芜湖职业技术学院

毕业论文

**题 目:基于Android的即时通讯系统**

**院系名称: 信息工程学院**

**专业班级: 移动通信技术**

**学生姓名: 马山水**

**学 号: 1504020140**

**指导老师: 汤恒**

**2018年4月**

**摘要**

即时通信(IM)是指能够即时发送和接收互联网消息等的业务。自1998年面世以来，特别是近几年的迅速发展，即时通信的功能日益丰富，逐渐集成了电子邮件、博客、音乐、电视、游戏和搜索等多种功能。即时通讯软件功能的多样化带来了很多好处，但也有一些弊端，首先就是软件非常的臃肿，不管用户用得到的用不到的都堆积在软件上面，这会使软件运行速度降低，用户体验变差；其次大量的广告业务被放到app上，用户只能无条件的接收大量的广告推送；基于以上的原因，我决定做一款单纯实现聊天功能的软件，这款软件能够注册帐号，通过帐号添加好友，能够上传一部分基本信息保存在云端，聊天的内容包括图像，文字，数据，语音等方面，随时随地的与人交流。为了增加用户体验这里还添加了天气模块，毕竟这个模块是人人都需要的。

此系统手机端使用Android开发，后端服务器使用java web开发，在后端服务器的开发中使用了SSM（Spring+SpringMVC+MyBatis）框架，这使得前后端的业务实现分离，更加利于维护和扩展，也使得项目更加的简洁。数据库使用的是MySql，这种轻量级的数据库更符合我们的需求。

关键字:Android，Java，MySql，SSM，即时通讯，简洁

目录

[第一章 绪论 3](#_Toc4314)

[1.1 开发背景 3](#_Toc18802)

[1.2 开发意义 3](#_Toc28737)

[1.3 开发目标 3](#_Toc11772)

[1.4 系统描述 4](#_Toc12934)

[第二章 系统实现相关技术 4](#_Toc12073)

[2.1 开发框架技术 4](#_Toc4140)

[2.1.1 手机端 4](#_Toc18256)

[2.1.2 服务器端 5](#_Toc13960)

[2.2数据库技术 5](#_Toc360)

[2.2.1手机端 5](#_Toc5089)

[2.2.2 服务器端 6](#_Toc3352)

[2.3 开发环境 6](#_Toc12731)

[2.3.1 手机端 6](#_Toc27755)

[2.3.2 服务器端 6](#_Toc30272)

[第三章 需求分析 7](#_Toc20484)

[3.1 系统业务需求 7](#_Toc11809)

[3.2 系统用户需求 7](#_Toc7680)

[3.3 系统功能需求 7](#_Toc2671)

[3.3.1 用户用例 7](#_Toc16624)

[3.3.2 用例说明 8](#_Toc23090)

[3.3 时序图 10](#_Toc1931)

[第四章 系统分析与设计 13](#_Toc17871)

[4.1 系统功能的总体设计 13](#_Toc25355)

[4.2 负责模块设计 14](#_Toc17012)

[4.2.1 注册登录 14](#_Toc334)

[4.2.2 获取位置信息和天气 16](#_Toc26158)

[4.2.3 获取用户信息 16](#_Toc26936)

[4.2.4 添加好友 17](#_Toc18764)

[4.3数据库设计 18](#_Toc13031)

[4.3.1 手机端数据库设计 18](#_Toc18494)

[4.3.2 服务器端数据库设计 20](#_Toc692)

[第五章 系统的实现 22](#_Toc20756)

[5.1 系统模块介绍 22](#_Toc25882)

[5.2 模块的具体实现 23](#_Toc581)

[5.2.1 登录注册模块的实现 23](#_Toc23289)

[5.2.2 显示天气模块的实现 27](#_Toc19565)

[5.2.3 获取用户信息模块的实现 29](#_Toc8401)

[5.2.4 添加好友模块的实现 31](#_Toc5911)

[第六章 系统测试 34](#_Toc20707)

[6.1 测试目的 34](#_Toc10455)

[6.2 测试用例 34](#_Toc16225)

[总结 36](#_Toc21826)

[致谢 36](#_Toc10514)

[参考文献 37](#_Toc18303)

[附录 38](#_Toc20495)

# 绪论

## 1.1 开发背景

即时通信可以说是继电子邮件、WWW之后，互联网上最具杀伤力的应用。艾瑞咨询的一项调查显示，2003年中国即时通信软件的最高在线人数已达550万人，到2004年底有望突破740万人，2006年则将突破1200万人。相对于传统的电话、E-mail等通信方式来说，即时通信不仅节省费用，而且效率更高。例如企业的即时通信系统可以随时查看各部门在线人员、沟通各分支机构、即时传输文件、进行远程视频会议、群发手机短信等。因此，即时通信很有可能成为继安全软件、ERP软件后又一企业必备的工具，市场潜力不可忽视。

同样，在消费领域，越来越多的个人用户也在转向即时通信应用，有专家认为，从某种意义上说，经营即时通信软件的企业正在发展成一个虚拟的电信运营商。因此，即时通信已经成为现有电信运营商必须介入的领域。

## 1.2 开发意义

由于即时通信软件的兴起，能够进行即时互通的“内容”正迅速由语音全面扩展到图像、文字、数据等方面，即时通信不仅具有多功能，能够跨越互联网、手机、固定电话等多个平台进行通信才是即时通信未来的价值所在。即时通信正向更为广义的方向发展，未来的即时通信软件可以随时随地和任何人进行任何方式的沟通，不仅是语音，还包括图像、资料、数据等等，不仅在电脑上，还可以在手机、固定电话等任何终端上。

## 1.3 开发目标

通过观察以及网上调查发现。目前即时聊天软件当下已经是非常的多了。但是大多都充斥着垃圾广告以及各种无用的功能。不能提供简单快捷的服务。为此我们希望可以自己开发一个软件为我们自己提供方便快捷的服务。这个软件能够通过移动终端实现通信，包括图像，文字，数据，语音等方面，随时随地的与人交流。

## 1.4 系统描述

本文是在Android平台上开发即时手机通讯系，它能够使用户在手机上方便的、迅捷的收发及时消息、传输图片。此外，根据现今用户对于即时通讯软件的特殊要求，比如：在聊天的基础之上还要有一些增值服务功能，以满足不同用户的需求。例如，即时文本消息，图片的传输，语音聊天等，可以对好友传送过来的图片随时进行浏览，从而实现即时通讯系统的功能多样化。

本系统为客户端/服务器端的系统架构，服务器端由自身服务器和环信服务器组成，其中，自身服务器和环信服务器负责处理所有用户的请求、访问数据库和消息的传递；而客户端是基于Andrnid平台开发的，负责连接并登录服务器，通过服务器来实现客户端与客户端之间的即时通讯。

对于用户端，当用户输入用户名密码后点击登录，客户端向服务器端发起请求，验证账号信息后即可登录成功。对于登录成功的用户具有：浏览好友、收发信息，获取位置天气，添加好友等功能模块

# 

# 第二章 系统实现相关技术

本章节对即时通讯系统应用的技术做简单的介绍，该系统的相关技术包括两个方面：手机端和服务器端。本章主要针对这两个方面使用的框架、数据库等相关技术进行介绍。

## 2.1 开发框架技术

### 2.1.1 手机端

手机端并没有使用当下流行的MVC、MVP或者MVVM设计，但是其中使用了很多第三方的框架。网络请求方面使用了okhhtp，他的优点是通过连接池,减少了请求延迟, 减少了对数据流量的消耗。Android中的事件传递使用了EventBus，通过使用EventBus减少了系统大量的接口，使得代码更简洁,性能更好。除此之外，手机端的定位、查看天气、聊天等功能分别使用了百度地图SDK、和风天气SDK和环信SDK，他们的引入使该系统更加的好用，功能更加多样。

### 2.1.2 服务器端

服务器端使用的是SSM（Spring+SpringMVC+MyBatis）框架。其中spring是一个轻量级的控制反转（IoC）和面向切面（AOP）的容器框架。SpringMVC分离了控制器、模型对象、分派器以及处理程序对象的角色，这种分离让它们更容易进行定制。MyBatis是一个支持普通SQL查询，存储过程和高级映射的优秀持久层框架。使用SSM框架可以让项目的代码更加简洁、易于维护、更加的稳定，同时也更加易于测试和修改。

## 2.2数据库技术

### 2.2.1手机端

手机端使用的是Android系统自带的SQLite数据库，SQLite是一种嵌入式数据库，它的数据库就是一个文件。由于SQLite本身是C写的，而且体积很小，所以，经常被集成到各种应用程序中，甚至在iOS和Android的App中都可以集成。为了更加方便的使用SQLite，在系统还集成了LitePal，LitePal是一款开源的Android数据库框架，它对SQLite进一步封装，它采用了对象关系映射(ORM)的模式，并将我们平时开发时最常用到的一些数据库功能进行了封装，使得不用编写一行SQL语句就可以完成各种建表、増删改查的操作。相较于SQLite冗余的代码量和升级数据库时复杂的操作逻辑，LitePal显然更加的方便。

### 2.2.2 服务器端

服务器端采用的是MySQL数据库，MySQL是一种关系数据库管理系统，关系数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。

MySQL所使用的 SQL 语言是用于访问数据库的最常用标准化语言。MySQL 软件采用了双授权政策，分为社区版和商业版，由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，一般中小型网站的开发都选择 MySQL 作为网站数据库。

## 2.3 开发环境

### 2.3.1 手机端

手机端是在Android Studio环境下开发的，Android Studio 是一个Android集成开发工具，基于[IntelliJ IDEA](https://baike.baidu.com/item/IntelliJ%20IDEA" \t "_blank). 类似 [Eclipse](https://baike.baidu.com/item/Eclipse/61703" \t "_blank) [ADT](https://baike.baidu.com/item/ADT" \t "_blank)，Android Studio 提供了集成的 Android 开发工具用于开发和调试。

### 2.3.2 服务器端

服务器系统是在Eclipse环境下开发的web应用系统，服务器采用的是Apache Tomcat 9.0版本，数据库采用的是MySql 5.7。

Tomcat 服务器是一个免费的开放源代码的Web 应用服务器，属于轻量级应用[服务器](https://baike.baidu.com/item/%E6%9C%8D%E5%8A%A1%E5%99%A8" \t "_blank)，在中小型系统和并发访问用户不是很多的场合下被普遍使用，是开发和调试程序的首选。

注：服务器是部署在腾讯云服务器上，外网的IP是139.199.158.151

# 

# 第三章 需求分析

## 3.1 系统业务需求

随着科技的发展，电话和短信在现代社会已经不是那么主流了，网络通信渐渐取代他的位置成为使用最多通讯方式，相较于电话短信这样传统的通讯方式，网络通讯的使用成本更低，更加的便捷。所以，制作一款自己趁手的聊天工具就有了很大的应用前景。系统要求通过移动终端实现通信，包括图像，文字，数据，语音等方面，随时随地的与人交流。

## 3.2 系统用户需求

对于用户而言最大的需求就是实现两个人之间的通讯，所有的需求都围绕这个需求展开，具体可划分成如下需求：

* 注册：用户可以自行注册账号
* 登录：用户可以通过账号登录软件
* 发送文字：发送文字消息
* 发送语音：发送语音消息
* 发送图片：发送图片消息
* 添加好友：用户添加好友
* 退出登录：用户退出登录

## 3.3 系统功能需求

### 3.3.1 用户用例

用例模块如图3-1所示，从图中可以看出用户具有四个用例模块，他们分别是：注册功能，登录功能，添加好友功能，查看好友功能，发消息功能，收消息功能，退出登录功能。

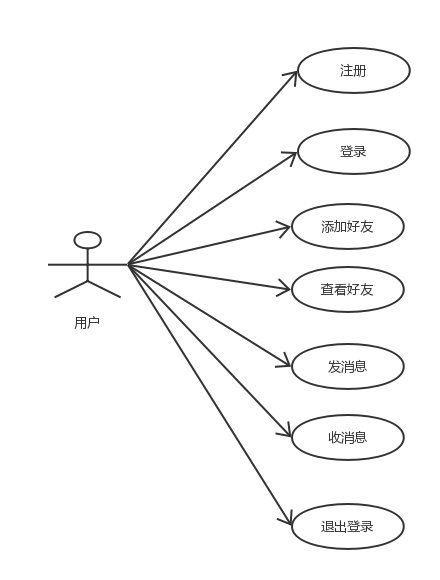


图3-1 用户用例图

### 3.3.2 用例说明

用户注册用例说明如表3-2所示

表3-2 用户注册用例说明

|  |  |
| --- | --- |
| 用例名称 | 用户注册 |
| 标识符  用例描述  参与者  前置条件  后置条件  基本操作流程  可选操作流程 | UC001  注册成为用户  用户  无  注册成为用户   1. 点击注册进入注册页面 2. 输入相关信息 3. 提交信息到数据库中   如果输入的用户名有重名，则返回重新注册，或取消，终止用例 |

用户登录用例说明如表3-3所示

表3-3 用户登录用例说明

|  |  |
| --- | --- |
| 用例名称 | 登录 |
| 标识符  用例描述  参与者  前置条件  后置条件  基本操作流程  可选操作流程 | UC002  登录系统  用户  登录到相应系统  无   1. 在登录界面输入用户名和密码 2. 系统验证用户名和密码，若正确，则登录到相应系统中   如果用户输入无效的用户名和密码，系统显示错误信息，并返回重新提示用户输入用户名和密码：或者取消登录或者终止登录。 |

用户添加好友用例说明如表3-4所示

表3-4 用户添加好友用例说明

|  |  |
| --- | --- |
| 用例名称 | 添加好友 |
| 标识符  用例描述  参与者  前置条件  后置条件  基本操作流程  可选操作流程 | UC003  添加好友  用户  登录到相应系统  无   1. 在添加页面输入好友账号 2. 在填写备注页面填写备注 3. 将请求数据提交到服务器 4. 添加成功等待对方验证   如果好友账号不存在或者已经是好友，系统会提示相应的错误信息。 |

### 3.3 时序图

登录：如图3-5所示，用户在登录页面输入账号和密码，信息会发送到服务器验证账号密码的正确性，如果成功则跳转到主页面，失败则提示用户错误原因。

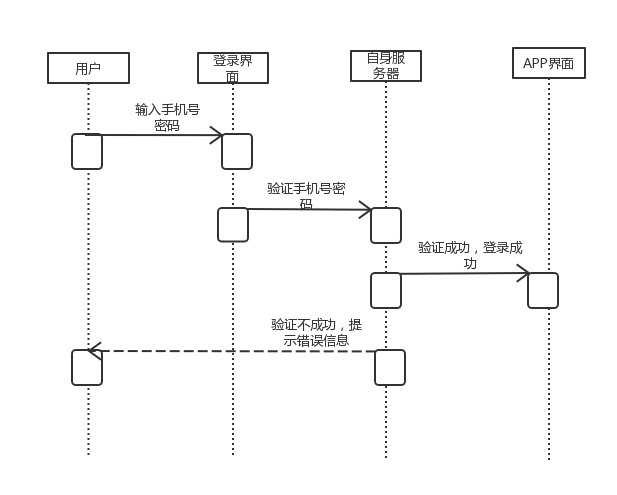


图3-5 登录时序图

注册：如图3-6所示，用户输入手机号作为注册的账号，将账号发送到服务器，服务器向网易云服务器请求发送验证码，验证码返回到手机端，用户输入验证码发送到服务器，服务器向网易云服务器发送请求验证验证码的正确性，验证成功后返回到手机端，手机端跳转到填写用户名密码和上传头像的页面，填写完成后存到服务器数据库，返回用户注册成功。

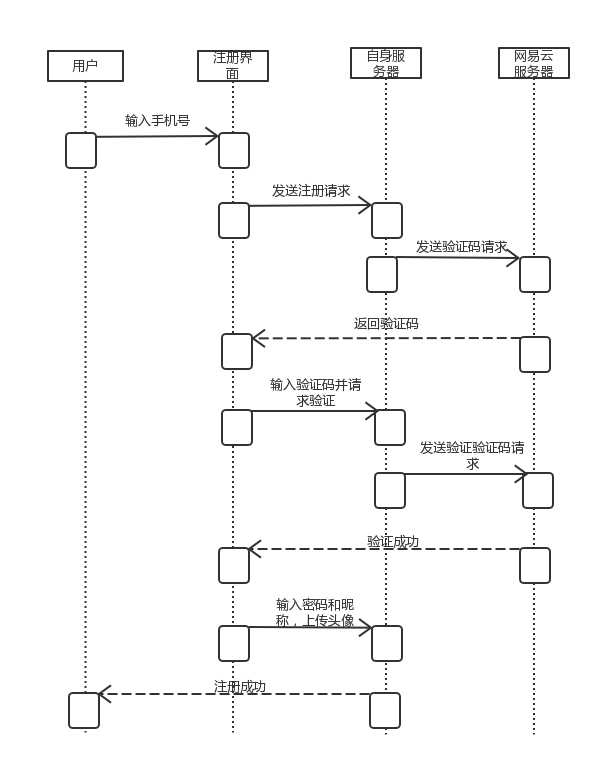


图3-6 注册时序图

添加好友：如图3-7所示，用户输入好友账号选择添加，好友账号发送到服务器，服务器判断账号是否可以添加，不能添加则返回错误原因，可以添加则提示用户输入备注信息，将备注信息发送到服务器，服务器返回添加成功等待好友验证的结果；好友收到请求选择同意或者不同意，同意后添加好友成功，用户收到成功通知，不同意后添加好友失败，用户收到失败通知。

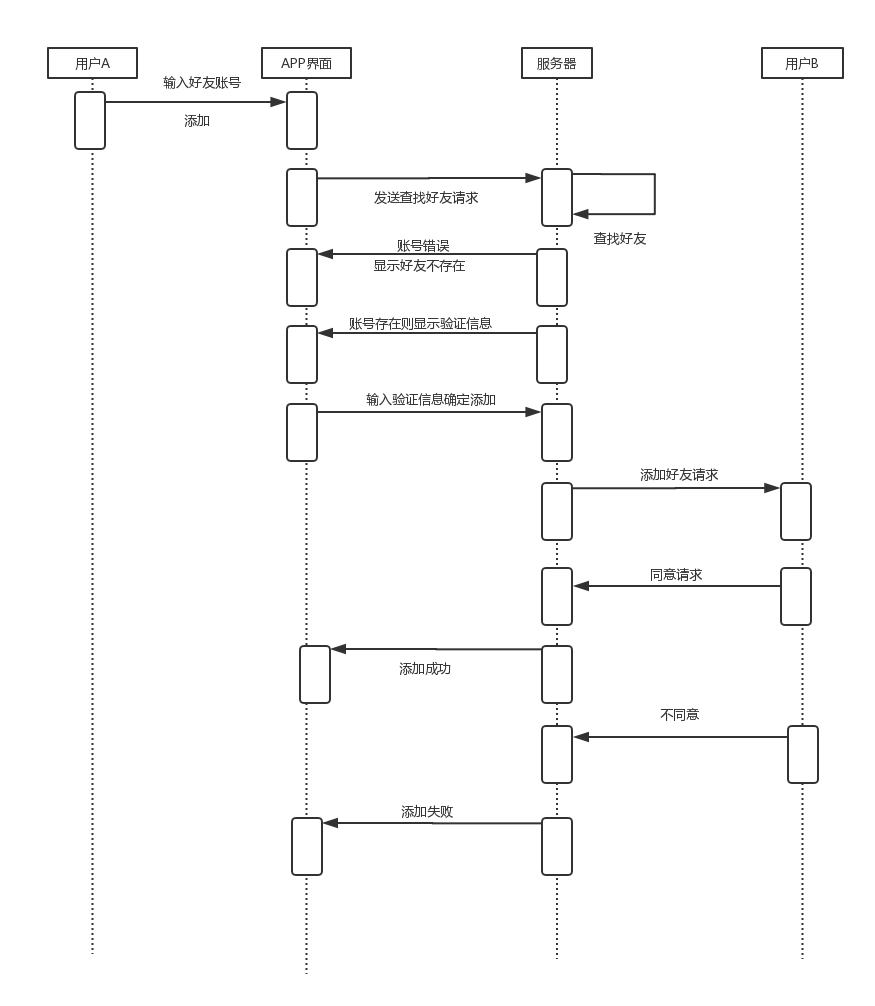


图3-6 添加好友时序图

# 

# 第四章 系统模块分析与设计

## **4.1 系统功能的总体设计**

本系统是一个即时通讯系统，既然是通讯必然有用户管理模块，其中应该包括用户的注册、登录，以及用户个人信息的管理；然后用户间想要进行通信必须要知道对方是谁，因此这里使用注册时的帐号作为用户的唯一标识，这样就可以进行添加好友的操作了；知道了对方是谁，接下来就可以的发送消息，发送消息时需要找到联系人，因此需要联系人模块供用户查找联系人列表；我们向对方发送消息之后需要查看消息，并且接收到消息后也需要查看消息内容，这里就需要一个查看消息的模块供用户查看和接收消息；到这里即时通讯系统的总体设计就完成了。

系统功能总体设计图如下：

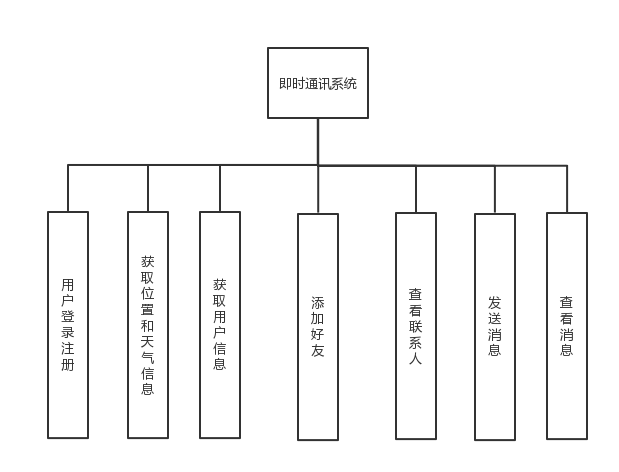


图4-1 系统功能总体设计

## 4.2 负责模块设计

### 4.2.1 注册登录

注册登录模块是一个系统最基础的模块，是整个系统的入口，这部分的设计虽然简单，但是对系统来说至关重要。

首先从注册讲起，注册最重要的就是一个帐号，并且这个帐号要是唯一的，能够唯一标识一个用户，就拿QQ来说吧，每个人申请的帐号都是不同的，QQ的帐号是后台来生成分配和管理的，为了系统的简易我们这里使用手机号作为用户的帐号，用手机号做帐号的好处就是首先他是唯一的，其次他可以直接获取到用户的手机号，方便接下来的短信验证码验证。注册时首先让用户输入手机号，接下来发送验证码，用户收到并填写验证码之后开始填写昵称和密码，为了密码的准确性这里让用户输入两遍，再接着用户上传头像，为了方便用户选取这里跳转到用户的相册选取，用户的图片一般会比较大，但是显示头像时并不需要很高的分辨率，而且图片过大会在上传时浪费更多的时间和流量，也会加大服务器的负载，所以这里我们对图片进行一个剪切和压缩，到这里注册基本完成。

再来说说登录，相较于注册，登录就简单多了，首先在登录页面输入帐号，这里为了用户体验在输入完成帐号之后自动加载用户的头像显示在界面上，这样用户就不会输错帐号，或者在输错帐号后不知情，然后输入密码登录就完成了。

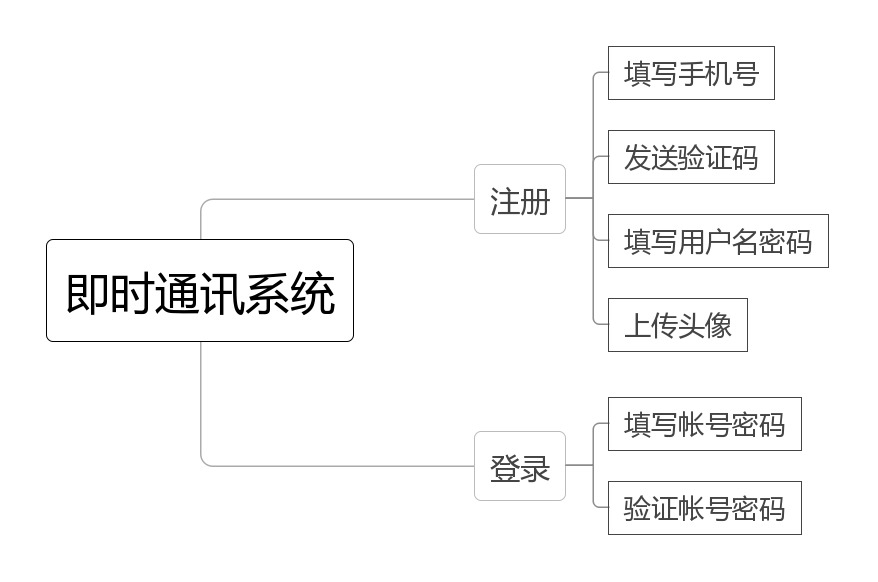


图4-2-1 登录注册

### 4.2.2 获取位置信息和天气

获取位置天气是系统中的一个辅助模块，虽然只是一个简单的功能但实现起来还是很复杂的。

首先要确定获取那里的天气信息，如果只能获取固定位置的天气信息那就没有什么意义了，所以这里使用到了定位，本系统选择的是“百度地图SDK”作为数据源。位置信息获取到后我们还需要一个获取天气的接口，通过网上的搜索，我们决定使用“和风天气”作为数据源。

两个数据源的问题解决后我们看一下具体流程，首先通过百度地图获取到当前设备位置的经纬度信息，接着将经纬度信息传递给天气模块，天气模块通过接口将位置信息发送到“和风天气”的后台，后台返回给我们当前位置的天气信息，我们把天气信息展示在侧滑抽屉的界面上。

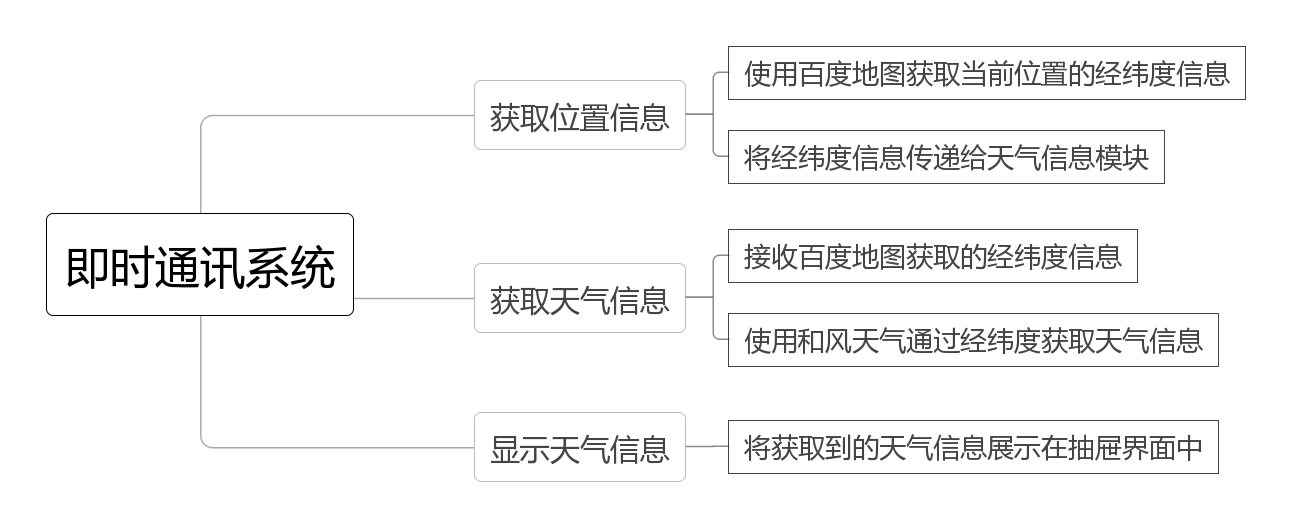


图4-2-2 获取位置和天气信息

### 4.2.3 获取用户信息

获取用户信息是系统中非常关键的模块，因为很多模块都要依赖他才可以正常工作。

在登录的时候，在我们输入手机号时系统会自动获取用户头像信息展示在界面上，在登录成功之后首先进入消息页面（主页面），包括发送消息时页面展示的信息都是通过这个模块获取，当我们滑出抽屉时，抽屉里会展示当前用户的昵称和头像，联系人页面展示好友的昵称和头像。

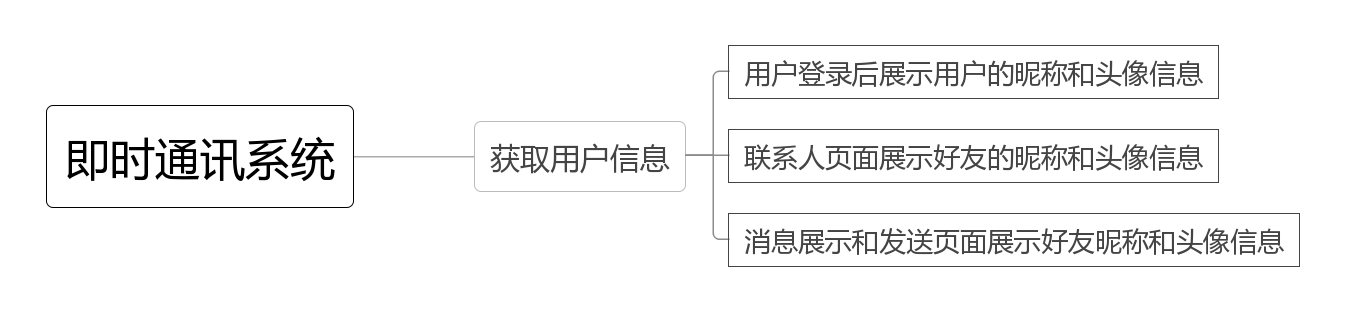


图4-2-3 获取用户信息

### 4.2.4 添加好友

添加好友是通讯的开端，这部分的设计需要处理很多问题，比如帐号的验证、好友收到请求后的展示，好友同意请求后的回复。下面我们具体设计一下这个模块。

首先我们要知道和输入好友的帐号（因为之前注册的时候帐号是唯一的，所以这里直接使用帐号作为添加好友的依据），接着发送到后台，后台会验证这个帐号是否存在，是否已经是好友，然后我们会收到一个结果，如果帐号没有问题的话会进入下个页面输入添加理由，这个理由会在好友查看时展示，接着确定就发送成功了。

发送之后如果我们不记得了，想要查看向谁发送了请求，请求有没有被同意，以及用户收到好友请求时查看谁添加自己，同意还是拒绝等等这些操作都需要一个展示页面，我们把这个页面放在联系人页面的头部方便用户查看，这样添加好友的设计就完成了。

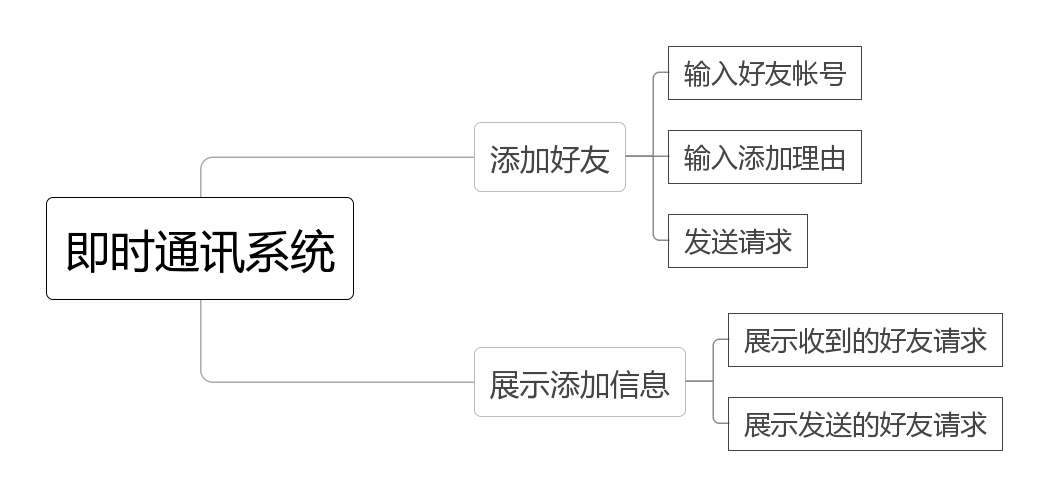


图4-2-4 添加好友

## 4.3数据库设计

### 4.3.1 手机端数据库设计

1.数据库分析

根据系统需求，对应数据表的设计及功能如下。

（1）用户表：存放用户注册时的个人信息

（2）用户好友表：存放用户添加的好友信息

（3）好友请求表：存放用户发送以及接收的好友请求信息

（4）消息表：存放用户发送和接收的消息

（5）天气表：存放获取的天气信息

2.数据字典

1）用户表db\_user的说明如表4-1所示：

表4-1 用户表（db\_user）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 字段长度 | 字段说明 |
| ID（主关键字） | Int |  | 自动编号 |
| account | Varchar | 15 | 账户 |
| username | Varchar | 15 | 用户名 |
| iconURL | Varchar | 50 | 头像地址 |

1. 用户好友表db\_user\_list的说明如表4-2所示：

表4-2 用户好友表（db\_user\_list）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 字段长度 | 字段说明 |
| ID（主关键字） | Int |  | 自动编号 |
| friend\_account | Varchar | 15 | 好友账户 |
| username | Varchar | 15 | 好友用户名 |
| iconURL | Varchar | 50 | 好友头像地址 |

1. 好友请求表db\_invite\_message的说明如表4-3所示：

表4-3 好友请求表（db\_invite\_message）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 字段长度 | 字段说明 |
| ID（主关键字） | Int |  | 自动编号 |
| status | Int |  | 添加状态 |
| form | Varchar | 15 | 添加人 |
| time | Long | 30 | 添加时间 |
| reason | Varchar | 50 | 添加理由 |

1. 消息表db\_msg\_list的说明如表4-4所示：

表4-4 消息表（db\_msg\_list）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 字段长度 | 字段说明 |
| ID（主关键字） | Int |  | 自动编号 |
| Msgtype | Int |  | 消息类型 |
| Content | Varchar | 200 | 消息体 |
| From | Varchar | 30 | 发送人 |

1. 天气表db\_weather的说明如表4-5所示：

表4-5 天气表（db\_weather）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 字段长度 | 字段说明 |
| ID（主关键字） | Int |  | 自动编号 |
| account | Varchar | 15 | 账户 |
| tmp | Varchar | 15 | 天气温度 |
| txt | Varchar | 30 | 天气状况 |
| city | Varchar | 15 | 城市 |
| dir | Varchar | 15 | 风向 |
| sc | Varchar | 15 | 风力 |

1. 手机端数据库E-R图

如图4-6所示，描述了手机端的数据库E-R图，这张图展示了手机端数据库之间的联系，让数据库之间的关系更加的清晰。

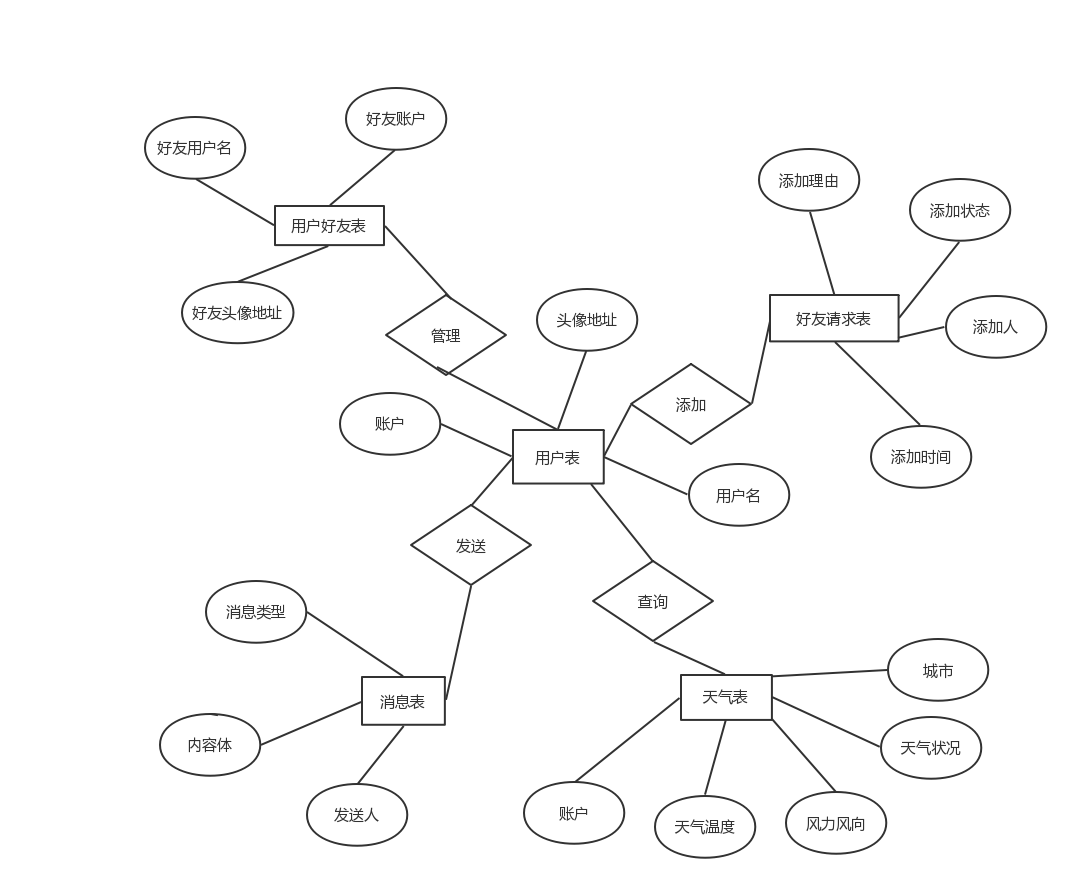


图4-6 数据库E-R图

### 4.3.2 服务器端数据库设计

1.数据库分析

根据系统需求，对应数据表的设计及功能如下。

（1）用户表：存放用户注册时的个人信息

（2）用户好友表：存放用户添加的好友信息

2.数据字典

1）用户表qg\_user的说明如表4-1所示：

表4-1 用户表（qg\_user）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 字段长度 | 字段说明 |
| ID（主关键字） | Int |  | 自动编号 |
| account | Varchar | 15 | 账户 |
| username | Varchar | 15 | 用户名 |
| iconURL | Varchar | 50 | 头像地址 |
| password | Varchar | 50 | 密码 |
| hx\_account | Varchar | 15 | 环信帐号 |

2）用户好友表qg\_user\_friend\_list的说明如表4-2所示：

表4-2 用户好友表（qg\_user\_friend\_list）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 字段长度 | 字段说明 |
| ID（主关键字） | Int |  | 自动编号 |
| friend\_account | Varchar | 15 | 好友账户 |
| account | Varchar | 15 | 好友账户 |

1. 服务器端数据库E-R图

如图4-3所示，描述了服务器端的数据库E-R图，这张图展示了服务器端数据库之间的联系，让数据库之间的关系更加的清晰。

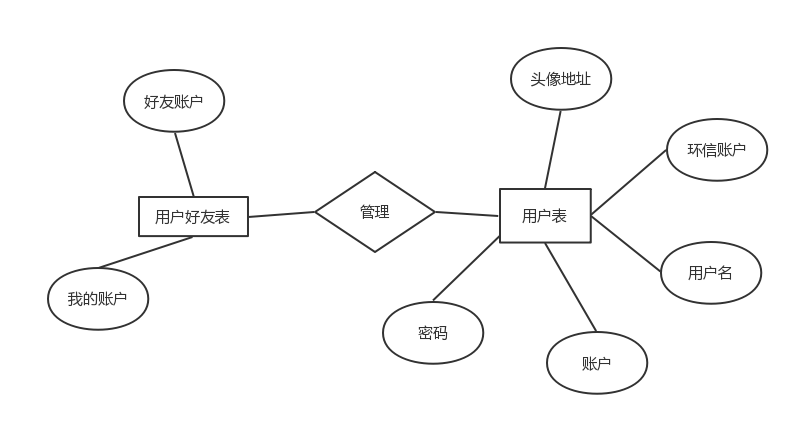


图4-3 数据库E-R图

# 系统的实现

## 5.1 系统模块介绍



图5-1 系统总模块

如图所示，本系统分为：登录注册、显示天气、获取用户信息、添加好友、查看联系人、发送消息和查看消息共七个模块。

其中我负责登录注册、显示天气、获取用户信息和添加好友四个模块，首先登录注册是整个系统的开端和入口，它的实现关系到系统后续的稳定性，一切工作都是从它展开；其次显示天气信息是系统的一个重要扩展功能，它包含了获取位置信息的内容，位置信息是使用非常广泛和普遍的信息，这对以后系统的扩展提供了基石；获取用户信息模块是一个相对于用户很不起眼的一个模块，用户感知不到它的存在但却时刻都在使用它，任何需要展示用户信息的页面都依赖了它，比如显示消息时的发送人信息、登录时的用户自身的信息、联系人页面的好友信息等等；最后添加好友这个模块是系统核心功能（通讯）的开端，只有添加了好友才能将信息准确的发送出去，好友才能接收到用户发送的消息。

## 5.2 模块的具体实现

### 5.2.1 登录注册模块的实现

登录注册的具体流程之前已经说过，下面详细介绍他的每个流程的具体实现。

注册：首先输入手机号（帐号），点击之后会将这个手机号发送到后台服务器，服务器会判断这个帐号是否已经存在，如果不存在后台会向网易云信后台发送一个发送验证码的请求，并返回给手机端一个可以创建的返回值，手机端收到这个返回值之后进行下一步，这里是输入手机号收到的验证码的页面，输入收到验证码点击确定会向服务器发送一条验证验证码是否正确的请求，后台收到这条请求后会向网易云信发送请求验证验证码的正确性，在接收到请求后将结果返回给手机端，如果手机端收到验证码正确的返回值会进入填写昵称和密码的页面，这里会直接将用户输入的值发送给服务器，服务器直接将它插入到数据库，并将操作的结果发送给手机端，这时手机端进入注册的最后的一个步骤上传头像，这个页面有打开相机和相册两个选择供用户选择，用户可以使用相机拍照或相册选择的照片作为头像，用户选择之后会跳到图片剪切页面（为了系统的统一，系统使用的都是正方形的照片），图片剪切之后会直接对图片进行压缩并上传到服务器，服务器会接收上传的图片并保存到硬盘中，在保存成功之后生成一个图片路径（外网可以访问）保存到数据库中，到这里注册的全部实现就完成了。

登录：登录的流程相对而言比较简单，首先输入帐号和密码，点击登录按钮之后会将信息发送到后台，后台验证信息的正确性之后返回给手机端结果，手机端判断后决定登录成功还是失败。

系统注册图见附录 图1-1

系统登录图见附录 图1-2

关键代码：

private void login() {

account = edtAccount.getText().toString().trim();

password = edtPassword.getText().toString().trim();

if (TextUtils.isEmpty(account)) {

edtAccount.setShakeAnimation();

setToast("请输入账号");

return;

}

if (TextUtils.isEmpty(password)) {

edtPassword.setShakeAnimation();

setToast("请输入密码");

return;

}

showBufferDialog();

EMClient.getInstance().login(account, password, new EMCallBack() {

@Override

public void onSuccess() {

EMClient.getInstance().chatManager().loadAllConversations();

EMClient.getInstance().groupManager().loadAllGroups();

handler.sendEmptyMessage(LOGIN\_SUCCESS);

}

@Override

public void onProgress(int progress, String status) {

}

@Override

public void onError(int code, String error) {

handler.sendEmptyMessage(LOGIN\_FAIL);

}

});

}

private void start() {

/\*\*

\* 将是否登录设置为true

\* 将帐号密码保存在本地

\*/

dismissBufferDialog();

SharedPreferences.Editor editor = preferences.edit();

editor.putString("account", account);

editor.putString("password", password);

editor.apply();

Intent intent = new Intent(AtyLogin.this, AtyMain.class);

intent.addFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK |

Intent.FLAG\_ACTIVITY\_CLEAR\_TASK);

startActivity(intent);

}

private void uploadIcon() {

String filePath = imageUri.getEncodedPath();

String imagePath = Uri.decode(filePath);

Map<String, String> param = new HashMap<>();

String filename = ((AtyRegister) getActivity()).phoneNumber;

param.put("account", filename);

((AtyRegister) getActivity()).showBufferDialog();

HttpUtil.uploadImage(HttpUtil.uploadImageURL, param, imagePath, filename, new Callback() {

@Override

public void onFailure(Call call, IOException e) {

Message message = new Message();

message.what = REQUEST\_FAIL;

handler.sendMessage(message);

}

@Override

public void onResponse(Call call, final Response response) throws IOException {

Message message = new Message();

String responseData=response.body().string();

Gson gson=new Gson();

StatusResponse statusResponse=gson.fromJson(responseData,StatusResponse.class);

if (statusResponse.getCode().equals("200")) {

message.what = REQUEST\_SUCCESS;

handler.sendMessage(message);

} else if (statusResponse.getCode().equals("400")) {

message.what = REQUEST\_FAIL;

handler.sendMessage(message);

}

}

});

}

### 5.2.2 显示天气模块的实现

显示天气模块的功能虽然简单但是实现起来还是比较复杂的，因为里面牵扯到了百度地图和离线加载。

首先我们把它的界面放在抽屉里，每当用户打开抽屉时去加载天气数据，接着在系统中集成百度地图SDK，获取到用户位置的经纬度信息之后保存到内存中，当用户打开抽屉时会自动获取位置信息并通过和风天气的API接口将位置信息发送到和风天气的后台，它的后台会将该位置的天气信息以json字符串的形式返回给我们，我们把数据解析出来加载到界面上并保存到数据库中，这样当用户处在没有网络环境下也能显示之前加载的天气信息，这样天气模块的实现就完成了。

天气显示图见附录 图1-3

关键代码：

/\*\*

\* 百度地图定位监听

\*/

public class MyLocationListener extends BDAbstractLocationListener {

@Override

public void onReceiveLocation(BDLocation bdLocation) {

lon = bdLocation.getLongitude(); //获取经度信息

lat = bdLocation.getLatitude(); //获取纬度信息

}

}

/\*\*

\* 加载天气信息

\*/

private void loadWeather() {

if (AccessNetwork.getNetworkState(AtyMain.this) != AccessNetwork.INTERNET\_NONE && lat != 0 && lon != 0) {

String url = HttpUtil.getWeatherURL.replace("CITY", lon + "," + lat);

HttpUtil.sendOkHttpRequest(url, new Callback() {

@Override

public void onFailure(Call call, IOException e) {

}

@Override

public void onResponse(Call call, Response response) throws IOException {

String result = response.body().string();

Weather weather = BeanUtil.handleWeatherResponse(result);

String account = preferencesUtil.getAccount();

DBUtil.saveWeather(account, weather);

runOnUiThread(new Runnable() {

@Override

public void run() {

updateWeather();

}

});

}

});

} else {

updateWeather();

}

}

private void updateWeather() {

DBWeather weather = DataSupport.findFirst(DBWeather.class);

if (weather != null) {

weatherArea.setText(weather.getCity());

weatherState.setText(weather.getTxt());

weatherTemp.setText(weather.getTmp() + "°");

weatherQuality.setText(weather.getDir() + weather.getSc());

}

}

### 5.2.3 获取用户信息模块的实现

获取用户信息模块是本系统中依赖最大的一个模块，几乎所有的模块都依赖它的实现。

首先要获取到用户的帐号信息，因为获取信息是通过帐号来获取的，所以需要使用这个模块的时候地方一定要能够获取到帐号信息。有了帐号信息之后就可以通过将帐号发送到后台获取后台返回的用户详细数据。虽然实现起来很简单但却是系统中至关重要的一环。

系统获取用户信息图见附录 图1-3

关键代码：

SharedPreferences preferences = getSharedPreferences("qgchat", MODE\_PRIVATE);

account = preferences.getString("account", "");

if (AccessNetwork.getNetworkState(AtyMain.this) != AccessNetwork.INTERNET\_NONE) {

String url = HttpUtil.getUserMessageURL + "?account=" + account;

HttpUtil.sendOkHttpRequest(url, new Callback() {

@Override

public void onFailure(Call call, IOException e) {

}

@Override

public void onResponse(Call call, Response response) throws IOException {

final String result = response.body().string();

if (!result.equals("failed")) {

runOnUiThread(new Runnable() {

@Override

public void run() {

UserBean bean = BeanUtil.handleUserBeanResponse(result);

DBUtil.saveUser(bean);

setDrawerHander(bean.getIconURL(), bean.getUsername());

}

});

}

}

});

} else {

DBUser bean = DataSupport.findFirst(DBUser.class);

setDrawerHander(bean.getIconURL(), bean.getUsername());

}

### 5.2.4 添加好友模块的实现

添加好友模块又分为两个子模块：添加和显示。这也不难理解，我们发送了添加好友的请求后需要一个展示页面显示我添加了谁、谁添加了我，否则只发送请求别人却不知道就没有什么意义了。

添加的实现并不复杂，首先输入要添加好友的帐号，然后发送到后台服务器，后台会首先判断这个帐号是否存在、是否已经是好友，如果可以添加后台会将信息发送到环信的服务器并返回给手机端结果。

显示的实现需要数据库的支持，因为我们要保存历史信息，不能收到好友请求后没有查看，再次登录这条数据就消失了，所以我们在收到好友请求时将数据保存到数据库，每次打开页面时都从数据库中加载并显示出来。

系统添加好友图见附录 图1-4、1-5

关键代码：

/\*\*

\* 判断用户是否可以添加

\*/

private void checkFriend() {

String url = HttpUtil.checkFriendURL + "?ownerAccount=" + ownerAccount + "&friendAccount=" + friendAccount;

HttpUtil.sendOkHttpRequest(url, new Callback() {

@Override

public void onFailure(Call call, IOException e) {

Message message = new Message();

message.what = INTERNET\_FAIL;

handler.sendMessage(message);

}

@Override

public void onResponse(Call call, Response response) throws IOException {

Message message = new Message();

String responseData = response.body().string();

Gson gson = new Gson();

StatusResponse statusResponse = gson.fromJson(responseData, StatusResponse.class);

Log.e(TAG, "onResponse: " + statusResponse.toString());

if (statusResponse.getStatus().equals("ok")) {

message.what = REQUEST\_SUCCESS;

} else if (statusResponse.getStatus().equals("fail")) {

Log.e(TAG, "onResponse: " + statusResponse.getDescription());

message.what = REQUEST\_FAIL;

message.obj = statusResponse.getDescription();

}

handler.sendMessage(message);

}

});

}

private void showDialog() {

AlertDialog.Builder dialogBuilder = new AlertDialog.Builder(AtyAddFriend.this);

dialogBuilder.setMessage("确认添加");

dialogBuilder.setPositiveButton("确定", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

showBufferDialog();

new Thread(new Runnable() {

@Override

public void run() {

try {

EMClient.getInstance().contactManager().addContact(friendAccount, message);

runOnUiThread(new Runnable() {

@Override

public void run() {

dismissBufferDialog();

setToast("等待对方验证");

finish();

}

});

} catch (final HyphenateException e) {

runOnUiThread(new Runnable() {

@Override

public void run() {

dismissBufferDialog();

setToast(e.getMessage());

}

});

}

}

}).start();

}

});

dialogBuilder.setCancelable(true);

AlertDialog dialog = dialogBuilder.create();

dialog.show();

}

# 系统测试

## 6.1 测试目的

软件测试的目的是以较少的人力、物力、时间找出系统中潜在存在的各种缺陷，通过减少Bug和改正缺陷，从而提高软件的健壮性，尽快尽早的发现系统缺陷，使得开发工作进展和尽可能的满足用户的需求。

系统测试的原则主要如下：

1.系统测试是找Bug，不是验证系统的正确性。

2.尽早不断地进行测试。

3.重视无效并且和预期结果不符的数据进行测试。

4.程序员应该尽量不要检查自己的系统程序，以免影响测试效果。

5.应该充分注意测试中的集群现象。

6.测试用例要不断修改、不断完善。

7.考虑经济损失。

## 6.2 测试用例

由于该系统功能模块较多，写起来测试篇幅较长，本文只介绍用户注册登录模块测试用例，具体如下表6-2所示：

表6-3用户注册登录模块

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 功能特性 | 用户注册登录模块 | | | | |
| 测试目的 | 验证用户是否能成功完成注册并成功登录 | | | | |
| 测试数据 | 管理员用户名：18297707817,17754015630 密码：12345678，abc12345678 | | | | |
| 测试内容 | 操作描述 | 数据 | 期望结果 | 实际结果 | 测试状态 |
| 情况1 | 输入用户名（手机号）为空，点击注册。 | 用户名：空。 | 界面无反应无法跳转 | 界面无反应无法跳转 | 与期望结果相同 |
| 情况2 | 输入用户名点击注册。 | 用户名：17754015630 | 转跳界面获取短信验证：已注册无短信接受！ | 转跳界面无获取的短信验证码短信。 | 与期望结果相同 |
| 情况3 | 输入未注册用户信息 | 用户名：18297707817 | 跳转界面获取短信验证，接受并填写成功 | 正确接受验证码并成功填写 | 与期望结果相同 |
| 情况4 | 输入要设定的昵称和密码 | 用户名：啊嘿嘿，密码：12345678 | 用户输入昵称和设定的密码成功跳转界面 | 正确设定信息跳转界面 | 与期望结果相同 |
| 情况5 | 输入用户名或者密码为空，点击登录。 | 用户名：空，或密码：空 | 转跳界面提示：用户名和密码不能为空。 | 转跳，提示“用户名和密码不能为空” | 与期望结果相同 |
| 情况6 | 输入用户名和密码，点击登录 | 用户名：17754015630密码：abc12345678 | 转跳界面提示：密码错误，请重新登录！ | 转跳，提示“密码错误，请重新登录” | 与期望结果相同 |

# 总结

本文通过以上章节的介绍，描述了“基于Android的即时通讯系统”的全过程。每一个章节都是相互独立却又相互依存的，每一个章节也是实现该系统必不可少的环节，至于实现本系统的重点以及难点，为了方便用户理解，都有图表说明，从本文的结构上分析，在章节上都有循序渐进的变化，从系统简单入门到系统的深入理解，阐述的比较透彻。

在去年暑假的时候我打算做一个系统的全面的应用来将我大学所学的知识集中展现出来，同时也是为了巩固所学的知识，再经过一段时间的考量之后我决定做一个基于Android的即时通讯系统，这个系统会用到Socket，多线程，服务器，Android，数据库等方面的知识，能够对我进行全面的锻炼，经过一个暑假之后这个系统基本成型，能够在两个手机端运行并进行通讯，当我想进一步扩展的时候发现一个重大的问题，那就是在做这个系统之前我并没有对Socket进行深入的研究，而且也没有使用成熟的Socket框架，导致系统极其不稳定和难以扩展，于是我放弃了自己实现Socket转而使用市场上成熟的即时通讯框架，这样做的好处是省去了Socket的实现，并且极其稳定，之后再对这个系统进行了完善。这个系统可以说将我大学所学完全体现了出来，再实现这个系统的过程中我收获到很多书上没有的东西，这可能也就是俗称的经验吧，当然过程中也有艰辛，每当出现一个bug但却无从下手的时候是最煎熬的，好在最后都顺利解决了。当然这个系统也有很多遗憾，比如Android端没有使用MVP框架开发，用户不能修改信息，在以后时间和条件允许的情况下我会将这些功能完善。

# 致谢

毕业设计之所以顺利完成，受益于给予我帮助的老师、室友和组员们，尤其是我的指导老师汤恒老师。

汤老师给予我技术方面非常多的指导，在校期间的教学内容，让我技术方面有足够的基础作为铺垫，也在写设计的过程中咨询过老师相关的技术实现上的难题，得到了老师的指导，为后续的设计过程提供了最直接的帮助。

感谢室友对我的帮助，在多方面的协助探讨，以及系统编写过程中的一些建议等等。

组员们给予的支持，让我在设计的过程中能够静下心来，一些资料的整合和设计思路都得益于两个组员伙伴。使我在设计过程中能够认真的静下心来面对各种技术上的难点。

除此之外，还要感谢信息工程学院所有老师、帮助过我的同学以及在背后默默支持我的父母，因为你们，毕业设计才能顺利完成

# 

# 参考文献

[1] 郭霖. Android第一行代码[M]. 人民邮电出版社,2016.11

[2] 任玉刚.Android开发艺术探索[M]. 北京.电子工业出版社,2015.9

[3] 李刚. 疯狂Ajax讲义[M].北京. 电子工业出版社,2013

[4] 徐宜生. Android群英传[M]. 电子工业出版社,2015.9

[5] 何红辉、关爱民. Android 设计模式源码分析[M]. 人民邮电出版社,2015.11

[6] Eric Freeman, Elisabeth Freeman, Kathy Sierra and Bert Bates. Head First 设计模式[D]. 中国电力出版社出版引入出版,2007.9

[7] sun公司. Effective Java[M]. 机械工业出版社,2009.1

[8] 埃克尔.Think in Java[M]. 机械工业出版社,2007.6

[9] Bignerdranch.Android Programming: The Big Nerd Ranch Guide[M]. 人民邮电出版社,2014.4

[10] Jonathan Simon. Head First Android Development[M]. 电子工业出版社,2011.8

# 附录



图1-1 注册

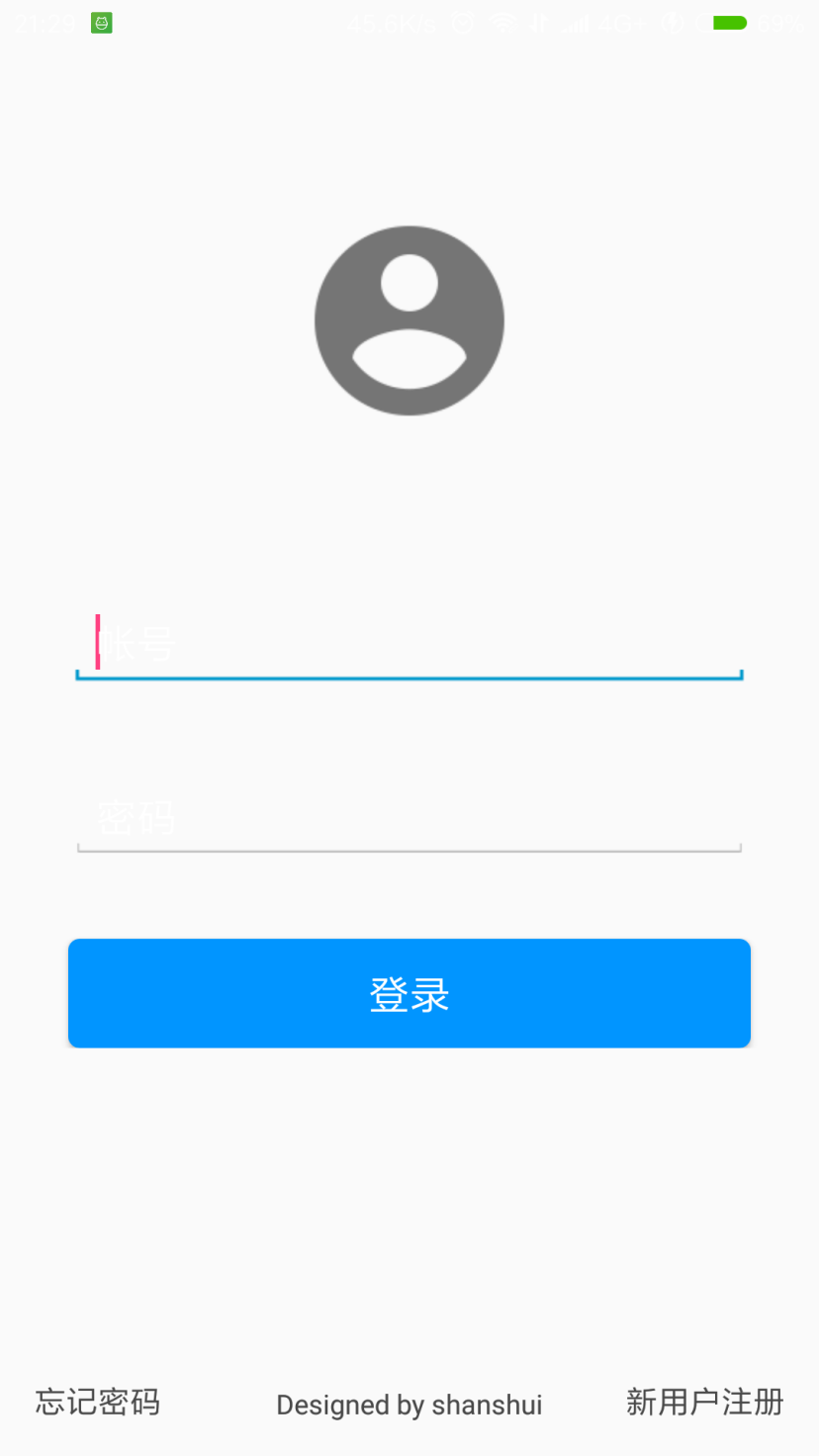


图1-2 登录

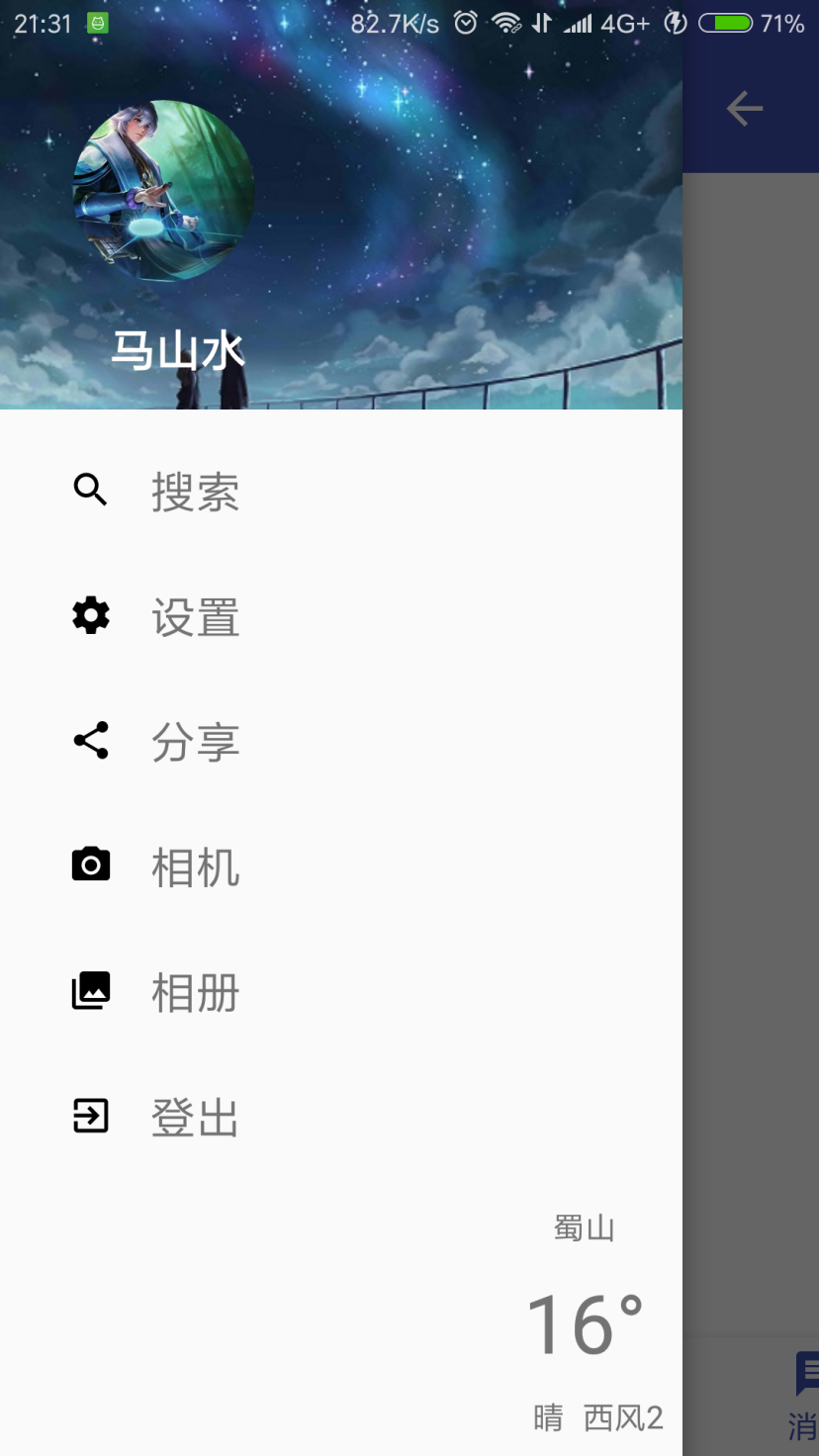


图1-3 天气和用户信息

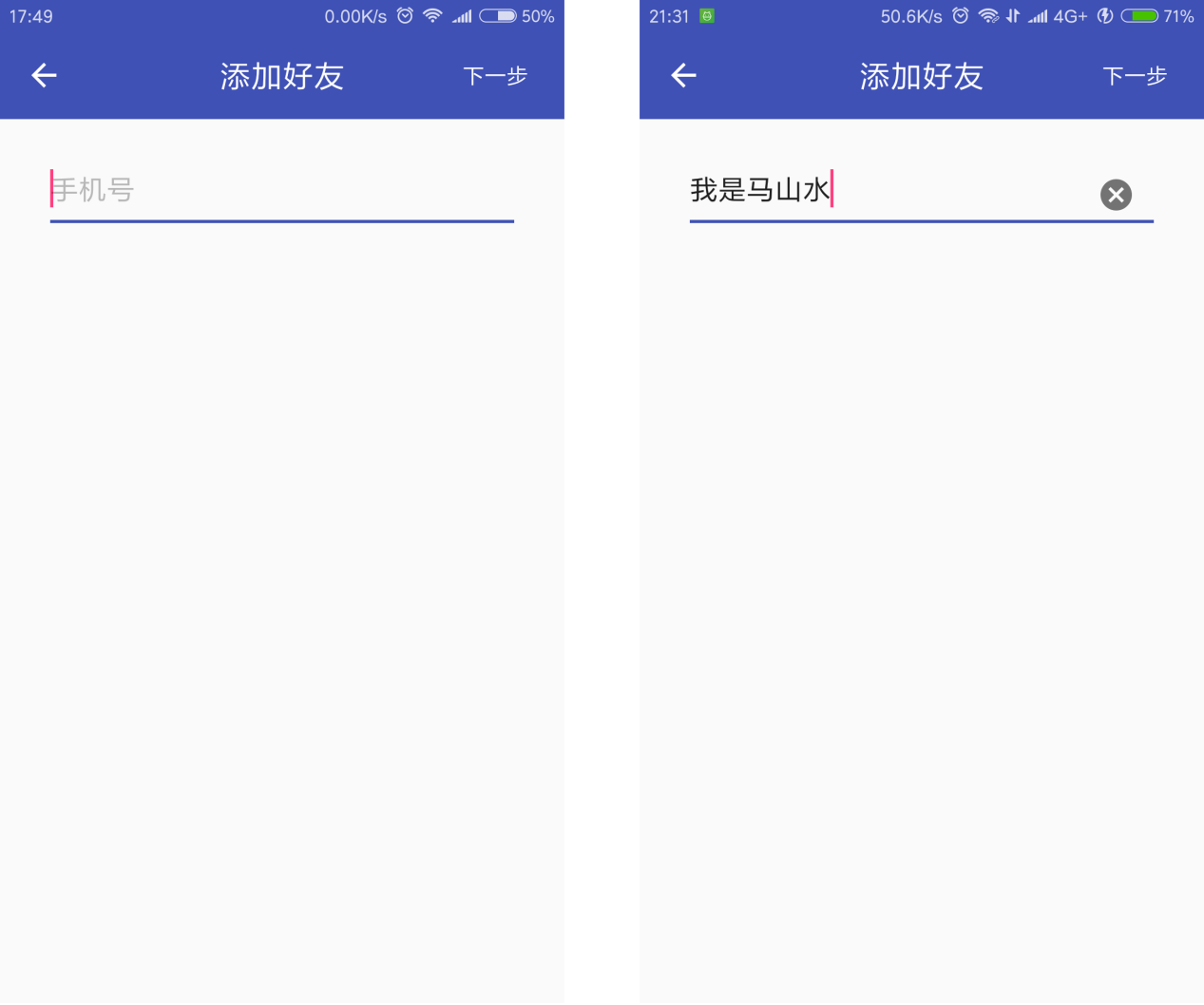


图1-4 添加好友

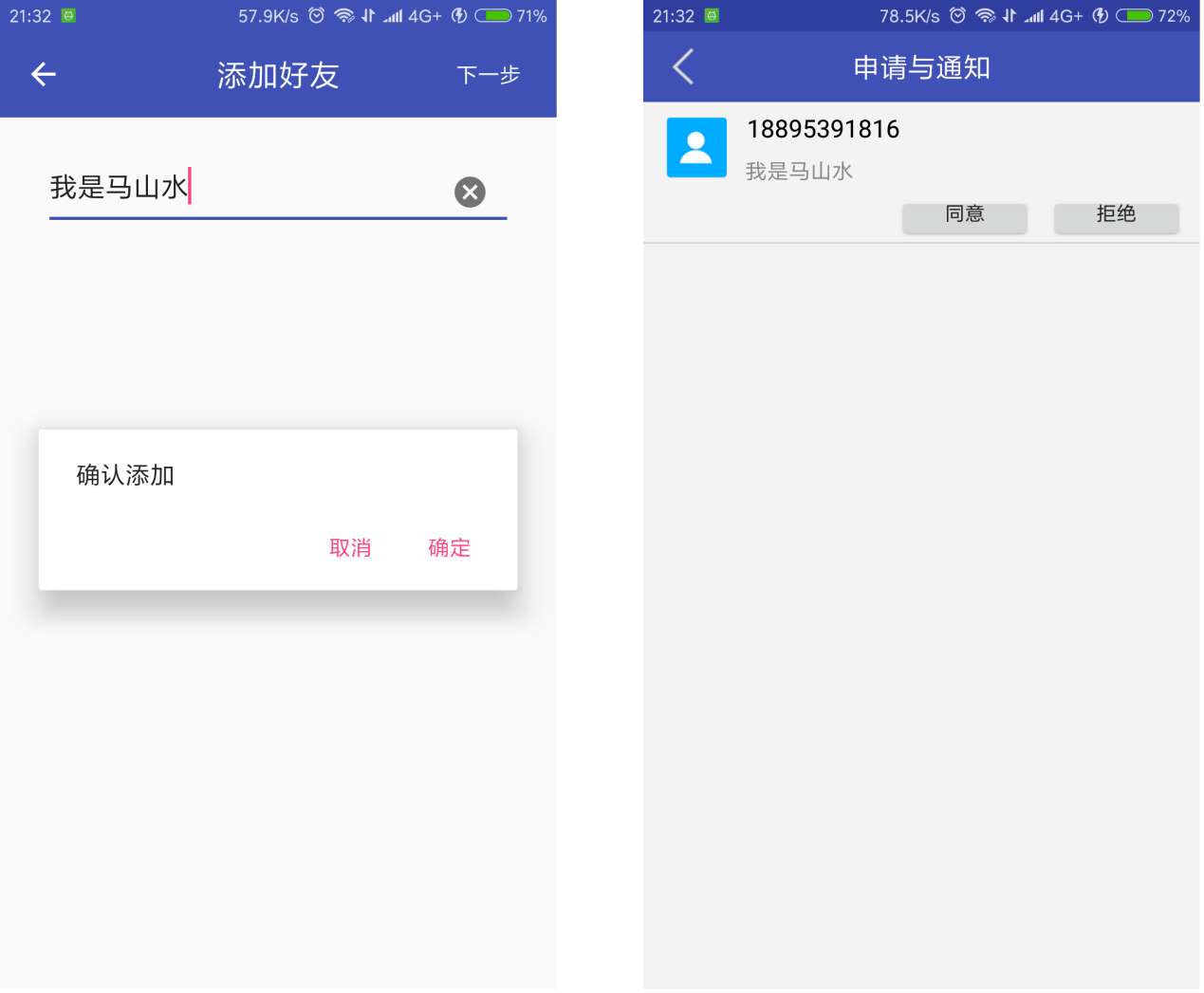


图1-5 添加好友