Gaël AMIOT

C# report

# I- Functions implemented/Not implemented

1. Create a profile (login, password) and save it : Done
2. Login : Done
3. List topics : Done
4. create topics : Done
5. join topics (only one - bonus : many at the same time) : Done (include the bonus)
6. send messages to all chatters on a specific topic : Done
7. send private messages : Not done

All functions were done, except the send private messages.

For the send private message, in order to implement this, we could track every user connected to the server. When a user wants to send a message to another, we show him the list of the connected user, he chooses one user and it opens a new window. When he sends a message, it contains the emitter, the receiver and the text message. The server then sends it to the correct user. The user receives it and a new window opens where he can see the message and can chat with the emitter.

To be able to do this, we would need a specific port on our server side, dedicated to receive and send message. Another port would be used to keep track of who is connected. User sends a specific message when they login or logout so it’s easy to implement. The redirection part would also be easy to do, we just must store a list containing a Username with the Socket associated with that client.

Other improvement :

* Save and retrieve the topic from the database (currently, only the login and password of the users are stored)
* When a client click on the disconnect button on the TopicList form, all the TopicChat form opened are then closed.

# II- Design

## Sequence diagram:

First, login and password of user are stored in the database, when we launch our server, they are retrieved from the database. The database is no longer needed, as the credentials are stored in the memory. When we close the server, the database is then updated if new user has registered. I created a simple database on Wamp and connects to it in localhost.

## To create the database

Create a table named “csharp” and run the following script :

Port 3306

Username: root

Ip address of database used: 127.0.0.1

-- phpMyAdmin SQL Dump

-- version 4.8.3

-- https://www.phpmyadmin.net/

--

-- Hôte : 127.0.0.1:3306

-- Version du serveur : 5.7.23

-- Version de PHP : 7.2.10

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

SET AUTOCOMMIT = 0;

START TRANSACTION;

SET time\_zone = "+00:00";

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8mb4 \*/;

--

-- Base de données : `csharp`

--

-- --------------------------------------------------------

--

-- Structure de la table `account`

--

DROP TABLE IF EXISTS `account`;

CREATE TABLE IF NOT EXISTS `account` (

`login` varchar(100) NOT NULL,

`password` varchar(200) NOT NULL,

PRIMARY KEY (`login`)

) ENGINE=MyISAM DEFAULT CHARSET=utf32;

--

-- Déchargement des données de la table `account`

--

INSERT INTO `account` (`login`, `password`) VALUES

('admin', 'admin'),

('test', 'test'),

('hello', 'hello'),

('azerty', 'azerty'),

('qwerty', 'qwerty');

COMMIT;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

On the client side, everything is made with graphical interface. It’s easier to be able to chat in multiple topic for example and wouldn’t be possible with a console interface.

We have the loginForm which is our first form. We can basically login or register on this form. When we have logged in, the TopicList opens, which displays all the topic available. We can double-click on a topic to opens a TopicChat form, which allows us to chat with other members on that topic. When a TopicChat is opened, the TopicList doesn’t close, so we can open another TopicChat to have conversations about different topic. On the TopicList, we can also disconnect, which closes the connection, we can reload the topic to see the latest topic available, and we can also add a topic, which opens a form that allows us to enter the name of a topic.

On the server side, one thread that handles the login or registering part, one thread that sends the list of topics when a client asks for it, a thread that only listens for topic to be added, and one thread per topic. Basically, we have one server per functionality and topic, that each run on a unique port.

When we launch the server, the main in the Program.cs is launched. A Server is created and started. We then launch our server, one that handles login and register request, one that handles the request of the list of topic, one that handles the add topic request, and one server per topic, for the moment, I’ve only added 3 topic at the beginning, so only 3 servers of topic are launched.

The Server class contains a UserManager, a TopicManager and a topicMutex.

The UserManager contains every login and password of User and gives method to login and register. It contains a list of User, a mutex and a connection to the database. The mutex is used when we go through the list, so only one thread can access the list of users at a given time.

The TopicManager contains a list of Topic and a mutex. When the server creates a topic, it uses AddTopic method, which creates a new topic with the name given by a user. It then creates a thread and launched the server of that topic in the thread. The thread then waits for connection of clients. When a client connects to a topic, a thread is created where the client is placed, where he can interact by sending messages or by disconnecting.

A Topic contains a Name, a Port, a List of the connected client to that topic. It also contains a static int nextPort which is used to give every topic a unique port, so the server doesn’t override and create errors. There is also a mutex which is used to allow only one thread to access the list and modify it. In the Topic, you can addClient, removeUser and sendMessage that are sent to every client.

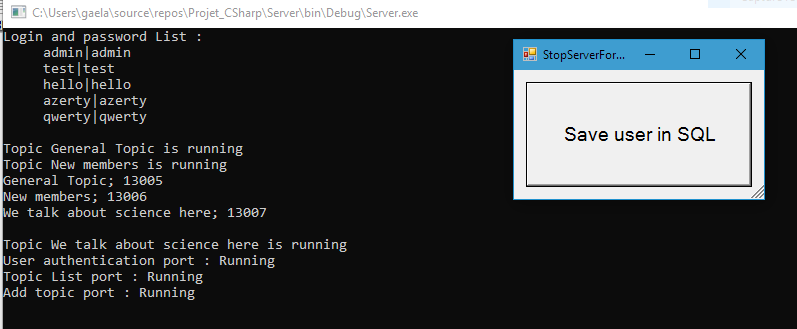
We also have the User class, that contains a login and a password, and which is Serializable. We have UserRegister, UserLogin and UserDisconnect that are inherited from User. Those 3 class are used when a client communicates with the server, it then performs the action according the name of the class.

When the server is launched, a simple form with a button opens, which allows us to save the list of User to the database when we close our server.

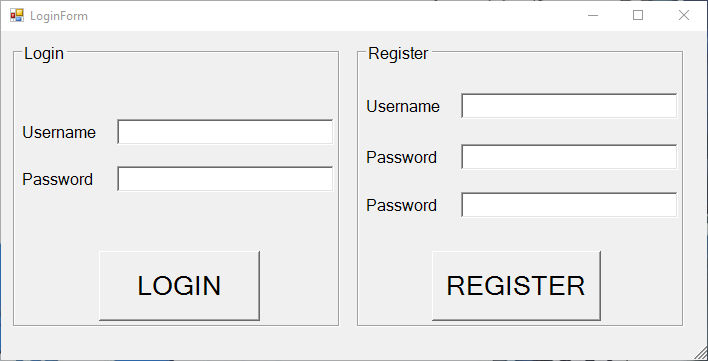
There is also a class called Message that is serializable that is used to send simple text message between client and server.

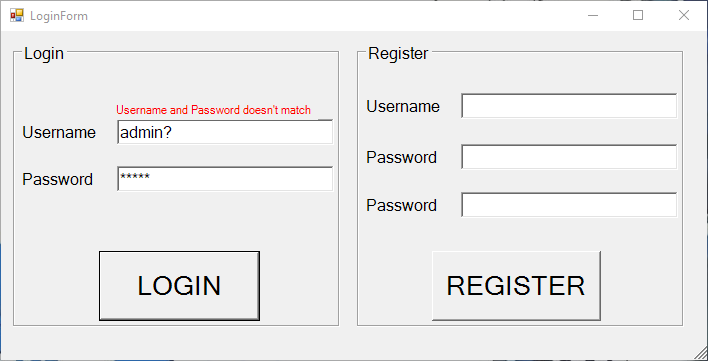
# III- Screenshot of the application

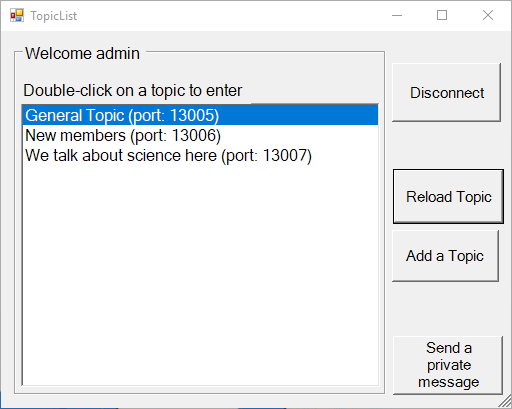
First, we launch the server.



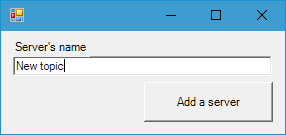
A client wants to login (the console on the client side has no purpose, but keeps showing up, we will focus on the GUI)

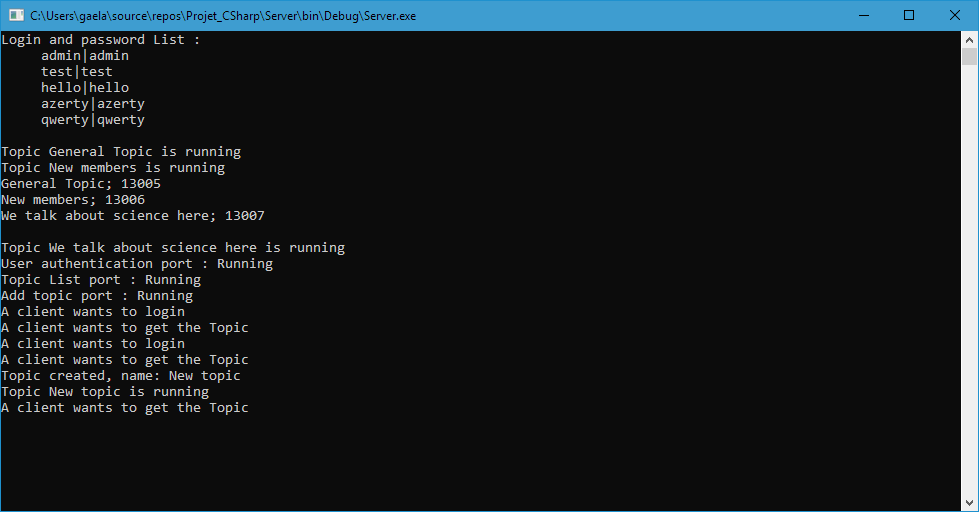
  
Basic error, such as “wrong username and password”, “Password not the same” or “Username already taken”, are handled.

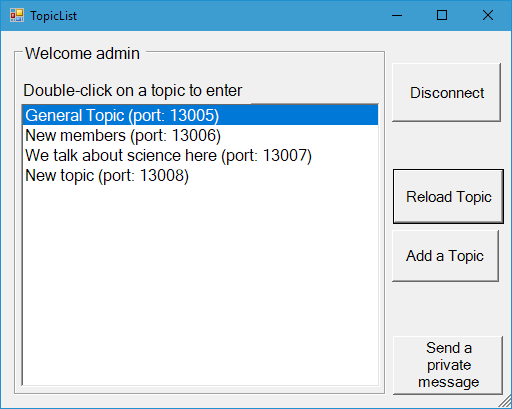


We then input a correct Username and password and we are redirected to the TopicList form

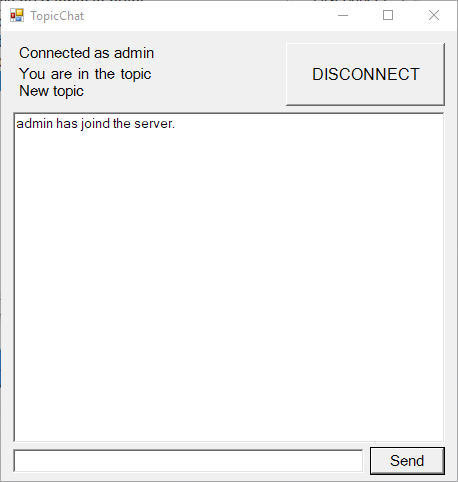
We can click on add Topic, which opens a form. We then input the name of a new topic and click on the add button.



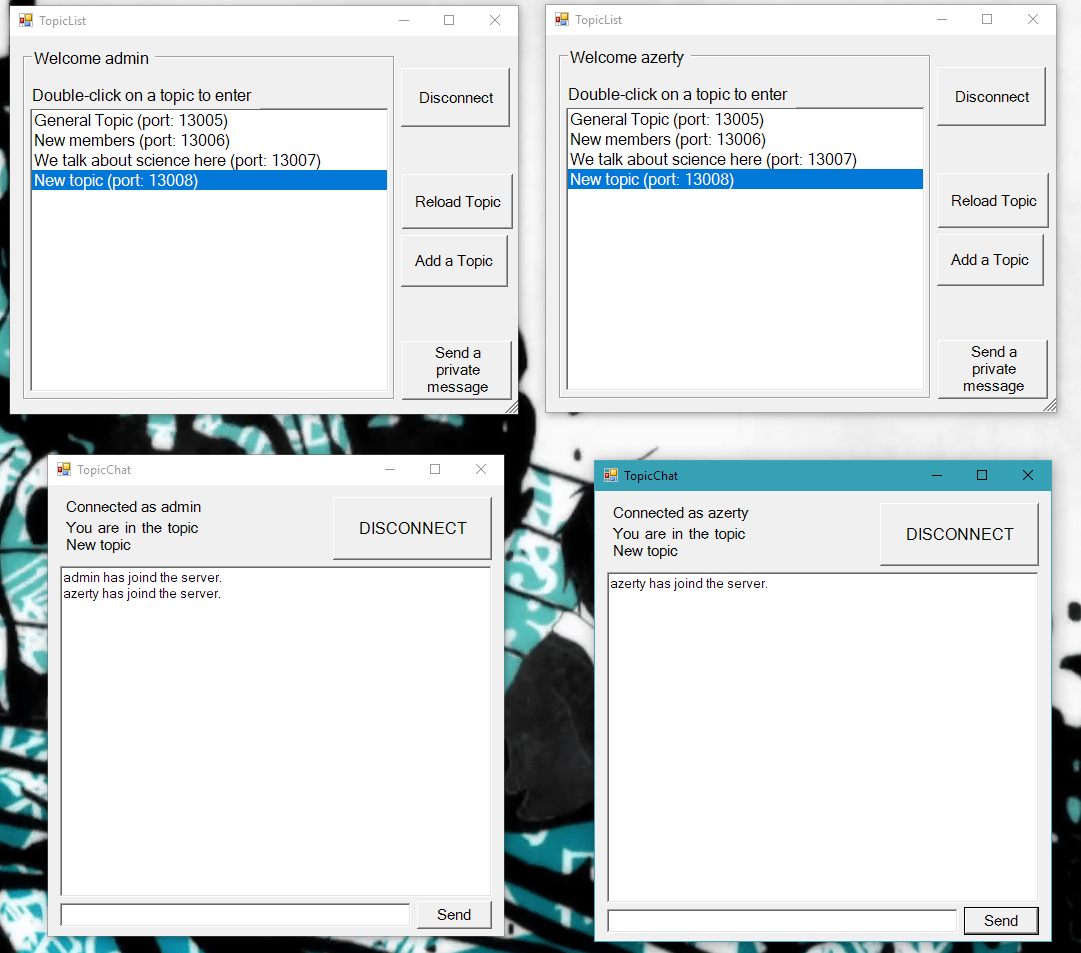
We then click on Reload Topic, to see that our new topic has been created.



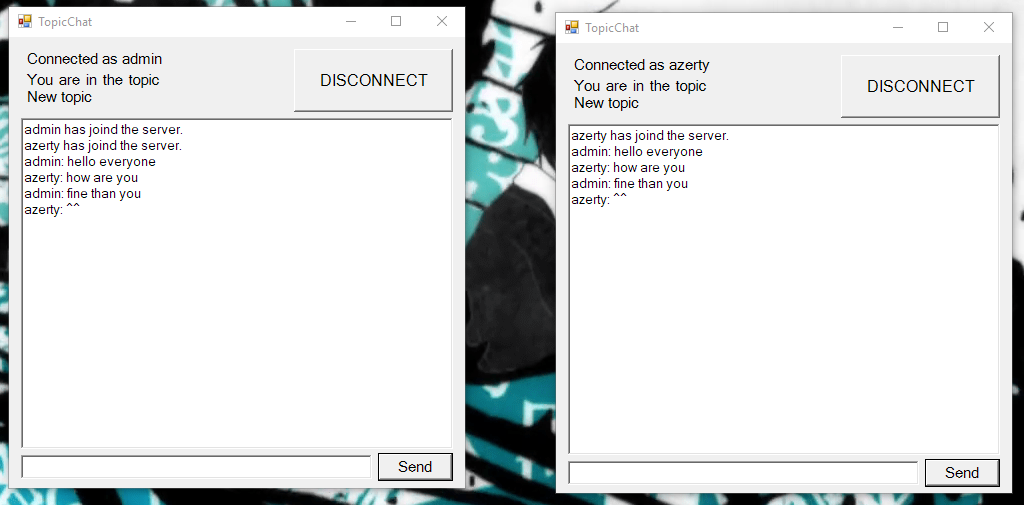
We can then double click on our New topic to access the chat of that topic.



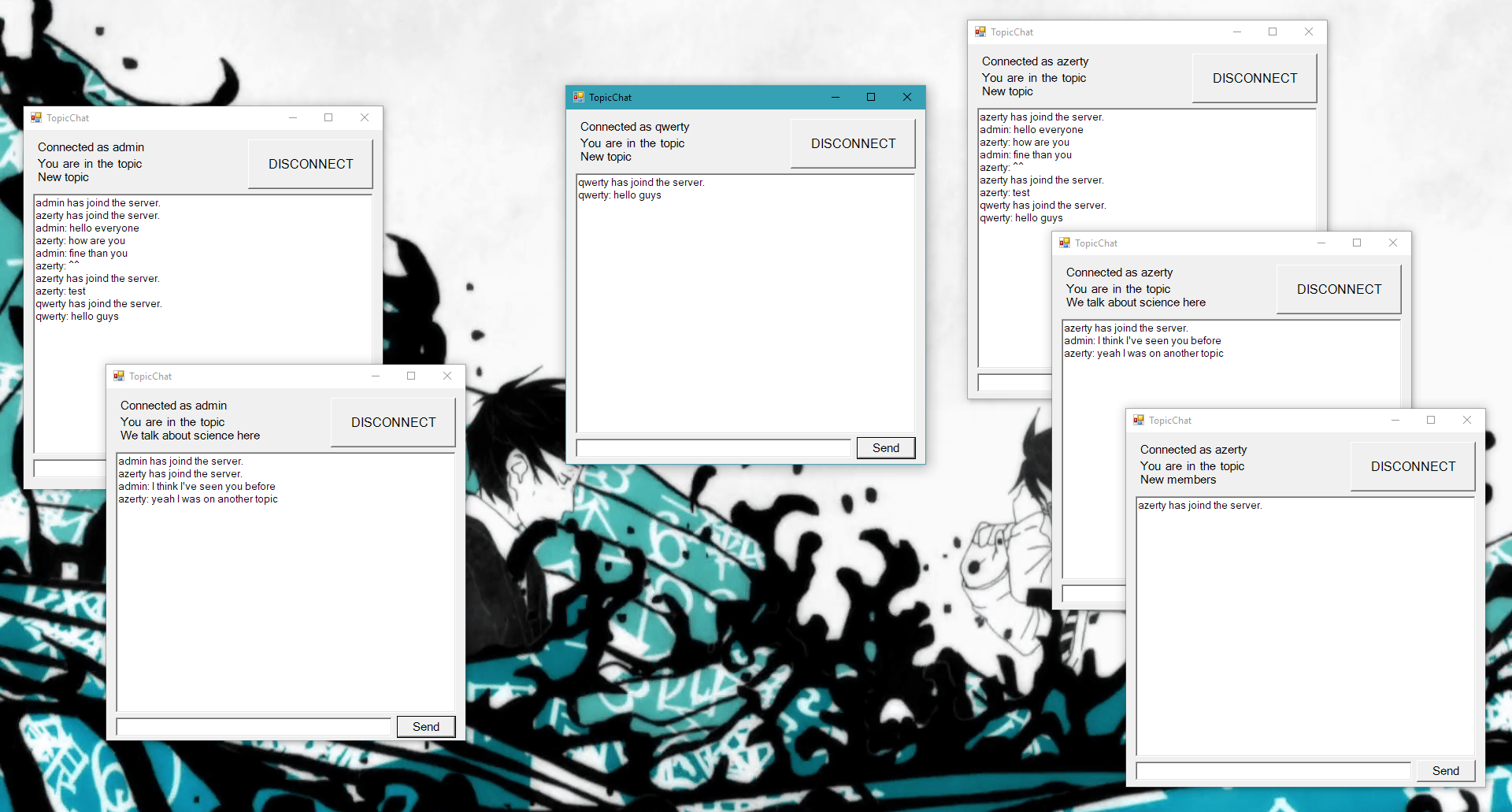
I’m going to open a new client so they can chat between them



We can see that they are able to communicate in the topic



They also can connect on multiple topic at the same time (I have connected a third client to demo purposes)

 We can see one client on 2 topic on the left, one client on 1 topic in the middle, and one client on 3 topic on the right that are able to chat on different topic.

They can then disconnect from a topic or close the form to send to the server a UserDisconnect, so they are disconnected from the topic and removed from the connectedUser list of that topic. They can also add topic while chatting on topic.