

# Spyke (Babel Project)

La Pintade

5 Octobre 2015 - 8 Novembre 2015

## Contents

<b>1</b>	<b>CLIENT</b>	<b>2</b>
1.1	GUI . . . . .	2
1.2	NETWORK . . . . .	4
1.3	Network Client Handler . . . . .	5
1.4	Audio Encoder . . . . .	5
1.5	Audio Handler . . . . .	5
1.6	Thread . . . . .	5
1.7	Contact . . . . .	5
<b>2</b>	<b>SERVER</b>	<b>6</b>
2.1	LoginWidget . . . . .	6
2.2	MainWidget . . . . .	7
2.3	SignupWidget . . . . .	8
2.4	Home . . . . .	8
2.5	Contact . . . . .	8
2.6	Conversation . . . . .	8
2.7	Notifications . . . . .	8

# 1 CLIENT

## 1.1 GUI

---

```
class PTUser
{
private:
    class User
    {
        std::string _username;
        std::string _password;
        std::string _objectId;
        std::list<Contact*> _contactList;
    public:
        User();
        ~User();
        const std::string &getUsername() const; // Get the username of
            the current user
        const std::string &getPassword() const; // Get the password of
            the current user
        const std::string &getObjectId() const; // Get the objectId of
            the current user
        void setUsername(); // Set the username of a new user
        void setPassword(); // Set the password of a new user
        void setObjectId(); // Set the object ID gotten from the server
        std::list<Contact*> getContactList() const; // Return the
            contact list of the current user
        const std::string &getLocation() const; // Return the location
            info of the current user
        void addContact(Contact&); // Add a new contact to the contact
            list of the current user
    };
    User _currentUser;
    NetworkServerHandler server;
    std::string _ipServer;
public:
    PTUser();
    ~PTUser();
    User& currentUser();
    template<typename T>
    void signUp(T &obj, void(T::*call)(int), const std::string
        &username, const std::string &password, const std::string &i);
    void logUser(T &obj, void(T::*call)(int), const std::string
        &username, const std::string &password, const std::string &ip);
    // The function logUser will call the server to check if the
        user exist, if not it will call the callback function with
        an error value set to 1, else it will call with an error
        value set to 0.
}
```

```
};  
extern PTUser g_PTUser;
```

---

## 1.2 NETWORK

---

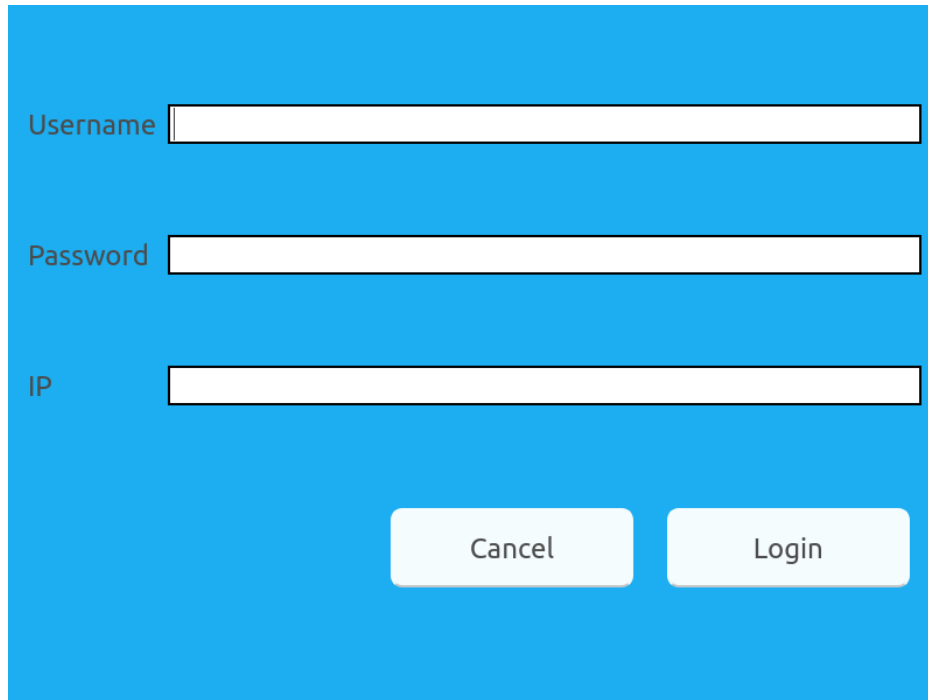
```
class NetworkServerHandler : public QObject
{
    Q_OBJECT
private:
    QObject *parent;
    QTcpSocket *_socket;
    std::string _host;
    unsigned int _port;
public:
    NetworkServerHandler(QObject *parent = 0);
    ~NetworkServerHandler();
    int start(const std::string &host, unsigned int port); // Launch the
        connection between server and client /\ Return -1 if failed, 0
        if success.
    void setHost(const std::string &); // Set the host of the server
    void setPort(unsigned int); // Set the port of the server
    const std::string &readSocket(); // Read data from server
    void writeSocket(const std::string&); // Write data on server
private slots:
    void readyRead(); // Callback function from QT /\
    void connected(); // Callback function from QT /\ Is called when
        connection has succeed
};
```

---

- 1.3 Network Client Handler**
- 1.4 Audio Encoder**
- 1.5 Audio Handler**
- 1.6 Thread**
- 1.7 Contact**

## 2 SERVER

### 2.1 LoginWidget



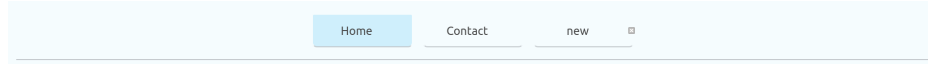
A login form titled "LoginWidget" with a blue background. It contains three input fields: "Username", "Password", and "IP". Each field is a white rectangle with a black border. Below the fields are two buttons: "Cancel" and "Login", both with rounded corners and a light blue background.

Username

Password

IP

## 2.2 MainWidget

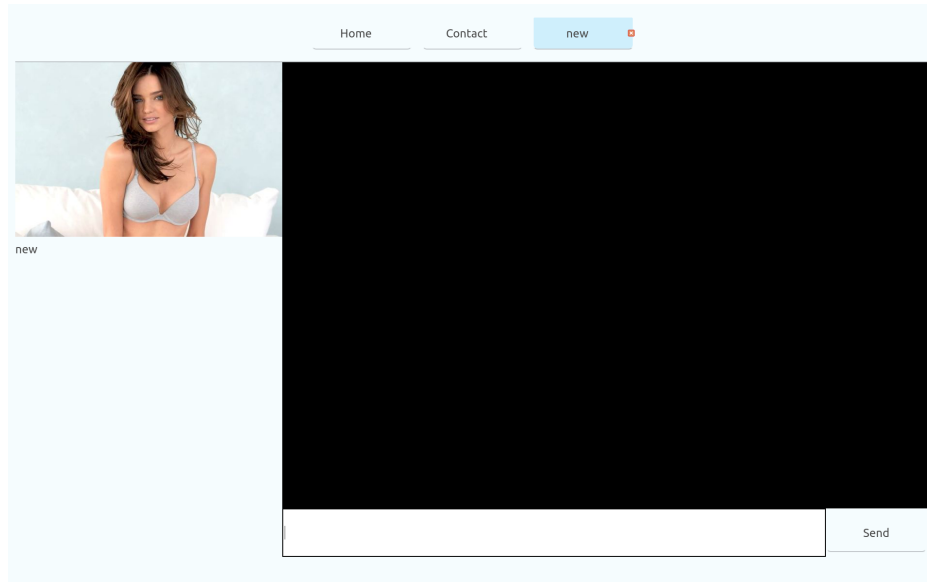


### 2.3 SignupWidget

### 2.4 Home

### 2.5 Contact

### 2.6 Conversation



### 2.7 Notifications