**重庆大学Project报告**

Project题目： 基于Java Socket的多人聊天系统

学 院： 计算机学院

专业班级： 计算机科学与技术05、04、06班

年 级： 2020级

姓 名： 杨奎、何锐、付作涛

学 号： 20204166、20204156、20204232

完成时间： 2022 年 12 月 20 日

成 绩：

指导教师： 陈自郁

重庆大学教务处制

Project教师评定成绩表

A. 课程设计的成绩构成

|  |  |  |  |
| --- | --- | --- | --- |
| 课程设计项目 | 该项目标准分值 | 评定分值（100分） | |
| 项目完成质量（含检查） | 项目报告 |
| **多人聊天系统** | 10 | 80% | 20% |

B. 课程设计项目的成绩评定标准

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 项目 | 得分情况 | | | | |
| 优秀  (90～100) | 良好  (80～89) | 中等  (70～79 | 及格  60～69 | 不及格  0～59 |
| 参考标准 | 参考标准 | 参考标准 | 参考标准 | 参考标准 |
| 项目完成质量 | 1.分工明确，独立完成课程设计项目，有源码  2.设计合理，功能完善，满足课程设计全部要求，且有拓展  3.界面友好，Bug极少，针对异常情况有处理  4.汇报检查时，讲解清晰，演示流畅，能正确回答问题 | 1.分工明确，独立完成课程设计项目，有源码  2.设计合理，功能完善，满足课程设计全部要求  3.界面良好，Bug较少，针对异常情况有处理  4.汇报检查时，讲解清晰，演示流畅，能正确回答问题 | 1.独立完成课程设计项目，有源码  2.功能较完善，基本满足课程设计要求  3.界面一般，Bug较多，无异常情况有处理  4.汇报检查时，讲解较清晰，完成演示，基本能正确回答问题 | 1.独立完成课程设计项目，有源码  2.功能基本完成，基本满足课程设计要求  3.界面一般，Bug很多，无异常情况有处理  4.汇报检查时，能基本讲清楚，主要功能能演示，基本能正确回答问题 | 1.抄袭或被抄袭以0分计，没有参与课程设计检查以0分计；  2.未能独立完成课程设计项目；  3.不能通过检查，不能进行课程设计汇报，主要功能无法演示，不能正确回答问题。 |
| 项目报告 | 1.结构严谨，逻辑清晰  2.文档规范，文字、图表表达清楚，用词专业，完全符合规范化要求  3.计算机打印  4.有完整的项目设计方案，采用的理论技术合理正确，结果及分析正确，功能有拓展  5.报告内容与课程设计检查结果相符 | 1.结构合理，符合逻辑；  2. 文档规范，文字、图表表达较清楚，用词专业，符合规范化要求  3.计算机打印  4.有较完整的项目设计方案，采用的理论与技术正确，结果及分析正确  5.报告内容与课程设计检查结果相符 | 1.结构合理，符合逻辑  2. 文档较规范，文字、图表表达较清楚，基本达到规范化要求  3.计算机打印  4.项目设计方案基本完整，采用的理论与技术基本正确，结果及分析基本正确。  5.报告内容与课程设计检查结果相符 | 1.结构基本合理,逻辑基本清楚  2. 文档较规范，文字、图表表达较清楚，勉强达到规范要求  3.计算机打印  4.项目设计方案有欠缺，采用的理论与技术基本可行，没有测试结果与结果分析  5.报告内容与课程设计检查结果相符 | 1.抄袭与被抄袭以0分计；  2.结构混乱，内容空泛，文字表达不清，错别字较多，达不到规范化要求  3.人工书写或计算机打印  4.报告内容与课程设计检查结果不相符 |

说明：评定分值要折算成项目标准分值。课程设计项目得分 = 课程设计项目的评定分值\*10%

指导教师评定成绩：

指导教师签名： 年 月 日

**目 录**

[1 分工说明 1](#_Toc22527)

[2 需求分析 2](#_Toc27531)

[3 系统设计 2](#_Toc13246)

3.1 服务器端.........................................................................................................2

3.2 客户端.............................................................................................................3

3.3 类图.................................................................................................................3

[4 关键代码（技术）描述 4](#_Toc31885)

[4.1 服务器端关键代码（技术） 4](#_Toc6454)

4.1.1 Server类的重要属性..................................................................................................4

4.1.2 Server类开启服务的方法..........................................................................................4

4.1.3 服务器端发送信息给指定用户的方法....................................................................6

4.1.4 实现消息群发与私发的方法....................................................................................7

4.1.5 ServerFileThread类的重要属性.................................................................................9

4.1.6 ServerFileThread类创建Socket连接........................................................................9

4.1.7 实现文件的私发与群发..........................................................................................10

[4.2 客户端关键代码和技术 1](#_Toc19820)2

4.2.1 建立Socket连接......................................................................................................12

4.2.2 消息接收与添加功能的实现..................................................................................13

4.2.3 消息发送功能的实现..............................................................................................14

4.2.4 文件接收与存储功能的实现..................................................................................14

4.2.5 文件发送功能的实现..............................................................................................15

4.3 用户实体类....................................................................................................16

4.3.1 用户实体类的重要属性..........................................................................................16

4.3.2 UserDao类的重要方法............................................................................................16

[5 系统测试运行 1](#_Toc25065)7

5.1 服务器端调试与测试...................................................................................17

5.2 客户端调试与测试.......................................................................................18

5.3 测试总结.......................................................................................................19

[6 项目运行效果 1](#_Toc25093)9

6.1 客户端界面...................................................................................................19

6.2 服务器端界面...............................................................................................20

6.3 群聊功能实现效果.......................................................................................20

6.4 私聊功能实现效果.......................................................................................22

6.5 文件/图片发送功能实现效果......................................................................23

6.6 服务器端的强制下线功能...........................................................................25

7 项目总结................................................................................................................................................................27

7.1 项目的收获...................................................................................................27

7.2 项目存在的不足...........................................................................................27

# 

# **1 分工说明**

课题任务分工说明

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 姓名 | 学号 | 角色 | 承担的任务 | 备注 |
| 1 | 杨奎 | 20204166 | 组长 | 1. 负责服务器端聊天、连接数据库等功能的实现 2. 参与实现服务器端和客户端的文件传输功能   （2）整合项目各部分代码，保障整体运行  （3）撰写PPT提纲，修改完善PPT  （4）进行项目答辩讲解  （5）负责项目报告的分工说明、系统设计部分，并且整合项目报告 |  |
| 2 | 何锐 | 20204156 | 组员 | （1）负责服务器端和客户端的前端界面开发  （2）参与聊天功能与文件传输功能的实现  （3）负责项目报告的关键代码描述、系统测试报告、运行效果、总结部分  （4）负责项目的调试与debug |  |
| 3 | 付作涛 | 20204232 | 组员 | （1）参与实现客户端聊天功能的实现  （2）根据PPT提纲制作PPT  （3）撰写项目报告的需求分析部分  （4）参与项目的调试工作 |  |

# **2 需求分析**

使用 Java Socket API 实现一个能部署在Internet上的多人聊天系统

（1）实现基于服务器转发的多个客户间的数据共享与交换，共享和交换的数据包括文字、图片和文件；

（2）服务器端记录当前在线客户列表，客户端能够获取该客户列表，并实时刷新；

（3）任一个客户可以和任意其它的客户进行交互，即支持一对一通信、组通信；

（4）拥有良好的用户图形界面，便于系统的使用；

（5）服务器端拥有强制下线的功能，并将下线信息通报给其他所有用户端；

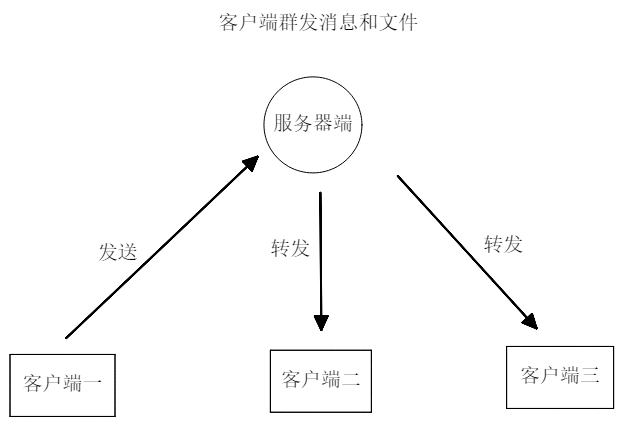
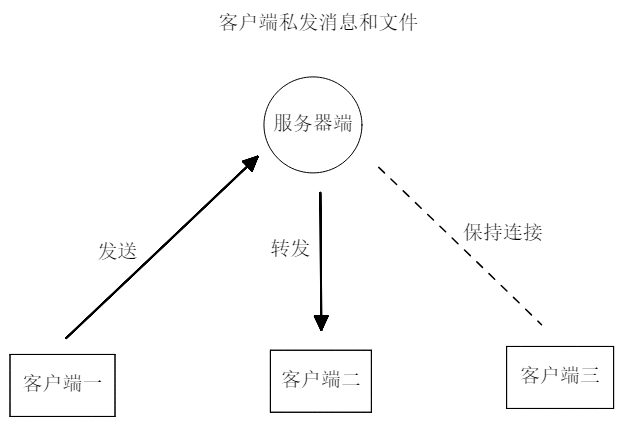
（6）基于传输层采用TCP协议的SOCKET编程实现；

（7）消息传输和文件传输线程互不干扰，可以同时进行；

（8）消息采用JSON格式，特定字段携带特定含义。

# **3 系统设计**

系统实现的是基于TCP协议的多人聊天系统，在此系统中，服务器端负责客户端的数据共享与交换。

3.1 服务器端

**A.消息的转发**

服务器端总是监听着信道，当有消息到达时，提取出用户字段，确认消息发送的对象（可以为所有用户或者某个特定用户），并将其转发至对应的客户端。同时，服务器端将消息的相关信息呈现在服务器的信息显示区域，包括信息发送时间、内容、发送人、发送对象。

当用户上线时，向服务器端发送上线通知，服务器接收通知并更新在线用户列表，然后将该用户上线通知转发给其他所有用户。当用户下线时，与之类似。

当用户被强制下线时，在在线用户列表中删除对应用户，向其他所有用户发送强制下线通知。

**B.文件的转发**

# 服务器端总是监听信道，当有文件二进制流到达时，先读取出文件发送的目的用户，（可以为所有用户或者某个特定用户），并将其转发至对应的客户端。

# 服务器端不存放文件的具体信息，只进行发送的筛选。

**C.维护在线用户列表**

服务器端维护用户名列表与用户列表。

用户名列表用于服务器端在界面上显示出在线用户的名字

用户列表的每一个元素是封装后的用户类，服务器端对该用户列表进行上线、下线、发送消息状态的更新、查询操作。

3.2 客户端

**A.消息的发送**

用户可以通过双击界面上的在线用户列表（双击自己没有作用）切换到对应聊天界面，在消息框输入消息后进行发送。

用户上线时，会给服务器端发发送上线通知，下线时同理发送下线通知。

用户发送消息前，先设置JSON对象的时间、发送对象等字段，然后提取用户输入的消息设置消息字段，写入输出流。

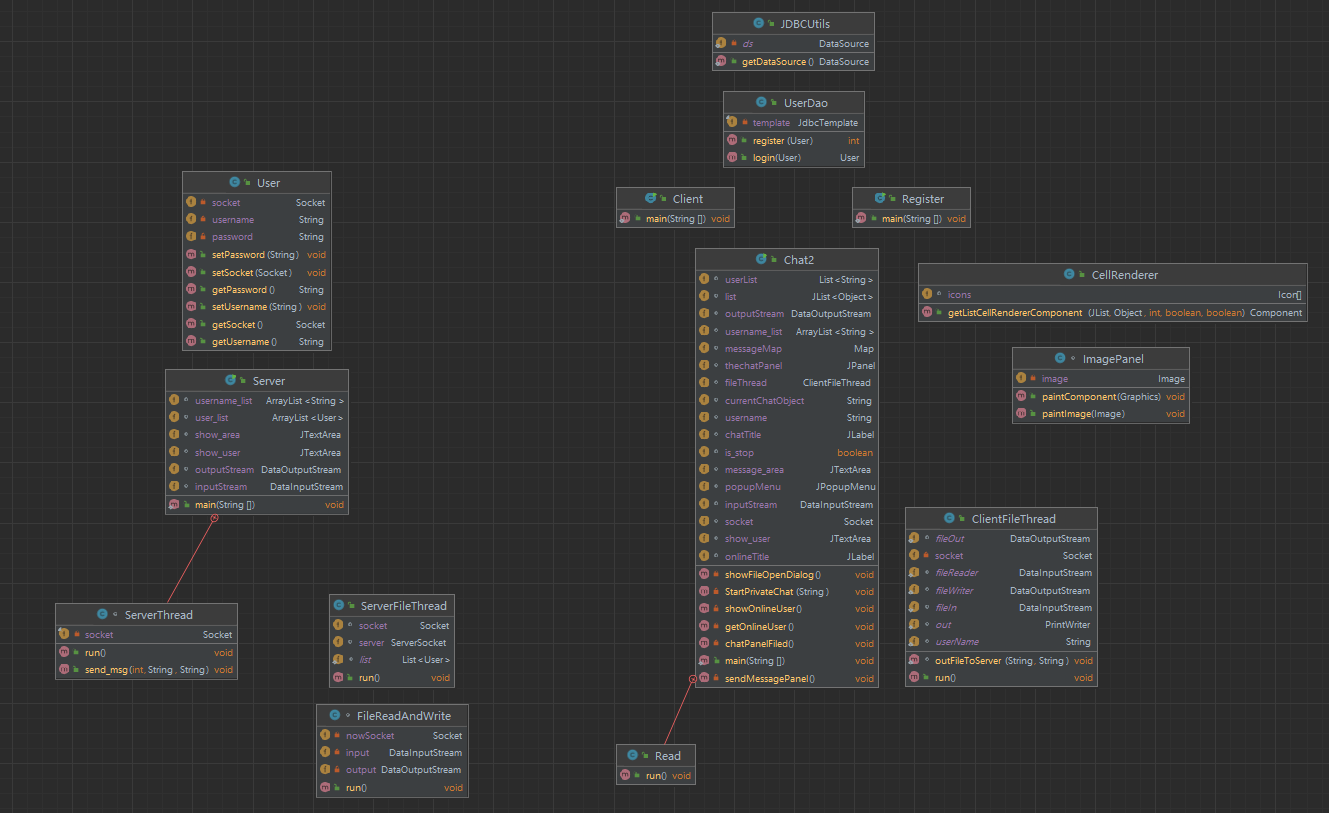
**B.文件的发送**

用户可以自由选择发送文件的位置与类型，客户端有专门处理文件发送与接收的线程，文件的传输以二进制的形式进行。

在进行文件的正式内容传输之前，要先进行文件名字、发送对象和文件长度写入流中。

3.3 类图

类与类之间的相互关系如下所示:



# **4 关键代码（技术）描述**

### 4.1 服务器端关键代码（技术）

4.1.1 Server类的重要属性

//用户列表，用于存放连接上的用户信息ArrayList<User> **user\_list** = **new** ArrayList<>();  
//用户名列表，用于显示已连接上的用户ArrayList<String> **username\_list** = **new** ArrayList<>();  
//socket的数据输出流  
DataOutputStream **outputStream** = **null**;  
//socket的数据输入流  
DataInputStream **inputStream** = **null**;

4.1.2 Server类开启服务的方法

创建Socket端口后持续监听端口，将成功连接的端口添加到连接列表中

ServerSocket serverSocket = **new** ServerSocket(11111);  
//Sim信息显示区打印服务器启动时间  
pleDateFormat sdf = **new** SimpleDateFormat(**"YYYY年MM月dd日 HH:mm:ss"**);  
String time = sdf.format(**new** Date());  
**show\_area**.append(**"服务启动时间 "** + time + **"\n"**);  
//持续接收连接  
**while** (**true**) {  
 //接收连接  
 Socket socket = serverSocket.accept();  
 //创建用户对象  
 User user = **new** User();  
 //判断是否连接上  
 **if** (socket != **null**) {  
 //获取输入流  
 **inputStream** = **new** DataInputStream(socket.getInputStream());  
 //读取输入流  
 String json = **inputStream**.readUTF();  
 //创建信息对象  
 JSONObject data = JSONObject.fromObject(json);  
 //信息显示区打印用户上线  
 **show\_area**.append(**"用户 "** + data.getString(**"username"**) + **" 在"** + **new** Date() + **"登陆系统"**+**"\n"**);  
 //创建新用户  
 user = **new** User();  
 //存储socket对象  
 user.setSocket(socket);  
 //获取输入流用户名  
 user.setUsername(data.getString(**"username"**));  
 //添加进用户列表  
 **user\_list**.add(user);  
 //添加进用户名列表  
 **username\_list**.add(data.getString(**"username"**));  
  
 //刷新在线人数  
 **show\_user**.setText(**"人数有 "** + **username\_list**.size() + **" 人\n"**);  
 //刷新在线用户  
 **for** (String s : **username\_list**) {  
 **show\_user**.append(s + **"\n"**);  
 }  
 }  
 //封装信息对象  
 JSONObject online = **new** JSONObject();  
 //设置接收信息对象  
 online.put(**"user\_list"**, **username\_list**);  
 //设置信息内容  
 online.put(**"msg"**, user.getUsername() + **" 上线了"**);  
 online.put(**"private"**,**"groupChat"**);  
 //依次遍历，将信息广播给所有在线用户  
 **for** (User value : **user\_list**) {  
 //获取输出流  
 **outputStream** = **new** DataOutputStream(value.getSocket().getOutputStream());  
 //给所有用户输出上线信息  
 **outputStream**.writeUTF(online.toString());  
 }  
 //开启新线程，持续接收该socket信息  
 **new** Thread(**new** ServerThread(socket)).start();

4.1.3 服务器端发送信息给指定用户的方法

首先构建JSONObject对象然后封装信息至该对象中，然后通过输出流发送对象

**public void** send\_msg(**int** i, String msg, String pri) {  
 //构建对象  
 JSONObject data = **new** JSONObject();  
 //封装信息  
 data.put(**"user\_list"**, **username\_list**);  
 data.put(**"msg"**, msg);  
 data.put(**"private"**,pri);  
 //获取目标对象  
 User user = **user\_list**.get(i);  
 **try** {  
 //获取输出流  
 **outputStream** = **new** DataOutputStream(user.getSocket().getOutputStream());  
 //写信息  
 **outputStream**.writeUTF(data.toString());  
 } **catch** (IOException e) {  
 //如果没有找到，则说明该用户已经下线  
 User out\_user = **user\_list**.get(i);  
 //重复删除操作  
 **user\_list**.remove(i);  
 **username\_list**.remove(out\_user.getUsername());  
 //重新构建信息  
 JSONObject out = **new** JSONObject();  
 out.put(**"user\_list"**, **username\_list**);  
 out.put(**"msg"**, out\_user.getUsername() + **" 下线了\n"**);  
 //将其下线通知广播给所有用户  
 **for** (User value : **user\_list**) {  
 **try** {  
 **outputStream** = **new** DataOutputStream(value.getSocket().getOutputStream());  
 **outputStream**.writeUTF(out.toString());  
 } **catch** (IOException ex) {  
 ex.printStackTrace();  
 }  
 }  
 }  
}

4.1.4 实现消息群发与私发的方法

获取输入流后接收信息，根据信息中的内容选择群发与私发的方法

//获取输入流  
DataInputStream inputStream = **new** DataInputStream(**socket**.getInputStream());  
//持续接收信息  
**while** (**true**) {  
 //获取传递进来的信息  
 String json = inputStream.readUTF();  
 //封装成json格式  
 JSONObject data = JSONObject.fromObject(json);  
  
 //通过json里面的private判断是否私发  
 **boolean** is\_private = **false**;  
 //私发处理  
 **for** (**int** i = 0; i < **user\_list**.size(); i++) {  
 //找到私发对象  
 **if** (**user\_list**.get(i).getUsername().equals(data.getString(**"private"**))) {  
 //构建私发信息内容  
 String msg = data.getString(**"username"**)+**" "** + data.getString(**"time"**) + **"\n"** + data.getString(**"msg"**);  
 String pri = data.getString(**"username"**);  
 //用该方法指定对象发送信息  
 send\_msg(i, msg, pri);  
  
 //将发送成功反馈给原用户  
 **for** (**int** j = 0; j < **user\_list**.size(); j++) {  
 //找到发信息用户  
 **if**(**user\_list**.get(j).getUsername().equals(data.getString(**"username"**))){  
 //构建反馈信息内容  
 String msg2 = data.getString(**"username"**)+**" "**+  
 data.getString(**"time"**)+**"\n"**+data.getString(**"msg"**);  
 String pri1 = **user\_list**.get(i).getUsername();  
 //用该方法指定对象发送信息  
 send\_msg(j,msg2, pri1);  
 }  
 }  
 //将该操作打印到服务器监视窗  
 **show\_area**.append(data.getString(**"username"**) +data.getString(**"time"**)+ **"私发给"** + data.getString(**"private"**) + **":\n"** + data.getString(**"msg"**) + **"\n"**);  
 //判断是私发  
 is\_private = **true**;  
 **break**;  
 }  
 }  
 //非私发的情况  
 **if** (!is\_private) {  
 //构建信息内容  
 String msg = data.getString(**"username"**) + **" "** + data.getString(**"time"**) + **"\n"** + data.getString(**"msg"**);  
 //添加到服务器显示  
 **show\_area**.append(msg + **"\n"**);  
 //依次发给所有在线用户  
 **for** (**int** i = 0; i < **user\_list**.size(); ) {  
 String pri2 = **"groupChat"**;  
 send\_msg(i, msg, pri2);  
 i++;  
 }  
 }  
}

4.1.5 ServerFileThread类的重要属性

//存储的客户端类

**static** List<User> *list* = **new** ArrayList<User>();

4.1.6 ServerFileThread类创建Socket连接

**server** = **new** ServerSocket(8090);  
System.***out***.println(**"ServerFileThread启动成功"**);  
**while**(**true**) {  
 System.***out***.println(**"添加socket中..."**);  
 **socket** = **server**.accept();  
 //获取输入流DataInputStream in = **new** DataInputStream(**socket**.getInputStream());  
 //读取输入流String json = in.readUTF();  
 //创建信息对象JSONObject data = JSONObject.*fromObject*(json);  
 //创建新用户User user = **new** User();  
 //存储socket对象user.setSocket(**socket**);  
 //获取输入流用户名  
 user.setUsername(data.getString(**"username"**));  
 //添加进用户列表  
 list.add(user);  
 System.**out**.println(list);  
  
 System.**out**.println(**"与服务器端socket连接成功"**);  
 //开启文件传输线程  
 FileReadAndWrite fileReadAndWrite = **new** FileReadAndWrite(**socket**);  
 fileReadAndWrite.start();  
}

4.1.7 实现文件的私发与群发

**input** = **new** DataInputStream(**nowSocket**.getInputStream()); // 获取输入流  
**while** (**true**) {  
 //获取文件名字和文件长度  
 String textName = **input**.readUTF();  
 String people = **input**.readUTF(); //私发对象  
 System.**out**.println(**"私发对象"** + people);  
 **long** textLength = **input**.readLong();  
 System.**out**.println(**"文件名"**+textName+**" 私发对象"**+people+**" 文件长度"**+textLength);  
 **if** (people.equals(**"groupChat"**)){  
 System.**out**.println(**"进行文件群发"**);  
 **for**(User u: ServerFileThread.list) {  
 **if** (u.getSocket()!=**nowSocket**){  
 **output** = **new** DataOutputStream(u.getSocket().getOutputStream()); // 输出流  
 **output**.writeUTF(textName);  
 **output**.flush();  
 **output**.writeLong(textLength);  
 **output**.flush();  
 }  
 }  
 System.**out**.println(**"文件群发完毕"**);  
 }**else**{  
 System.**out**.println(**"进行文件私发"**);  
 //发送文件名字和文件长度给所有客户端  
 **for**(User u: ServerFileThread.list) {  
 **if** (u.getUsername().equals(people)) { //私发对象  
 **output** = **new** DataOutputStream(u.getSocket().getOutputStream()); //输出流  
 **output**.writeUTF(textName);  
 **output**.flush();  
 **output**.writeLong(textLength);  
 **output**.flush();  
 }  
 }  
 }  
  
 **byte**[] buff = **new byte**[1024];  
 **int** length = -1;  
 **long** curLength = 0;  
 **if** (people.equals(**"groupChat"**)){  
 System.**out**.println(**"进行文件群发"**);  
 **while** ((length = **input**.read(buff)) > 0){  
 curLength += length;  
 **for**(User u: ServerFileThread.list) {  
 **if** (u.getSocket()!=**nowSocket**){  
 **output** = **new** DataOutputStream(u.getSocket().getOutputStream()); //输出流  
 **output**.write(buff, 0, length);  
 **output**.flush();  
 }  
 }  
 **if** (curLength >= textLength) { //强制退出  
 System.**out**.println(**"文件群发完毕"**);  
 **break**;  
 }  
 }  
 }**else**{  
 System.**out**.println(**"进行文件私发"**);  
 **for**(User u: ServerFileThread.list) {  
 **if** (u.getUsername().equals(people)) { //私发对象  
 **output** = **new** DataOutputStream(u.getSocket().getOutputStream()); //输出流  
 }  
 }  
 **while** ((length = **input**.read(buff)) > 0){  
 curLength += length;  
 **output**.write(buff, 0, length);  
 **output**.flush();  
 **if** (curLength >= textLength) { //强制退出  
 System.**out**.println(**"文件发送完毕"**);  
 **break**;  
 }  
 }  
 }  
}

### 4.2 客户端关键代码和技术

4.2.1 建立Socket连接

//建立socket连接**try** {  
 **socket** = **new** Socket(**"127.0.0.1"**, 11111);  
 **inputStream** = **new** DataInputStream(**socket**.getInputStream());  
 **outputStream** = **new** DataOutputStream(**socket**.getOutputStream());  
 **outputStream**.writeUTF(data.toString());  
 **new** Thread(**new** Read()).start();  
 System.**out**.println(**"建立连接成功"**);  
} **catch** (IOException e) {  
 **message\_area**.append(**"服务器无响应"**);  
 JOptionPane.showMessageDialog(**null**,**"服务器无响应"**,**"提示"**,  
 JOptionPane.**WARNING\_MESSAGE**);  
}

4.2.2 消息接收与添加功能的实现

**while** (**true**) {  
 String json = **inputStream**.readUTF();  
 JSONObject data = JSONObject.fromObject(json);  
 String msg = data.getString(**"msg"**);  
 **if** (msg.contains(**"踢出"**) && msg.contains(**username**)) {  
 System.**out**.println(**"已被强制下线"**);  
 **is\_stop** = **true**;  
 **message\_area**.append(**username** + **",你已经被强制下线\n"**);  
 JOptionPane.showMessageDialog(**null**,**"你已经被强制下线"**,**"提示 "**+**username**,  
 JOptionPane.**WARNING\_MESSAGE**);  
 System.exit(0);  
 } **else** {  
 String pri = data.getString(**"private"**);  
 **currentChatObject** = pri;  
 System.**out**.println(pri);  
 StartPrivateChat(pri);  
 JTextArea temp= (JTextArea) **messageMap**.get(**currentChatObject**);  
 temp.append(msg + **"\n"**);  
 System.**out**.println(**"内容："**+msg+**"添加到"**+**currentChatObject**);  
 **message\_area**.setText(temp.getText());  
 **messageMap**.replace(**currentChatObject**, temp);  
 //获取所有用户  
 **username\_list**.clear();  
 JSONArray jsonArray = data.getJSONArray(**"user\_list"**);  
 **for** (Object o : jsonArray) {  
 **username\_list**.add(o.toString());  
 }  
 //在线总人数**show\_user**.setText(**"人数有 "** + jsonArray.size() + **" 人\n"**);  
 **onlineTitle**.setText(**"用户列表("**+jsonArray.size()+**")"**);  
 **for** (String s : **username\_list**) {  
 **show\_user**.append(s + **"\n"**);  
 }  
 **userList**.clear();  
 **for** (**int** i=**username\_list**.size()-1;i>=0;i--) {  
 **userList**.add(**username\_list**.get(i));  
 }  
 getOnlineUser();  
 }  
}

4.2.3 消息发送功能的实现

SimpleDateFormat sdf = **new** SimpleDateFormat(**"MM月dd日 HH:mm"**);  
String time = sdf.format(**new** Date());  
String msg = send\_area.getText().trim();  
**if** (!msg.equals(**""**)) {  
 JSONObject data = **new** JSONObject();  
 data.put(**"username"**, **username**);  
 data.put(**"msg"**, msg);  
 data.put(**"time"**, time);  
  
 **try** {  
 data.put(**"private"**, **currentChatObject**);  
  
 } **catch** (ArrayIndexOutOfBoundsException e1) {  
 data.put(**"private"**, **""**);  
 }  
 **outputStream**.writeUTF(data.toString());  
}

4.2.4 文件接收与存储功能的实现

String textName = fileIn.readUTF();  
**long** totleLength = fileIn.readLong();  
**int** result = JOptionPane.showConfirmDialog(**new** JPanel(), **"是否接受？"**, **"提示 "**+userName,  
 JOptionPane.**YES\_NO\_OPTION**);  
**int** length = -1;  
**byte**[] buff = **new byte**[1024];  
**long** curLength = 0;  
//提示框选择结果，0为确定，1位取消  
**if**(result == 0){  
 File userFile = **new** File(**"/Users/herui/ChatWAY/接收文件/"**+userName);  
 **if**(!userFile.exists()) { //新建当前用户的文件夹  
 userFile.mkdir();  
 }  
 File file = **new** File(**"/Users/herui/ChatWAY/接收文件/"** + userName + **"/"** + textName);  
 System.**out**.println(textName);  
 fileWriter = **new** DataOutputStream(**new** FileOutputStream(file));  
 **while**((length = fileIn.read(buff)) > 0) { //把文件写进本地  
 fileWriter.write(buff, 0, length);  
 fileWriter.flush();  
 curLength += length;  
 **if**(curLength == totleLength) { //强制结束  
 **break**;  
 }  
 }  
 System.**out**.println(**"文件存放地址：\n"** + **"/Users/herui/ChatWAY/接收文件/"** + userName + **"/"** + textName);  
 //提示文件存放地址  
 JOptionPane.showMessageDialog(**new** JPanel(), **"文件存放地址：\n"** +  
 **"/Users/herui/ChatWAY/接收文件/"** +  
 userName + **"/"** + textName, **"提示"**, JOptionPane.**INFORMATION\_MESSAGE**);  
}  
**else** { //不接受文件  
 **while**((length = fileIn.read(buff)) > 0) {  
 curLength += length;  
 **if**(curLength == totleLength) { // 强制结束  
 **break**;  
 }  
 }  
}  
fileWriter.close();

4.2.5 文件发送功能的实现

System.**out**.println(**"客户端发送文件"**);  
File file = **new** File(path);  
fileReader = **new** DataInputStream(**new** FileInputStream(file));  
  
fileOut.writeUTF(file.getName()); //发送文件名字  
fileOut.flush();  
  
fileOut.writeUTF(currentChatObject); //发送私发对象  
fileOut.flush();  
System.**out**.println(**"发送文件的对象"** + currentChatObject);  
  
fileOut.writeLong(file.length()); //发送文件长度  
fileOut.flush();  
**int** length = -1;  
**byte**[] buff = **new byte**[1024];  
System.**out**.println(**"正在发送..."**);  
**while** ((length = fileReader.read(buff)) > 0) { //发送内容  
 fileOut.write(buff, 0, length);  
 fileOut.flush();  
}  
System.**out**.print(**"发送完毕"**);

4.3 用户实体类

4.3.1 用户实体类的重要属性

**private** String **username**; //用户名  
**private** String **password**; //密码  
**private** Socket **socket**; //socket

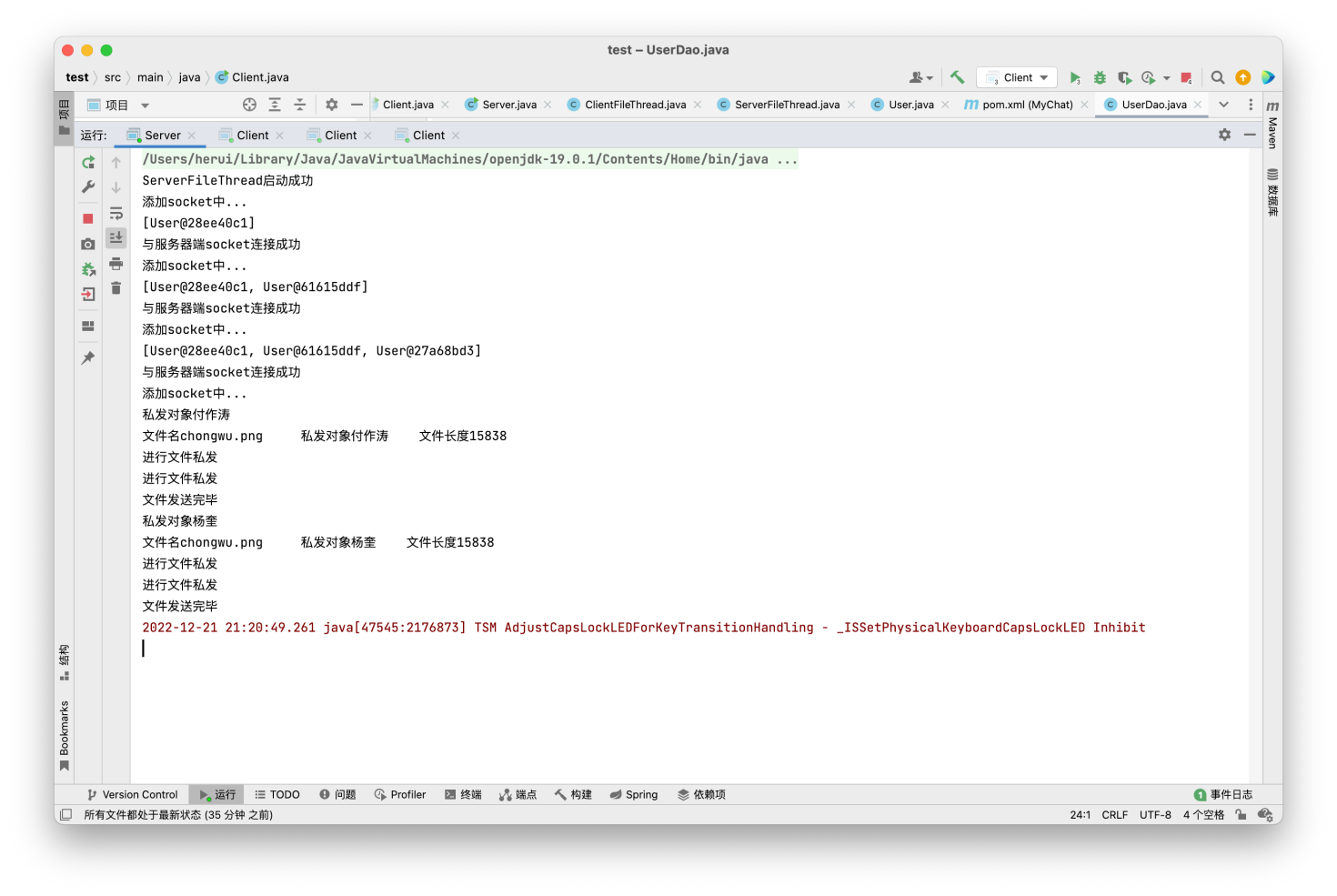
4.3.2 UserDao类的重要方法

**private final** JdbcTemplate **template** = **new** JdbcTemplate(JDBCUtils.*getDataSource*());

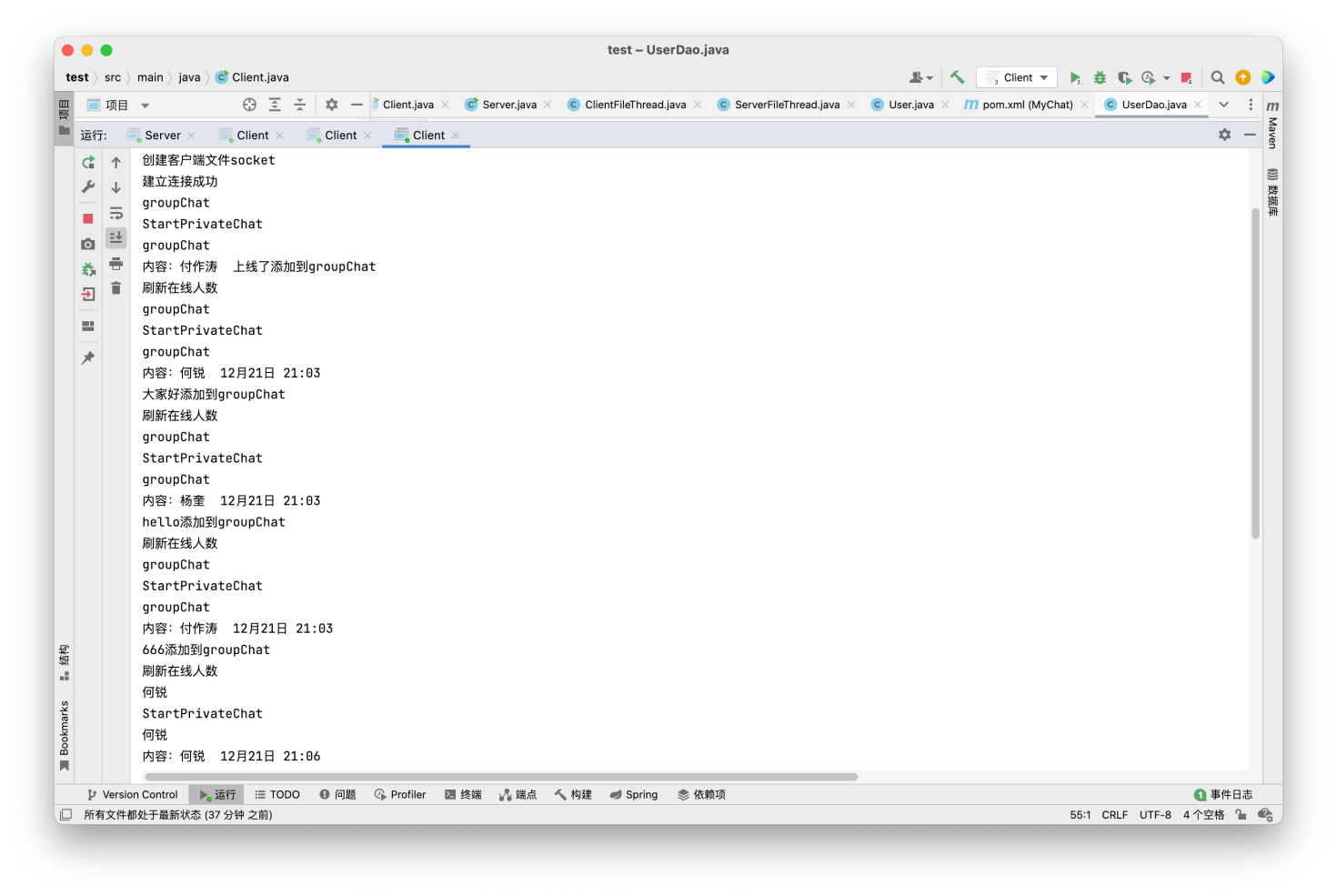
//创建JdbcTemplate对象,依赖于DataSource  
**public** User login(User login\_user) {  
 **try** {  
 //编写sqlString sql = **"select \* from user where username = ? and password = ?"**;  
 User user = **template**.queryForObject(sql,  
 **new** BeanPropertyRowMapper<User>(User.**class**),  
 login\_user.getUsername(), login\_user.getPassword());  
 **return** user;  
 } **catch** (DataAccessException e) {  
 e.printStackTrace();  
 **return null**;  
 }  
}  
  
**public int** register(User register\_user) {  
 **try** {  
 String sql = **"insert into user values (?,?)"**;  
 //返回行数**int** count = **template**.update(sql,register\_user.getUsername(),register\_user.getPassword());  
 **return** count;  
 } **catch** (DataAccessException e) {  
 e.printStackTrace();  
 **return** 0;  
 }  
}

# **5 系统测试运行**

5.1 服务器端调试与测试



5.2 客户端调试与测试



5.3 测试总结

服务器端能够正常创建Socket，客户端能够成功地与服务器端成功连接。客户端与服务器的通信与数据传输能够正常进行，JSON文件、图片文件和其他任意类型的文件能够正常传输。客户端对消息文件的处理能够正常进行，私聊与群聊功能、图片选择、文件选择、文件保存等功能能够按照期望运行。服务器端对消息的转发功能经过测试也能够正确转发。

# **6 项目运行效果**

6.1 客户端界面



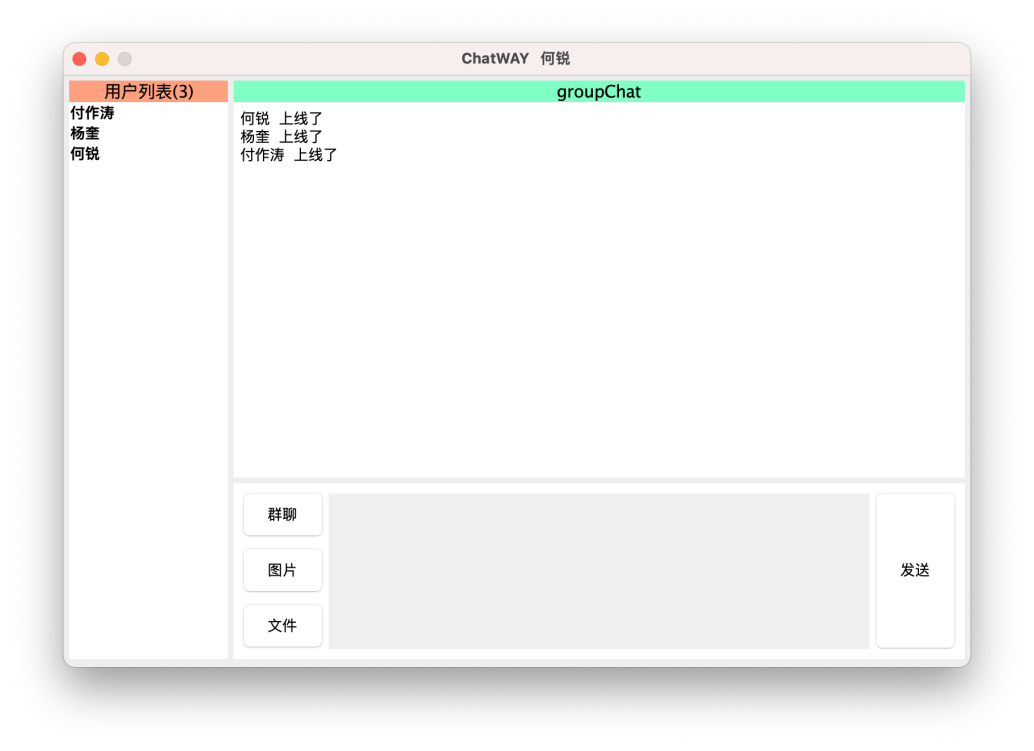
图 客户端登录界面 图 客户端注册界面

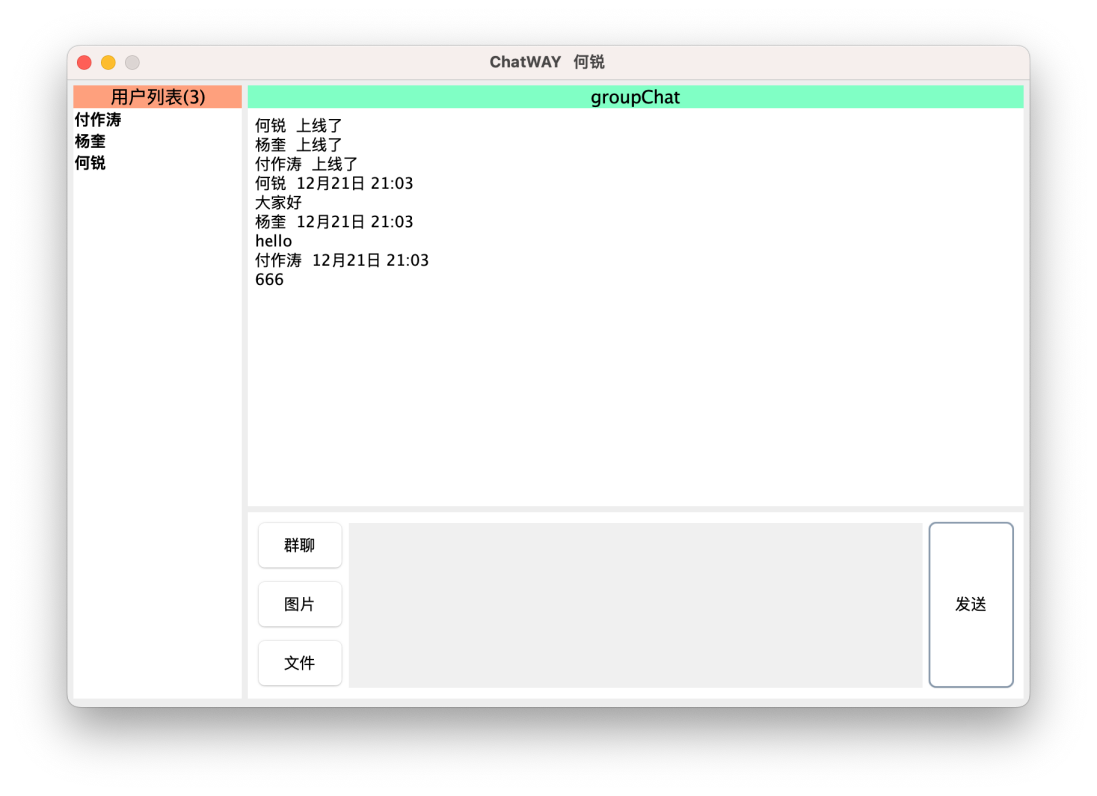
图 客户端聊天界面

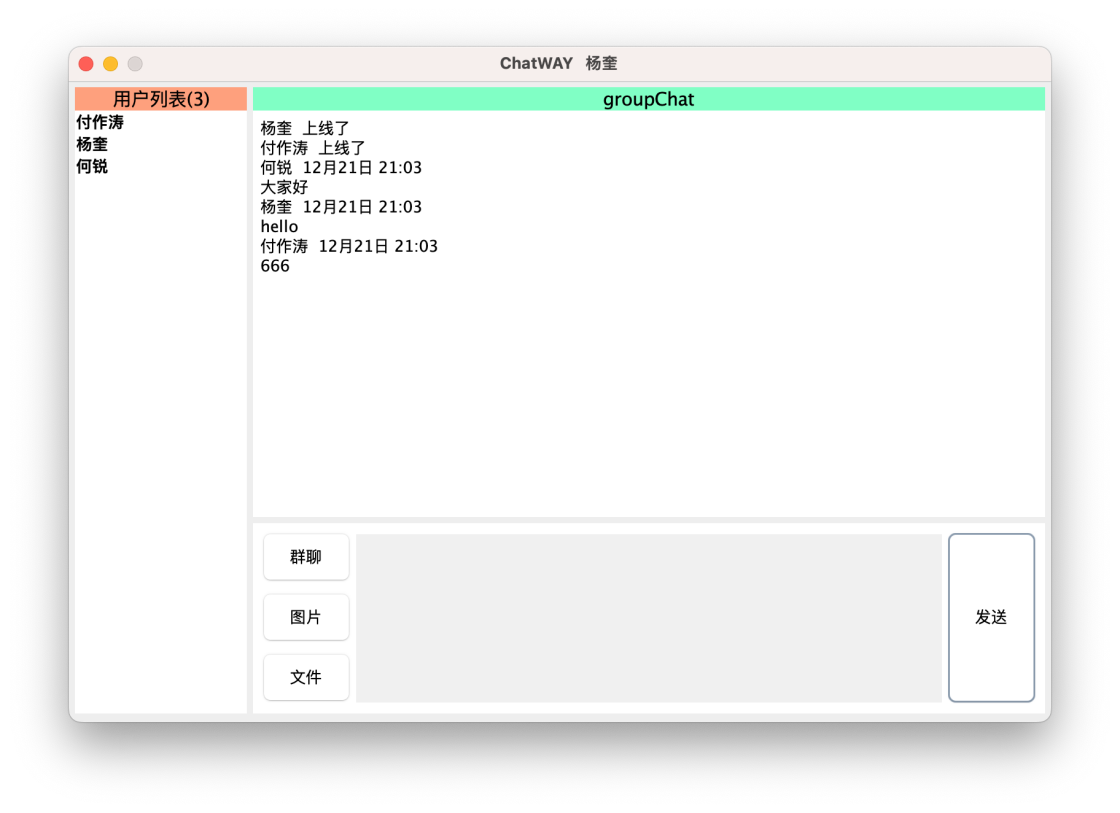
6.2 服务器端界面

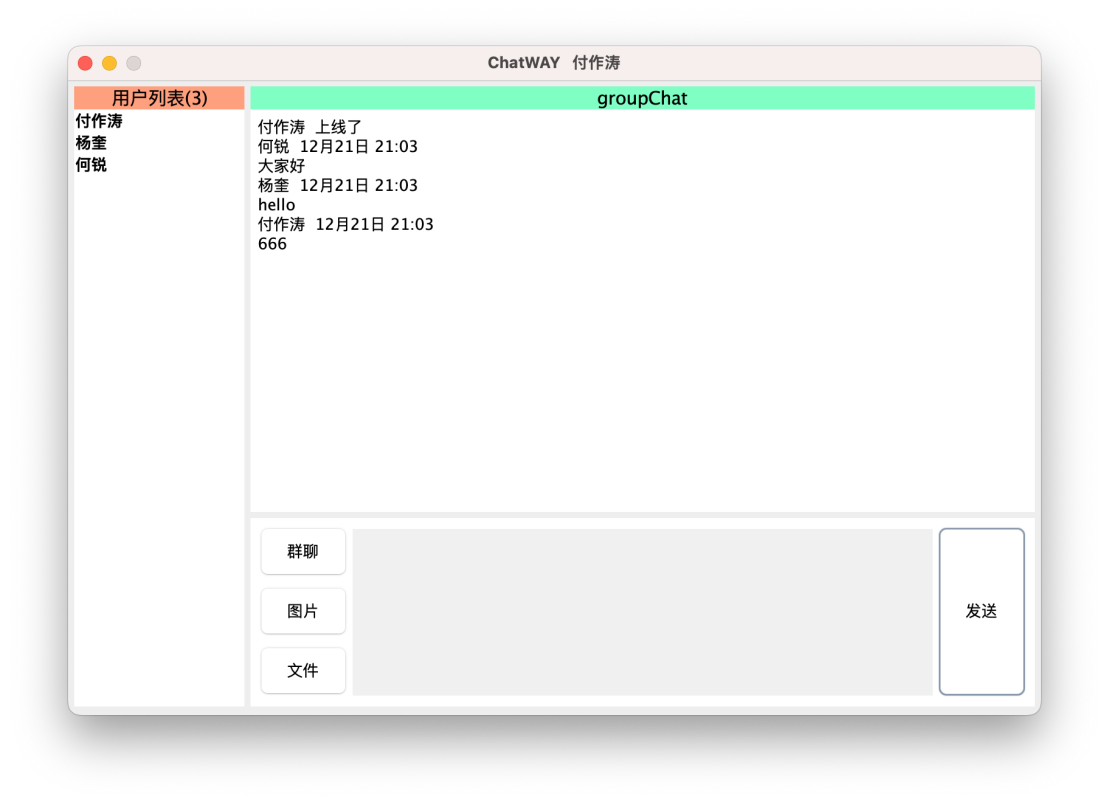


图 服务器端可视化界面

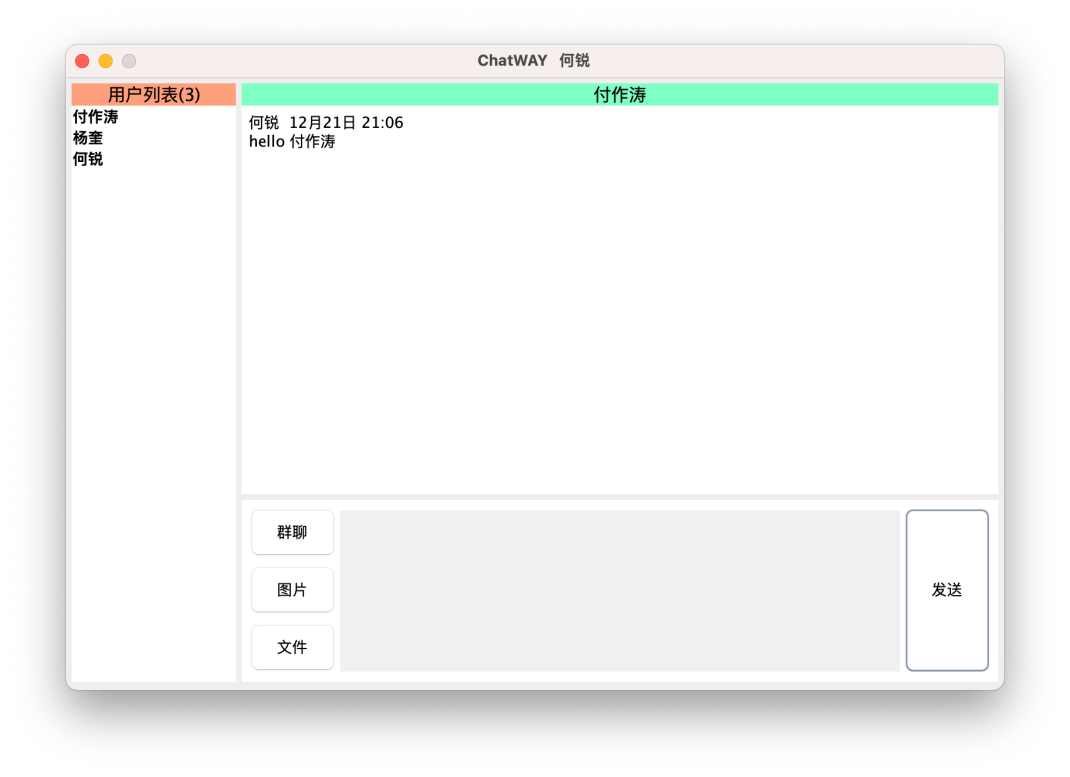
6.3 群聊功能实现效果





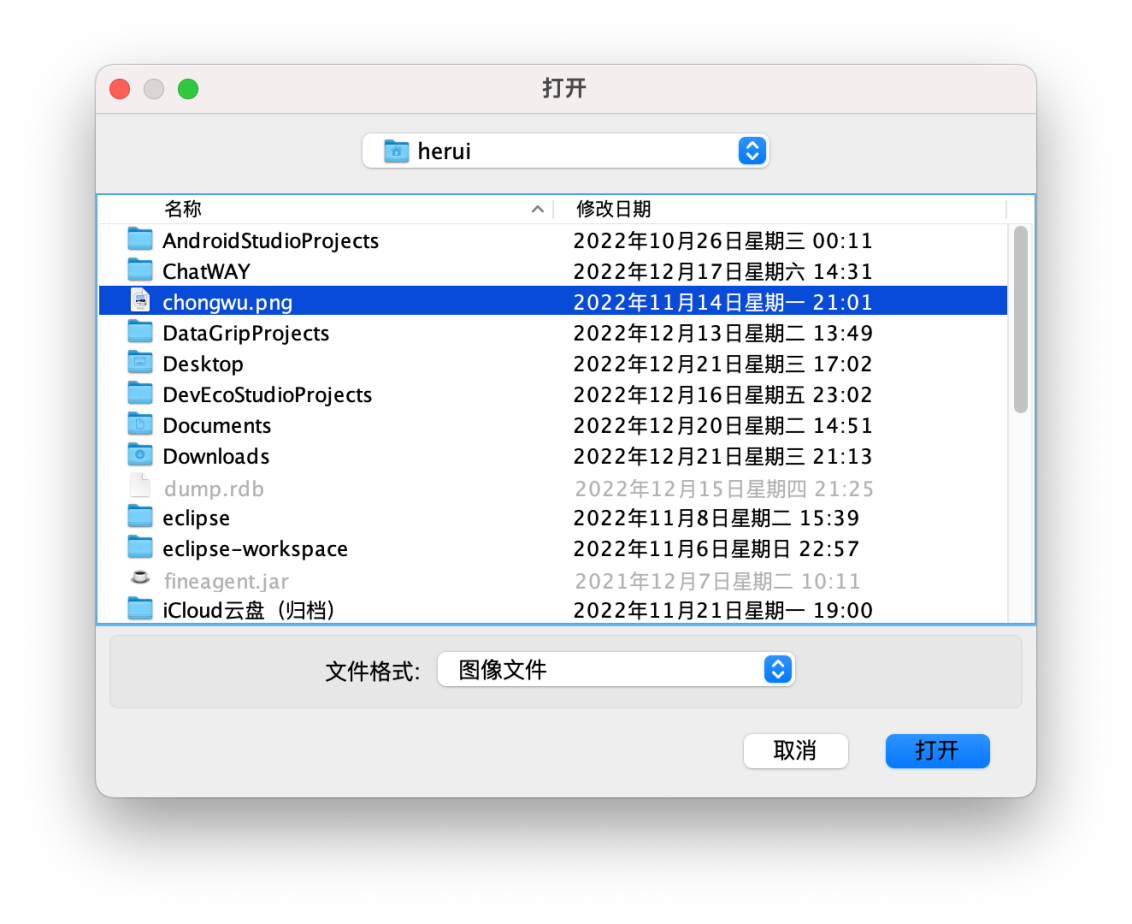


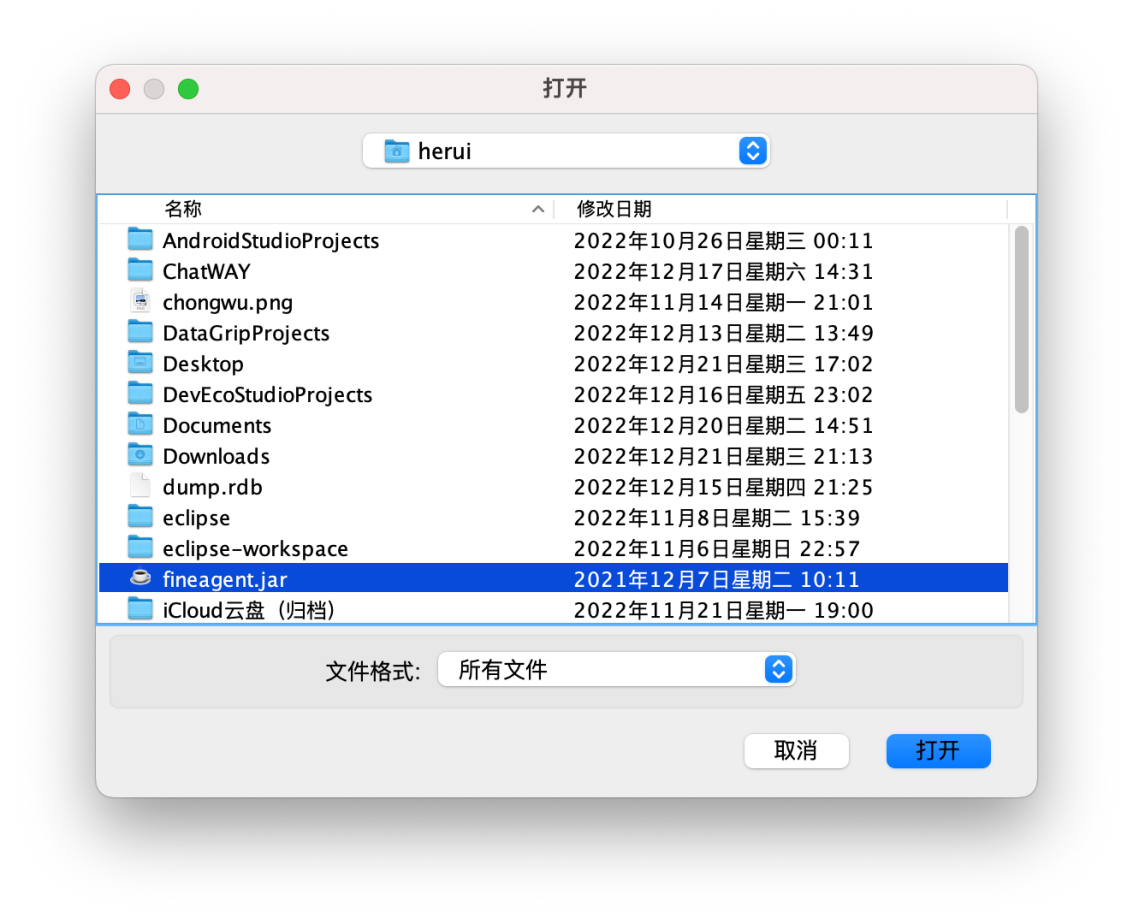
6.4 私聊功能实现效果



# 截屏2022-12-21 21.06.47

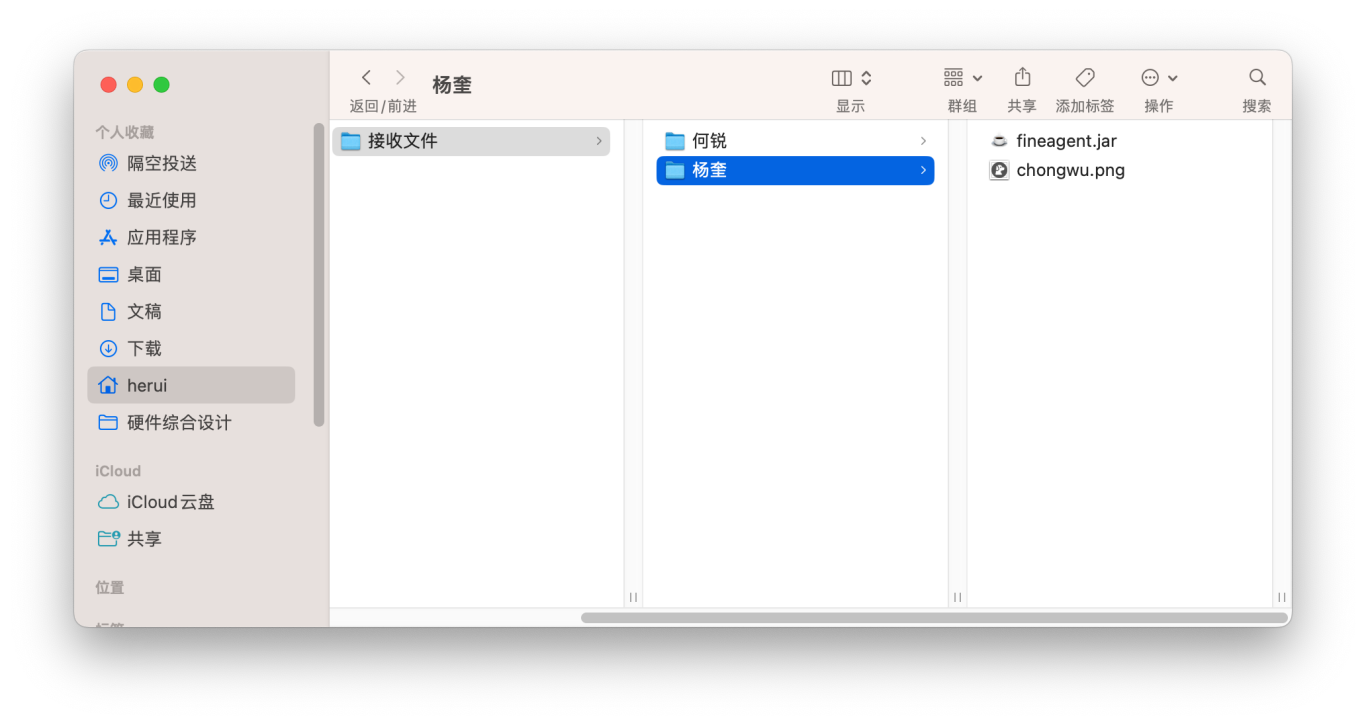
6.5 文件/图片发送功能实现效果



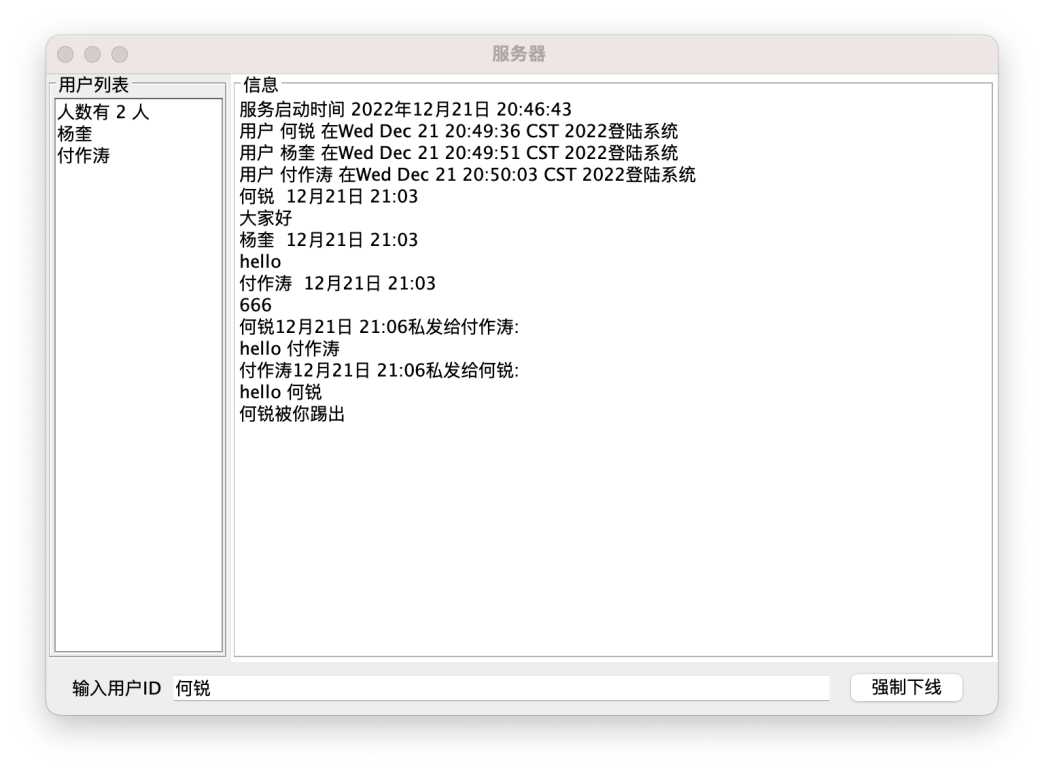








6.6 服务器端的强制下线功能



# 截屏2022-12-21 21.21.20



# **7 项目总结**

7.1 项目的收获

1. 学习到使用 Swing 工具包进行图形用户界面开发
2. 学习到使用 JAVA org.springframework.jdbc 包连接数据库
3. 学习到使用 JAVA SOCKET API 实现网络通信
4. 学习到服务器与多客户端通信的原理及具体实现方式
5. 学习到使用多线程、长连接技术保障通信的稳定和有序

7.2 项目存在的不足

1. 使用 Swing 包开发图形界面耗时耗力，下次可以采用其他框架或平台开发前端界面
2. 时间仓促，没有对系统进行非常完善的极端测试
3. 对通信规则的定义仍然不够细致、全面，但对实现本次项目已然足够