.NET Environment

EFREI Paris
Development in C Sharp
Project

Project Report

Student :
Gabin Salabert

Teacher :
Christian KHOURY

Sommaire

1 Introduction

Application coded in C with Visual Studio. All the source code is commented and each method is explicated.

It's a network based multithreaded client-server chat app. Chatters establish a connection with the server and can chat in channels or by private messages.

There are several functionnalities asked :
- Create a profile (login, password) and save it
- login
- List topics
- create topics
- join topics (only one - bonus : many at the same time)
- send messages to all chatters on a specific topic
- send private messages
- ....

Multithreading, MVC pattern, mutual exclusion, stream, serialization, NotifyPropertyChanged and TCP library are the most complicated tools used in this application.

The user interface has been done with the VS designer.

2 Functions implemented

2.1 All functions

All Functionalities asked were implemented and work.

1. Graphical interface for the whole application.
2. Register on the server.
3. Login on the server.
4. All clients are serialized in a binary file.
5. Create a channel.
6. Interface is instantly edited when new client or new channel.
7. List every channels.
8. All channels are serialized in a binary file.
9. Join one or many channels at the same time.
10. Send messages to every chatters on a channel.
11. Send direct messages.
12. Receive a notification when new message on channel / in private.
13. Disconnecting by closing the interface.

2.2 Possible improvements

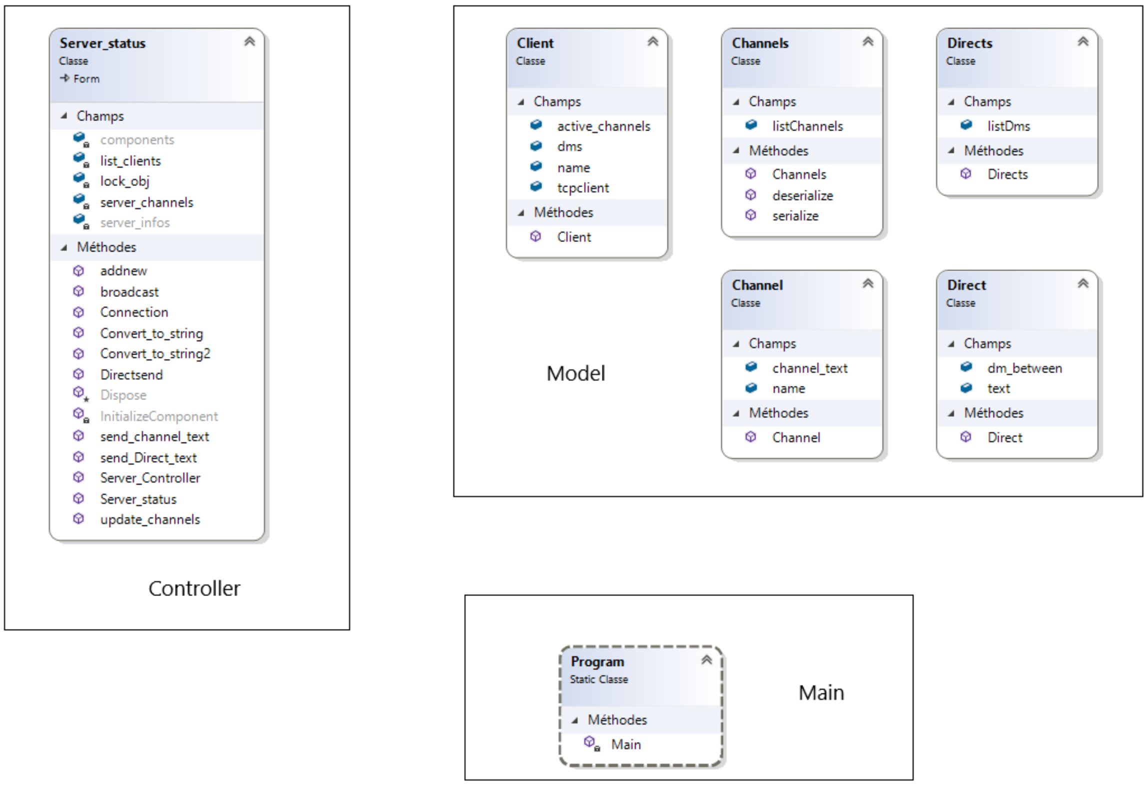
I would like that every notification for a new message appears next to the corresponding channel or private message and get a sound alert when in the same time.

Improve the interface to be more user friendly and allow lots of channels and private message without reading issues.

Add some smileys to the chat and animations to the interface, just to try out and get some experience. Tired of doing really ugly things (Swing, etc) when coding interfaces for PC apps.

3 Design (UML)

3.1 This is the server side project class diagram :



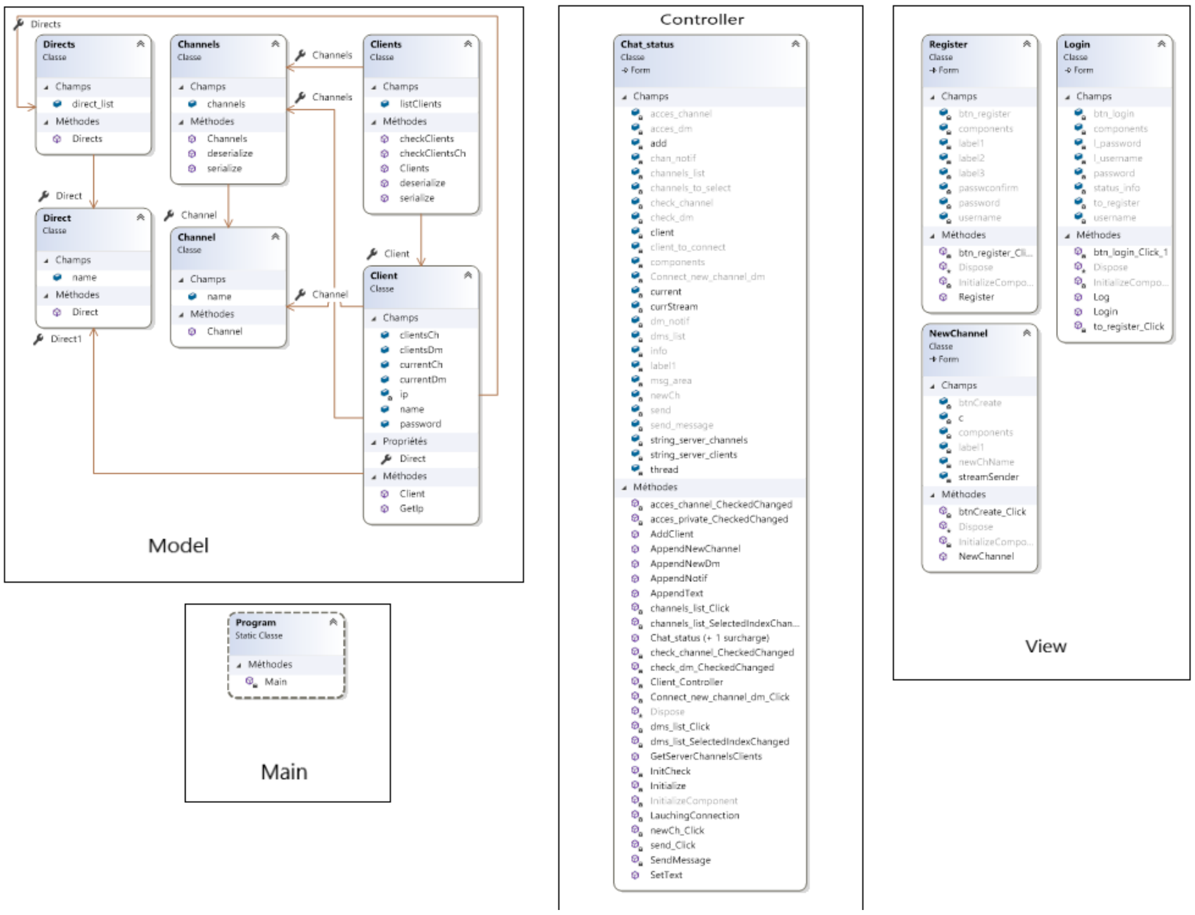
As you can see, there is the MVC pattern, to centralized the sending and the receiving of encrypted data on streams between the server and the client. Data are composed of 2 part : Action and information.

Each time, the controller get the action and use the according method with the information given.

Directs and Channels are classes like collection which allow the serialization.

Locks are used to not use the same object at the same time, for example with the list of connected clients which is usually used by different threads.

3.2 This is the client side project class diagram :



There is also the MVC Pattern, like the server side.

But here there is a NotifyPropertyChanged notion, it means that when events like a new connection, or a channel's creation, the interface is automatically update in real time. You receive a notification alert when you get a message as well.

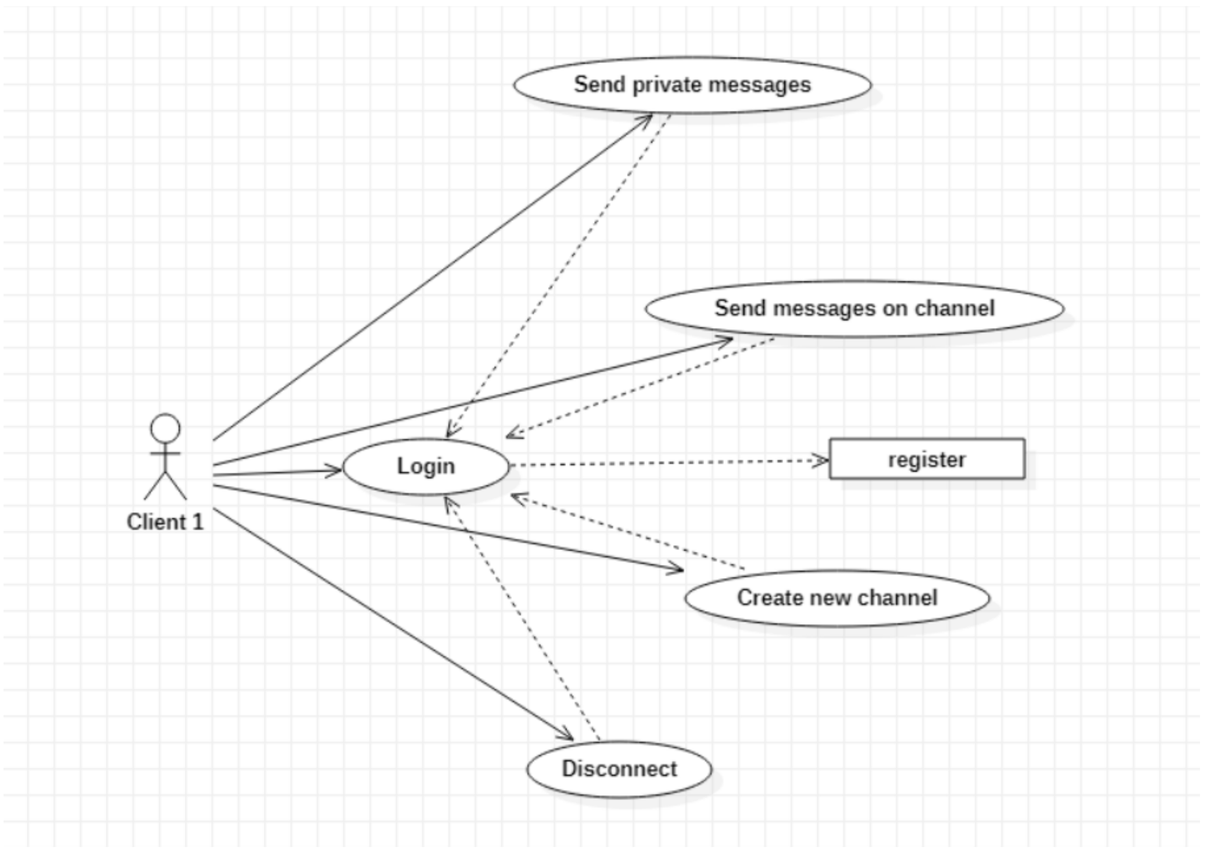
There are 3 interfaces in this side :

1. Register : To create a new client profile and save it.
2. Login : To login an existing profile, errors like wrong passwords are catch and treated.
3. NewChannel : To create a new channel and save it.

Directs, Channels and clients are collection to allow serialization.

Client got a current private chatter and a current channel.
But also a private chatters list and a list of channels he belongs.

3.3 use case diagram of the app :



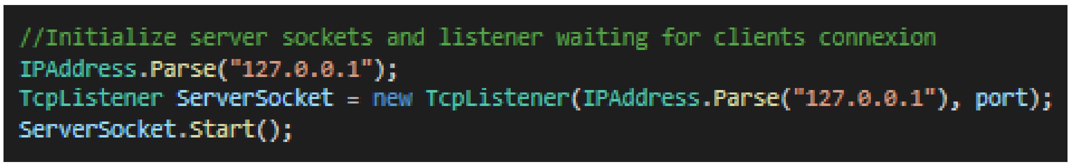
A use case diagram has to be simple and only show the global and main function of the app, that's why I put the basic and fundamental functionalities.
It's required to be register to login, of course.

And then it's required to be logged to use the interface.

4 Gobal Implementations

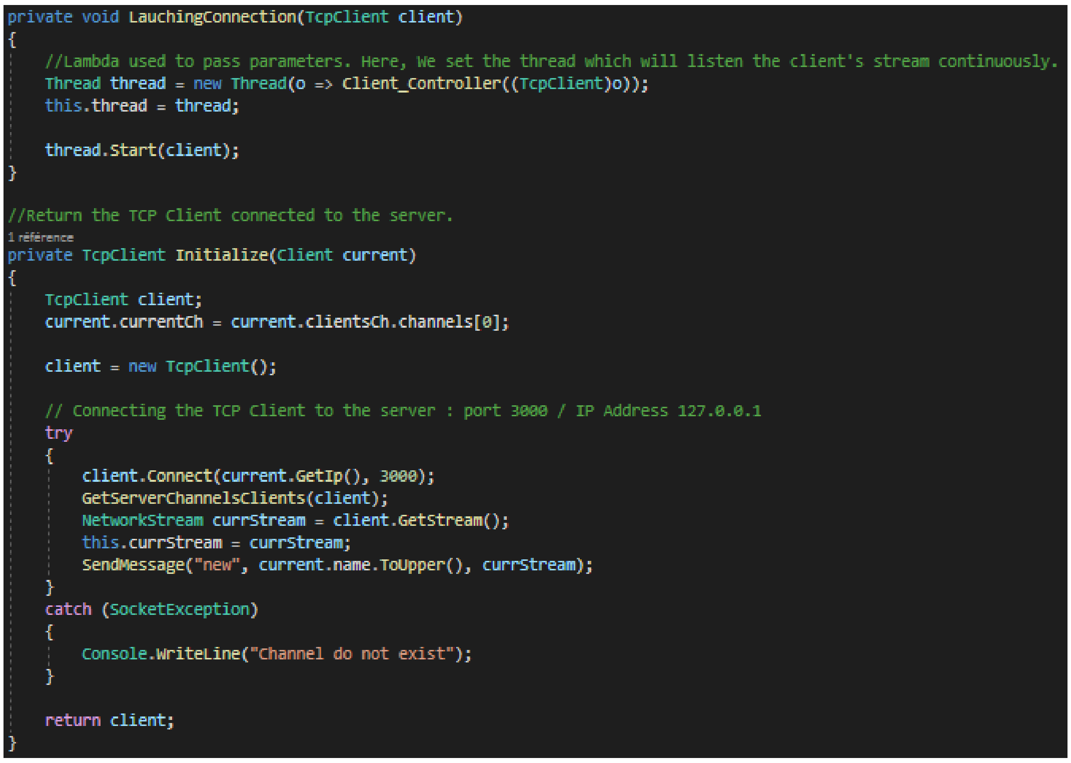
4.1 Server and Client connexions + multithreading :

That's the way I launch my server, as you recommanded in your example on moodle. After that, it's waiting for connexions.



```
//Initialize server sockets and listener waiting for clients connexion
IPAddress.Parse("127.0.0.1");
TcpListener ServerSocket = new TcpListener(IPAddress.Parse("127.0.0.1"), port);
ServerSocket.Start();
```

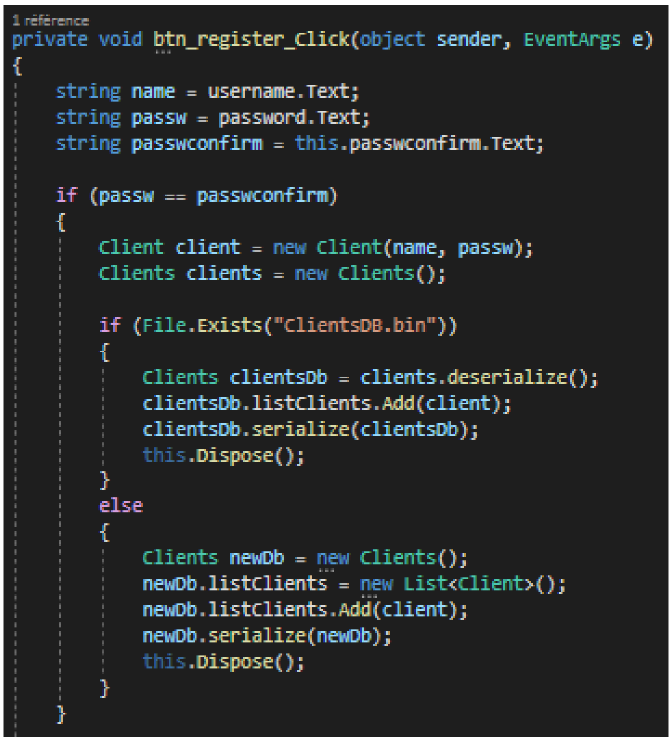
And that's how I use the multithreading system to get several clients. Moreover, there is the way I create a TCP Client and connect him to the server.



```
private void LauchingConnection(TcpClient client)
{
    //Lambda used to pass parameters. Here, We set the thread which will listen the client's stream continuously.
    Thread thread = new Thread(o => Client_Controller((TcpClient)o));
    this.thread = thread;

    thread.Start(client);
}

//Return the TCP Client connected to the server.
1 référence
private TcpClient Initialize(Client current)
{
    TcpClient client;
    current.currentCh = current.clientsCh.channels[0];

    client = new TcpClient();

    // Connecting the TCP Client to the server : port 3000 / IP Address 127.0.0.1
    try
    {
        client.Connect(current.GetIp(), 3000);
        GetServerChannelsClients(client);
        NetworkStream currStream = client.GetStream();
        this.currStream = currStream;
        SendMessage("new", current.name.ToUpper(), currStream);
    }
    catch (SocketException)
    {
        Console.WriteLine("Channel do not exist");
    }

    return client;
}
```

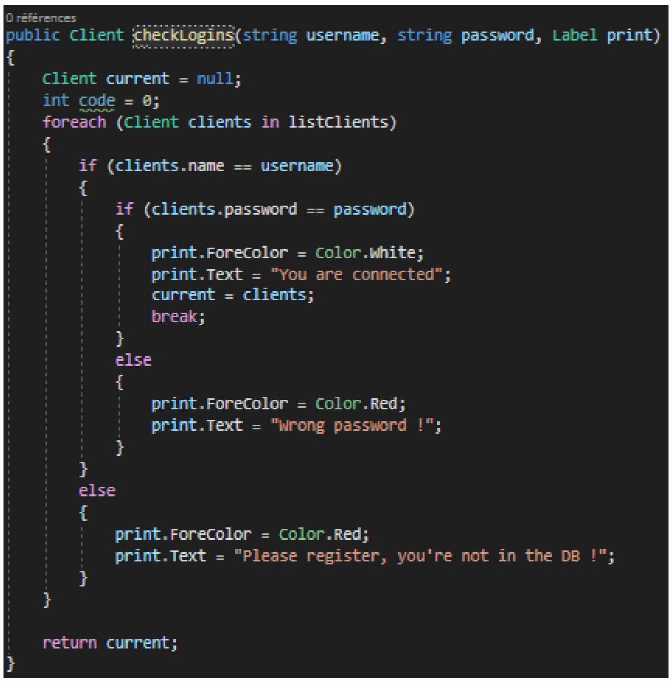
4.2 Register and Login :

This is how I register and then serialize a new profile.



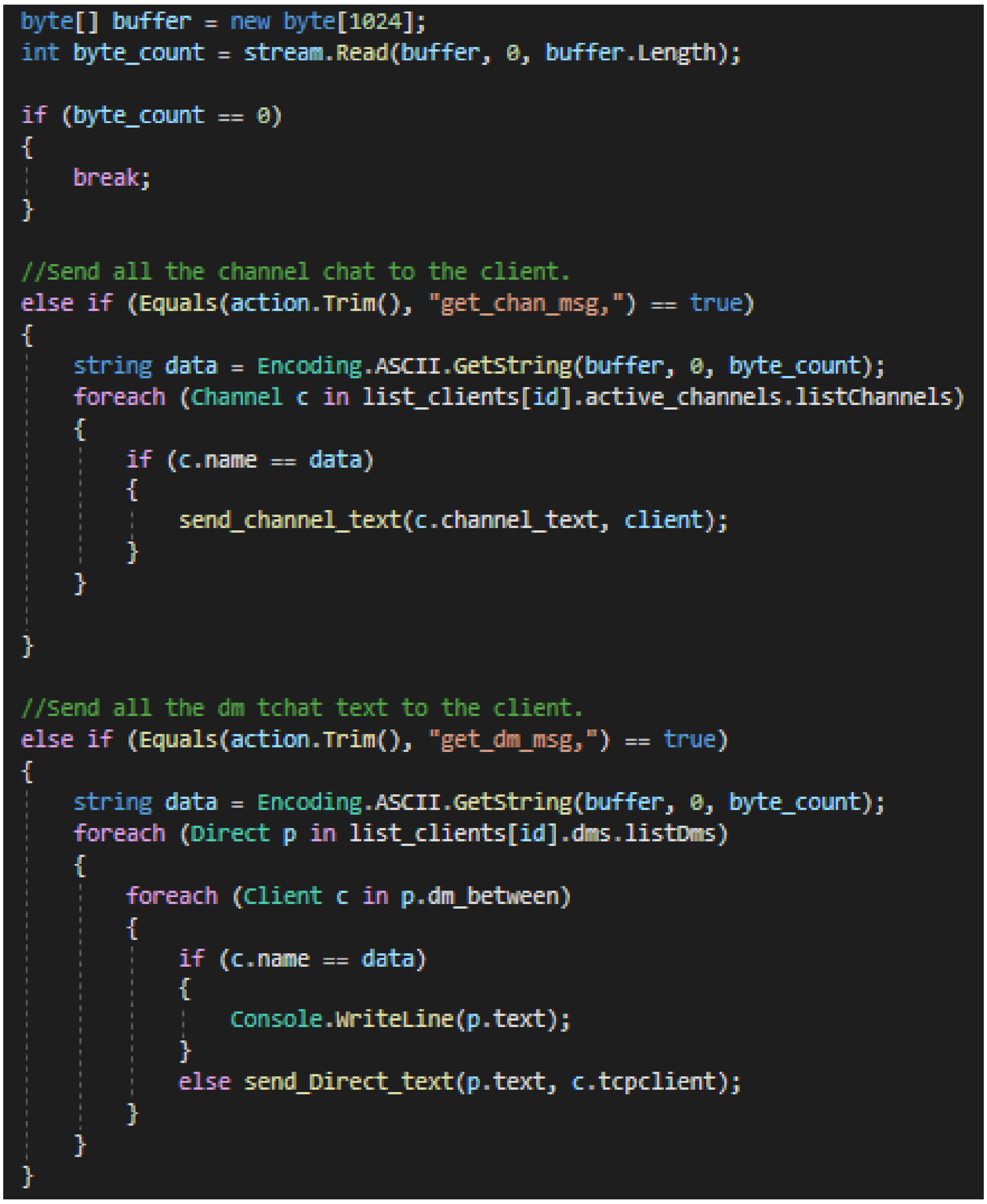
```
1 référence
private void btn_register_Click(object sender, EventArgs e)
{
    string name = username.Text;
    string passw = password.Text;
    string passwconfirm = this.passwconfirm.Text;

    if (passw == passwconfirm)
    {
        Client client = new Client(name, passw);
        Clients clients = new Clients();

        if (File.Exists("ClientsDB.bin"))
        {
            Clients clientsDb = clients.deserialize();
            clientsDb.listClients.Add(client);
            clientsDb.serialize(clientsDb);
            this.Dispose();
        }
        else
        {
            Clients newDb = new Clients();
            newDb.listClients = new List<Client>();
            newDb.listClients.Add(client);
            newDb.serialize(newDb);
            this.Dispose();
        }
    }
```

This is how a I check if logins are good and return the corresponding client in database to log.



```
0 références
public Client checkLogins(string username, string password, Label print)
{
    Client current = null;
    int code = 0;
    foreach (Client clients in listClients)
    {
        if (clients.name == username)
        {
            if (clients.password == password)
            {
                print.ForeColor = Color.White;
                print.Text = "You are connected";
                current = clients;
                break;
            }
            else
            {
                print.ForeColor = Color.Red;
                print.Text = "Wrong password !";
            }
        }
        else
        {
            print.ForeColor = Color.Red;
            print.Text = "Please register, you're not in the DB !";
        }
    }

    return current;
}
```

4.3 Controller's work (equal for each side) :



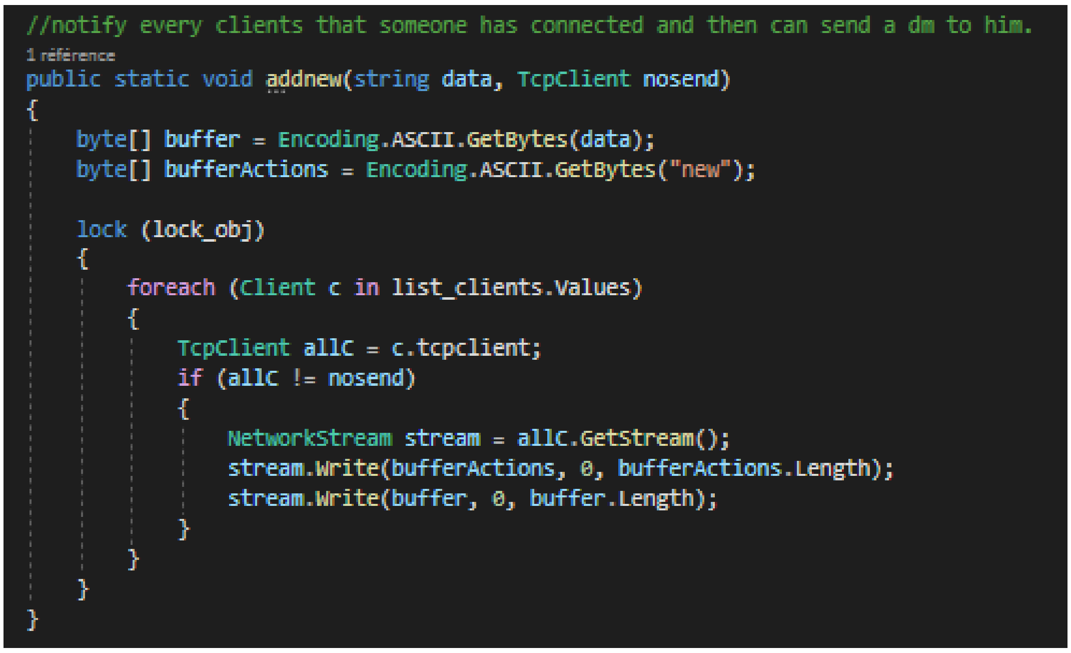
```
byte[] buffer = new byte[1024];
int byte_count = stream.Read(buffer, 0, buffer.Length);

if (byte_count == 0)
{
    break;
}

//Send all the channel chat to the client.
else if (Equals(action.Trim(), "get_chan_msg,") == true)
{
    string data = Encoding.ASCII.GetString(buffer, 0, byte_count);
    foreach (Channel c in list_clients[id].active_channels.listChannels)
    {
        if (c.name == data)
        {
            send_channel_text(c.channel_text, client);
        }
    }

}

//Send all the dm tchat text to the client.
else if (Equals(action.Trim(), "get_dm_msg,") == true)
{
    string data = Encoding.ASCII.GetString(buffer, 0, byte_count);
    foreach (Direct p in list_clients[id].dms.listDms)
    {
        foreach (Client c in p.dm_between)
        {
            if (c.name == data)
            {
                Console.WriteLine(p.text);
            }
            else send_Direct_text(p.text, c.tcpclient);
        }
    }
}
```

That's the way my controller work (Server side here). He's listening on the stream, and get an action to do from the client. Then, according to this action, it will do something and send the information "asked" to the client.

Precision : here I send data without "action" variable because the client is asking for some text. Then I send the text and of course the client's stream.

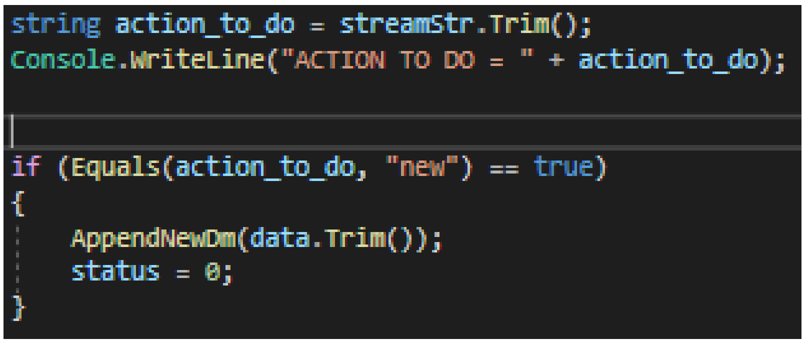
4.4 NotifyPropertyChanged methods :

This is the example where the server send a message to every clients when someone has just connected.



```
//notify every clients that someone has connected and then can send a dm to him.
1 référence
public static void addnew(string data, TcpClient nosend)
{
    byte[] buffer = Encoding.ASCII.GetBytes(data);
    byte[] bufferActions = Encoding.ASCII.GetBytes("new");

    lock (lock_obj)
    {
        foreach (Client c in list_clients.Values)
        {
            TcpClient allC = c.tcpclient;
            if (allC != nosend)
            {
                NetworkStream stream = allC.GetStream();
                stream.Write(bufferActions, 0, bufferActions.Length);
                stream.Write(buffer, 0, buffer.Length);
            }
        }
    }
}
```

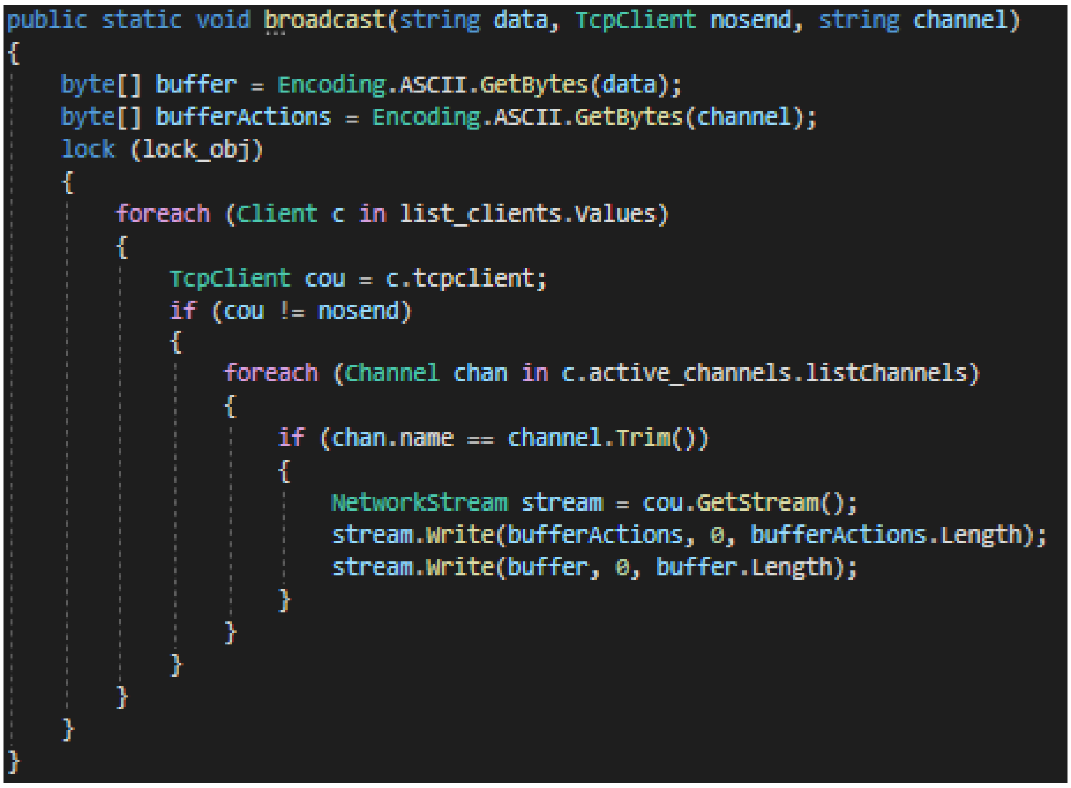
When the client got the info, then it makes an update of all the client interfaces to allow them to send a private message to the new chatters connected.



```
string action_to_do = streamStr.Trim();
Console.WriteLine("ACTION TO DO = " + action_to_do);


if (Equals(action_to_do, "new") == true)
{
    AppendNewDm(data.Trim());
    status = 0;
}
```

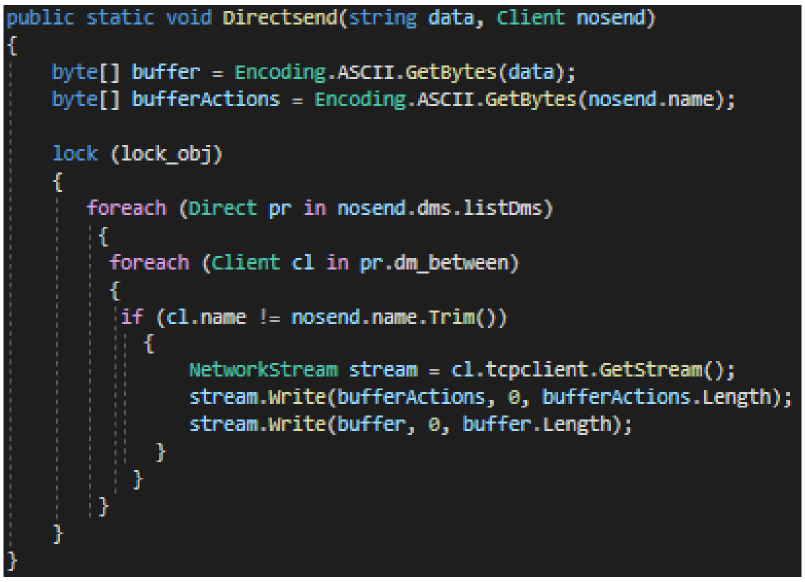
4.5 Sending messages :

To send messages on channels, I simply check every clients connected. And for each of them, I check if one currently belongs to the corresponding channel. If yes, then I send the message on the client's stream on the right channel.



```
public static void broadcast(string data, TcpClient nosend, string channel)
{
    byte[] buffer = Encoding.ASCII.GetBytes(data);
    byte[] bufferActions = Encoding.ASCII.GetBytes(channel);
    lock (lock_obj)
    {
        foreach (Client c in list_clients.Values)
        {
            TcpClient cou = c.tcpclient;
            if (cou != nosend)
            {
                foreach (Channel chan in c.active_channels.listChannels)
                {
                    if (chan.name == channel.Trim())
                    {
                        NetworkStream stream = cou.GetStream();
                        stream.Write(bufferActions, 0, bufferActions.Length);
                        stream.Write(buffer, 0, buffer.Length);
                    }
                }
            }
        }
    }
}
```

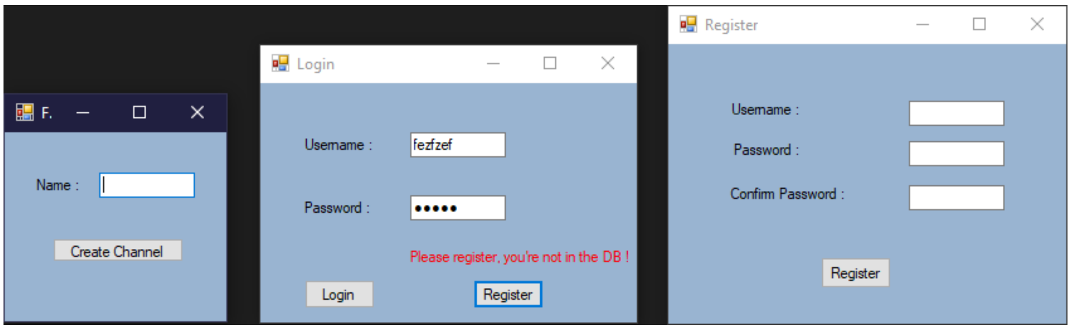
For direct messages, I browse the client's private messages list and then check if is currently chatting with someone (dmbetween). Then I send to the corresponding chatters but not including the sender.



```
public static void Directsend(string data, Client nosend)
{
    byte[] buffer = Encoding.ASCII.GetBytes(data);
    byte[] bufferActions = Encoding.ASCII.GetBytes(nosend.name);

    lock (lock_obj)
    {
        foreach (Direct pr in nosend.dms.listDms)
        {
         foreach (Client cl in pr.dm_between)
         {
          if (cl.name != nosend.name.Trim())
            {
                NetworkStream stream = cl.tcpclient.GetStream();
                stream.Write(bufferActions, 0, bufferActions.Length);
                stream.Write(buffer, 0, buffer.Length);
            }
          }
        }
    }
}
```

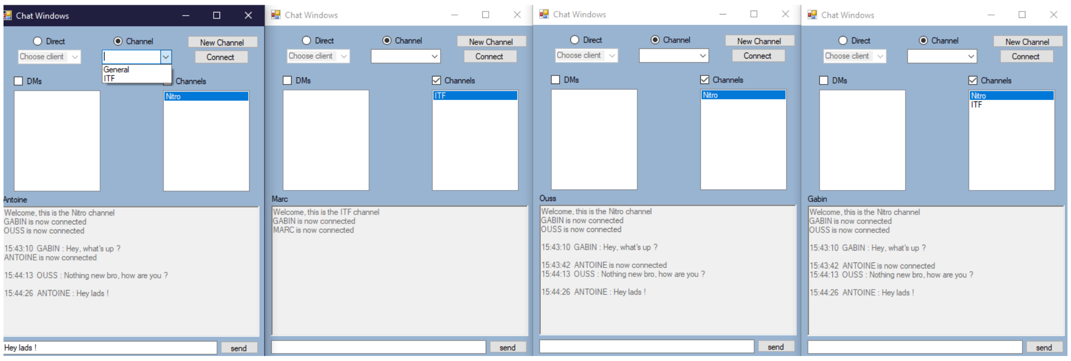
5 Screenshots of the app

5.1 Creation interfaces :



Username : fezfzef
Password : •••••
Please register, you're not in the DB !

5.2 Channels chat :



```
Antoine
Welcome, this is the Nitro channel
GABIN is now connected
OUSS is now connected

15:43:10  GABIN : Hey, what's up ?
ANTOINE is now connected

15:44:13  OUSS : Nothing new bro, how are you ?

15:44:26  ANTOINE : Hey lads !

Hey lads !
```

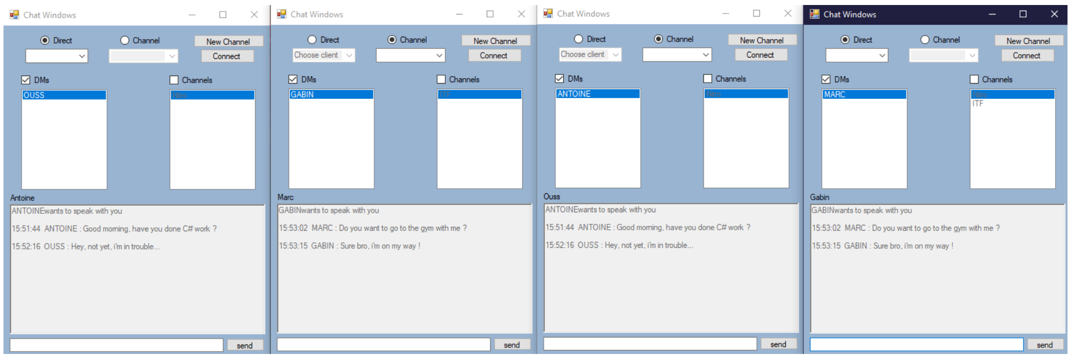
```
Marc
Welcome, this is the ITF channel
GABIN is now connected
MARC is now connected
```

```
Ouss
Welcome, this is the Nitro channel
GABIN is now connected
OUSS is now connected

15:43:10  GABIN : Hey, what's up ?

15:43:42  ANTOINE is now connected
15:44:13  OUSS : Nothing new bro, how are you ?

15:44:26  ANTOINE : Hey lads !
```

```
Gabin
Welcome, this is the Nitro channel
GABIN is now connected
OUSS is now connected

15:43:10  GABIN : Hey, what's up ?

15:43:42  ANTOINE is now connected
15:44:13  OUSS : Nothing new bro, how are you ?

15:44:26  ANTOINE : Hey lads !
```

Channel lists: General, ITF, Nitro

5.3 Private chat :



```
Antoine
ANTOINEwants to speak with you

15:51:44  ANTOINE : Good morning, have you done C# work ?

15:52:16  OUSS : Hey, not yet, i'm in trouble...
```

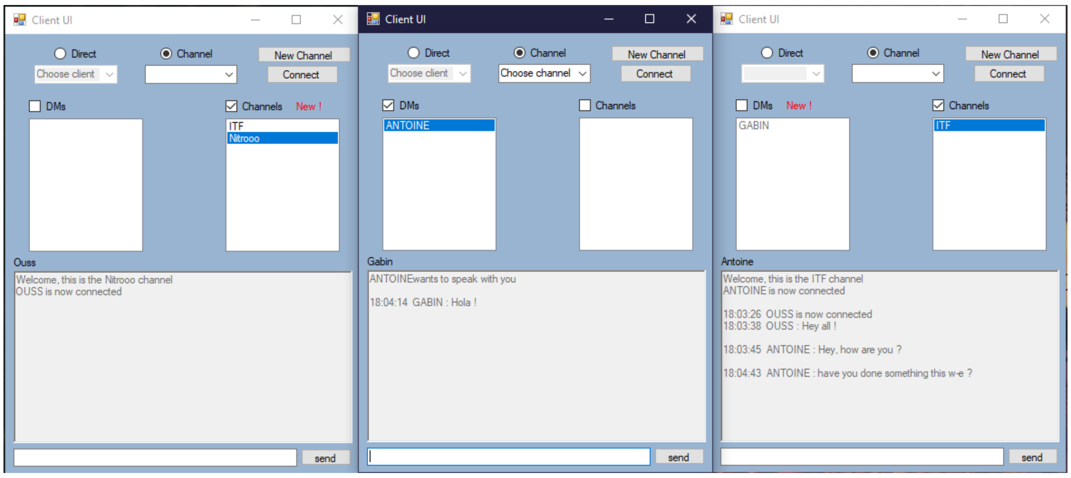
```
Marc
GABINwants to speak with you

15:53:02  MARC : Do you want to go to the gym with me ?

15:53:15  GABIN : Sure bro, i'm on my way !
```

```
Ouss
ANTOINEwants to speak with you

15:51:44  ANTOINE : Good morning, have you done C# work ?

15:52:16  OUSS : Hey, not yet, i'm in trouble...
```

```
Gabin
GABINwants to speak with you

15:53:02  MARC : Do you want to go to the gym with me ?

15:53:15  GABIN : Sure bro, i'm on my way !
```

DMs: OUSS, GABIN, ANTOINE, MARC

5.4 Notifications :



Ouss

Welcome, this is the Nitrooo channel
OUSS is now connected

Gabin

ANTOINEwants to speak with you

18:04:14 GABIN : Hola !

Antoine

Welcome, this is the ITF channel
ANTOINE is now connected

18:03:26 OUSS is now connected
18:03:38 OUSS : Hey all !

18:03:45 ANTOINE : Hey, how are you ?

18:04:43 ANTOINE : have you done something this w-e ?