东软睿道

项目实训总结报告

项 目 名 称: WeTalk

学 校 名 称: 北京理工大学

项 目 讲 师: 崔浩

姓 名: 陈耀宇

学 号: 1120200944

实 训 日 期: 2022年8月15日-8月25日

**1 引言**

1.1项目概述

本项目旨在设计一款基于QtQuick，QML，Qt，SQLite和云服务器的支持广域网的聊天应用，为用户提供方便快捷的远程通信和自由交友，能够跨越距离的限制，能够在办公，娱乐，日常生活等多场景下使用，时刻保持信息畅通。在项目中我们主要借鉴了QQ和Wechat的相关功能进行开发。

1.2 需求分析

在互联网时代，依靠电话和短信的通信手段，远远满足不了人们的需求且效率较低，成本较高，在许多场景下，需要文件和多人信息传输，此外，生活水平的日益提高，人们对于交友和娱乐的追求也随之提高，为此我们开发了WeTalk应用，支持自由交友，广域网连接，建立群聊，好友单聊，设计个人名片等能够为人们日常生活，办公提供便利的功能。

1.3 运行环境

客户端：Windows/Linux/Mac

服务端：Ubuntu 20.04 LTS

**2 项目设计**

2.1 设计思路

项目分成客户端、服务器端和数据库端，其中服务器端放在公网下的云端服务器上，达到公网即时通讯的目的。项目系统的整体架构和各架构下的内容如图2-1所示，以服务器端为开发中心，连接三个部分的通讯。

开发过程采用前后端分离技术，并写好对应的接口函数和信号，最后合并时进行连接调整。这有利于我们分别独立进行开发，分工明确，开发效率高。

本人负责通信模块与架构。本团队在最开始就打算做广域网即时通信，所以一个好的架构尤为重要。

经过一天的学习，借鉴，思考，我得出了基于Udp报文+Json命令解析的服务器中转模式，其中，不同类型的命令Json报文中，其command参数都不同，对应不同的函数调用。

除此之外，本团队使用QML技术作为前端，这加大了通讯的难度，我除了客户端与服务端的通信以外，还负责QML的前后端通信，以及中后期的项目整合。经过实践，本团队成功实现了注册，登录，以及各种功能，并且成功在云服务器上部署，运行，即使大家处在天南海北，也能实现多人聊天。

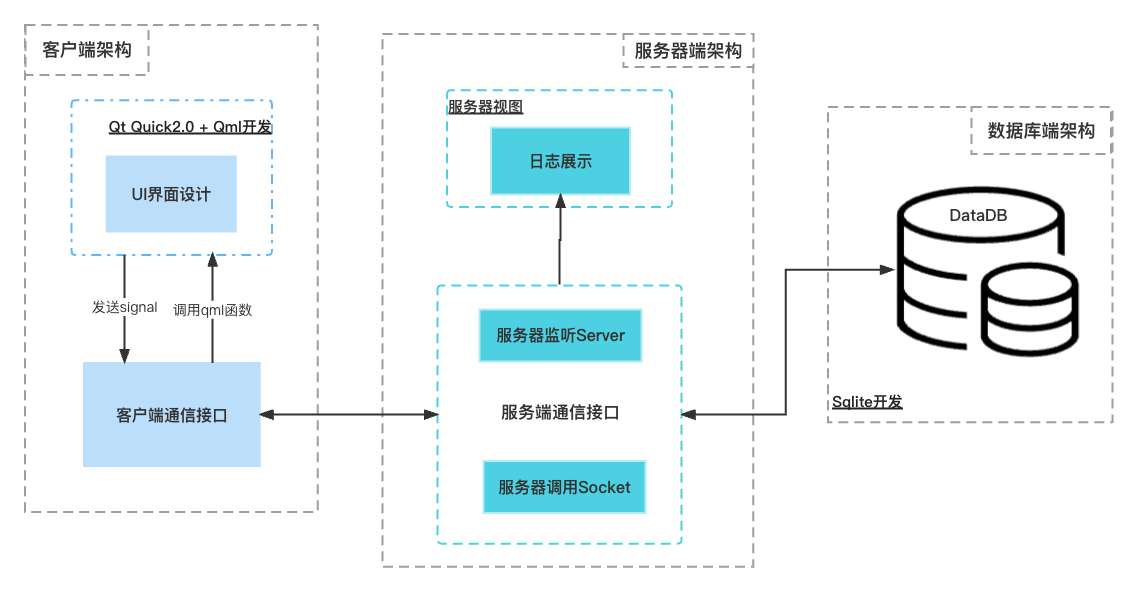


图2-1 项目系统整体架构图

2.2 模块功能介绍

客户端模块：

1、登录注册模块：用户注册新账号登录，并需要通过邮箱验证码验证；或者直接登录已有账号。

2、主界面模块：用户可以通过主界面的各个按钮切换到不同的功能模块或界面，包括聊天界面和好友/群聊列表，并实现了点击头像查看对应用户或群组的详细信息。

3、聊天界面模块：用户可以在该界面内进行私聊、群聊的即时聊天功能。

4、联系人模块：用户可以在该界面查看好友/群聊列表，并可以点击聊天切换到聊天界面。

5、添加好友/群聊模块：用户可以通过搜索对应用户/群聊的id，从而添加新好友。

6、用户/群组信息详细展示模块：用户可以查看自己、他人或群组的详细信息。特别的，用户可以修改自身的详细信息，也可以修改本人为群主下的群组信息。

7、创建群聊功能：用户可以向服务器端申请创建一个群聊，并返回群聊id判断群聊是否创建成功。

服务端模块：

1.登录注册响应模块：接收客户端发来的登录和注册请求，注册时向数据库写入用户信息，登陆时与数据库进行验证对比，并返回是否登陆成功。

2.查找信息模块：接收来自客户端的搜索请求，按照不同的请求类型（查找群或者查找用户），从服务器调取并向客户端返回查找的信息。

3.更改信息模块：接收来自客户端的个人和群信息，并写入数据库完成更新。

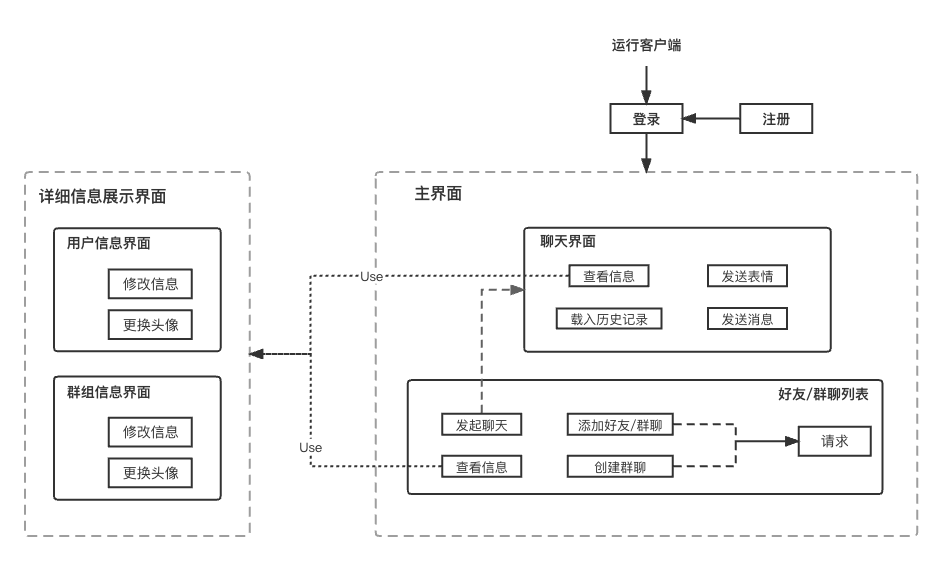
4.消息传输模块：接收来自客户端发出的消息，判断聊天性质（私聊或者群聊），向目标客户端转发消息，并完成聊天记录向数据库的存储。

5.搜索响应模块：接收来自客户端的检索请求，在数据库中查找对应id的群或者用户信息，向客户端返回搜索结果。

6.添加好友或群模块：接收来自客户端的添加好友或群的信息，向客户端返回添加成功或者失败信息，并更新数据库。

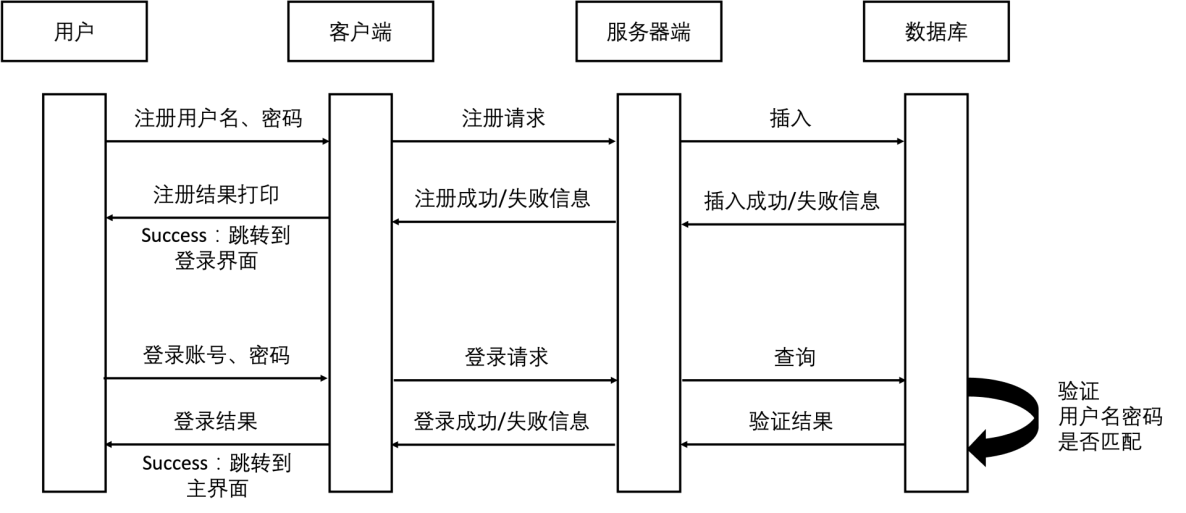
7.列表刷新响应模块：接收服务端的刷新请求，解析请求的性质（刷新好友列表或者群列表），并向服务端返回数据库中的最新信息。

2.3 模块结构图

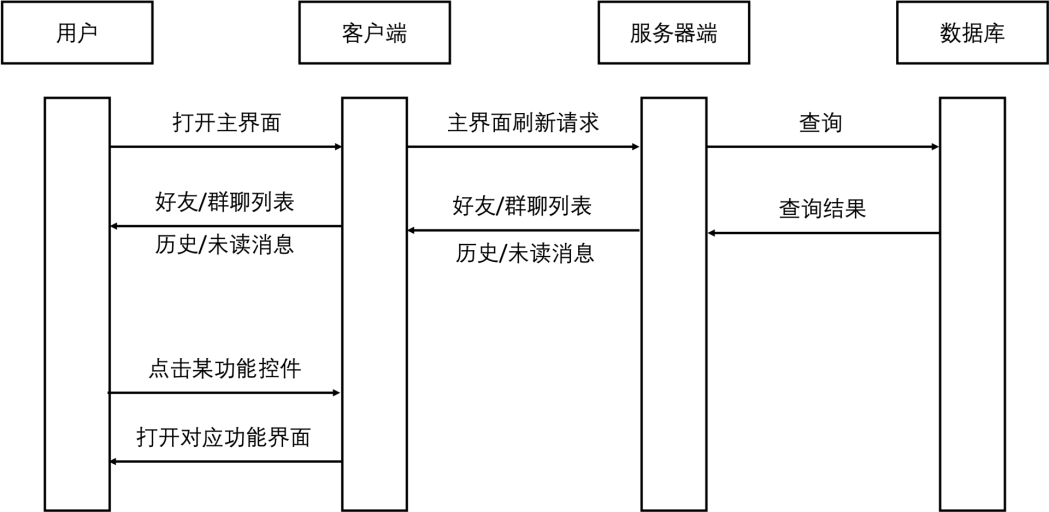


2.4 程序流程图

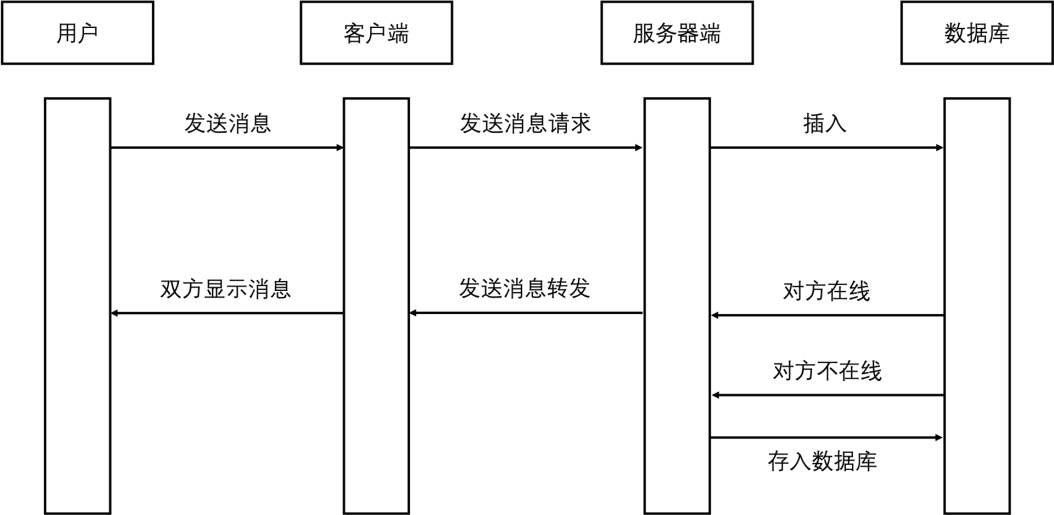
2.4.1 登录注册模块



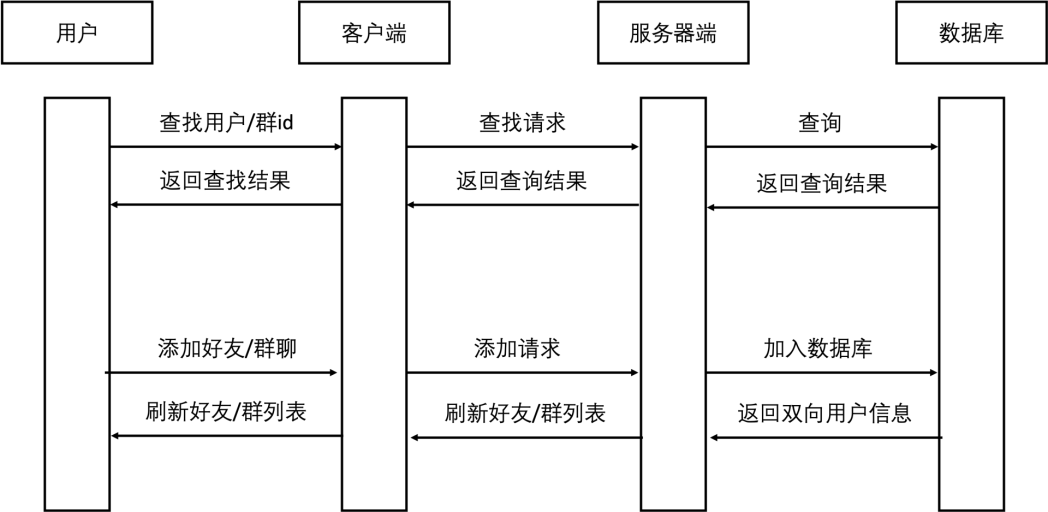
2.4.2主界面模块



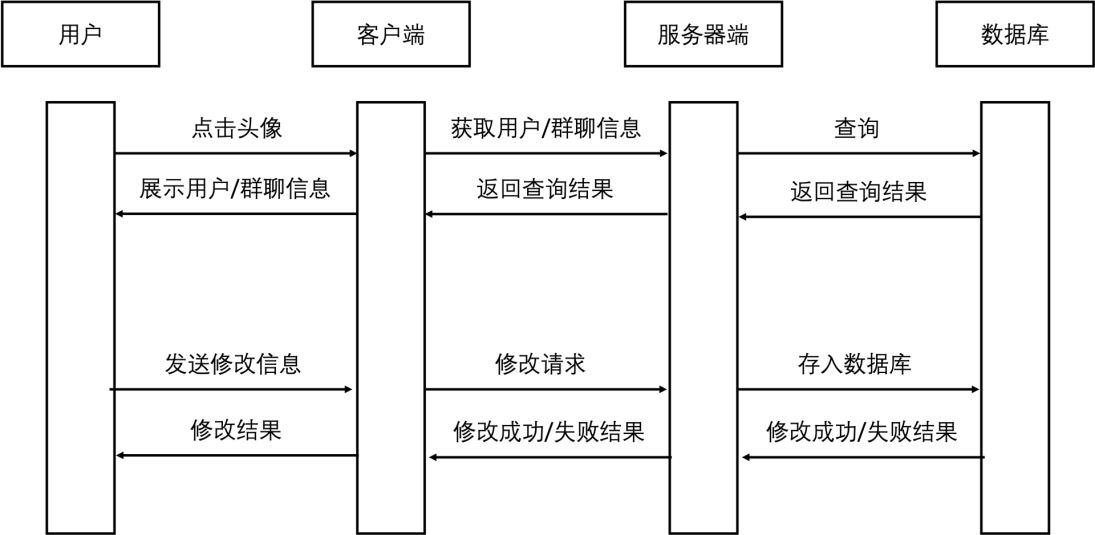
2.4.3 聊天界面模块



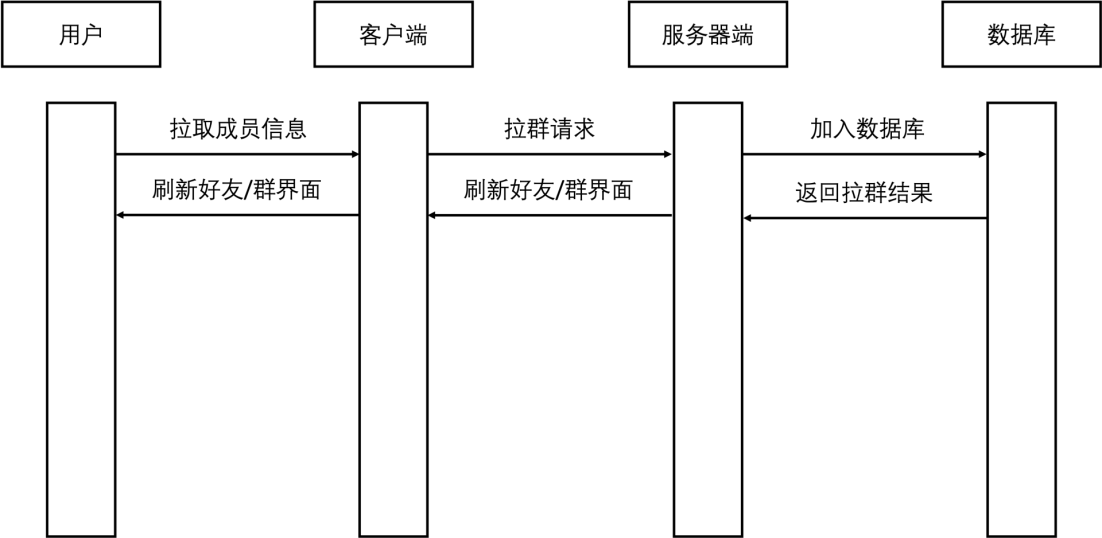
2.4.4 添加好友/群聊模块



2.2.4 详细信息显示界面



2.2.5 创建群聊界面



2.5 功能设计分工

服务器端：陈耀宇、罗家安

DataDB：刘恩泽

设计美化客户端ui界面：刘翎、沈宇航

界面跳转：刘翎、沈宇航

自定义控件：刘翎、沈宇航

编写客户端功能：刘翎、沈宇航

客户端与服务器端接口函数：陈耀宇、沈宇航、罗家安

服务器端与数据库接口函数：陈耀宇、刘恩泽

测试：陈耀宇、沈宇航、刘翎、刘恩泽、罗家安

项目文档创建与修改：刘翎、刘恩泽、罗家安、沈宇航、陈耀宇

本人xxx

**3 详细设计**

本项目采用Qt5和qml开发，采用典型的服务器-客户端架构，依托Qt的跨平台特性可以运行在绝大多数Qt支持的平台上。服务器放在云端公网上，与客户端之间的通信主要使用socket实现。通过服务器启动的各项配置参数可以适配Qt支持的主流数据库。核心功能为用户的注册、登录，私聊或群聊中即时消息的收发，历史记录的缓存。

在客户端的设计中主要包含两个方面，一个是负责与服务器端通信的客户端通信模块，一个是负责整个客户端的ui绘制与逻辑处理的客户端ui模块，整个客户端包含多个ui界面，每个ui界面中会包含自定义控件用来显示各种个人信息，或聊天信息，通过信号与槽的来与客户端通信模块交互往服务器端请求信息。客户端的界面包含登录界面、注册界面、主界面、聊天界面、信息界面、通讯录界面、添加好友/群聊界面。

在服务器端的设计中主要考虑了三个方面。第一个方面是负责接收来自客户端的请求，并对该请求进行解析。当客户端发出捕获到用户点击的信号之后，会调用用户端与服务端的接口函数，将传输的信号封装成json字段，通过udp传输协议向服务端发送请求。每个json字段中的command键值对应不同的请求类型。第二个方面是负责与数据库方面的交互，实现从数据库中拉取个人信息、好友信息、群信息和历史消息。服务器与数据库一起部署在云端，服务器通过调用数据库的各种函数接口实现对数据的增删改查操作。第三个方面是负责向客户端传回响应，返回对客户端请求的回应。服务端将传回的数据封装在json字段中中，通过udp协议传回客户端，以command字段存储不同的回应类型，客户端收到json字段后，对json进行解析并调用不同函数来回应用户的请求。

打开客户端时，首先会进行与服务器的连接，若不成功会给出警告；注册界面与登录界面是相关联的，注册界面包含有邮箱验证码和密码匹配提示功能，注册成功后会分配ID并自动将ID和密码填充入登录框中，点击登录后会与服务器进行通信确认登录信息，进入主界面后显示聊天界面，并显示历史聊天记录。点击聊天界面内个人头像或好友头像，弹出用户详细信息界面，若为个人信息界面，还有设置功能键，点击后可以修改个人信息和头像，点击确认后更新信息和头像。从主界面可以切换到通讯录界面，点击好友/群聊列表的头像，可以进入好友/群聊信息界面；点击对应好友/群聊的聊天按钮，自动切换到对应的聊天界面，进行聊天。聊天时可以选择表情、发送文字。同时，在通讯录界面包含添加好友/群聊，以及创建群聊功能按钮，点击后实现或进入对应功能界面。

**4 实现方案**

**前端**

前端使用QML + QtQuickControl 2.0设计，实现了UI层和逻辑层的分离。 QML通过信号与C++部分通信，发送响应请求；C++部分处理完毕后调用QML层预定义函数进行页面的更新。

在整体的代码组织上，前端页面基于页面结构进行组织。整体包含 Window -> Screen -> Widget 三级。其中：

* Window代表具体的某一窗口，每个Window为独立的小窗口。如注册Window：ResgisterWindow
* Screen代表窗口中某一功能部分，通过布局在Window中排列组织。如聊天Screen：ChatScreen
* Widget代表更具体的小组件，用于实现具体的功能点，比如导航栏组件VerticalTabWidget

详细如下：

RegisterWindow

* + 注册Window，包含输入用户名、密码、发送邮件、校验邮箱验证码。其中，发送邮件的具体处理程序部署于另一台服务器中，通过提供Http接口实现前端的调用。前端QML页面通过XmlHttpRequest API发送请求并处理响应，根据返回的状态码给出具体的错误提示或完成校验。
  + 当接收到注册成功的信号时，会自动跳转到LoginWindow并自动填充服务端下发的用户账号，用以下一步登录

LoginWindow

* + 登录Window，包含输入用户账号与密码，并进行登录的功能。对于各种异常情况（如空输入、错误输入、错误响应等）均给出了良好的异常提示。
  + 当接收到登录成功的信号时，会自动跳转到MainWindow，登录者的id将持久化保存在本地。同时向服务端拉取账户的基本信息。

MainWindow

* + 主Window，各种基本操作均在此处完成。
    - VerticalTabWidget
      * 自定义组件，基于StackView实现右侧页面的切换。
    - HistoryMessageScreen
      * 历史对话列表，以ListView的形式展示用户的历史对话列表，包括好友消息与群聊消息，展示的信息包括对方的头像、名称、最后一条消息的内容和发送时间。
      * 点击任意一项即可快速打开与对方的聊天窗口并自动从云端拉取聊天记录，同时此项高亮显示。
      * 当新的聊天消息发出或被接受时时，此列表的内容会被同步更新。
    - ChatScreen
      * 聊天页面，整体包含上半部分的消息列表以及下方的输入和操作Panel。
      * 消息列表由云端返回，包含所有离线和在线消息
      * 点击聊天列表中的头像可以查看对应用户的信息，如果查看的是“我”的信息，可以设置
      * 聊天列表将自动判断是否是“我”发出，并完成左右两侧的定位展示
      * 输入框可以选择表情并添加对应的文本表情，按下键盘的Enter键或者点击右侧的发送按钮均可发送消息
    - ContactScreen
      * 联系人页面，包含好友和群组，也可以在这里进行添加好友/群组和建群的操作
      * 好友列表和群组列表均从云端获取，收到加好友请求后会自动刷新

AddFriendWindow

* + 添加好友/群聊界面
    - SearchWidget
      * 搜索栏，包括搜索输入框和搜索按钮，可输入好友/群聊的id，查询返回搜索结果。
    - InfListItem
      * 搜索结果显示，打印搜索id对应的用户/群聊的头像、用户名，以及添加按钮发送添加。
  + 用户/群聊信息界面
    - PersonalInf
      * 用户详细信息展示界面，自动扒取数据库中存储的信息，如用户名、个性签名、性别、出生日期等信息。
      * 若为本人的用户信息界面，增加设置按钮，可以对个人信息和头像进行修改。
    - GroupInf
      * 群聊详细信息展示界面，自动扒取数据库中存储的信息，如群名、群公告、群简介。

**程序运行图片**



注册页面



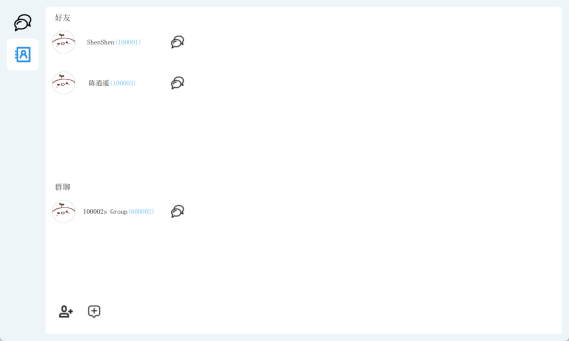
输入错误



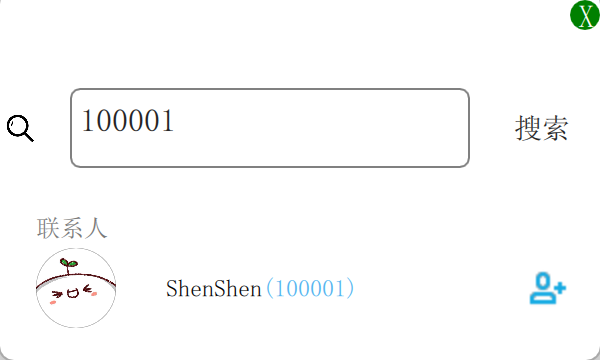
验证码发送



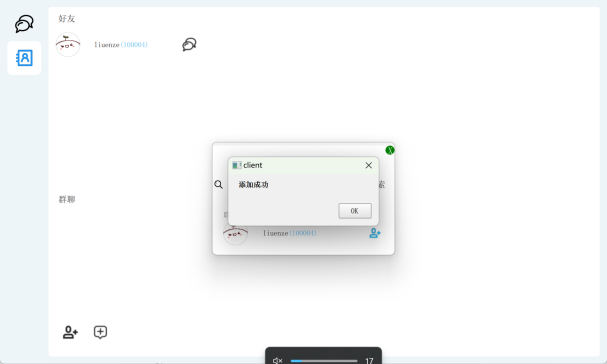
登录页面（自动跳转，自动填充内容）



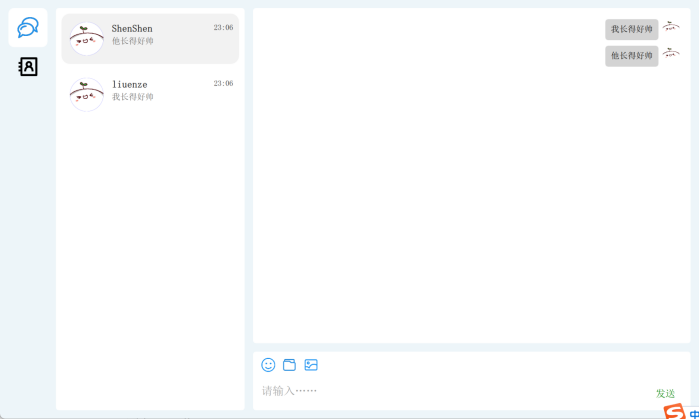
好友和群聊页面



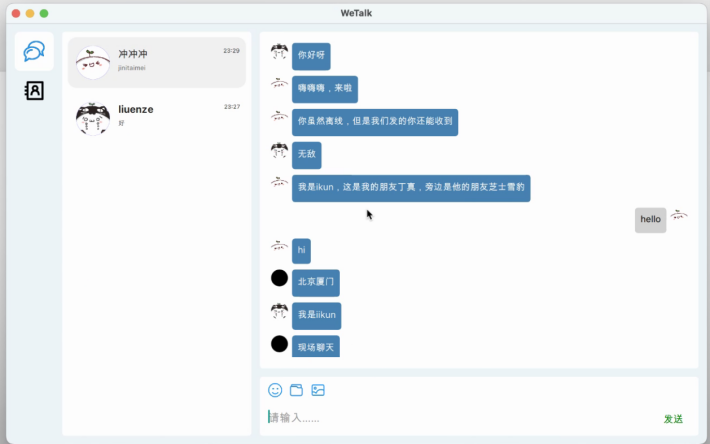
搜索添加好友/群聊



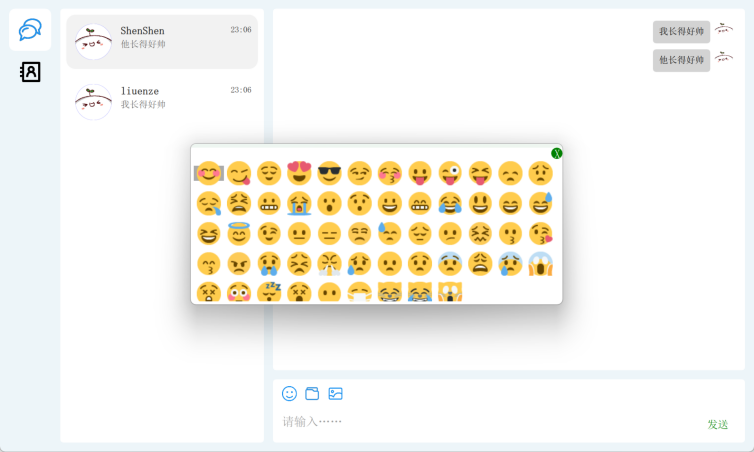
加好友成功弹窗



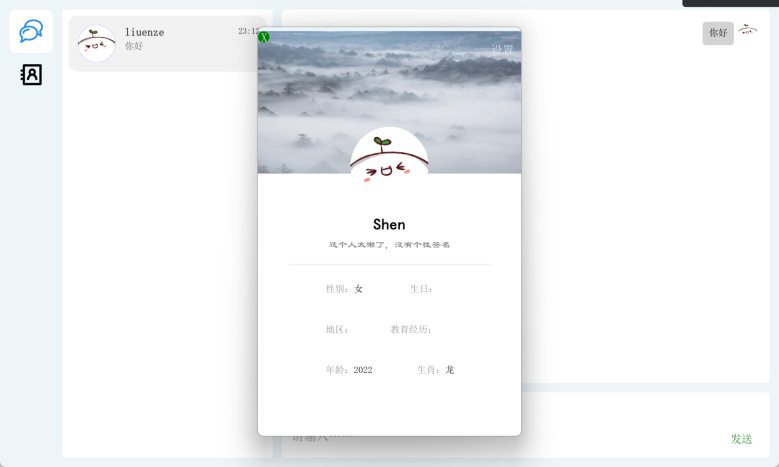
聊天页面和历史消息页面（广域网，支持离线信息收取）



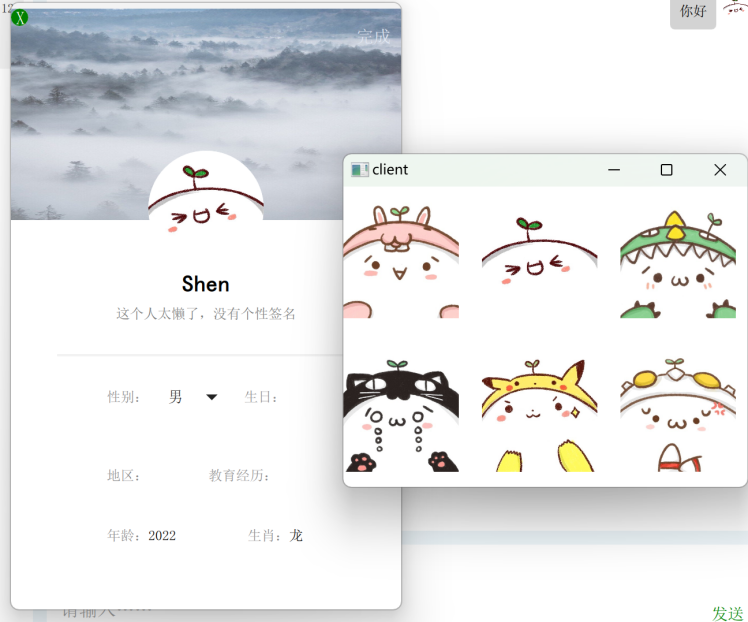
群聊演示



发送表情



点击头像可以查看个人信息（包括聊天列表、好友列表均可）



个人信息修改（所示信息均可修改，修改生日自动计算年龄和生肖。头像可选择）



设置完成后

**5 系统测试**

|  |  |  |
| --- | --- | --- |
| 客户端注册登录界面是否稳定 | 打开client | 窗口能打开且保持稳定 |
| 注册信息是否能发到服务端 | 打开client，尝试注册，查看服务端是否接收到消息 | 服务端成功接到消息 |
| 注册信息是否存入数据库 | 打开client，输入注册信息，查看服务端的数据库是否存入了数据 | 服务端的数据库内有刚刚注册的相关信息 |
| 登录信息是否能发到服务端 | 打开client，尝试登录，查看服务端是否接收到消息 | 服务端成功接到消息 |
| 服务端能否返回注册、登录正确信息 | 打开client，尝试注册、登录，查看客户端是否接到消息 | 客户端成功接到服务端返回的信息 |
| 服务端能否返回登录错误信息 | 打开client，输入错误的登录信息，查看客户端能否接到登录错误的相关信息 | 客户端成功显示错误信息 |
| 客户端能否对错误的注册信息作出回应 | 打开client，输入错误的注册信息，客户端能否作出回应 | 客户端弹出错误信息窗口 |
| 主界面是否稳定 | 登录完成后，是否出现主界面且主界面是否稳定 | 主界面在不操作的情况下可以出现且能稳定 |
| 好友列表能否正常显示 | 登录完成后，切换至好友列表界面，观察是否正常 | 好友列表正常显示且没有错误 |
| 群列表能否正常显示 | 显示登录界面后，切换到群列表界面，观察是否正常 | 群列表正常显示且没有错误 |
| 是否能发送私聊请求 | 点击好友信息部分，右端出现与该好友聊天的历史记录 | 记录显示正常 |
| 是否能够发送添加好友请求 | 点击添加好友按钮，输入id，可以发送添加好友请求 | 服务端收到添加好友请求 |
| 是否可以发送改变头像的信息 | 客户端点击头像，可以发送改变头像信息 | 服务端收到改变头像信息 |
| 是否能够更新头像 | 服务端收到改变头像的信息后，发送一个ID，客户端的头像可以变为指定ID的图片 | 客户端的头像改变 |
| 服务端能够返回新的好友列表 | 服务端收到刷新好友列表的消息后，返回新的好友列表 | 客户端收到新的好友列表 |
| 客户端能否将信息发至服务端 | 客户端点击”发送“按钮，服务端可以收到客户端发送的信息 | 服务端成功接到客户端发出的消息 |
| 服务端能否将一个客户端发送的消息发到另一个客户端 | 服务端可以将收到的客户端的信息转发到另一个客户端上 | 客户端可以收到服务端发出的另一个客户端的信息 |
| 客户端是否可以发送表情 | 客户端点击表情按钮选择表情，另一个客户端可以收到表情 | 另一个客户端可以接收到表情 |
| 客户端是否可以发送加入群组请求 | 客户端点击加入群组，输入群组名字，可以加入群组 | 服务端接收到客户端发出的加入群组信息 |
| 服务端能否发送更新群组列表的信息 | 服务端发送更新群组列表信息，客户端可以接到这个消息 | 客户端的群组列表更新 |
| 客户端是否能够发送创建群组请求 | 客户端点击创建群组信息，可以被服务端接受 | 服务端收到客户端发出的创建群组信息 |
| 客户端能否在群组中发送消息 | 客户端发送消息被服务器接收后，消息可以被服务器发送到信息栏 | 客户端的历史信息的位置出现客户端发送的消息 |
| 服务器可以对客户端发送的不同信息进行不同处理 | 接收到客户端的信息后，服务器可以完成不同的响应步骤 | 客户端可以得到相应的回应 |
| 服务器端可以与客户端连接，连接不成功返回错误信息 | 客户端可以连接服务端，若连接错误则会弹出错误信息 | 客户端可以收到服务端的相应信息 |

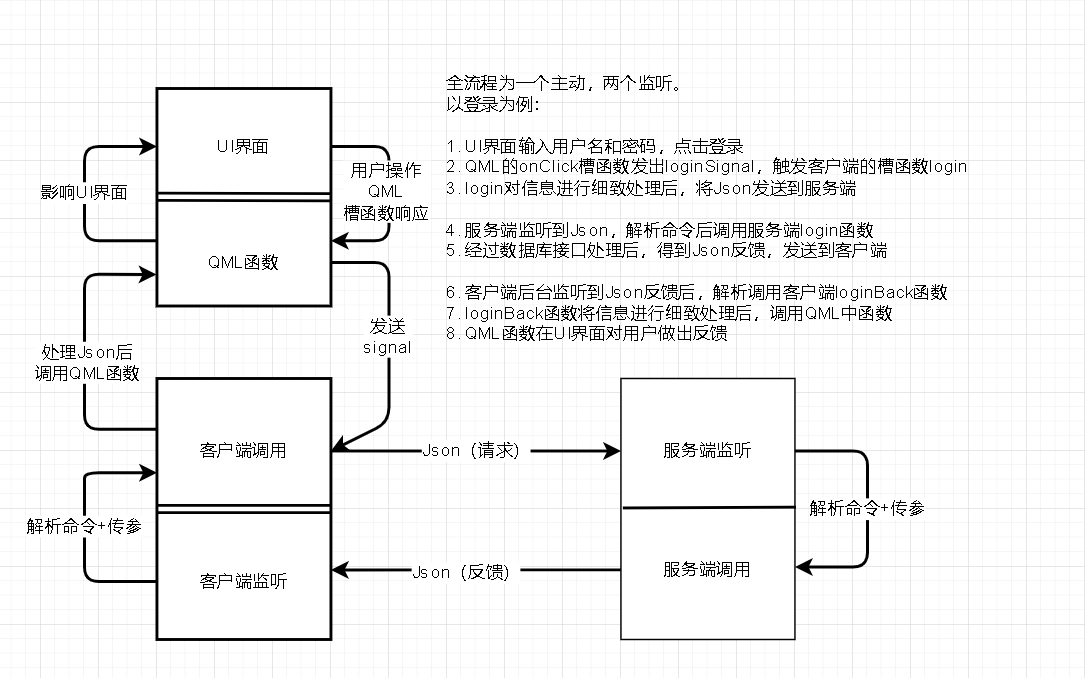
**6 心得体会**

这次实训，我有三点印象深刻。首先是通信工程师/架构师的工作。其次是，很高兴我能有一个强大的团队，最后是，好辛苦。

首先是通信工程师/架构师的工作。

刚开始我选择做通信仅仅是因为我没做过，我想学点新东西。但是从项目进行的第一天开始，我就产生了一股紧迫感，如何能实现客户端与服务端的通信困扰了我一晚上，辗转反侧后，我明白这个问题的本质。

从本质上来说，通信就是如何实现A客户端对B服务端的函数调用，这让我不由得联想到了Qt的信号与槽机制，我是否可以传输一个signal呢？说干就干，我直接无中生有，构思出了基于Json+command字段的通信架构：



经过一天的激动的测试，终于成功地实现了本地局域网中转的注册与登录。虽然只有两个功能，但是实际上已经走完了复杂的流程。从此以后，一切的功能，都可以由以上架构实现，本来我们的团队还在犹豫做局域网还是广域网，经过我的实践，团队达成一致：冲击广域网。

1. 我很早就醒来，写了一天的架构，将各种功能的接口，通信都测试出来，并且和QML前端同学将注册和登录链接起来。看着优美的UI进行各种变化，我明白，支撑其运行的是我的通信架构，UI为表，通信为骨，两相结合，相得益彰。
2. 我在QML里写各种信号，然后进行前后端通信，将项目不断整合。不得不说，github真的好用，虽然因为我们手法的稚嫩导致偶尔会有bug，但是我也切切实实地体会到了github在大项目中的作用，手动合并对项目的影响是灾难性的，git完美的解决了这个问题。经过一天的工作，项目被推进了一大截。看着我主导的架构被不断完善，成就感是很强的。

其次是，很高兴我能有一个强大的团队。说实话，我们能走到这一步已经超出了我的预料。我们的项目有多个亮点：

1）基于云服务器，我们实现了真正地广域网即时多人聊天

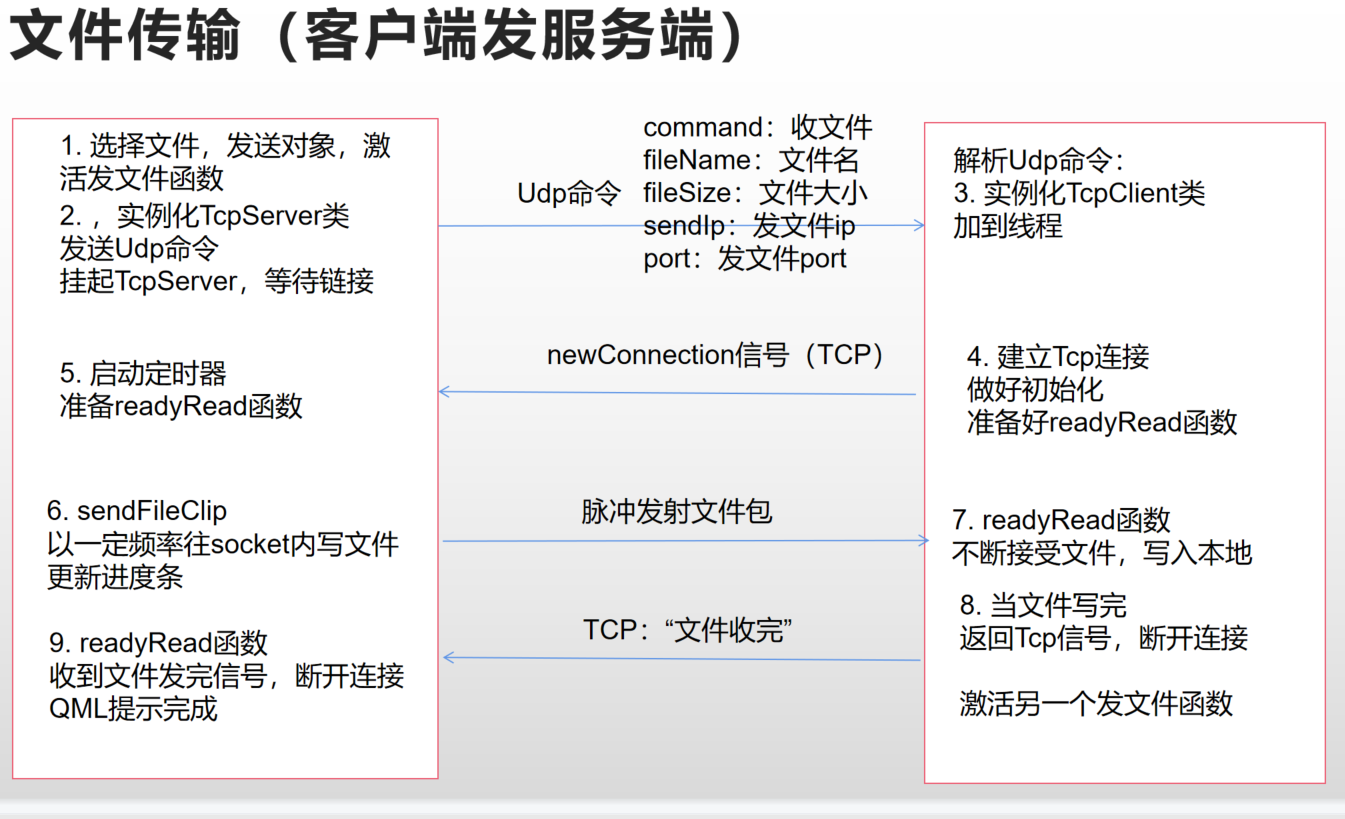
2）前端使用Qt Quick2.0， 以qml为开发语言进行代码开发

3）使用服务器中心化的数据库，提供历史消息漫游，离线也可以接受消息

4）良好的输入合法性检验，完善的反馈通信机制

其中，云服务器由我构思，而前端界面是采用QML架构，根据我的观察，采用QML技术的组非常稀少，很多人甚至都没听说过QML，而我的两个前端队友却能提出并践行，可见其能力强劲。后端这块，我们的强度也不低，除我之外，另外两个队友都有丰富的开发经验，包括服务器配置等经验，这也为我们的成功埋下伏笔。总之，即使我一个人可以干的很好，但是想干到这种程度，真的需要几个强大的队友联手才能实现。

最后是，好辛苦。虽然项目完善的比较好，但是其中免不了有很多bug，尤其是通信方面，甚至会出现udp的缓冲区不足导致报文截断，进而导致Json解析失败的bug，还有一些离谱的bug，与逻辑无关，是被底层牵连，比如明明signal能发出去，QML对象能抓取到，connect也没有报错，结果就是断了通讯的bug。因为各种意外，我们最后还是没有完成文件，即使我们在最后一天熬到了第二天4.30，见到了第二天的太阳，我们还是没能写完文件，我们只能尽可能地把功能完善。虽然没有写文件，但是我对文件的构思已经比较完善了，可惜天意弄人，学校提前安排答辩，哪怕给我半天时间，我也可以把文件传输实现。



小学期时间太紧张了，只有5天不到的项目时间，压力很大，希望以后能有所改善。