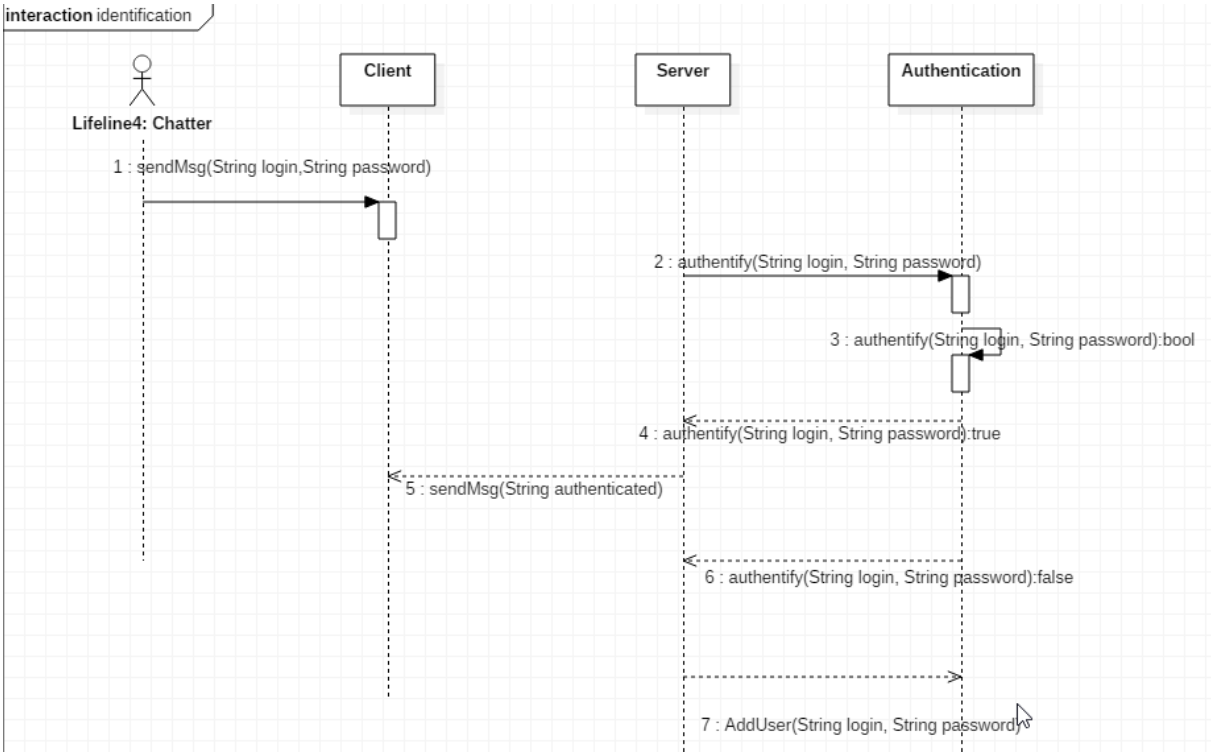
Class diagram

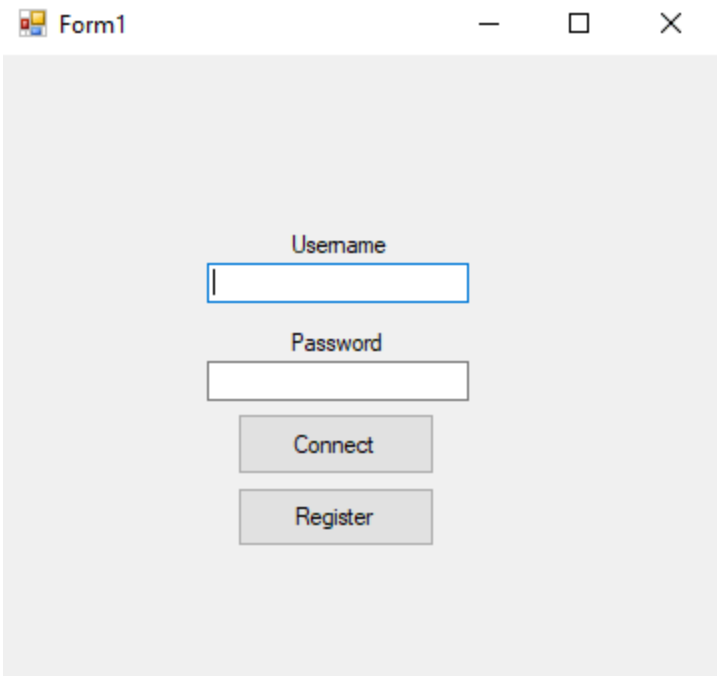




For the server, we followed the implementation given in the subject. The server has 2 modes: A TreatClient mode, which processes all the client's requests and a Connection mode, which listens on the port 81 in an infinite loop. If a client connects to this port, the server accepts it, makes a clone of the client and creates a new thread, in the TreatClient mode. Then, the server resumes listening for a new connection.

Authentication process

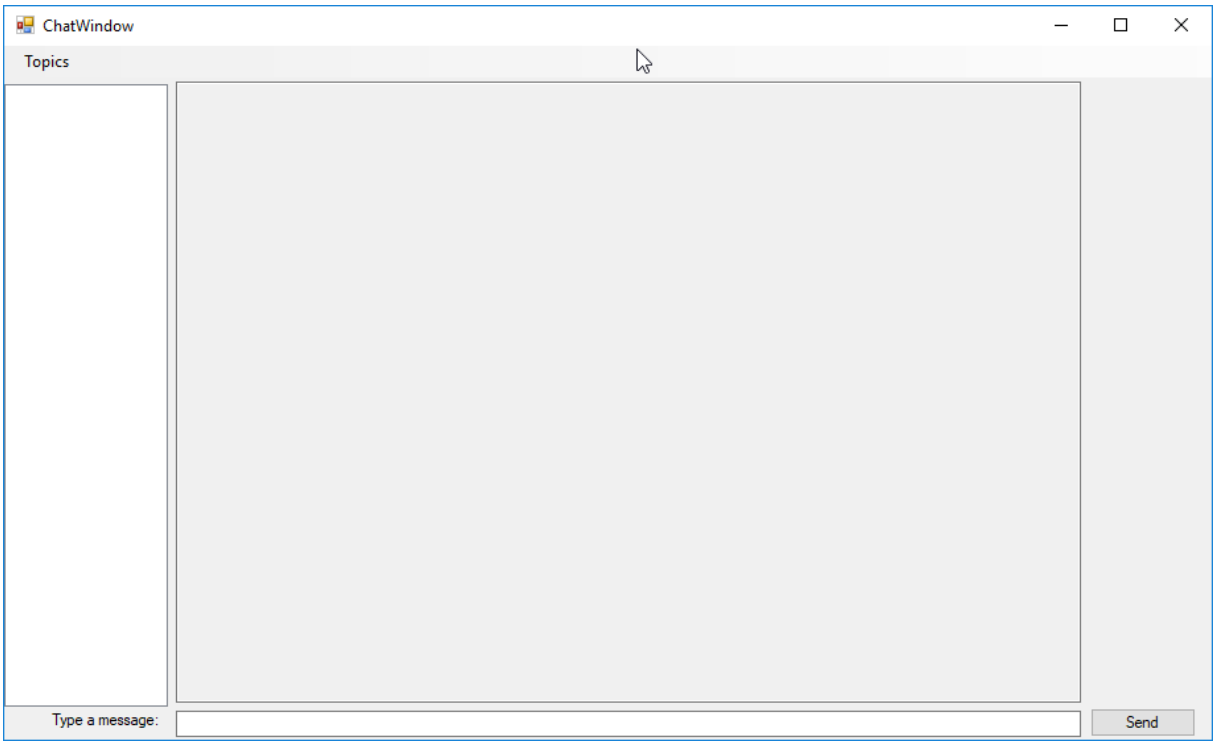




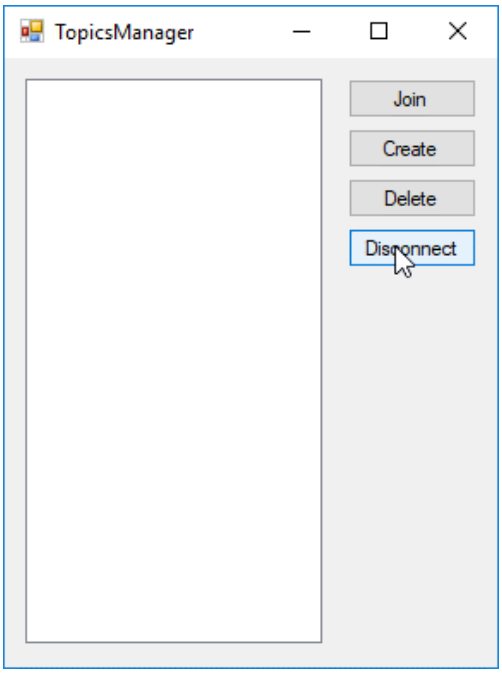
Authentication process diagram and window. If the user is not registered, the button Register creates a user with the information the user typed in the textboxes.

Basic features

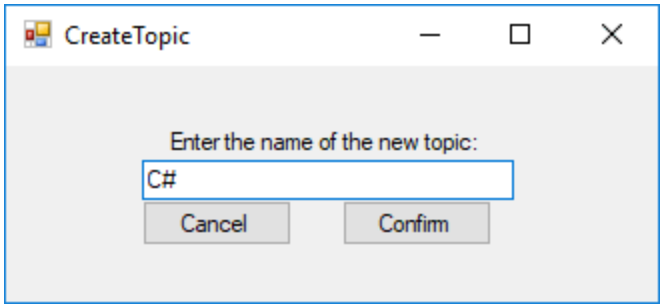
Create a topic



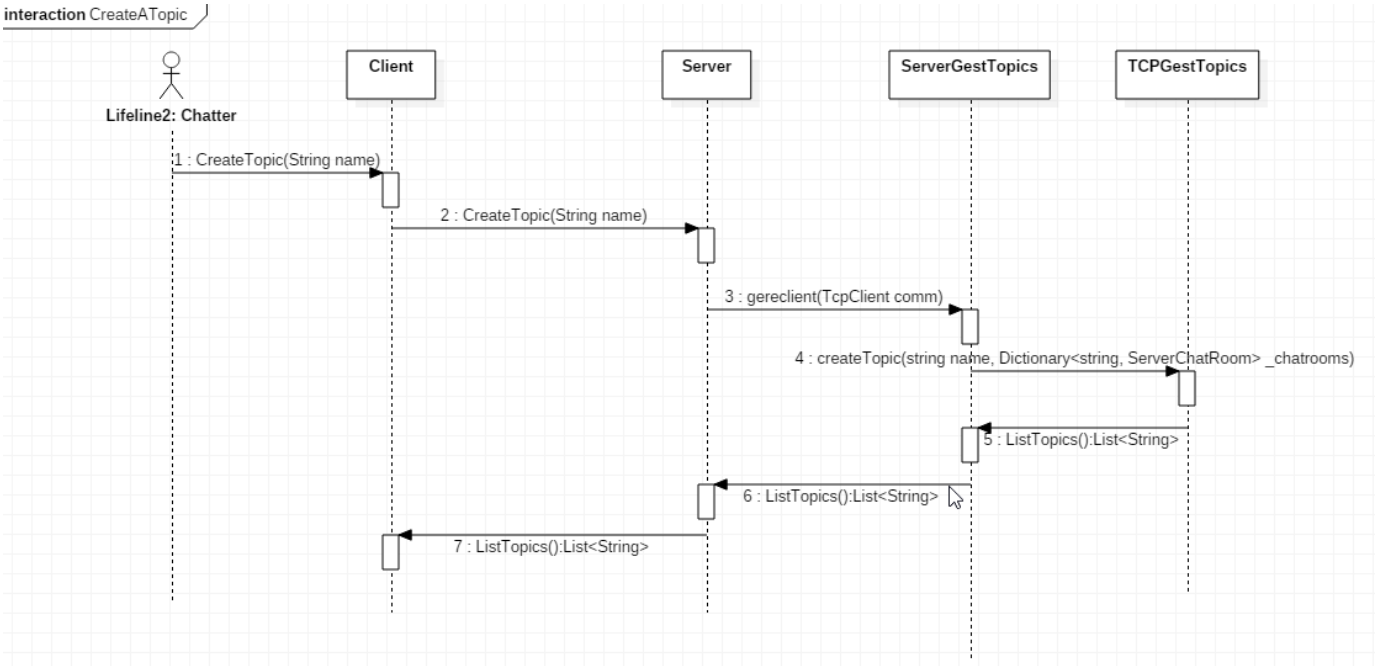
After the authentication, the program displays an empty chat window. To manage all the topics, we need to click on the Topics strip menu item.



This window manages all the topics. The ListBox on the left will be filled with the names of all the existing topics.

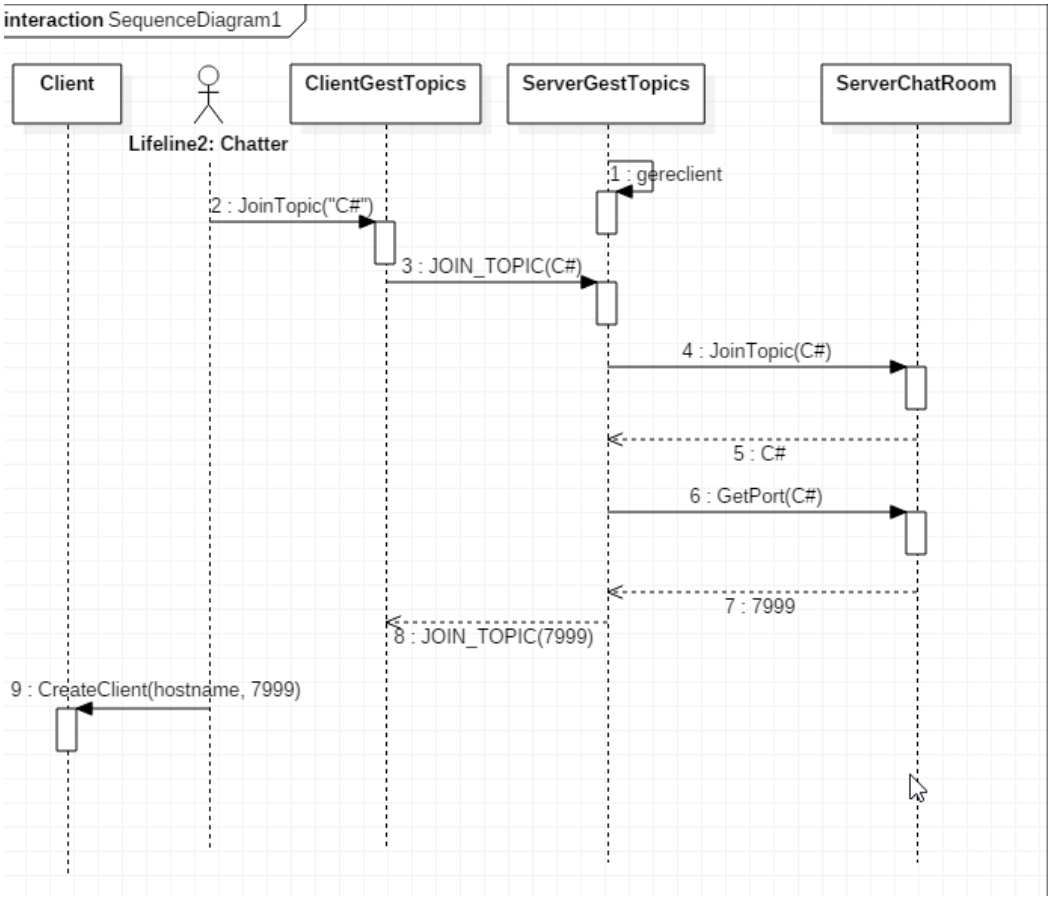


To create a new Topic, type the name of the topic and press confirm.

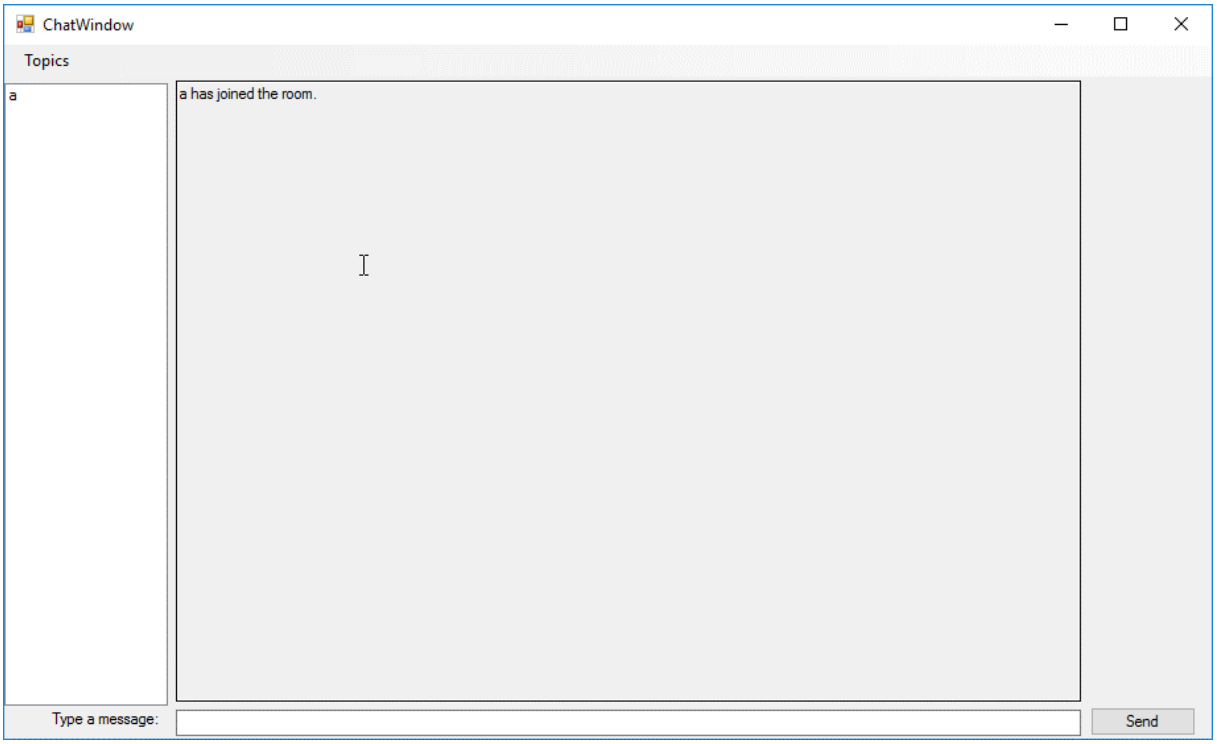


In the TCPGestTopics, we create a new TCPServer on a new port. Then we run it on a new thread and catch an exception in case the port is already taken. Finally, we send back the new list of topics.

Join a topic

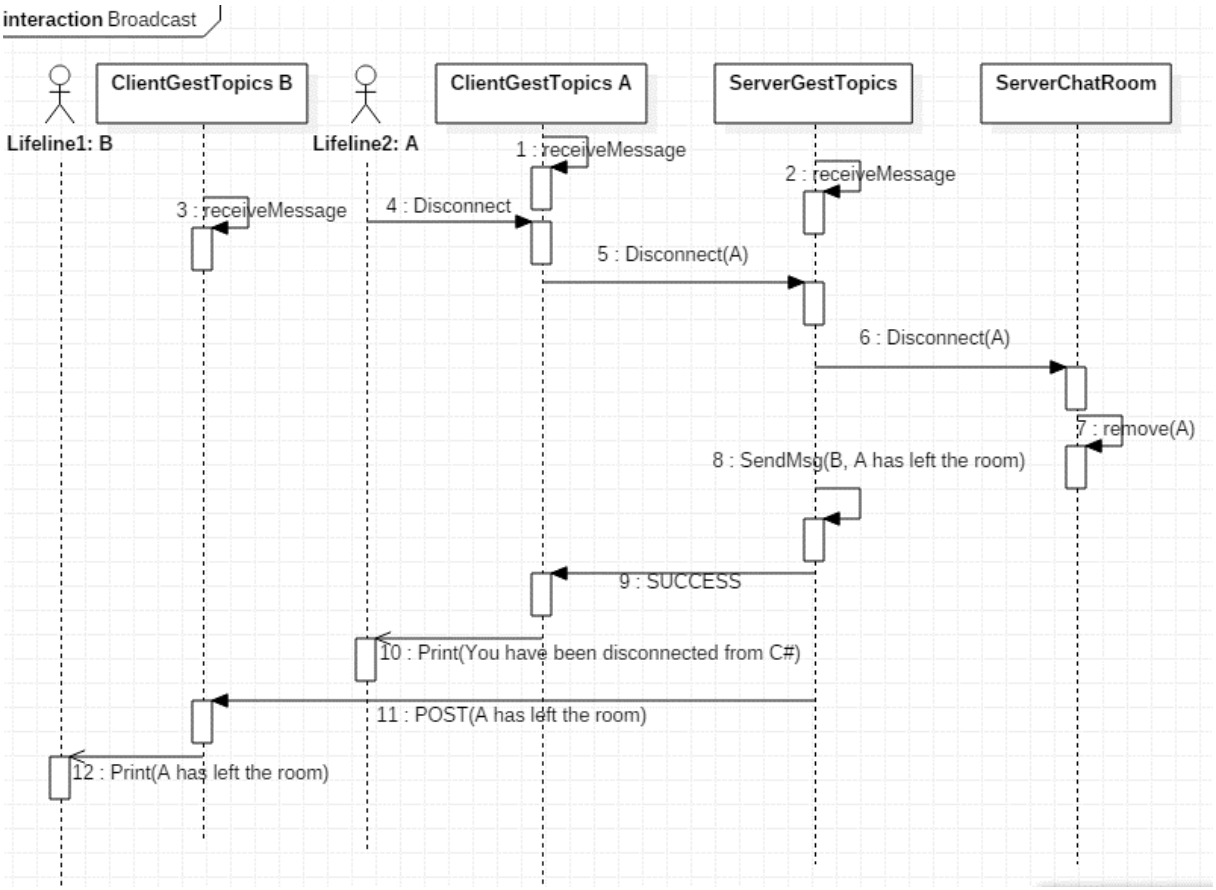


*The client sends a request with the **JOIN_TOPIC header** along with the name of the topic he wants to join. The ServerGestTopics gets the right chatroom and calls the getport method to get the port of the chatroom. Then, a reply is sent to the ClientGestTopics with the port he needs to connect to. A new instance of a client is created, with a new connection to the chatroom.*

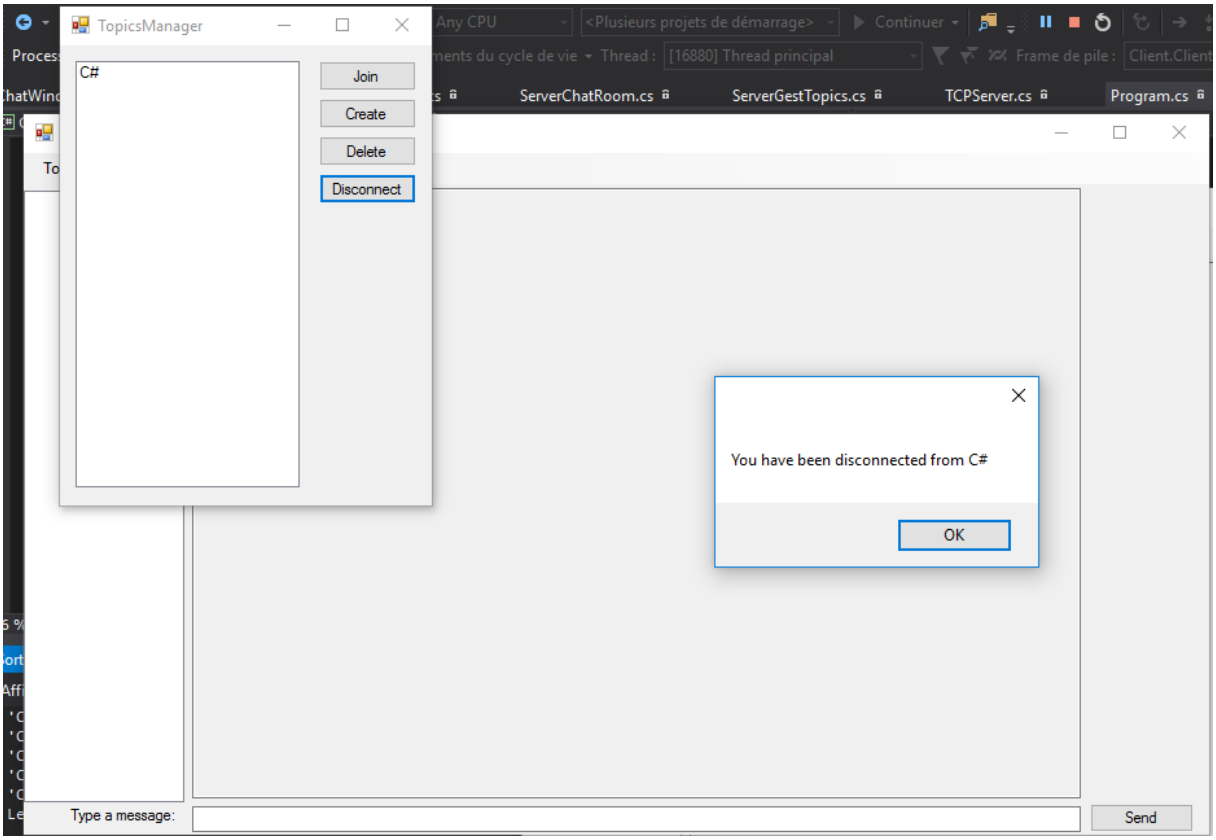


*The user gets redirected to the **ChatWindow**. The listbox on the left hand lists all the user currently connected in the chatroom. The textbox in the middle displays the messages sent by users in the chatroom and join/quit notification from the server. The textbox and the listbox are updated through a **delegate** with the **Invoke** method.*

Disconnect from a topic

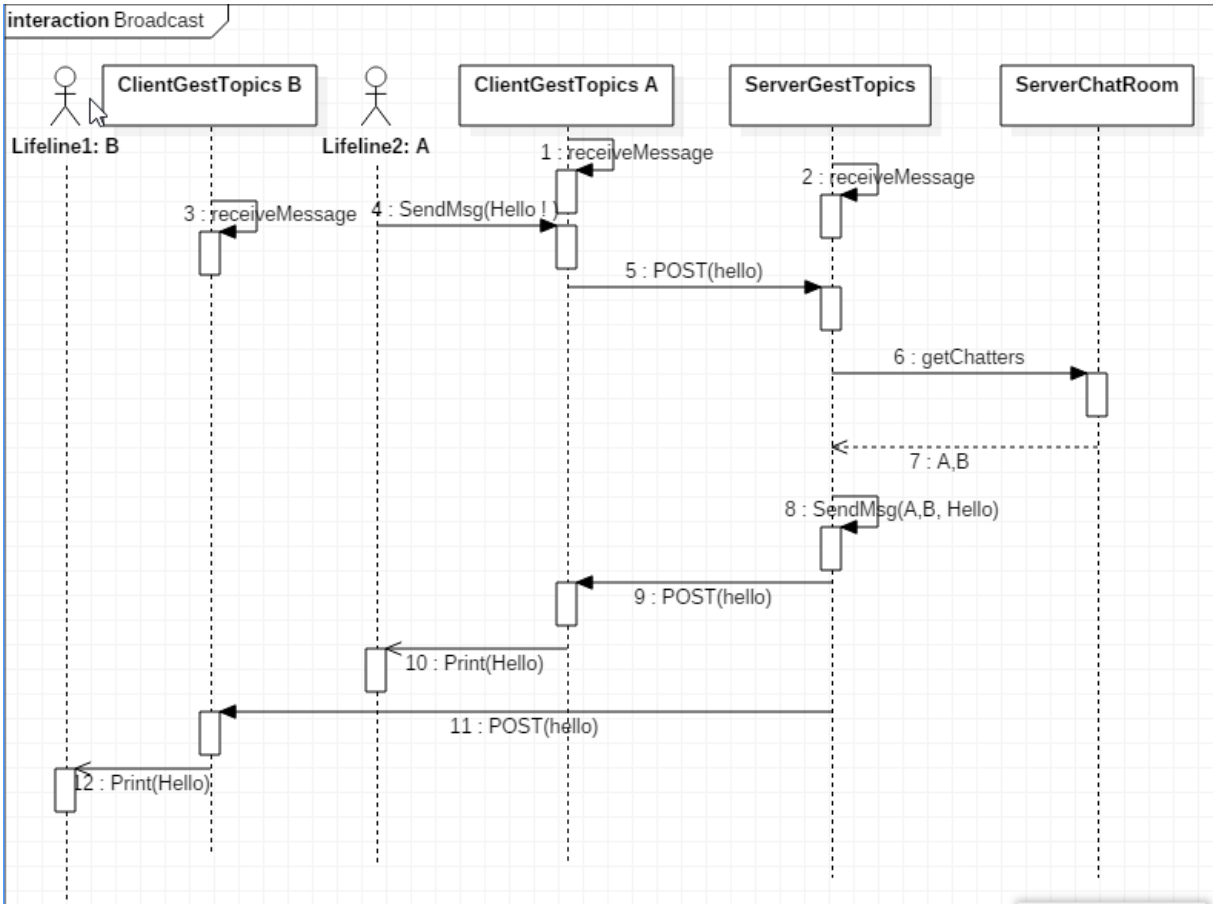


If A wants to disconnect from C#, he sends a disconnect request to the server. ServerGestTopics removes A from the users' list of the Chatroom C#. Then the server sends a SUCCESS message to notify A that he has been disconnected from the chatroom without any error. Finally, the server notifies all the chatters that A has left the room.

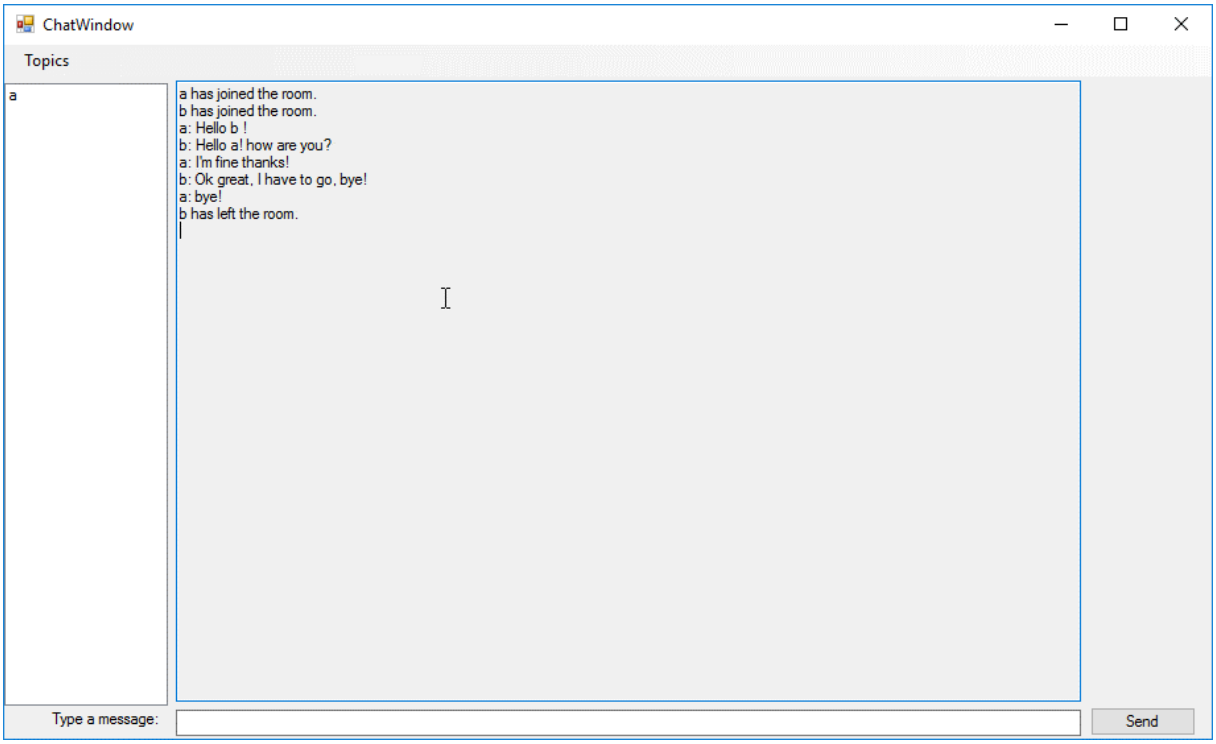


The user has been disconnected from the C# chatroom.

Discussion in a chatroom



In this diagram, actor A and actor B are discussing with each other. User A wants to send the message "hello" to user B. The clientGestTopics in the instance of A sends a POST request to broadcast a message. The request gets received by the ServerGestTopics, which gets the list of all the chatters. The ServerChatRoom returns that A and B are in the chatroom. We now send the message Hello to A an B.



Discussion between a and b

Missing features and possible improvements

First, we haven't implemented an administration system. Any user can register and manage all topics. We tried to handle as many exceptions as possible. However, a few exceptions are still thrown: for example, a user can run another instance of the client and connect to an account on two clients at the same time. As for the possible improvements, we could implement a private message system which will allow two users to discuss with each other without creating a public chatroom.