

C# .net Chat Application

Marc Belleperche

M1 ITF 2021

Table des matières

[Functionalities 3](#_Toc29753366)

[Improvements 4](#_Toc29753367)

[Intellectual Contribution 5](#_Toc29753368)

[Client side 6](#_Toc29753369)

[Login 6](#_Toc29753370)

[Register 7](#_Toc29753371)

[Chat Window 8](#_Toc29753372)

[Server side 9](#_Toc29753373)

# Functionalities

My application provides the following functionalities:

* Connections between clients and server
* Sending message on a channel
* Connecting to many channels
* Clicking on a channel’s name you are connected show you the historic of the conversation since the beginning
* Connecting to person to have private conversation
* Clicking on a private conversation name you are connected show you the historic of the conversation
* You can disconnect from a server by sending “exit” in the conversation you want to leave
* Using serialisation to store users and their passwords so you can log
* Login and register interface
* A complete chat interface allowing you to send and receive message from as many people as you want.

# Improvements

Those are the few improvements I could have done:

* Create a new channel from a client side (I could do it, but I created already 5 channels, so I didn’t catch the utilities to do it)
* Create a new channel and add people directly inside of it
* Block a private connection (even if you can exit it)
* Notification system

# Intellectual Contribution

Very interesting projects where we can understand the basics of C# programming. Using serialization, multi-threading, lock, Tcp connections, Streams, switch case, …

I have spent approximatively 35 hours on this project and really enjoyed it.

If you have any question about the code, please Email me at [marc.belleperche@efrei.net](mailto:marc.belleperche@efrei.net).

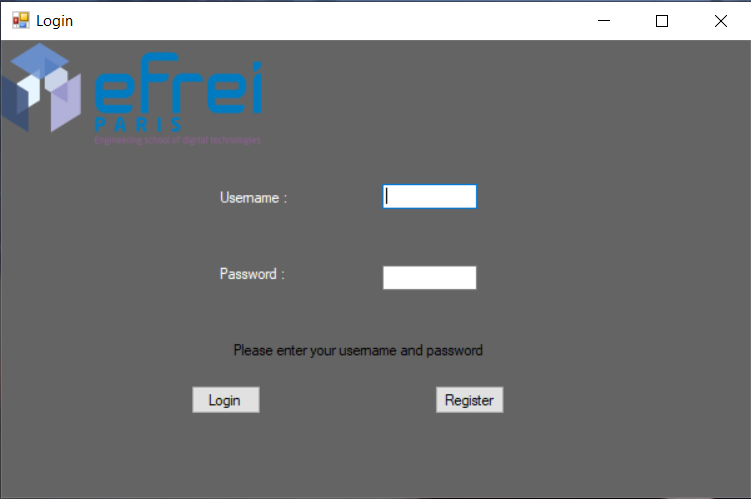
You can see the whole progress of my project on my GitHub : <https://github.com/MarcBelleperche/Chat_Application/> .

You may read the ReadMe file before launching my application.

# Client side

The client-side project has been created to handle the client connection to the server. For that I have created 3 windows and 6 classes. We are going to go threw them.

## Login



Username textbox

Register button

Information label

Password textbox

Login button

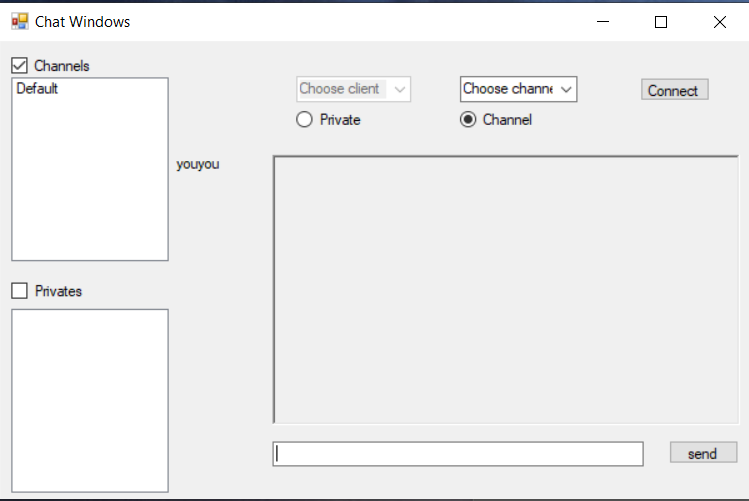
This window has been created allow the client to connect with a name and a password. I am using the classes **client** and **clients**. The program check with the function check which is in the **clients** class. This function deserialise the list of clients stored in the project (Clients.bin). Then I check if the client exists for the name entered then if the password associated is the good one. If it is all good the function returns the client object, then launch the chat window. If the name is not good a message is printed as well as if the password associated is not good. You also have an error message if you didn’t launch the server project before. If you want to connect with a new client, you can Register by clicking on the button register.

## Register

This window has been created to allow the user to create a new client. Here, the principal function deserialise the Clients.bin file then add the new client to the list before reserialising it. To add a new client, you have to enter a username and a password then verify your password.

If the name is entered and the two passwords are the same you are good to go, the new client has been created and the window will automatically close.

## Chat Window



Text to send

Send button

List of channels from the server

List of clients connected to the server

person list which you are connected

Channels list which you are connected

The Chat window is the base of the project. First I defined a special constructor taking a **Client** in parameter (it is the client return by the check function in **Clients**). Here I first connect the client to the server using the IPAdress 127.0.0.1 and the port 5000. If the connexion is established, the client aske to the server the list of channels and the list of clients connected to store them in the following combo boxes creates to that (channel\_to\_connect & client\_to\_connect). Then the client launch a thread in which, it is waiting for incoming messages from the server.

When I click on the button send, the message in the textbox is send to the server with other information. If the checkbox channels is selected, then it sends the message witch the name of the channel, the message itself and the pre-message “msg” to inform the server that a message is incoming. If the privates checkbox is check then the name of the client you want to communicate is send instead of the channel name.

If the word “exit” is entered, then you are disconnected from the channel.

Each client possesses a list of channels, a list of privates (conversation with only one person), and an active\_channel and a active\_private attribute. When you selected a channel or a private in the list box objects (left of the window), it sends a request to the server to get the conversation of this channel or private.

If you want to connect a new channel or someone, you can select it in the combo boxes then press connect. It sends a request to the server to inform you want to make a new connexion.

Each message as a pre send then one or several other send. The pre send is used to inform what type of action the server has to execute.

It is the same for the client side, it is waiting for incoming data to know which action it has to execute.

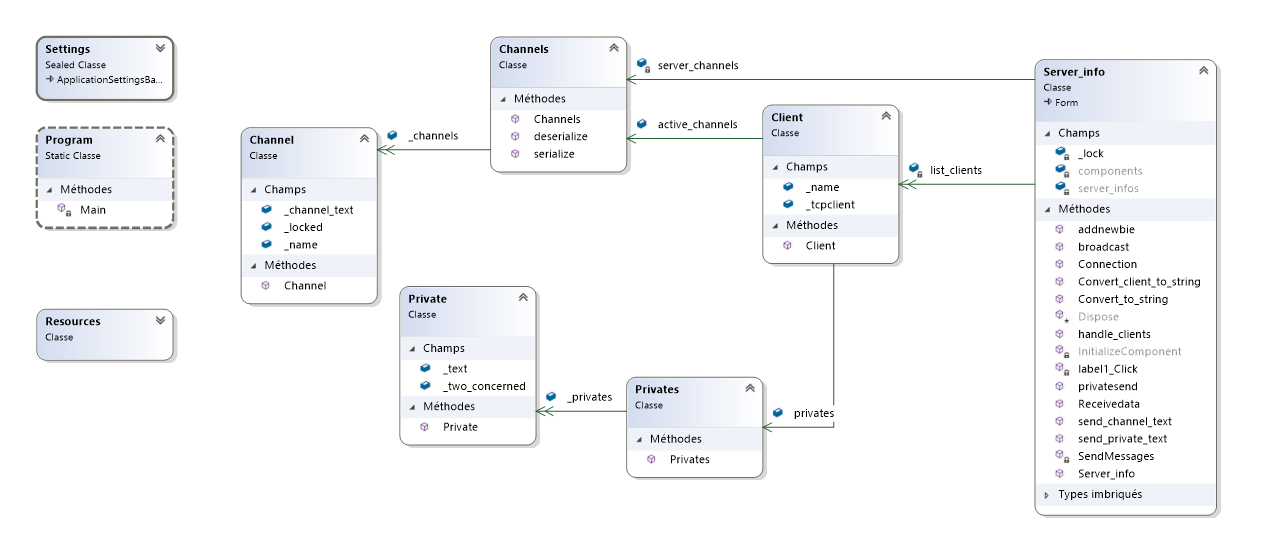
# Server side

The server side has no graphic interface. It uses the console output to show informations. It’s always waiting for new incoming client connection. Once a client connects to the server, it TcpClient is stored in a client class stored itself in a dictionary and so associated to a number. A thread is launched and is waiting for datas. The switch case is the main part of this project. Each time the client wants to execute a request to the server, it sends at least 2 messages. The first is interpreted and allows the switch to know which action he has to execute.

There is nothing more to say for the functionalities of the server side if I don’t want to run through the code that is already commented.

# Class Diagrams

Server side:



Client side:

