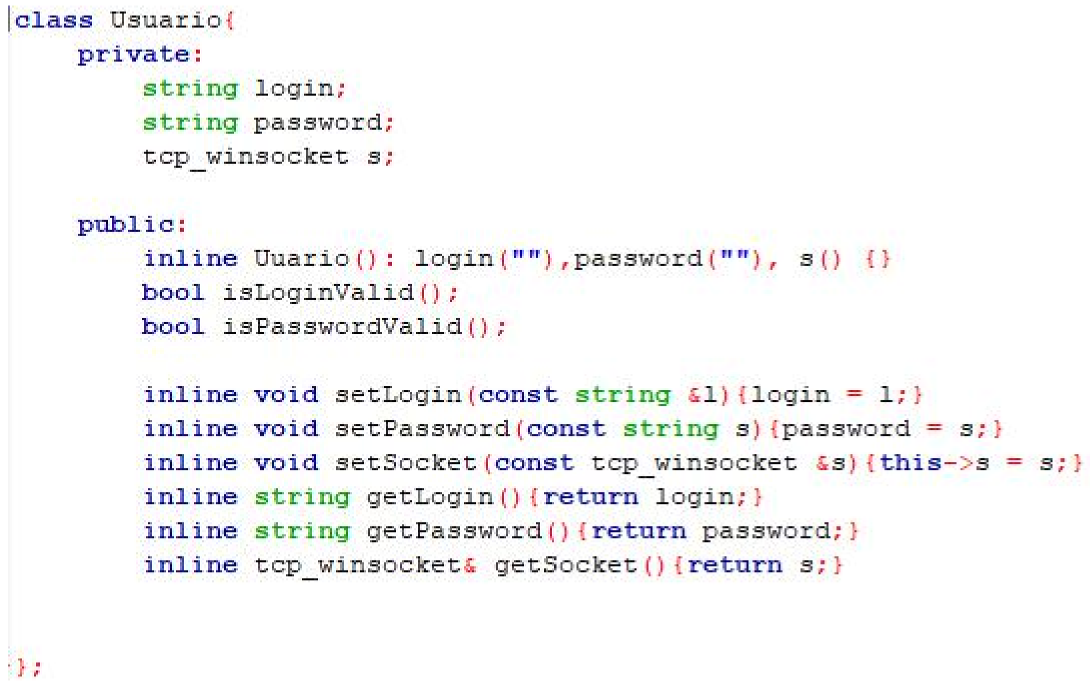
WhatsProg

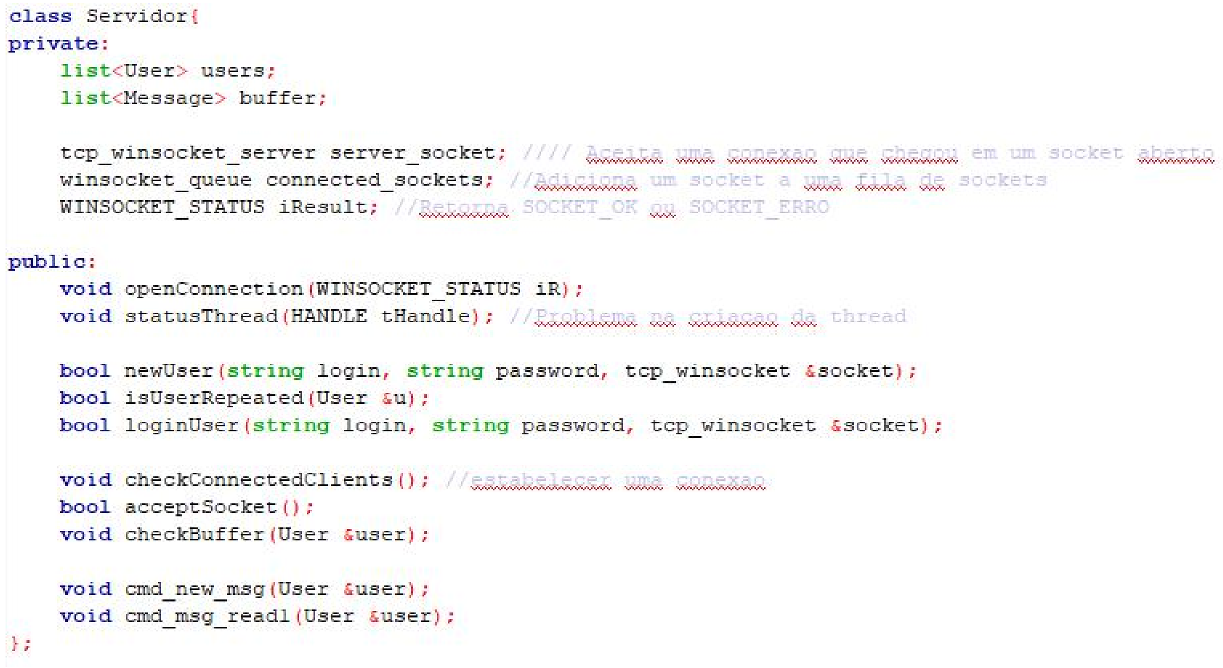
Rony de Sena Lourenço

A classe Usuário



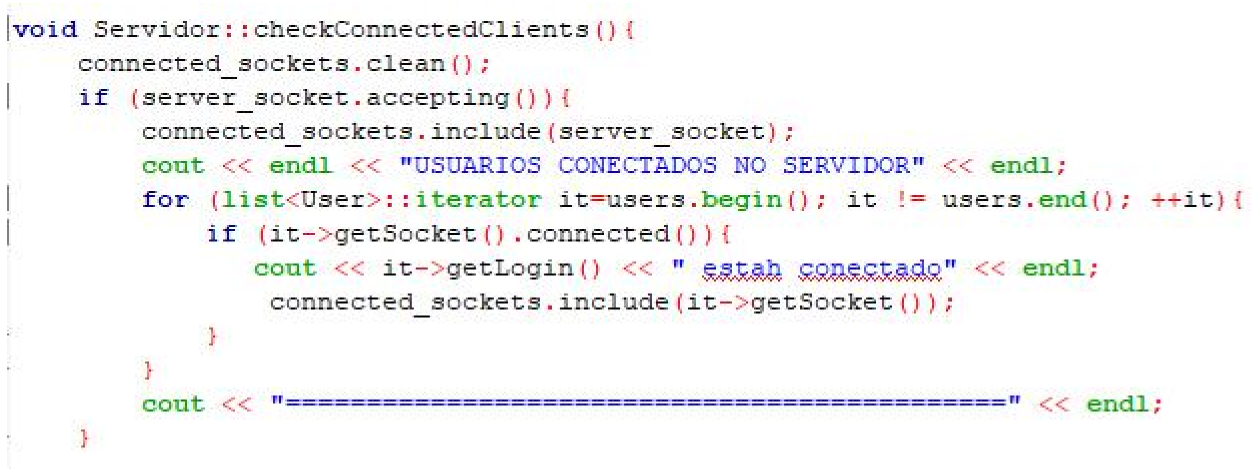
```
class Usuario{
    private:
        string login;
        string password;
        tcp_winsocket s;

    public:
        inline Uuario(): login(""),password(""), s() {}
        bool isLoginValid();
        bool isPasswordValid();

        inline void setLogin(const string &l){login = l;}
        inline void setPassword(const string s){password = s;}
        inline void setSocket(const tcp_winsocket &s){this->s = s;}
        inline string getLogin(){return login;}
        inline string getPassword(){return password;}
        inline tcp_winsocket& getSocket(){return s;}


};
```

A classe Servidor



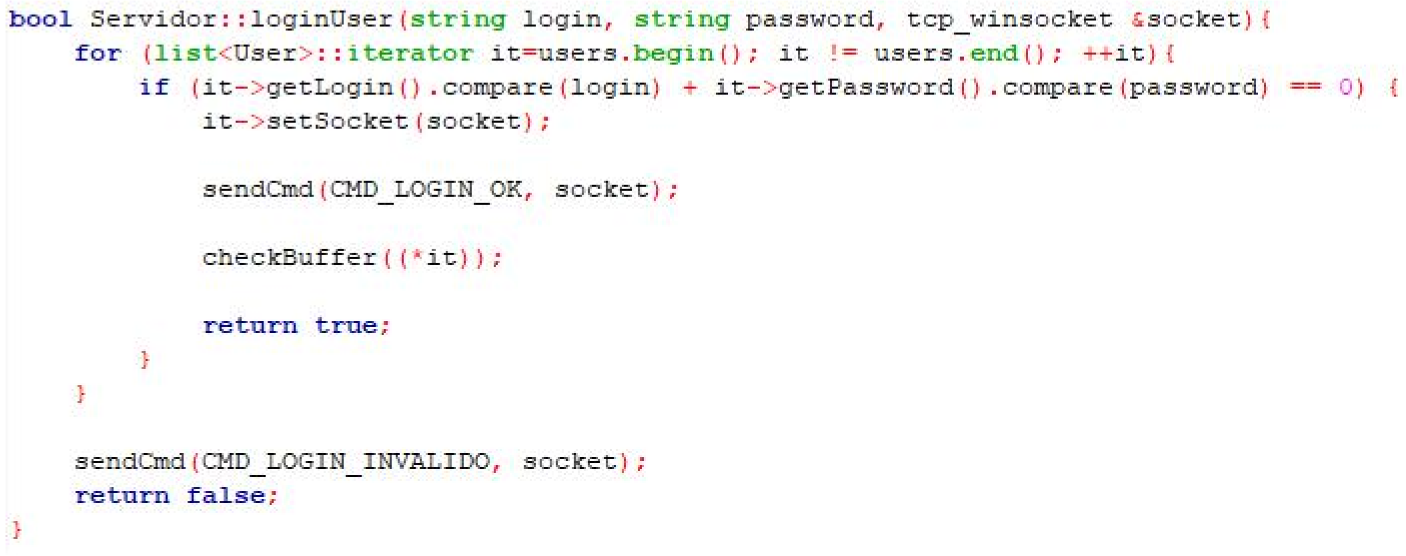
```
class Servidor{
private:
    list<User> users;
    list<Message> buffer;

    tcp_winsocket_server server_socket; //// Aceita uma conexao que chegou em um socket aberto
    winsocket_queue connected_sockets; //Adiciona um socket a uma fila de sockets
    WINSOCKET_STATUS iResult; //Retorna SOCKET_OK ou SOCKET_ERRO

public:
    void openConnection(WINSOCKET_STATUS iR);
    void statusThread(HANDLE tHandle); //Problema na criacao da thread

    bool newUser(string login, string password, tcp_winsocket &socket);
    bool isUserRepeated(User &u);
    bool loginUser(string login, string password, tcp_winsocket &socket);

    void checkConnectedClients(); //estabelecer uma conexao
    bool acceptSocket();
    void checkBuffer(User &user);

    void cmd_new_msg(User &user);
    void cmd_msg_readl(User &user);
};
```

Trecho de função



```
void Servidor::checkConnectedClients(){
    connected_sockets.clean();
    if (server_socket.accepting()){
        connected_sockets.include(server_socket);
        cout << endl << "USUARIOS CONECTADOS NO SERVIDOR" << endl;
        for (list<User>::iterator it=users.begin(); it != users.end(); ++it){
            if (it->getSocket().connected()){
               cout << it->getLogin() << " estah conectado" << endl;
                connected_sockets.include(it->getSocket());
            }
        }
        cout << "===============================================" << endl;
    }
```

Trecho de função



```
bool Servidor::loginUser(string login, string password, tcp_winsocket &socket){
    for (list<User>::iterator it=users.begin(); it != users.end(); ++it){
        if (it->getLogin().compare(login) + it->getPassword().compare(password) == 0) {
            it->setSocket(socket);

            sendCmd(CMD_LOGIN_OK, socket);

            checkBuffer((*it));

            return true;
        }
    }

    sendCmd(CMD_LOGIN_INVALIDO, socket);
    return false;
}
```