12/01/2020

C# Project
Chat App

Functions implemented:

- Create profile
- Login
- List topics
- Create topics
- Send messages to everyone

Functions not finished:

- Send messages to people in a specific topic

Functions not implemented:

- Send a private message

Possible improvements:

- Adding more operations to the database of user profiles (list profiles on the server interface, update username/password, delete)

Design

The application rests on a few structures.

```
enum MsgType
{
	createprofile = 0,
	login = 1,
	listtopics = 2,
	createtopic = 3,
	listusers = 4,
	sendmsg = 5,
	sendprivmsg = 6,
	switchtopic = 7,
	help = 8
}

struct Message
{
	public MsgType Mymsgtype { get; set; }
	public List<string> S { get; set; }
	public string Topic { get; set; }

	public Message(MsgType msgtype, List<string> listofString, string tpc = null)
	{
		this.Mymsgtype = msgtype;
		this.S = new List<string>(listofString);
		this.Topic = tpc;
	}
}



struct Client
{
	public TcpClient s;
	public string username;
	public string topic;

	public Client(TcpClient client, string usn, string tpc)
	{
		this.s = client;
		this.username = usn;
		this.topic = tpc;
	}
};
```

These are instantiated in the server as such:

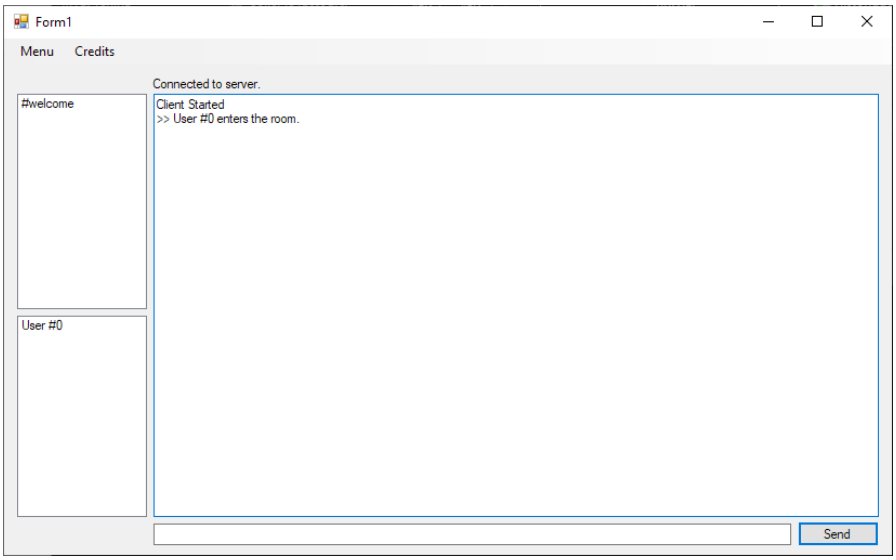
```
List<Client> clients = new List<Client>();
List<Profile> profiles = new List<Profile>();
Dictionary<string, List<Client>> topics = new Dictionary<string, List<Client>>();
```

Then, the server runs by waiting a TCP connection. Once it receives a connection, it creates a client associated to it and runs a thread dedicated to this client. This thread will then wait for any new instruction to execute by the server. When a client is closed, the exception is caught and the client is disposed of, as well from the list of clients.

On the client side, when the application is starting, it will try to connect to the server. On success, it will also create a subsidiary thread to listen to any instruction sent by the server. The instructions provided by the client are relayed to the server and then back to the same client, as well as the other clients if the instruction was a message to everyone, for example.

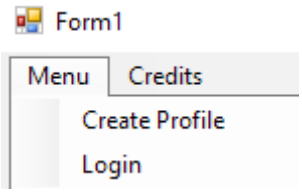
Screenshots

Client connection

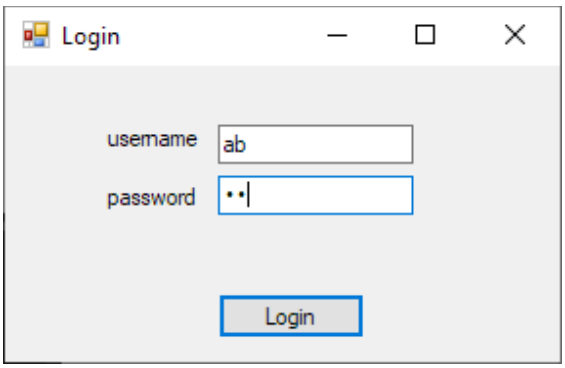


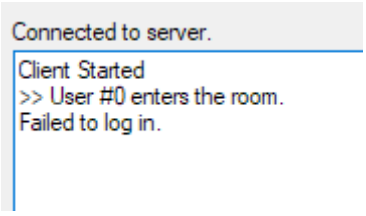
Login

From the toolbar, we can access the create profile and login.

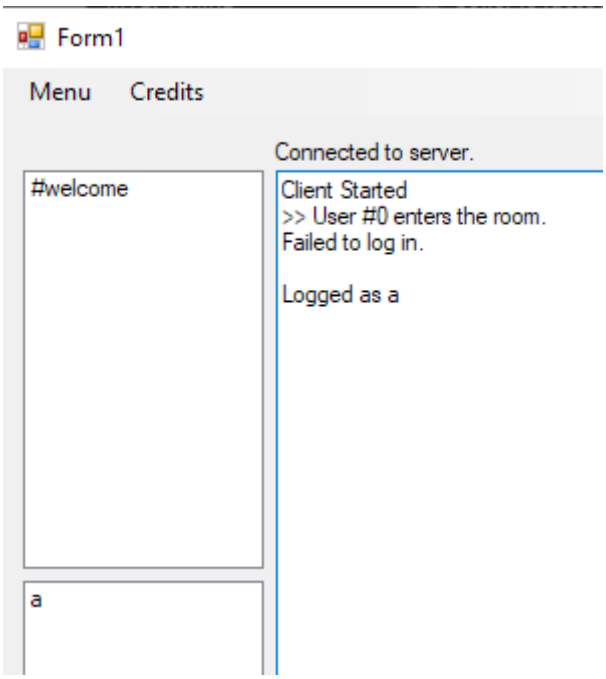


We try to log in as a user that does not exist in the database.



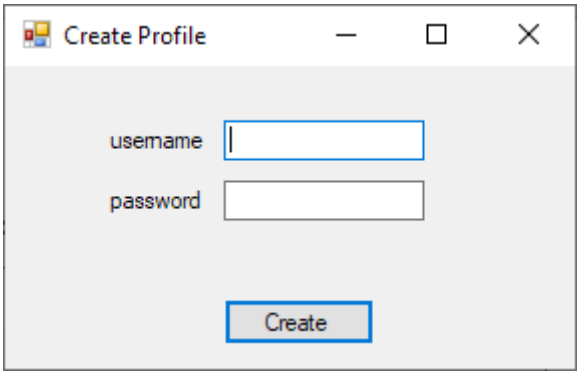


If the user exists and the password corresponds, we are logged as the user “a”. The listbox of users is updated as well.



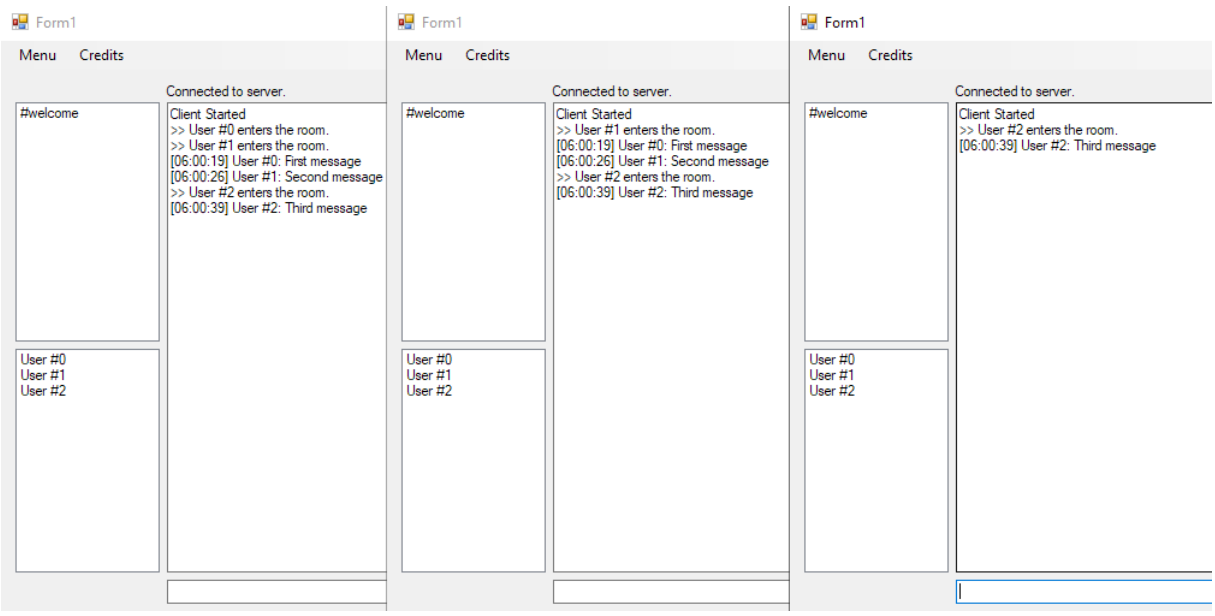
Create a profile

The layout is the exact same as the one for logging in. It is needed to create at least one profile to get the database up and thus doing the operations on this database. The database is in SQLite, thus if already installed, the database can be created on the spot.



Multiple users

As depicted below, the application can handle multiple clients sending to everyone messages.



Leaving is also a message sent to everyone. With the command "/createtopic", we can add a topic to the listbox of topics to everyone. However the development of the messages specifically for a channel is not over.