东软教育科技集团

项目实训总结报告

项 目 名 称: OnPointLink

学 校 名 称: 北京理工大学

项 目 讲 师: 薛静锋

姓 名: 王钟骐

学 号: 1120211096

实 训 日 期: 2023年8月21日-8月31日

**1 引言**

1.1项目概述

1.1.1 项目目的

设计一款基于linux操作系统的即时通讯软件，包括登录/注册、用户检索/分组、好友管理、一对一聊天/群聊、小型文件传输、消息通知、修改个人信息等基本功能；

软件分为客户端、网络模块、后端数据库三部分，通过局域网分发/接收数据实现功能，有较为完整的UI设计。

1.1.2 项目背景

通过前两年课程内的学习，我们初步掌握了程序开发所需的常用理论、部分算法；在此基础上，进行一定强度的项目开发，将理论应用于实践是必要的。

将即时通讯软件作为项目开发方向有如下原因。随着时代发展，人们对于通讯即时性的要求也越来越高；同时，即时通讯软件是业界开发重点之一，相关领域的积累也较为丰富，且可以用较为熟悉的编程语言与编译器（例如c++与Qt等）实现。

1.1.3 项目意义

1.体会软件整体设计的全流程，包括需求理解、概要设计、程序开发、测试、发布的整体流程。

2.了解linux系统上的软件开发方法。

3.使用并熟悉跨系统开发工具Qt的部分方法，包括多线程/多进程方法、socket通信编程等。

4.快速学习新知识并在项目中应用，既能独立解决问题，也能高效进行团队合作。

5.提高设计初级UI的能力。

1.1.4 参考资料

1.Qt Creater 快速入门，霍亚飞，2017.1，3.0

2.东软Qt5教程，东软睿道，1.0

3.Qt及Qt Quick开发实战精解，霍亚飞，2012，1.0

4.最新QT从入门到实战完整版|传智教育，传智教育，2019-06-04，1.0

5.Qt自带的Example与帮助文档

1.2 运行环境

运行系统：VMWare 16.0.0-Ubuntu 18.02、Windows11

开发工具：Qt 5.12.8 + MingW32 64bit/gdb+gcc + Qmake

硬件支持：全部Intel/AMD x86/x64的处理器

**2 需求分析**

**1.登录注册（担当者：王钟骐）**

**登录界面：**

1. ID框输入ID，并且有下拉菜单可以显示之前登录过的用户ID

（由于昵称不唯一，因此不能使用昵称登录）

2.密码框输入密码，但是不直接显示密码

3.复选框可以选择是否记住密码和是否自动登录

4.按下登录按钮登录

5.用户可以自行选择服务器的IP地址

6. 服务器无法链接时给予提示信息

7.按下注册按钮进入注册页面

**注册界面：**

1.注册界面要输入昵称、密码、确认密码，两次输入密码相同则注册成功

2.注册成功后显示ID（自动分配）、昵称、密码，跳转到登录界面

**2. 用户操作（担当者：申家芮）**

**用户检索：**

1.支持通过ID、昵称检索

2.用ID搜索，弹出对应用户信息，若无对应用户则报错

3.用昵称搜索，弹出符合条件的用户列表，点击某个用户即可弹出用户信息

**添加/删除好友：**

1. 在搜索到用户信息后，有发送好友申请选项，弹出发送好友申请窗口

2. 好友申请含打招呼消息、好友分组、好友备注。

3. 如果被申请人设置为“无需验证”，则申请直接通过；如果设置为“需要验证”，则会经过被申请人手动同意后申请通过。

4.如果被申请人设置为“回答问题”，则正确回答问题后自动通过，否则自动拒绝。

（以上两条在“设置”窗口进行设置）

5.主界面有“好友申请”按钮，收到申请时发出提示，点击按钮进入受理窗口，可以选择接受或拒绝

**好友分组/拉黑：**

1.设定黑名单，可以查看已经拉黑的用户，在这里可以解除拉黑。

2.添加黑名单方法类似添加好友，但是不用对方同意。

3.分组在好友列表当中进行操作，有一个默认分组。

**3.聊天（担当者：王启贤）**

**1对1聊天：**

1. 用户在好友列表当中选择用户，双击打开即可进入聊天窗口。一个聊天就是一个窗口。

2. 消息输入框输入文本后，点击发送按钮、或按下发送快捷键发送。消息输入框支持输入表情。

3. 发出的消息首先会存储在服务器上，然后如果对方用户在线，则进行消息同步；否则存储在服务器上一段时间，如果超出了存储时间仍然没有上线，则收不到该消息。

4.自己发送到消息和收到他人到消息以气泡形式显示在消息显示窗

**群聊：**

1. 点击创建群聊按钮创建群聊，需要至少三个用户，直接拉进群。创建后生成群聊ID，此后群聊的ID不变。

2. 建群者为群主，可以进行群主转让。只有群主可以进行群的解散操作。

3. 在群设置页面，群主可设置管理员，管理员人数有上限。群主、管理员均可进行加群验证、单人禁言、全体禁言、解除禁言、解除全体禁言、设置群名、设置群简介、管理群文件。

4. 群设置界面有查看群用户、消息记录、删除群聊等功能

5. 群文件设置群文件系统。

**文件传输：**

1. 在聊天消息框里面设置一个文件传输功能键，点击之后进入本地文件目录浏览界面，选择一个文件即可上传。

2. 文件与消息一样，会先保存在服务器，只不过文件的云存储时间较短。

3. 群聊中文件会有群文件系统管理。

4. 文件图标后面会有一个下载按钮，如果已下载则替换为一个打开文件目录的按钮。

5.文件下载在默认位置，在设置界面可以自行设置文件存储位置

**聊天记录：**

1. 有查找记录按钮，弹出查找对话框，支持查找功能，根据文本重合度、文件名重合度、文件名后缀名查找。

2. 支持多选+转发功能。

3. 聊天框当中点击一个按钮查看历史消息。

**消息通知：**

1. 如果接收到了新消息，则会进行闪烁提示。

2. 可以设置提示音/只接收不提示

**4. 用户设置（担当者：祝文轩）**

**基本信息设置：**

1.用户可以更改自己的昵称和密码

2.用户可以设置自己的身高、体重、年龄、学历、居住地等信息

（在设置界面）

**头像设置：**

1.头像在主界面和聊天对话框中显示，在主界面点击头像，可以自定义头像（系统内置头像或从本地文件上传）

**个性签名：**

1.在主界面显示个性签名，点击变成输入框，可以自定义用户签名

**隐私设置：**

1.用户可以设置其他人能否访问自己的隐私信息

2.隐私信息包括身高、体重、年龄、学历、居住地等个人信息

**5. 账号信息存储（担当者：黄森奕）**

**账号信息数据库：**

1. 使用mysql作为数据库引擎，存储账户信息。

**聊天记录存储：**

1. 存储的聊天信息包括但不限于：时间、发送人、对象（用户，群聊）、类型（文本消息、文件）、内容（文本的话就是文本，文件的话就是一个文件链接）

**服务器文件存储模块：**

1. 1对1聊天的文件为临时文件，存储按照时间划分文件夹在服务器当中进行存储。

2. 群聊的文件设在一个文件夹当中，当中按照群文件管理系统的方式存储。

3. 使用服务器自带的文件管理系统，服务器存储文件路径，并针对不同文件采用不同存储控制，如聊天信息定时删除等。

**6.附加功能（担当者：赵荣翰）**

1.在群聊界面，群主和管理员可以放歌，群员进入群聊页面则可以听歌

2.可以设置他人的备注，设置了则不显示他的昵称而是显示备注

**3 概要设计**

客户端：

· 客户端前端UI：负责显示内容在屏幕上，要求：美观、实用。

· 客户端核心：负责对内存的操作，连接程序的其他部分，任何信息流通都需要经过内存。

· 文件缓存系统：存储管理本地存储的信息，例如历史消息记录与传输的文件，并保存已有的账号密码信息。

网络传输：

· 网络模块：负责客户端和服务器之间的通信，包括但不限于对服务器端发起请求、接收从服务器端发出的信息。

· 传输模块：负责将信息在客户端与服务端的核心间原封不动地转移。

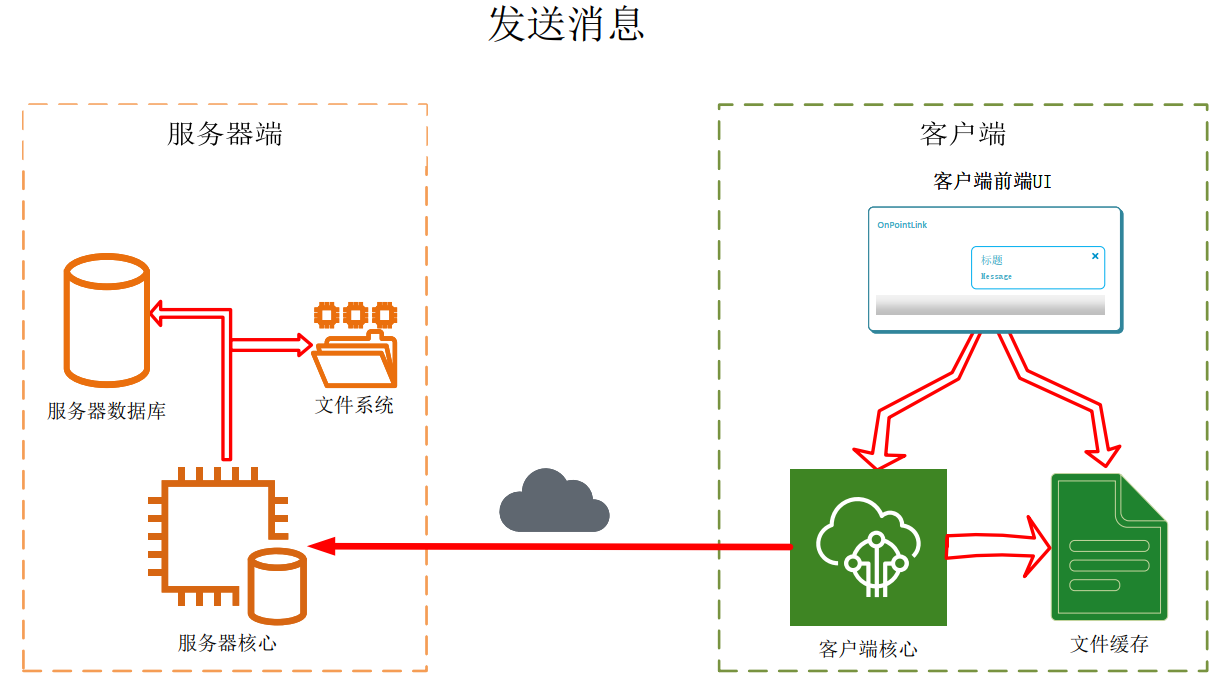
服务端：

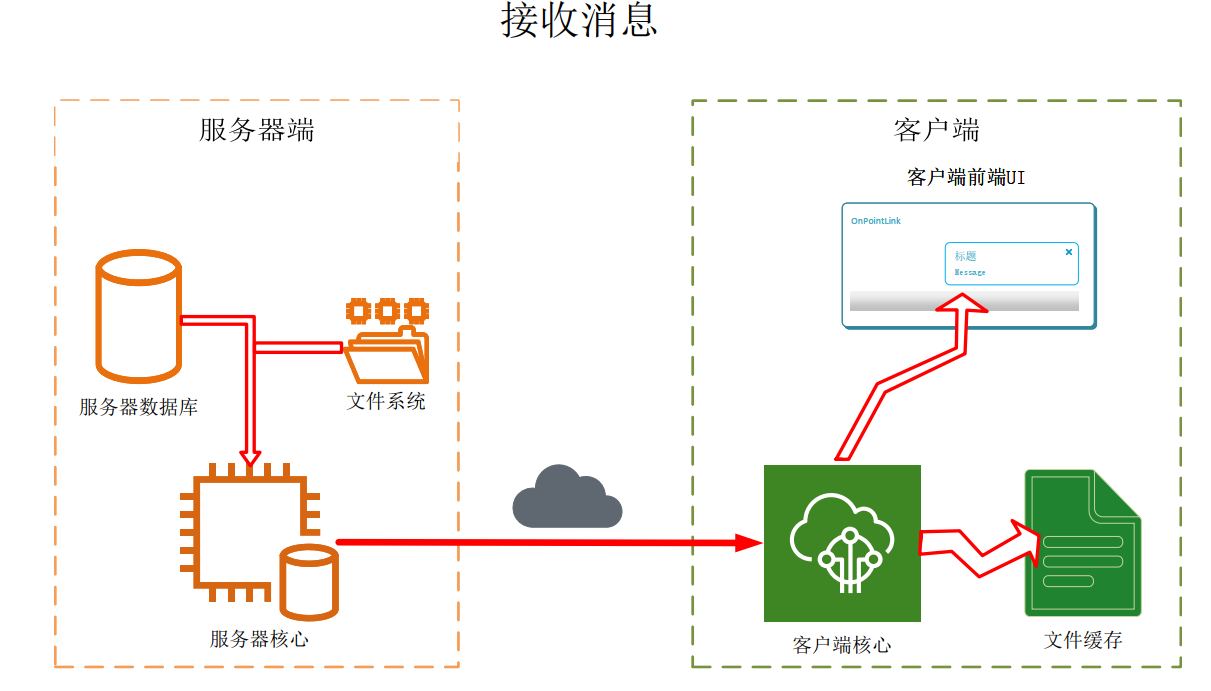
· 服务器数据库：存储所有用户的信息，用于登录、注册；并提供一定的云存储服务，以保存短时间内的消息记录

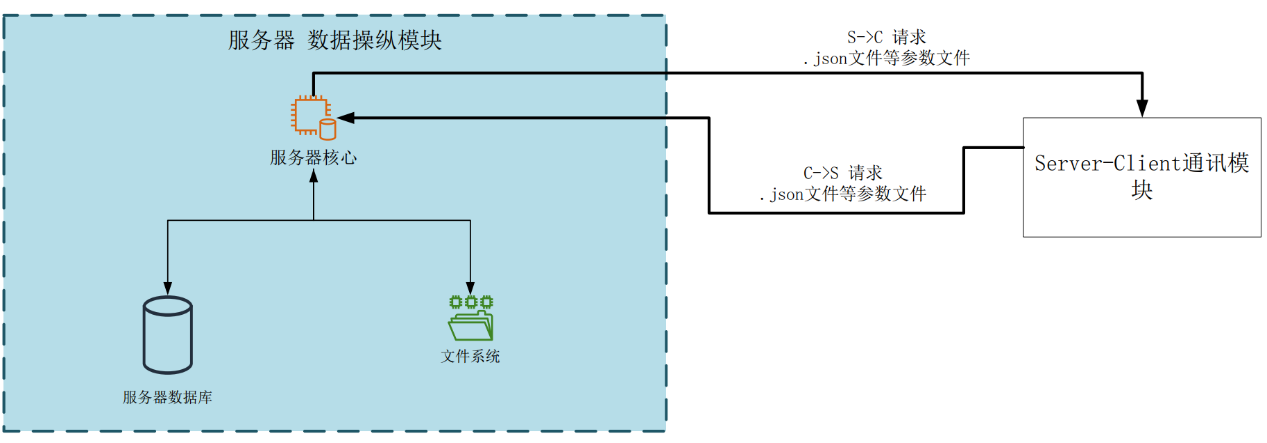
· 服务器文件系统：存储一定时间内的群文件与p2p传输文件。

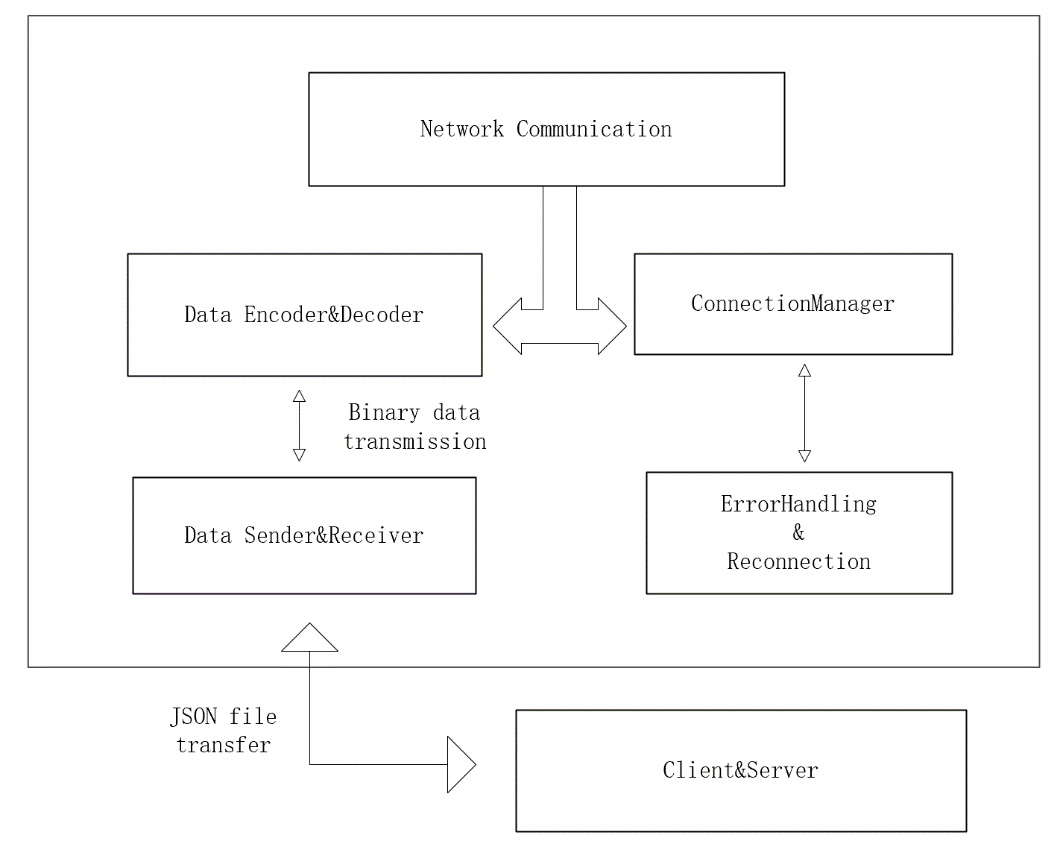
· 服务器核心：处理客户端的所有请求，以及即时向客户端发送其所需要的信息。

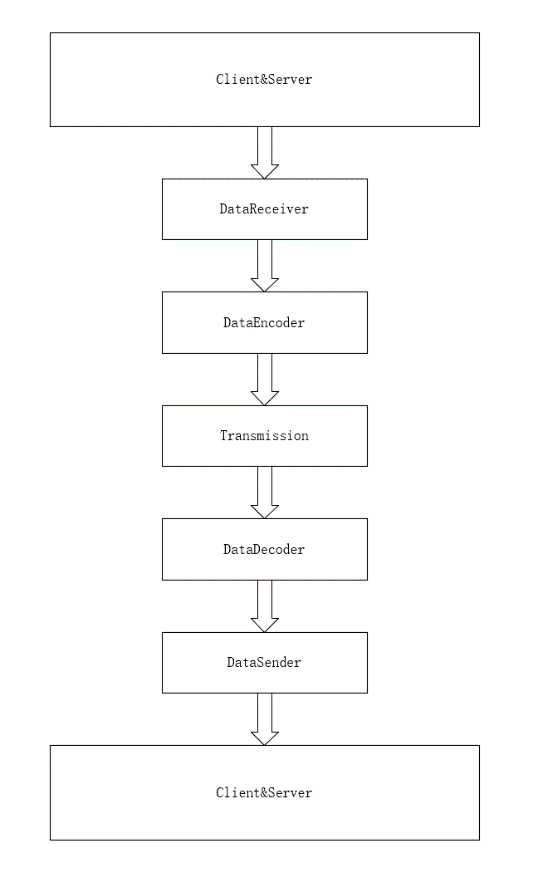
**图示：**









**

**客户端函数：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数名 | reconnectToServer | | | |
| 文件名 | login.h | | | |
| 功能 | 与服务器建立tcp链接 | | | |
| 记述形式 | void reconnectToServer(); | | | |
| 参数 | | | | |
| 类型 | 変数名 | | I/O | 说明 |
|  |  | |  |  |
| 返回值 | 型 |  | 说明 | |
| 详细说明 | | | | |
|  | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数名 | sendLogin | | | |
| 文件名 | login.h | | | |
| 功能 | 发出登录请求 | | | |
| 记述形式 | void sendLogin(QString ID, QString password, bool rememberPassword, bool autoLogin); | | | |
| 参数 | | | | |
| 类型 | 変数名 | | I/O | 说明 |
| QString | ID | | O | 用户ID |
| QString | password | | O | 用户密码 |
| bool | rememberPassword | | O | 是否记住密码 |
| 返回值 | 型 |  | 说明 | |
| 详细说明 | | | | |
|  | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数名 | sendSignUp | | | |
| 文件名 | sign\_up.h | | | |
| 功能 | 发出注册信号 | | | |
| 记述形式 | void sendSignUp(QString nickname, QString password); | | | |
| 参数 | | | | |
| 类型 | 変数名 | | I/O | 说明 |
| QString | nickname | | O | 昵称 |
| QString | password | | O | 密码 |
| 返回值 | 型 |  | 说明 | |
| 详细说明 | | | | |
|  | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数名 | sendNewIP | | | |
| 文件名 | switchserverip.h | | | |
| 功能 | 更改目标服务器IP地址 | | | |
| 记述形式 | void sendNewIP(QString IP); | | | |
| 参数 | | | | |
| 类型 | 変数名 | | I/O | 说明 |
| QString | IP | | O | 新的IP地址 |
| 返回值 | 型 |  | 说明 | |
| 详细说明 | | | | |
|  | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数名 | sendSearchRequest | | | |
| 文件名 | addfriend.h | | | |
| 功能 | 搜索用户 | | | |
| 记述形式 | void sendSearchRequest(QString ID, QString nickname); | | | |
| 参数 | | | | |
| 类型 | 変数名 | | I/O | 说明 |
| QString | ID | | O | 以用户ID搜索用户，否则空 |
| QString | nickname | | O | 以用户密码搜索用户，否则空 |
| 返回值 | 型 |  | 说明 | |
| 详细说明 | | | | |
|  | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数名 | acceptRequest | | | |
| 文件名 | dealfriendrequest.h | | | |
| 功能 | 接受好友申请 | | | |
| 记述形式 | void acceptRequest(QString ID); | | | |
| 参数 | | | | |
| 类型 | 変数名 | | I/O | 说明 |
| QString | ID | | O | 好友申请的ID |
| 返回值 | 型 |  | 说明 | |
| 详细说明 | | | | |
|  | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数名 | rejectRequest | | | |
| 文件名 | dealfriendrequest.h | | | |
| 功能 | 拒绝好友申请 | | | |
| 记述形式 | void rejectRequest(QString ID); | | | |
| 参数 | | | | |
| 类型 | 変数名 | | I/O | 说明 |
| QString | ID | | O | 好友申请的ID |
| 返回值 | 型 |  | 说明 | |
| 详细说明 | | | | |
|  | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数名 | sendMessage | | | |
| 文件名 | mainwindow.h | | | |
| 功能 | 发送一条消息 | | | |
| 记述形式 | void sendMessage(QString ID, QString content); | | | |
| 参数 | | | | |
| 类型 | 変数名 | | I/O | 说明 |
| QString | ID | | O | 发送对象ID |
| QString | content | | O | 发送内容 |
| 返回值 | 型 |  | 说明 | |
| 详细说明 | | | | |
|  | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数名 | sendAddFriendRequest | | | |
| 文件名 | mainwindow.h | | | |
| 功能 | 发送好友申请 | | | |
| 记述形式 | void sendAddFriendRequest(QString ID); | | | |
| 参数 | | | | |
| 类型 | 変数名 | | I/O | 说明 |
| QString | ID | | O | 申请对象ID |
| 返回值 | 型 |  | 说明 | |
| 详细说明 | | | | |
|  | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数名 | connectedToServer | | | |
| 文件名 | login.h | | | |
| 功能 | 成功连接到服务器 | | | |
| 记述形式 | void connectedToServer(); | | | |
| 参数 | | | | |
| 类型 | 変数名 | | I/O | 说明 |
|  |  | |  |  |
| 返回值 | 型 |  | 说明 | |
| 详细说明 | | | | |
|  | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数名 | cannotConnect | | | |
| 文件名 | login.h | | | |
| 功能 | 链接到服务器失败 | | | |
| 记述形式 | void cannotConnect(); | | | |
| 参数 | | | | |
| 类型 | 変数名 | | I/O | 说明 |
|  |  | |  |  |
| 返回值 | 型 |  | 说明 | |
| 详细说明 | | | | |
|  | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数名 | loginSuccess | | | |
| 文件名 | login.h | | | |
| 功能 | 登录成功 | | | |
| 记述形式 | void loginSuccess(); | | | |
| 参数 | | | | |
| 类型 | 変数名 | | I/O | 说明 |
|  |  | |  |  |
| 返回值 | 型 |  | 说明 | |
| 详细说明 | | | | |
|  | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数名 | loginFailed | | | |
| 文件名 | login.h | | | |
| 功能 | 登录失败 | | | |
| 记述形式 | void loginFailed(QString log); | | | |
| 参数 | | | | |
| 类型 | 変数名 | | I/O | 说明 |
| QString | log | | I | 失败原因 |
| 返回值 | 型 |  | 说明 | |
| 详细说明 | | | | |
|  | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数名 | signUpSuccess | | | |
| 文件名 | sign\_up.h | | | |
| 功能 | 注册成功 | | | |
| 记述形式 | void signUpSuccess(QString ID); | | | |
| 参数 | | | | |
| 类型 | 変数名 | | I/O | 说明 |
| QString | ID | | I | 分配到的帐号ID |
| 返回值 | 型 | int | 説明 | |
|  | 値 |  |  | |
|  |  |  |  | |
| 详细说明 | | | | |
|  | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数名 | signUpFailed | | | |
| 文件名 | sign\_up.h | | | |
| 功能 | 注册失败 | | | |
| 记述形式 | void signUpFailed(QString log); | | | |
| 参数 | | | | |
| 类型 | 変数名 | | I/O | 说明 |
| QString | log | | I | 失败原因 |
| 返回值 | 型 | int | 説明 | |
|  | 値 |  |  | |
|  |  |  |  | |
| 详细说明 | | | | |
|  | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数名 | getMessage | | | |
| 文件名 | mainwindow.h | | | |
| 功能 | 收到其他人发来的信息 | | | |
| 记述形式 | void getMessage(QString ID, QString content); | | | |
| 参数 | | | | |
| 类型 | 変数名 | | I/O | 说明 |
| QString | ID | | I | 发送者ID |
| QString | content | | I | 信息内容 |
| 返回值 | 型 | int | 説明 | |
|  | 値 |  |  | |
|  |  |  |  | |
| 详细说明 | | | | |
|  | | | | |

服务端函数：

/\*

\* servercore.h

\* 服务器核心模块，对接收到的消息进行处理并存储，

\* 对客户端的各个请求进行回应。

\*/

class **servercore** : public QObject

{

Q\_OBJECT

public:

/\*

\* 下列一部分的函数接口

\*/

explicit **servercore**(TcpServer \*ptcpserver, QObject \*parent = nullptr);

void **RegRequest**(QJsonObject &jsonData);

void **EnterRequest**(QJsonObject &);

void **AccountInfoRequest**(QJsonObject &);

void **FriendListRequest**(QJsonObject &);

void **GroupListRequest**(QJsonObject &);

void **UserInfoRequest**(QJsonObject &);

void **GroupInfoRequest**(QJsonObject &);

void **SendTxtMessageRequest**(QJsonObject &);

void **TargetMessageRequest**(QJsonObject &);

void **MessageReadedRequest**(QJsonObject &);

void **FileListRequest**(QJsonObject &);

void **FileRequest**(QJsonObject &);

void **returnRegResult**(int OID,bool Status,QString log,QString Name);

~***servercore***();

private slots:

void **switchFunction**(QTcpSocket \*psocket);

private:

TcpServer \*tp;

QSqlDatabase db;

};

/\*

\* tcpserver.h

\* TcpServer: 用于进行客户端与服务端通信的模块，负责接受和发送信息。

\*/

class **TcpServer** : public QObject

{

Q\_OBJECT

public:

explicit **TcpServer**(QObject \*parent = nullptr);

~***TcpServer***();

signals:

void **connected**(QTcpSocket \*);

void **received**(QTcpSocket \*);

public slots:

void **initServer**();

void **send**(QHostAddress &ip, quint16 &port, QByteArray &text);

private:

QTcpServer \* m\_s;

QList<QTcpSocket \*> connections;

QTime time;

|  |  |  |  |
| --- | --- | --- | --- |
| **信息类型名** | **方向** | **内容（注：所有json文件第一个都有一个"transType"属性，表示当前通讯讯息类型，这里就不一个一个加了）** | **备注（数据约束）** |
| RegRequest | C->S | regInfo.json: { "transType":"RegRequest", "Name":String, "Instruction":String(NAN), "Email":String(NAN)， "Area" :String(NAN), "Sex":String(NAN), "BirthDay":String(YYYY-MM-DD)(NAN), "Password":String}  HeadImage.png | "Name"长度不超过10个字符。 "Instruction"长度不超过500个字符。 "Email"需要符合Email命名规范:username@domain.com，其中username仅由数字字母下划线构成，domain.com为服务器域名。 BirthDay必须按照格式来，否则为空。 Password不低于6位，不多于20位。 HeadImage.png不超过1MB。 Sex必须是男或者女或者其他或空 |
| RegResult | S->C | regInfo.json: { "transType":"RegResult", "Status":bool, "OID":long int, "log":String(失败原因)(NAN)} | Status为True则表示成功，否则失败，失败才会有"log"。 |
| EnterRequest | C->S | enterInfo.json: { "transType": "EnterRequest", "OID":long int, "Password":String, "IP":String} | IP格式为"8位数字-8位数字-8位数字-8位数字" |
| EnterResult | S->C | enterResult.json: {"OID": ,"IP": ,"Status":bool, "log":String(失败原因)(NAN)} |  |
| AccountInfoRequest | C->S | accountInfoRequest.json: {"OID":long int} |  |
| AccountInfoResult | S->C | accountInfo.json: {"OID": , "Name":String, "Instruction":String(NAN), "Email":String(NAN)， "Birth":DateString(年-月-日)(NAN),}  HeadImage\_{OID}.png |  |
| FriendListRequest | C->S | friendListRequest.json: {"OID":long int} |  |
| FriendListResult | S->C | friendList.json: {"OID": ,"FriendList":[{"FriendOID": ,"Devide": },...]} |  |
| GroupListRequest | S->C | groupListRequest.json: {"OID": } |  |
| GroupListResult | C->S | groupListInfo.json: {"OID": ,"GroupList": ["GroupOID": ,"Name": ],...} |  |
| UserInfoRequest | C->S | userInfoRequest.json: {"OID": ,"FriendOID": } |  |
| UserInfoResult | S->C | userInfo.json: {"OID": ,"UserOID": ,"UserInfo": {"UserOID": ,"Name": ,"Instruction": ,"Email": ,"Birth": ,"Area": ,"HeadImage": }}  User\_HeadImage.png |  |
| GroupInfoRequest | C->S | groupInfoRequest.json: {"OID": ,"GruopOID": } |  |
| GroupInfoResult | S->C | groupInfo.json: {"OID": ,"GroupOID": ,"GroupInfo": {"Name": ,"HeadImage": ,"Instruction": ,"MembersOIDs": []}}  Group\_HeadImage.png |  |
| SendTxtMessageRequest | C->S | sendMessageRequest.json: {"SenderOID": ,"TargetOID": ,"Time": String,"Type": ,"Value": }  （一个文件） | Time为一个时间戳，精确到秒。 "Type"备选项："Txt","Docu"。 消息类型为文件时，Value为文件名。 Time为String，采用MYSQL DATETIME的时间格式："YYYY-MM-DD hh:mm:ss" |
| SendMessageResult | S->C | sendMessageResult.json: {"Status:","MID":int64, "Type": ,"SenderOID": ,"TargetOID": ,"Time": "Readed":bool, "Value": } | MID为一个64位哈希值（转化为16进制字符串只有16位），可以唯一标识一条消息。 Time依然为DATETIME类型。 Readed为True表示消息已读，否则表示未读。  消息发送成功后，Value会从原来的文件名，变为服务器上的文件路径名。 |
| TargetMessageRequest | S->C | targetMessageRequest.json: {"MID": ,"SenderOID": ,"TargetOID": ,"Time": ,"Type": ,"Value": } |  |
| TargetMessageResult | C->S | targetMessageResult.json: {"OID": "MID": ,"SenderOID": ,"TargetOID": ,"Time": ,"Type": ,"Value": ,"Status": } | 这里OID表示发起该请求的用户。（下面同） |
| MessageReadedRequest | C->S | messageReadedRequest.json: {"OID": ,"MID": ,"SenderOID": ,"TargetOID": , Time: , "Type": ,"Value": } |  |
| MessageReadedResult | S->C | messageReadedResult.json: {"OID": ,"MID": ,"SenderOID": ,"TargetOID": , Time: , "Type": ,"Value": ,"Readed": ,"Status": ,"Log": } | 一样，"Status"表示消息已读状态是否更改成功，"Log"表示失败后的日志 |
| FileListRequest | C->S | fileListRequest.json: {"OID": ,"GroupOID": } |  |
| FileListResult | S->C | fileListResult.json: {"OID": ,"GroupOID": ,"Values":[]} | Values对应的是一个字符串列表，内容是每个文件的群文件检索路径，经过前端群文件管理器整理后显示出一个文件系统。 |
| FileRequest | C->S | fileRequest: {"OID": ,"FilePath": } |  |
| SendFriendRequestToServer | C->S | {"OID1":,"OID2","RequestMessage",} |  |
| ProcessFriendRequestResult | S->C(Sender) | {"OID1","OID2","Status","log"} | 这个status记录是否成功申请，比如是否存在账号或者其他报错信息 |
| SendRequestToReceiverClient | S->C(Target) | {"OID1","OID2","RequestMessage"} |  |
| SendResultFromReceiverClientToServer | C(Target)->S | {"OID1","OID2","ReplyMessage"} |  |
| SendResultToApplicant | S->C(Sender) | {"OID1","OID2","Status","ReplyMessage"} | 这个status记录是不是同意申请 |

通信端函数：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数名 | send | | | |
| 文件名 | ../tcpserver.cpp | | | |
| 功能 | 将数据转发给特定的连接客户端 | | | |
| 记述形式 | void TcpServer::send(QHostAddress &ip, quint16 &port, QByteArray &text) | | | |
| 参数 QHostAddress &ip, quint16 &port, QByteArray &text | | | | |
| 类型 | 変数名 | | I/O | 说明 |
| void | - | | - | - |
| 返回值 | 型 | int | 说明 | |
|  | 値 |  |  | |
|  |  |  |  | |
| 详细说明 | | | | |
| 在一个TCP服务器中查找特定IP地址和端口号的客户端连接，并将指定的数据转发给该客户端。在找到匹配的连接后，它将数据写入套接字，然后通过调试输出确认数据已成功转发。 | | | | |

**4.2.2 模块内接口函数**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数名 | initServer | | | |
| 文件名 | ../tcpserver.cpp | | | |
| 功能 | 创建了一个TCP服务器并开始监听指定端口，以便接受客户端连接 | | | |
| 记述形式 | void TcpServer::initServer() | | | |
| 参数 | | | | |
| 类型 | 変数名 | | I/O | 说明 |
| void | - | | - | - |
| 返回值 | 型 |  | 说明 | |
|  | 値 |  |  | |
|  |  |  |  | |
| 详细说明 | | | | |
| 使用Qt的信号和槽机制来异步处理连接、数据接收和断开连接事件。此服务器可以在新客户端连接时发出信号，接收客户端发送的数据并处理客户端断开连接的情况。它还维护一个连接列表以跟踪所有连接的客户端套接字。 | | | | |

**4 实现方案**

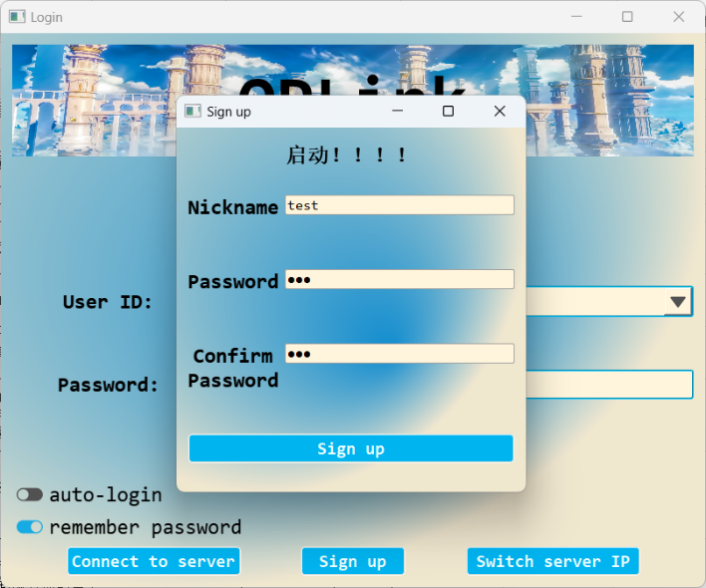
**登录界面**



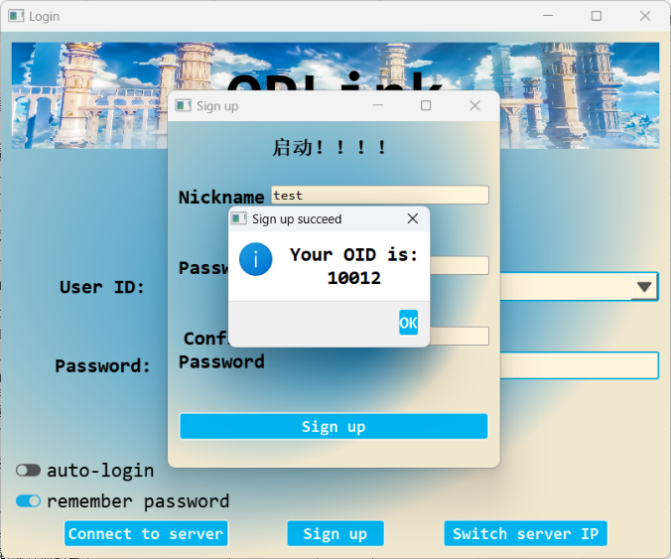
选择服务器IP



注册账号



为新账号分发ID



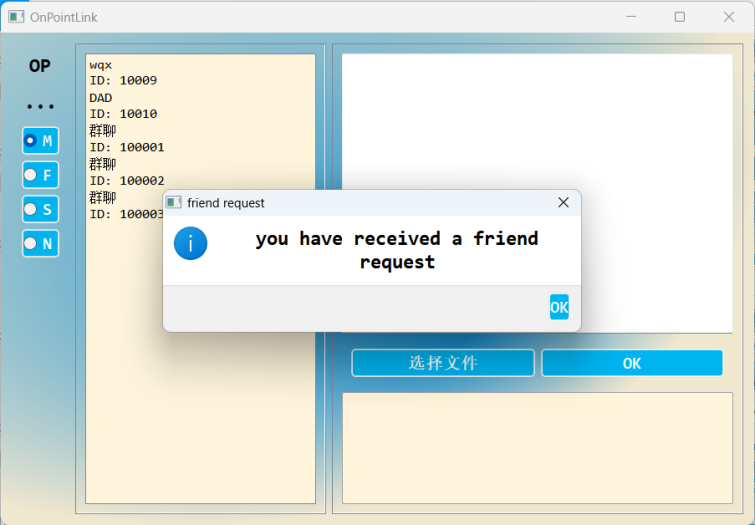
输入账号/密码、测试自动登录/记住密码功能



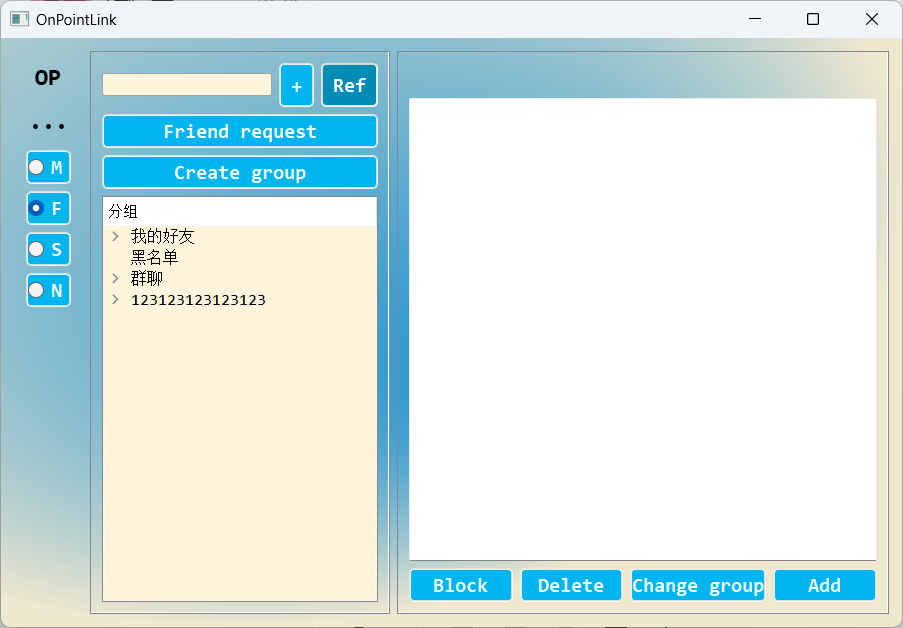
登录成功



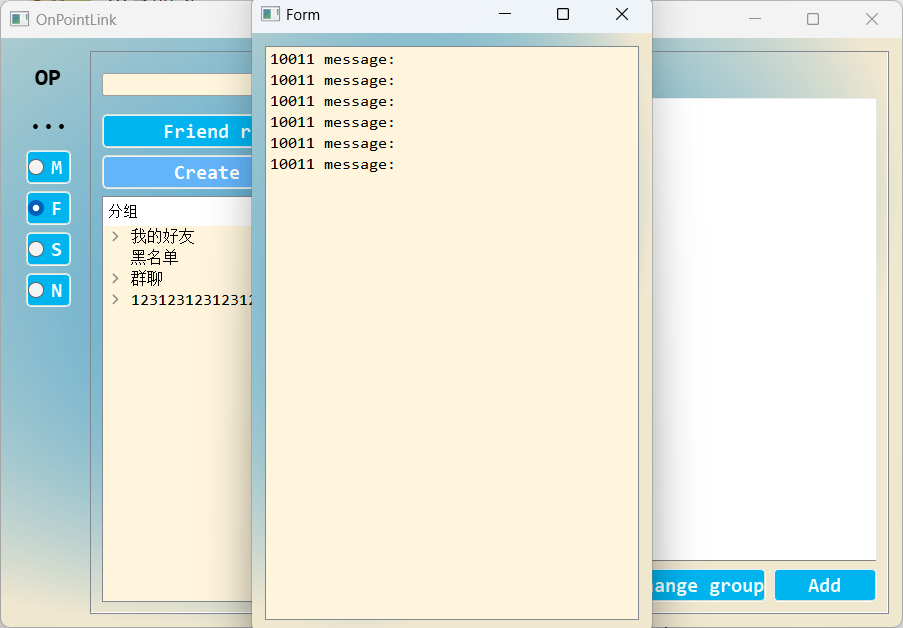
好友申请



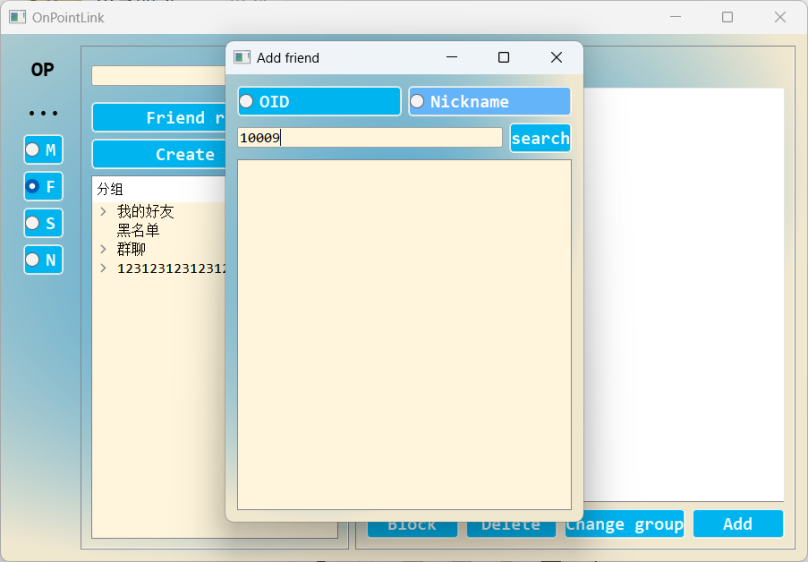
刷新功能【Ref按钮】



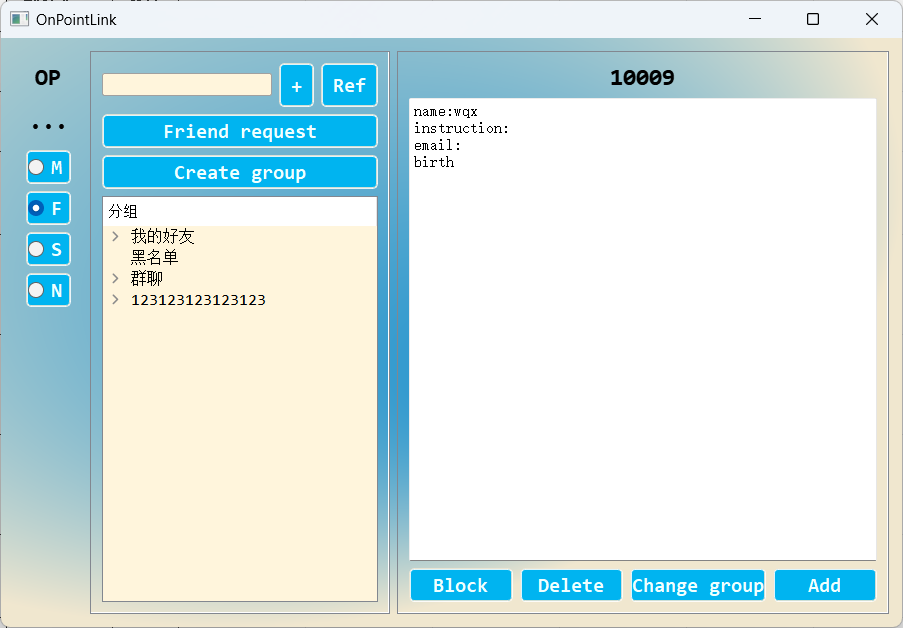
好友申请



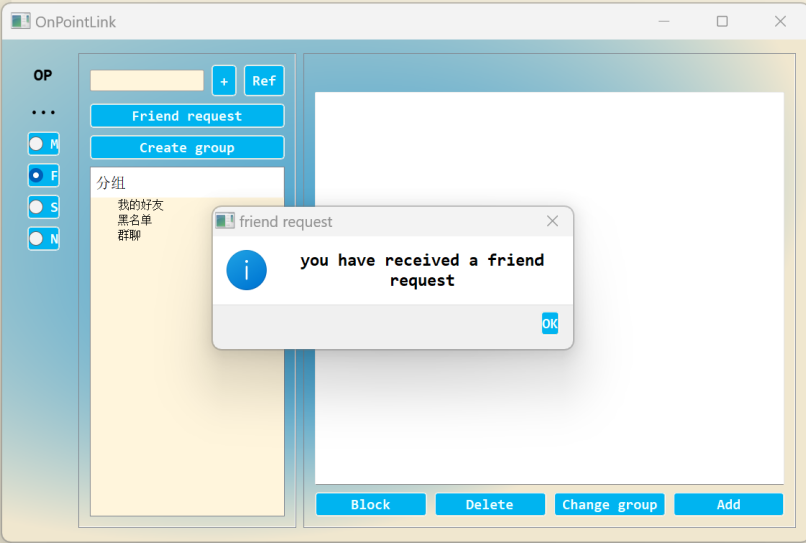
搜索用户



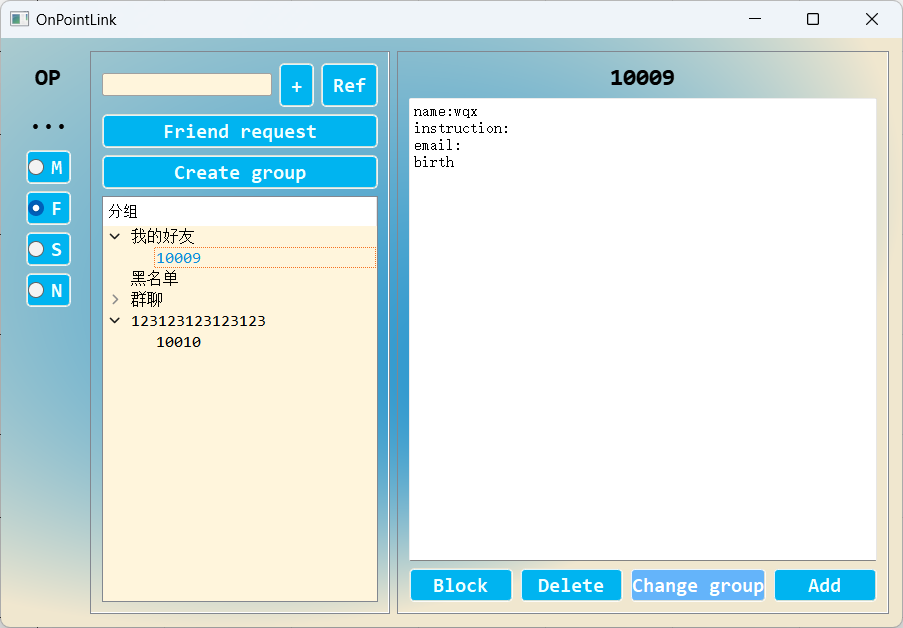
发出好友申请【包含个人信息】



接收好友申请



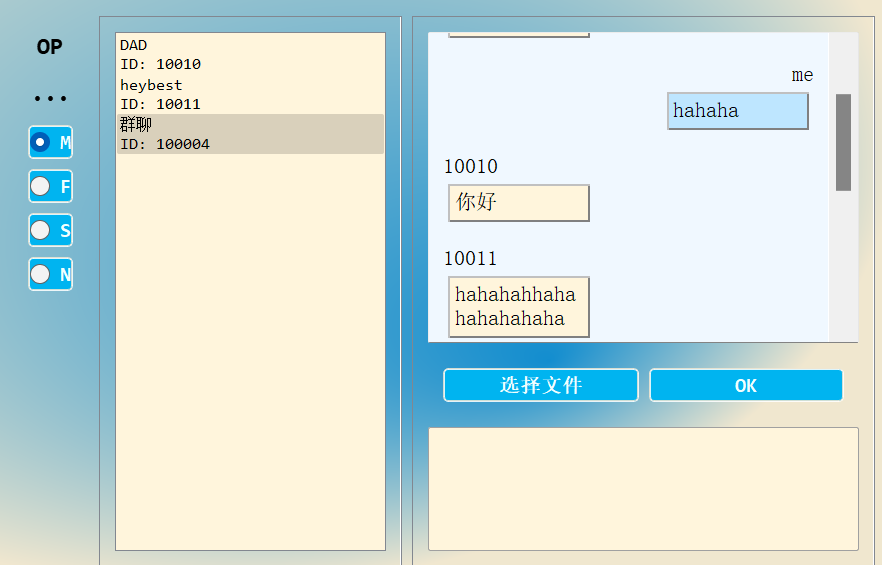
添加好友，并在好友列表中显示



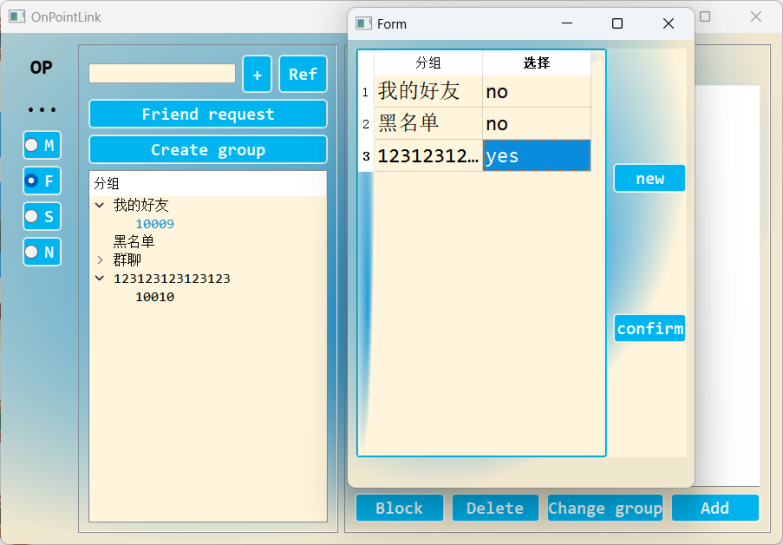
一对一聊天



多人群聊



好友分组/黑名单功能



创建新分组



**5 系统测试**

测试方法：小组事先撰写测试用例，测试员按用例表，打开程序依次测试。

测试描述：程序运行相对稳定，全部测试用例均通过，在信息接收/创建群聊时出现略微延迟，猜测为服务端网络不稳定，与程序本身无关。

预期结果：达到了全部测试用例通过的预期。

**6 项目经验及心得体会**

在这次实习中，我收获了许多。我巩固了Qt知识，锻炼了组织协调能力。虽然我们遇到了许多困难，比如功能实现不了，小组成员意见不统一等等；但是我们通力协作，克服了种种困难，最终形成了一个具有基本功能的软件，能够实现及时通信。希望之后还有更多有意义的实习。