**ProEP**

**Chat application with Peer-To-Peer connection**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

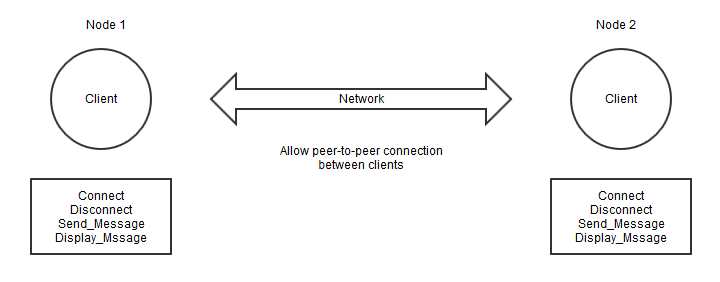
# 

# **First Increment**

## Goal of this increment

Create a simple chat application that works on a Peer-To-Peer network.  
Any communication protocol can be use.

## Architecture



For the first increment, we need only 2 nodes that know each other.

The minimum required to make a communication is 2 people, thus we need at least 2 clients.

A session is the period during the login and the logout of a client.

We don’t use the server as a bridge between clients.  
The network is just the bridge between clients. It doesn’t have any function except link together all logged in clients.

Function on each client :

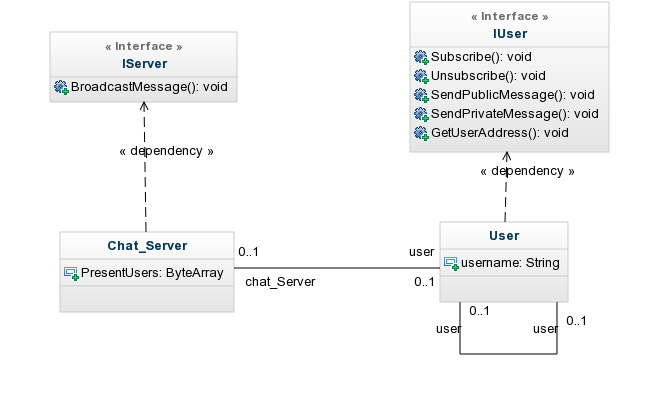
* Login / logout itself on the application
* Send messages to the other client
* Display all messages the client receive during this session from the other client.

## 

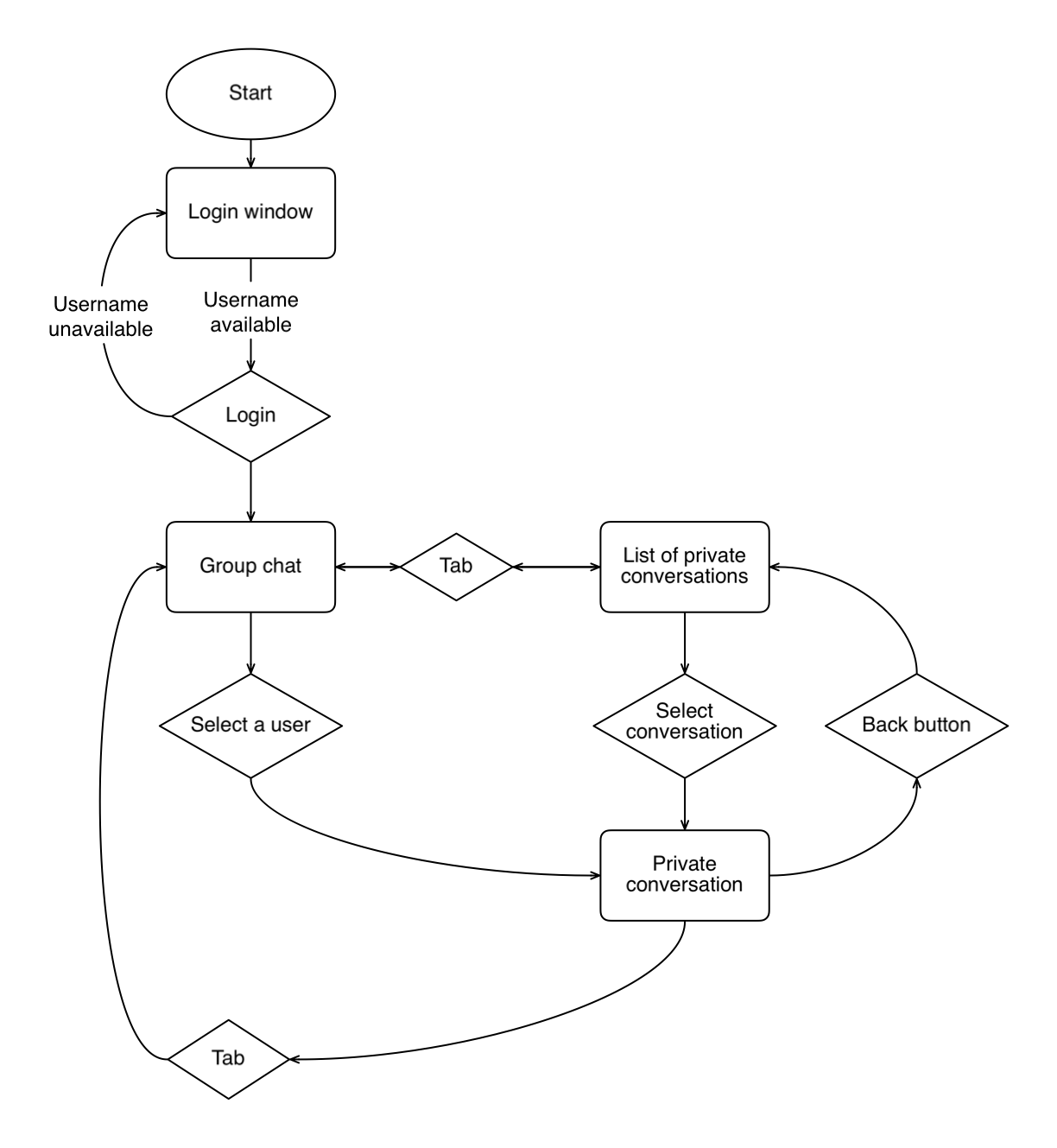
## Communication between nodes

For the first increment, we are going to use internet connection (Wi-fi).  
It’s easier and there is many guide / tutorials about it on internet

## Class Diagramm



## State Diagramm



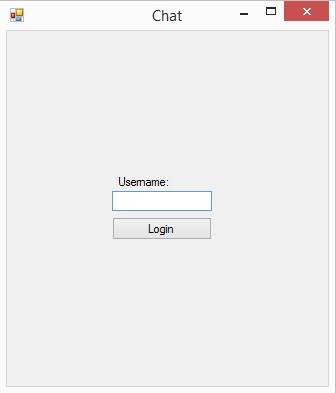
## Software parts

For the client, we are going to use :

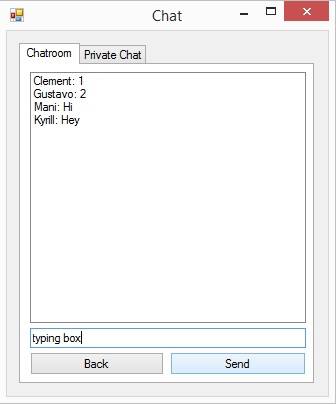
* Modules
  + ?
* Namespace
  + ?
* Classes
  + IClient => A client interface that contains everything required to create concrete client class
  + Local\_Client => The concrete client class that execute functions

## Graphic User Interface samples

Login to the application.



Main screen: public chat room.



If the user clicks on a username, it opens a new conversation with this person on the “Private Chat” tab.



If the user click on the “Private Chat” tab, all the previous private conversation the user had is present and he can choose one of them and continue it.  
