南京邮电大学

毕 业 设 计（论 文）

|  |  |
| --- | --- |
| 题 目 | 基于XMPP协议的Android即时通信软件的设计与实现 |
| 专 业 | 网络工程 |
| 学生姓名 | 徐阳 |
| 班级学号 | B13070134 |
| 指导教师 | 李养群 |
| 指导单位 | 物联网学院 |

日期： 2017 年 1 月 15 日至 2017 年 6 月 16 日

毕业设计（论文）原创性声明

本人郑重声明：所提交的毕业设计（论文），是本人在导师指导下，独立进行研究工作所取得的成果。除文中已注明引用的内容外，本毕业设计（论文）不包含任何其他个人或集体已经发表或撰写过的作品成果。对本研究做出过重要贡献的个人和集体，均已在文中以明确方式标明并表示了谢意。

论文作者签名：

日期： 年 月 日

摘 要

随着互联网在这些年内的飞速发展，即时通信已经风靡全球，并与人们的日常生活息息相关。各种各样的即时通信软件出现在人们的生活中，国内有流行的微信，国外有MSN。相比于传统的打电话，发短信等通信方式，这种方式更加便捷方便。但由于这些软件采用了不同的通信协议，不同软件间的通信变得异常艰难。但XMPP协议的出现改变了这一局面，由于其是基于XML的，灵活性和规范性得到了极大提升。而且，XMPP协议有助于形成一个统一的通信标准，提升软件的跨平台性，实现不同软件之间的会话。因此，基于XMPP协议的Android即时通信软件的实现有很好的前景。

利用XMPP协议的原理，本论文旨在设计与实现在Android环境下运行的即时通信软件，我完成了以下工作：

1.搭建Android环境，实现手机界面的设计，并选择Openfire服务器作为自己的服务端。

2. 运用基于XMPP的开源库Asmack，来完成客户端与服务端的交互流程，实现用户注册登录，好友管理，以及聊天等功能。

3. 完成在Android上运行的XMPP协议客户端的整体设计，并对成果进行测试，实验各项功能，找出问题和不足。

关键词：即时通信；可扩展通讯和表示协议(XMPP)；Android；Openfire

**ABSTRACT**

With the rapid development of the Internet in these years, instant messaging has swept the world, and people's daily life is closely related. A variety of instant messaging software in people's lives, there are popular WeChat, foreign MSN. Compared to the traditional call, send text messages and other means of communication, this approach is more convenient and convenient. But because of these software using a different communication protocol, the communication between different software has become extremely difficult. However, the emergence of the XMPP protocol has changed the situation, because it is based on XML, flexibility and standardization has been greatly improved. Moreover, XMPP protocol helps to form a unified communication standard, to enhance the cross-platform software, to achieve a different software between the session. Therefore, based on the XMPP protocol Android instant messaging software implementation has a good prospect.

Using the principles of the XMPP protocol, this paper aims to design and implement instant messaging software running in the Android environment, and I have done the following:

1. Build Android environment, to achieve the design of mobile phone interface, and select Openfire server as their own server.

2. The use of XMPP-based open source library Asmack, to complete the client and server-side interactive process, to achieve user registration login, friend management, and chat and other functions.

3. Complete the overall design of the XMPP protocol client running on Android, test the results, experiment with the features, identify problems and deficiencies.

**Key words:** instant messaging；Extensible communication and presentation protocols（XMPP）；Android；Openfire

目 录

第一章 绪论 1

1.1 课题研究的背景 1

1.2国内外研究现状 1

1.3研究内容及意义 2

1.4章节安排 2

第二章 相关技术简介 4

2.1 Java介绍 4

2.1.1 Java语言特点 4

2.2 Android介绍 4

2.2.1 Android的架构 4

2.2.2 Android之Activity生命周期 5

2.2.3 Android之布局 7

2.2.4 SQLite数据库 8

2.3 XMPP协议解析 9

2.3.1 XMPP协议简介 9

2.3.2 XMPP协议内容 10

2.3.3 XMPP地址格式 11

第三章 系统需求分析与设计 13

3.1系统需求分析 13

3.1.1系统整体需求分析 13

3.1.2用例分析 13

3.2系统设计 15

3.2.1系统总体结构设计 15

3.2.2功能模块设计 16

3.2.3数据库设计 17

第四章 系统设计与实现 19

4.1服务器设计 19

4.1.1系统启动流程 19

4.1.2消息处理 19

4.1.3数据库处理 20

4.2客户端设计 21

4.2.1连接服务器 21

4.2.2用户注册 21

4.2.3 用户登录 22

4.2.4 好友管理 22

4.2.5消息传递 24

第五章 功能测试 25

5.1用户注册测试 25

5.2用户登录测试 26

5.3好友测试 27

5.4聊天测试 28

第六章 总结与展望 30

6.1总结 30

6.2展望 30

结束语 31

致 谢 32

参考文献 33

附录A 基于XMPP的Android即时通信系统使用说明 34

# 第一章 绪论

## 1.1 课题研究的背景

随着互联网的迅速发展，PC端互联网已达到了瓶颈，更加方便廉价的移动互联网开始占据人们的生活。据有关数据统计，截止2016年底，中国使用手机上网的用户数量已经超过了总人口的一半，而且增长速度并没有随之下降，已经成为了人们使用互联网的主要方式。正是由于移动设备的普及，以及移动数据速度的大幅提升，人们更倾向于使用手机来接触互联网。Google为此专门推出了Android系统，这是专门在移动设备上使用的系统，由于一系列原因，越来越多的人在使用这个系统。

Android系统是如今世界上使用人数最多的手机操作系统，由于其开源性，越来越多的开发者投入了它的怀抱，产生了许许多多的开源代码，以及第三方的应用程序，使得Android系统在众多人的努力下茁壮成长，如今已经升级了7个Android的版本。

随着网络用户数量的飞速提高，通过互联网传递消息这个方式越来越受到人们的青睐。许多大公司也都推出了各自的即时通信软件，如QQ，微信等，这些软件的出现大大方便了我们的日常生活。相比于传统的通信方式，即时通信的功能愈发强大，文字，图片，语言等信息都能传递。而且，该通信方式最杰出之处便是其即时性，你无需等候太长时间，就能收到消息。英特网已经成为了传输信息的桥梁。

XMPP协议是基于XML的协议，它的出现促进了通信软件的成长，强化了服务器的重要作用和地位。而且，在业内形成了统一的标准，使用这个协议，用户可以跨平台地向其他用户发送消息。

因此，基于XMPP的Android即时通信软件获得了广泛的关注。由于其即时性，直观性，廉价性，人们可以很方便地完成信息的传递，节省了传统的打电话，发邮件来找人的麻烦。总之，即时通信软件拥有非常大的市场。

## 1.2国内外研究现状

即时通信，简单理解，就是人们之间可以实现在较短时间内进行消息的传递。不同于之前的联络方式，比如打电话，发邮件等，即时通信是即时的。而且，可以随时了解对方的在线状态，非常方便。无论是国内，还是国外，都有很多热门的应用。

如今，在中国最火的通信软件便是微信。腾讯于2011年上线了微信这个软件，这是一款免费软件。微信支持了即时通信中的大部分功能，如文字对话，视频见面。此外，还支持中国所特有的发红包功能哦。到了现在，中国已经有接近九成的手机用户在使用微信。而且在国外也有大量的人们使用，超过200个国家，存在着20种语言。

在国外，Skype最受欢迎。它不仅仅具备即时通信的大部分功能，比如发送文字图片等，而且还可以和他人进行视频对话，语音交流，连传统的打电话，发短信等服务也支持。

上面列举了目前主流的几款应用，虽然这些应用都取得了非常大的成绩，但他们用的通信协议都是自主开发的，是公司内部的协议，没有统一的标准。这使得使用不同软件的人们不能相互通信，必须使用相同的软件才能联络。而XMPP协议的出现，让人们可以在不同软件之间通过相同的协议进行通讯成为可能，极大地便利了民众。

## 1.3研究内容及意义

由于移动设备的普及，以及移动数据的提升，手机用户的数量增长飞快，如今已超过了PC端的网络用户。传统的打电话，发短信，发邮件等通信方式已满足不了日益增长的用户需求，为此，即时通信的概念被提了出来。即时通讯的出现，让人们可以通过英特网来发送信息，更加快速方便。

如今，即时通信软件比如微信，Skype都掀起了一股热潮。但这些软件的通信软件都存在着一个缺点，就是使用了自己公司内部的协议，这些协议各不相同，导致不同的软件之间不能相互通信。而XMPP协议的出现，有助于在业内产生一个统一的标准。

如今在手机系统方面，有两种：苹果系统和Android系统。由于Android的开源特性，越来越多的开发者投入了Android的怀抱，产生了许许多多的开源代码，以及大量的应用程序，使得Android系统在众人的努力下茁壮成长。

在研究了如今的课题背景后，本论文决定研究基于Android的XMPP的通信软件，主要做了如下任务：

（1）根据设计需求，利用Android studio搭建出开发环境，设计出用于注册，登录，好友和聊天的大体页面。

（2）利用基于XMPP的开源库Asmack，结合自己选择的开源服务器，实现用户注册，用户登录，好友管理，实时聊天等功能，并和上一步的设计出的页面相结合。

（3）进行该软件的测试工作，测试相关的功能模块，找出不足和bug。

## 1.4章节安排

本文主要讨论了XMPP协议的原理，以及根据XMPP协议，完成一个Android即时通信软件，来满足人们日常的通信需求，改善人们的工作效率。

第一章是绪论，简单描述了该论文所处的背景，国内和国外在这方面研究及成果，以及我进行研究的内容和意义。

第二章是相关技术简介，主要介绍了Java，Android的架构，活动等重要知识点，以及XMPP协议的相关知识点。

第三章进行了该通信软件的需求分析工作，并用用例图的方式来简单阐述了功能需求。

第四章是该通信软件的系统设计与实现，分别从客户端和服务端这两方面进行分析设计。

第五章则是对自己的成果进行功能测试，主要是针对客户端的用户注册，用户登录，好友管理，消息转发等功能进行测试。

第六章是对自己的总结和展望，讲述自己完成这篇论文的主要步骤，以及还存在哪些问题亟待解决。

# 第二章 相关技术简介

## 2.1 Java介绍

### 2.1.1 Java语言特点

Java是面向对象的，是在其他语言的基础上进行研究改进的，不仅汲取了其他语言的优势，而且将一些难懂的知识进行了简化，使得Java语言简单方便，灵活易用。而且，由于Java的开源特性，产生了许许多多和Java相关的开源库，以及开源框架，人们可以利用这些开源代码很轻松地就实现自己想要的功能。

Java语言的特性有三个：封装、继承、多态。

（1）封装：将所有的方法和属性封装在同一个事物里，通过新建此类对象可以访问到这个事物中的方法和属性。封装就是将对象的属性及其方法，构成一个不可分割的统一实体，用户不需要知道这些方法的实现细节，就可以直接来访问其方法，有效地保障了其安全性。

（2）继承：将全部共同的方法和属性集合起来形成一个类，这个类就被称为父类。其他类只要继承这个父类就可以获取这个类中的所有方法和属性，而且还可以加入自己所特有的方法和属性。不仅可以复用代码，而且可以实现将类进行划分类型。

（3）多态：多态是在封装和继承的基础上实现的。想要实现多态，就必须满足有三个必要条件中的一个，这三个分别是继承，重写和父类引用指向子类对象。子类继承父类，在这两个类的相同的方法中，参数相同，但内部的行为细节却有出入，这就称为多态。多态的使用，使代码更加灵活，更加方便。

## 2.2 Android介绍

### 2.2.1 Android的架构

Android是基于Linux的开源操作系统，是由Google开发实现的。Android的核心依赖Linux系统的内核，因此拥有很多Linux的优点，可以在ARM，X86，MAC等不同结构的机器上运行。  
 Android的系统架构是分层的，可分为四个层：应用程序、应用程序框架、系统运行库和Linux核心。

（1）应用程序

应用程序是Android中的主要构成部分。Android上的通话，短信，浏览器都属于是Android上的应用程序。正是由于Java的跨平台，应用程序可以不编译，就直接使用。而且，Android的开放性，也吸引了无数的Android开发者，编写了无数的手机程序，不仅提高了Android的可用性，而且充实了用户的日常生活。

（2） 应用程序框架

在这层中，有着开发Android所必要的一系列库，开发者可以使用这些核心的应用程序库提供的接口。这些接口中包含着许多模块，比如Android自带的视图View，访问数据的内容提供器，以及管理资源的资源管理器。开发者可以使用这些模块来更快速地实现自己的功能，用户也可以替换为自己喜欢的应用程序。

（3）系统运行库

a)程序库：Android虽然支持Java，但还是有些用C/C++语言实现的运行库。很多组件都在使用着这些运行库。比如媒体库，SQLite数据库都是C语言实现的。

b)Android 运行库：Android运行库里都是放着一些核心的东西。这些核心的东西都会在一个个互不干扰的虚拟机中运行，这种是Dalvik虚拟机，Android可同时运行多个Dalvik。

（4） Linux 内核

基于Linux的内核，Android形成了自己的核心优势，如高可靠性，稳定的网络通讯，优秀的内存机制等。除此以外，Android也对其做了一些修改，比如在进程通信，以及节省电力方面。如图2.1所示：

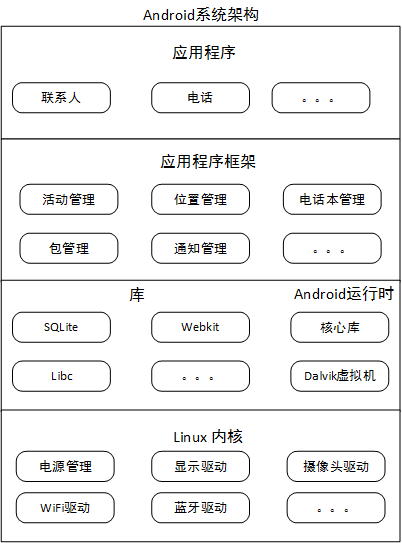


图2.1Android架构

### 2.2.2 Android之Activity生命周期

Android组件有最基本常见的四大组件，Activity是其中之一。Activity是用来和用户交互的，用户可以使用Activity来交互实现某项任务。在Android开发中，创建活动后，用户可以通过调用setContentView()这个函数，给这个Activity里指定一个布局界面View，这个界面的作用就是给用户实现交互的。Android系统中管理Activity是通过操作Activity栈来实现的，而Activity管理自己则是通过Activity自带生命周期的方法来进行操作的。下面就讲述了Activity的生命周期的大体过程。

1.onCreate()：Activity在第一次被创建时，会调用该方法。在这个方法里，你需要将一些初始化的操作完成：加载View，数据绑定到事件等。如果是第一次创建，接下去会调用onStart()方法，如果是停止后重新显示，那么会调用onRestart()这个方法。

2.onStart()：当前Activity对用户由不可见变为可见时，会调用该方法。当这个方法调用完，会接着调用onResume()这个方法。

3.onResume()：Activity会在用户将要和用户交互时，调用此方法。在进行该方法后，此时的活动会处在运行状态。

4.onPause()：当系统要启用其他活动或者恢复之前活动时，会调用这个方法。onPause()方法启动时，占用系统的资源比如CPU资源，会被回收释放，一些比较重要的数据则会被保留下来。

5.onStop()：当其他活动启动掩盖该Activity，该活动变得不可见时，会调用该方法。这个方法是紧接着onPause()之后执行的。如果当前活动恢复，会启动onRestart()方法；否则，会销毁掉。

6.onDestory()：调用该方法，活动会被销毁。

7.onRestart()：当活动经过onStop()方法处于停止状态时，之后如果用户再次需要该活动时，就会调用该方法。调用完后，该活动会恢复运行。

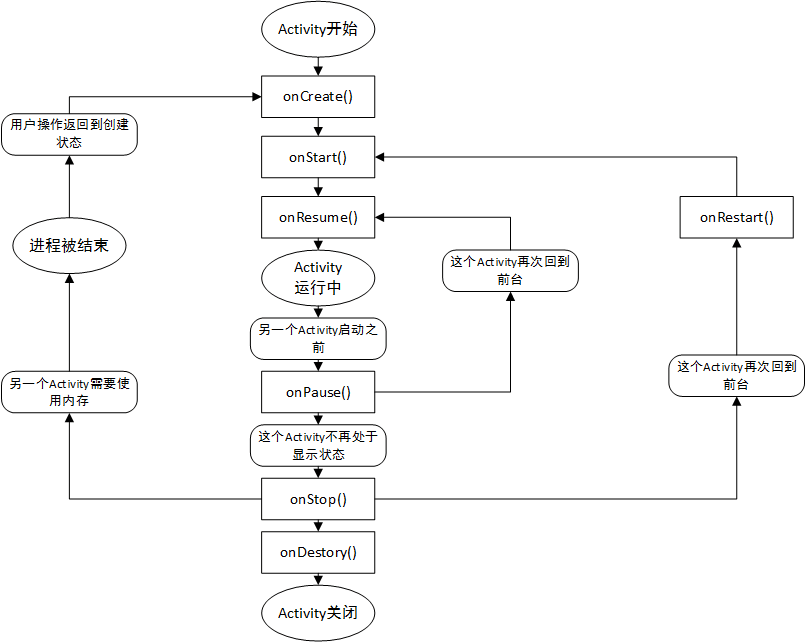


图2. 2 Activity生命周期

### 2.2.3 Android之布局

Android里有多种多样的控件，这些控件的有机结合，就可以形成丰富的Android界面。为了让这些控件在界面上按照自己的想法排列，Android引入了多种布局方式。布局是一种容器，它可以放置许多控件，控件的位置可以按照一定的规则进行改变，然后就可以将页面变得精美。而且，布局除了放置控件外，还可以在里面嵌套其他布局，通过多种不同的布局，就可以实现比单一布局更加复杂的页面。接下来，详细讲解了四种的基本布局。

（1）线性布局--LinearLayout

线性布局，从字面意思理解，就是从上往下一个一个的排列在界面上，就像html中的div标签一样。LinearLayout又可分为两种：垂直布局和水平布局。垂直布局的意思就是，每个元素都是按从上往下的方式排列的，每个元素独占一行；水平布局的意思是，所有元素都在同一行中，每个元素都是按照先后顺序在同一行进行排列的。在线性布局中，weight也是一个重要属性。在垂直布局时，代表了这个控件占总高度的比重；在水平布局时，代表的是该控件在宽度上占的比重；每个比重相当于该控件的weight除以所有的weight。

（2）帧布局--FrameLayout

帧布局应该算是所有布局里比较简单的。所以控件都会出现在左上角，越先加载的控件，越会出现在最下面，它们会相互重叠在一起。帧布局里自带的属性很少，提供了一种对其方式layout\_gravity,可以设置其为left 或right来实现控件左对齐，或右对齐。

（3）相对布局--RelativeLayout

相对布局就是说，相对于一个元素，可以按照这个元素为参照物，进行排列。可以以父类为参照物，也可以以同级元素为参照物，来进行上下左右各个方面的相对排列，比如居中，右下角等布局。相对于其他的排列方式而言，该布局灵活性大，有非常多的属性。

（4）百分比布局--PercentFrameLayout

百分比布局是Android为了解决控件大小问题而引入的。在上面三种布局中，只有线性布局中，有个weight属性可以按比例指定控件的大小。百分比布局针对上面的帧布局和相对布局，分别定义了PerencetFrameLayout和PerencetRelativeLayout，为这两个布局分别提供控制控件大小的功能。需要注意的一点是，这个布局需要自己引入，否则不能使用。

### 2.2.4 SQLite数据库

SQLite是Android系统中自带的数据库，无需自己引入。和MySQL数据库相比，SQLite虽然比较轻量，但结果却是更快的运行速度，更小的内存使用，在移动设备这种内存本来就不大的设备上使用，非常有优势。配置SQLite也非常简单，因为SQLite相当于零配置。但由此也带来了一些缺点，比如并发慢，对数据库进行增删改操作相比而言也比较慢。

SQLite数据库分为了三个模块：编译器(Compiler)，虚拟机(Virtual Machine)和后端(Back-End)。编译器的作用是对传进来的SQL进行检测，如果没有语法错误，会发送给虚拟机进行处理。虚拟机接收到这些数据后，会解释这些字节码，来进行数据的增删改查操作。而后端则是管理这些数据的，通过B+树，缓存等技术，来更迅速的实现各种操作。图2.3就是SQLite的架构

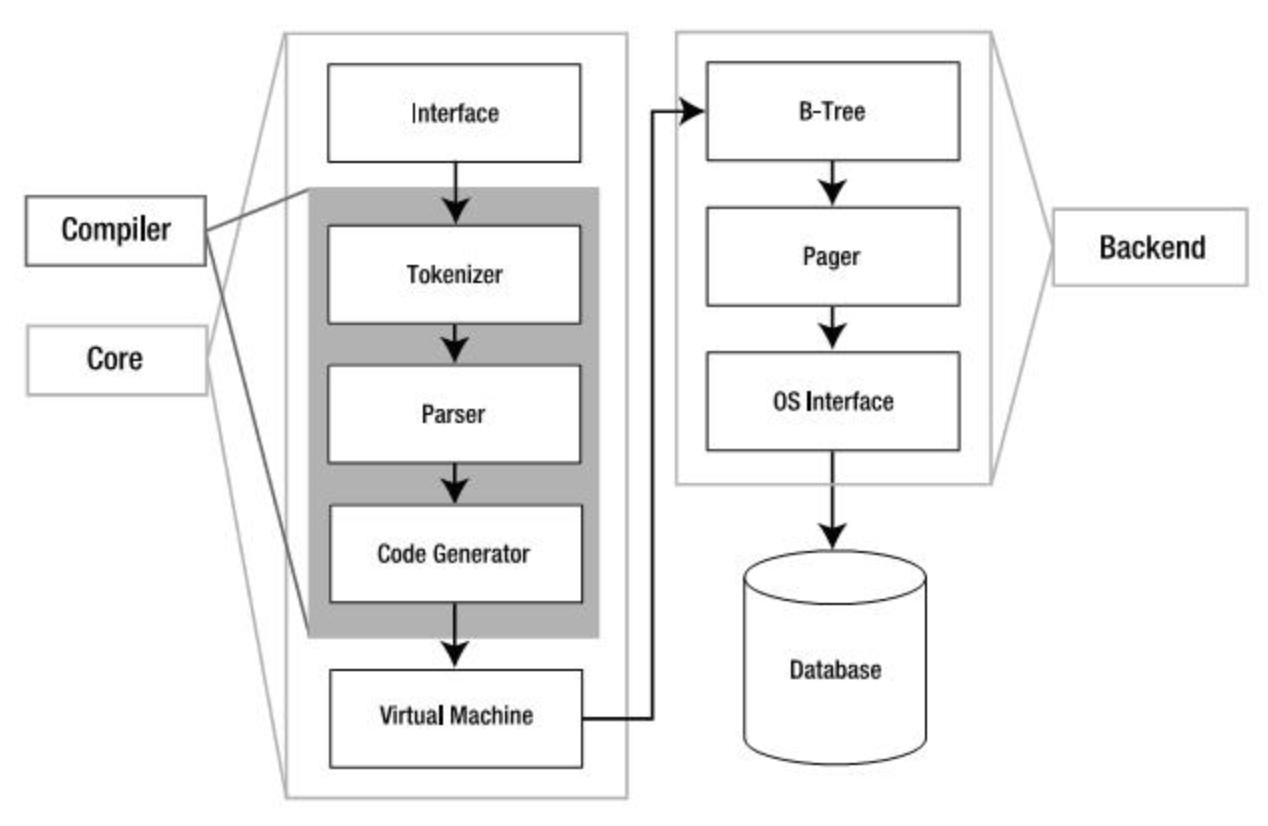


图2.3 SQLite架构

## 2.3 XMPP协议解析

### 2.3.1 XMPP协议简介

现在，存在着许许多多的即时通信协议，比较出名的有四个，其中之一就是XMPP协议。XMPP是由Jabber转变而来，是开源的通信协议。它的最大特点便是基于XML实现的 。它有以下优点：

(1).XMPP 协议是开源的，任何人都能得到XMPP的源码。经过无数人的验证和修改，XMPP协议变得更加完善，不足之处也在慢慢减少。而且，其开放性也让业界有了统一的规范。

(2). XMPP的扩展性非常好。由于XMPP协议是基于XML的，所以信息都是以XML形式存在的。XML的使用强化了数据的传输，用元素和属性来描述数据，而不使用其显示方法，致使无论是怎样的数据，都可以保存在信息中，提高了XMPP协议的扩展性。而且比起传统的方式少去了字符串解析步骤，节省了大量时间。

(3).分布式架构。XMPP 协议是基于C/S 架构的，和其他使用P2P的通信软件不同，消息主要是由服务器来传递的，这点和电子邮件的传递过程十分相似。而且，XMPP的三个元素之一的网关是异构的，使之实现了不同用户之间的互联互通。

(4). XMPP 使用范围广。XMPP 不仅仅可以使用在即时通信领域上，而且在游戏，视频对话，办公协同工具等领域也有很大的使用前景。

(5).安全性。XMPP协议会使用TLS和SASL协议来加密和解密进行消息转发的整个过程。TLS主要的作用是保证传输通道的安全性，SASL作用是鉴别用户的权限的。这两个协议的使用大大增强了XMPP的通信安全，在内部网之外的网络也有很强的保密性。图2.3展示了XMPP的安全性：

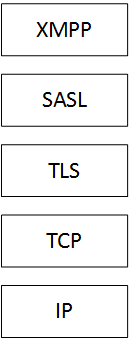


图2.3XMPP安全性

### 2.3.2 XMPP协议内容

XMPP是基于XML协议的，传输的是XML格式的数据。其他通信软件的传输内容则各不相同，比如腾讯QQ是使用二进制的数据进行传输的，MSN则是使用纯文本加其他符号来实现传送的。而XMPP传输的XML格式的信息，不但省去了传统的字符串解析工作，而且通俗易懂，能一眼看出每部分的用处和功能，简化了开发，方便了查错。由此可见，传输XML的协议是XMPP协议的基础，也是核心所在。

用户与用户之间的通信，都要经过服务器。服务器相当于一个中转站，所有的消息都要经过它来进行转发，才能到达各自的目的地。下面就讲解了用户和用户之间的消息传输过程：

(1).客户端要想进行下一步操作，第一步就要和服务器进行成功连接操作，这是先决条件。

(2).服务器收到上一步发送的连接请求，进行验证操作，无法通过则不会连接成功。

(3).确定想要交流的好友id，发送给服务器。并通过这个id知道对方的是否在线。

(4).服务器通过上一步发送过来的信息搜索其他客户端，连接和验证对方的身份。

(5).用户之间这时可以相互通信了。

XMPP开源到现在，拥有了许许多多的协议范本，别人也以XMPP为基础做了很多扩展，下面介绍几种重要的协议范本。

(1).RFC 3920：这是XMPP协议的核心部分。这个范本规定了XMPP使用XML协议来实现信息的传输，大大提高了XMPP的开放性以及扩展性，定义了一种统一的规范。这个范本也规定使用TLS和SASL来加密通信过程，提高了XMPP的安全可靠性。JID的定义，和错误样式也是在这部分定义的，所以，这个是核心所在，所以基于XMPP的都要实现该范本。

(2).RFC 3921：在用户连接服务器后，所做的操作比如好友的添加删除，消息发送都是该范本定义的。通过三种不同格式的XML字节来完成上面的这些功能，IQ节是用来请求响应的，Presence节是用来表明用户的在线或不在线状态的，Message节则是用来传递消息的。阻塞策略也是在该部分定义的，到如今已有多种方式。和RFC3920结合，就可以实现简单的即时通信软件了。

(3).除了上面这两种比较重要的范本外，还有许多。比如，XEP-0045是用来实现多人聊天的，XEP-0096是用来实现文件传输的，XEP-0167则是用来发送语音的，其他的还有很多，就不一一介绍了。

### 2.3.3 XMPP地址格式

每一个用户都有一个唯一标识符JID。JID标识符的作用很大，不仅可以表示一个用户，而且还可以表示类似聊天室等其他内容。JID的格式有严格的规定，必须由域名，节点以及源这三部分组成。格式也必须为 节点@域名/源 这种形式，前半部分和电子邮件的地址格式相类似。举个例子，当用户在服务器s上注册了一个名称为name的账号后，这个用户的JID就是name@s。以后每次与服务器进行通信，都会使用这个JID，后面的源则会跟着由于登录设备的不同而发生改变。用户的源可以有多种不一样的。

XMPP协议中定义了 三种XML元素: Message、Presence、IQ。Message主要负责消息的传输。用户每发送信息时，就会新增一个Message标签，这个标签里面放着用户想要说的话。Presence主要用来负责用户状态，随着用户的上线或下线，都会新增一个Presence标签，里面存放着用户的上线或下线的状态。IQ则是用来负责客户端与服务器之间的请求和响应的，服务端和客户端进行交互，都会产生IQ标签，存储请求或存储状态。下面详细介绍这三种元素：

(1).Message元素

主要是负责用户之间的消息的传输。Message首先会发送给服务器，服务器根据数据包中的的目的地址，查找到该用户，判断其状态，在线则会立即发送出去，否则会存储在本地，等其在线时，再发送出去。里面有三种属性。To :用来表示接收该消息的用户。from : 指发送该消息的用户。Text: 用来保存要发送的内容。

Message元素的格式如下所示:

<message to= [‘niyi@xy.cc/content’](mailto:%E2%80%98lily@jabber.org/contact%E2%80%99) type =’chat’>

<body>我有件事要请教你</body>

</message>

(2).Presence元素

主要是负责用户的上下线状态。当用户上线或下线时，都会产生Presence包，这个包里放着用户的在线信息。Presence中有三个属性：To和from属性和Message相似。status: 用来保存在线状态。

Presence元素的格式如下所示:

<presence>

From =‘niyi @ xy.cc/contact’

To = ‘xy @ xy.cc/contact'

<status> Online </status>

</presence>

(3).IQ元素

主要用来表示手机端与服务端之间的请求和响应信息。当服务端和手机端进行消息传递时，IQ元素被转发。例如，当一个手机端搜索一个朋友时，它向一个服务端发送一个IQ请求信息。在服务端处理完之后，它返回一个IQ响应信息。

IQ元素的属性有from，id，type三种。最重要的是type属性，用来区分是请求还是响应。From是该请求或响应的来源。Id则是该请求或相应的id

IQ元素的格式如下所示:

<iq from =‘niyi@ xy.cc/content’ id=’99121’ Type=’get’>

# 第三章 系统需求分析与设计

## 3.1系统需求分析

### 3.1.1系统整体需求分析

根据任务书的要求，我的最终目标是做出一个基于Android的即时通信软件。为了能更好地完成整个软件的设计，我先对该软件所需要具备的功能进行详细的研究分析，为下面进行设计打下基础。

(1).登录注册

用户只有在注册过后，才能登录此系统。

若用户没有注册账号，可以先进行注册，注册成功后会自动跳转到主界面，注册失败会给出相关提示。若用户己经注册过该系统，可以直接进行登录，登录成功会跳转到系统主界面，登录失败会给出相关提示。

当用户登录成功后，也可以退出该系统，选择其他账号登录。

(2).好友管理

a.添加好友

用户在登录成功后，可以搜索要添加的好友姓名，选择添加该好友，如果好友不存在，则会进行提示。在进行添加后，要等对方确定后，才会成为好友。

c.查看好友列表

用户可以查看自己所有的好友。好友列表展示每个好友的头像和姓名。

e.删除好友

如果不想要该好友，你可以选择删除该好友。

(3).即时通讯功能

a.发送消息

进入自己想要聊天的人的聊天页，在输入框中输入文字，图片等数据，然后发送出去。而且，自己发送的消息排布在屏幕的右边，别人的消息排布在屏幕的左边。

b.查看聊天记录

和好友的聊天记录会保存在各自的手机上，每次与该好友进行会话，都可以看到之前的信息。

### 3.1.2用例分析

对于软件来说，用例图的使用更加明确了软件的需求分析，对开发人员以后详细的设计思路进行了描述。用例视图描述了系统与用户的关系，用户该怎样与系统进行联系，系统为用户提供了哪些功能等。可以说，用例图的使用，大大缩短了软件开发的周期，在实际开发中被广泛使用。

为了更加清晰的展示我的系统，根据需求分析，我分析出系统的具体操作步骤及功能，从而确定了该系统的用例图。图3.2是系统的用例图：

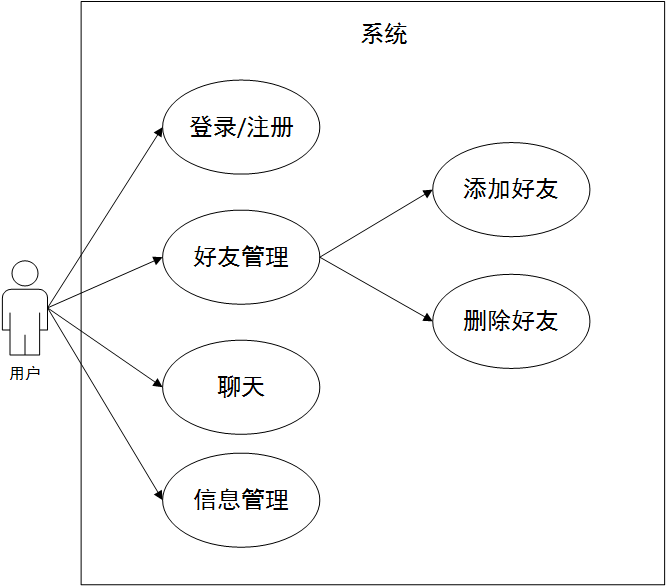


图3.2系统总体用例图

系统的各项功能描述如下：

(1).用例1

用例名：注册/登录

行为者：用户

目的：让用户有该系统的账号，让其可以登录。

过程描述：

a.进入登录页，点击“注册”按钮，可以进入到注册页；

b.进入注册界面后，填写账号，密码等信息；

c.数据无误后，用户注册成功，然后就可以进行登录。

(2).用例2

用例名：好友

行为者：用户

目的：可以添加好友，删除好友，来让用户可以选择人聊天。

过程描述：

a.登录进入首页，点击下方的通信录；

b.进入通信录管理界面；

(3).用例3

用例名：添加好友

行为者：用户

目的：让用户可以和更多的人聊天。

过程描述：

a.在好友列表页，点击“好友增加”按钮；

b.在搜索框中，输入关键字，进行查询找到该用户；

c.点击添加按钮，添加成功。

(4).用例4

用例名：删除好友

行为者：用户

目的：让用户删除不想聊天的好友。

过程描述：

a.点击该好友，进入聊天界面；

b.点击该好友头像，进入信息展示页，单击移出按钮；

c.删除好友成功。

(5).用例5

用例名：聊天

行为者：用户

目的：用户和用户之间可以即时通信。

过程描述：

a.在添加好友后，可以点击该好友；

b.进入聊天界面，你可以发送文字，图片和语音等；

c.你的好友可以收到，并进行回复。

(6).用例6

用例名：信息管理

行为者：用户

目的：对自己的信息进行修改。

过程描述：

a.进入我的界面；

b.可以修改自己的详细信息。

## 3.2系统设计

### 3.2.1系统总体结构设计

通过上面对自己毕设软件的需求分析，我大体了解了该软件需要具备哪些模块，在此，进行了一次整体的系统设计。图3.2是系统的总体架构图。

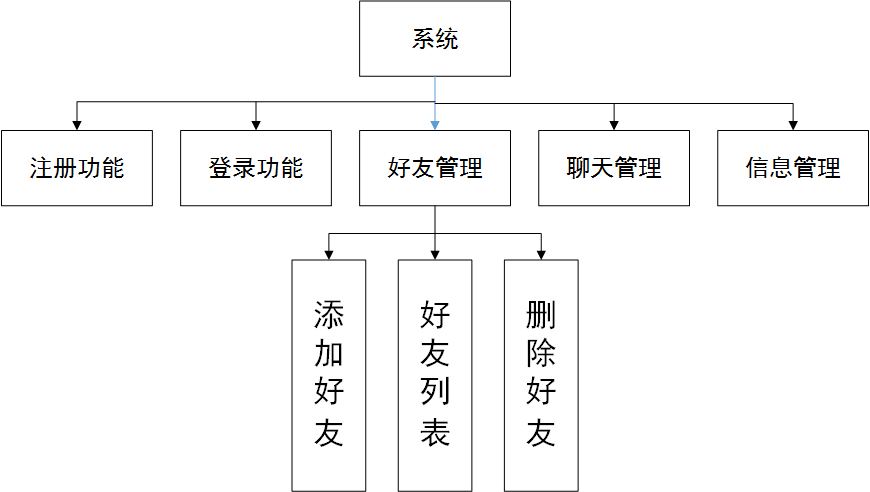


图3.2系统总体架构图

### 3.2.2功能模块设计

(1).注册和登录模块

注册模块：用户进入注册页面，输入账号，邮箱和密码这三个字段，密码要填两次进行确认。在提交到服务器之前，对这三个字段的格式和类型进行检测判断，比如邮箱的格式，以及密码的长度限制，在满足要求后，再提交到服务器进行注册功能。

登录模块：用户注册后，会把用户信息保存在服务器的数据库中，在进行登录操作后，会从服务器的数据库中查找是否有该用户存在，如果是，则登录成功，否则登录失败。这两个模块的内容如图3.1所示：

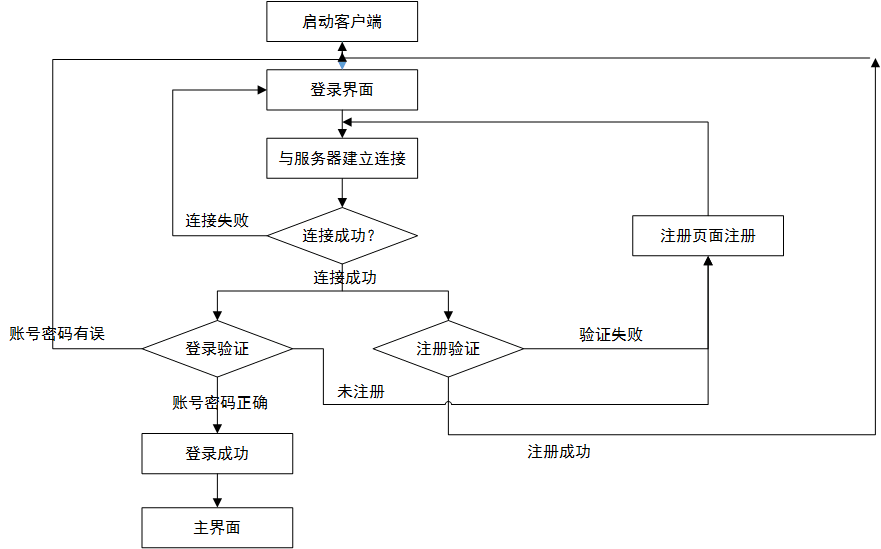


图3.1注册登录模块

(2).好友模块

好友模块大体有三大部分：好友列表，新增好友，删除好友等功能。登录成功的用户，可以在好友搜索页面添加自己想要加的好友姓名，如果存在，则可以显示出来，并添加为好友，否则显示该好友不存在。图3.2展示了好友模块的内容：

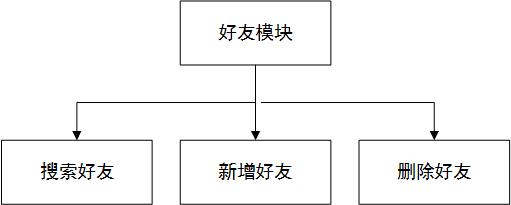


图3. 2好友模块

(3).消息模块

聊天模块：用户登陆后，在选择自己的好友后，可以进行一对一的聊天。用户可以发送文字，表情，语音，图片等，聊天记录保存在客户端。

a.发送信息

在添加完好友后，用户可以选择任意一个好友进行对话，点击要交谈的用户，进入聊天界面。然后在底部写入自己想要说的话，就可以发送出去。可以发送文字，图片，语音等不同类型的数据。

b.接收信息

登陆成功后，如果有其他好友发来的信息，会有提示音，提示用户查看，在首页，会出现其他好友发来的信息，显示条数。用户可以点开这个界面，查看具体信息。

c.处理历史信息

在聊天过程中，每次用户发出聊天的内容或者收到聊天的内容，都会保存在Android自带的SQLite数据库中，在下次进入到该聊天界面时，都会把之前聊天的数据显示在页面上。

(4).信息修改模块

用户登录后，可以修改自己的信息，主要包括两个：修改密码和修改基本信息。在修改密码时，首先需要输入自己的原有密码，然后输入2次相同的自己想改的密码。在改动基本信息时，用户可以将自己的昵称等信息进行二次改动。

### 3.2.3数据库设计

在设计该系统时，需要进行一些数据的持久化处理。Android内置了数据库SQLite。由于本系统是C/S架构，客户端和服务端都要在本地保存数据，而服务端选用的是开源的Openfire，会在MySQL中自动生成服务端所需数据库。客户端的数据则需要保存在本地的数据库SQLite中，表3.1，3.2，3.3分别是保存在客户端的信息表，好友表，新消息表：

表3.1信息存储表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 主键 | 描述 |
| id | number | 是 | 信息id |
| username | varchar | 否 | 对方的名称 |
| head | varchar | 否 | 头部信息 |
| msg | varchar | 否 | 消息 |
| sendDate | varchar | 否 | 发送时间 |
| inOrOut | number | 否 | 发送或接受 |
| whos | varchar | 否 | 谁的 |

表3.2好友关系表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 主键 | 描述 |
| id | number | 是 | 好友id |
| username | varchar | 否 | 对方名称 |
| sendDate | varchar | 否 | 发送时间 |
| isDeal | number | 否 | 删除状态 |
| whos | varchar | 否 | 谁的 |

表3.3新消息表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 主键 | 默认值 |
| id | number | 是 | 好友id |
| msgId | number | 否 | 消息id |
| msgCount | number | 否 | 消息数量 |
| whosMsg | varchar | 否 | 谁的消息 |

# 第四章 系统设计与实现

## 4.1服务器设计

XMPP 服务器拥有大量的实现方案，其中，较为成熟，且使用广泛的是ejabberd, jabberd 2.x, Openfire这几种服务器。Openfire不仅是用Java语言实现的，而且通过XML协议来管理信息。所以，我选择了Openfire这个服务器来作为自己的服务端。

Openfire的通信处理是基于Apache MINA框架实现。MINA的使用，简化了人们开发网络应用的过程，特别是那些要求比较高的应用。

### 4.1.1系统启动流程

系统在启动时，会调用ServerStarter类中的启动方法，然后通过Java中的反射技术，加载XMPPServer这个类，调用这个类中的启动方法，接着就会进行一系列所必需的操作，从加载，初始化，到启动系统模块。

然后，服务器会在5222这个端口上进行监听，判断客户端是否有信息发送。图4.1是服务器的启动过程：

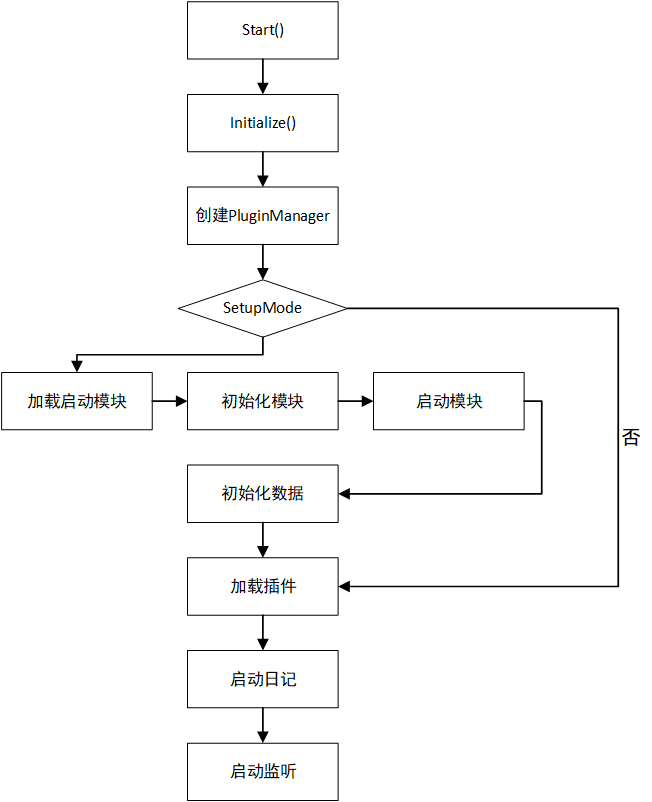


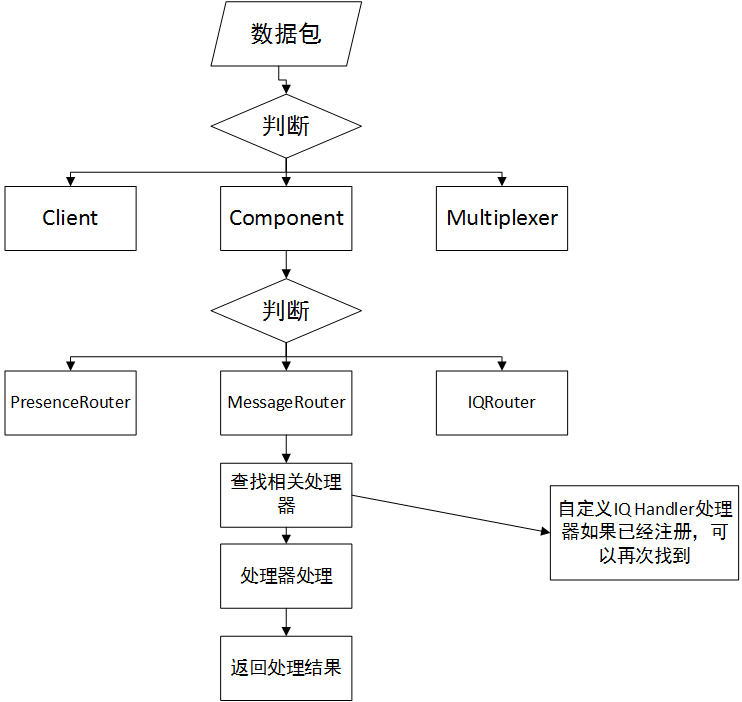
图4.1启动服务器

### 4.1.2消息处理

在服务器的消息处理中，为了监听客户端发送过来的数据，Openfire服务器会启动一个监听服务，这个监听服务是采用TCP/IP协议实现的。这个过程是由XMPPServer类的start()方法执行的。

在Openfire中会有一个服务端的接收器，专门用于接收信息的。让了防止接收到无用的数据，还拥有了一个过滤器来过滤无用的信息。只有有用的信息才会被保留下来。

接着，服务端会启动监听方法，在5222这个端口进行监听，对手机端的消息进行接收。图4.2是消息包接收处理的流程。

图4.2Openfire消息包接受处理流程

### 4.1.3数据库处理

Openfire通过Java中的JDBC方法来连接MySQL数据库的。在安装Openfire时，可以选用MySQL数据库。MySQL是一种关系型数据库，关系数据库将不同类型的数据保存在不同的数据表中，而不是将所有不同类型的数据放在同一个表内，这种改变提高了进行数据增删改查的速度。在图4.3页面填写完JDBC驱动程序，数据库URL，用户名，密码等相关参数后，然后再执行配置就可以成功配置数据库了。接着，Openfire服务器会在自己选中的MySQL数据库里自动新建所需的表和数据。



图4.3Openfire数据库配置

## 4.2客户端设计

为了更加方便的操作XMPP，我选择了Asmack这个开源库。Smack是以XMPP协议为基础实现的，与服务端进行通信的开源库。Asmack是Smack库的Android版本,虽然Smack在电脑端上适配很好, ,然而由于Android中缺少了Java的一些类库,而这些类库正是Smack所必需的,因此在Android平台上Smack的兼容性并不好。所以在客户端中，我采用了Asmack这个开源库来完成XMPP的相关操作。

### 4.2.1连接服务器

由于用户登录/注册，好友管理等功能都是需要与服务器进行交互的，因此，如何连接服务器是下面一系列实现的第一步。Asmack中提供了方法来连接服务器，此方法需要服务器IP，端口号等参数。如图4.4所示：

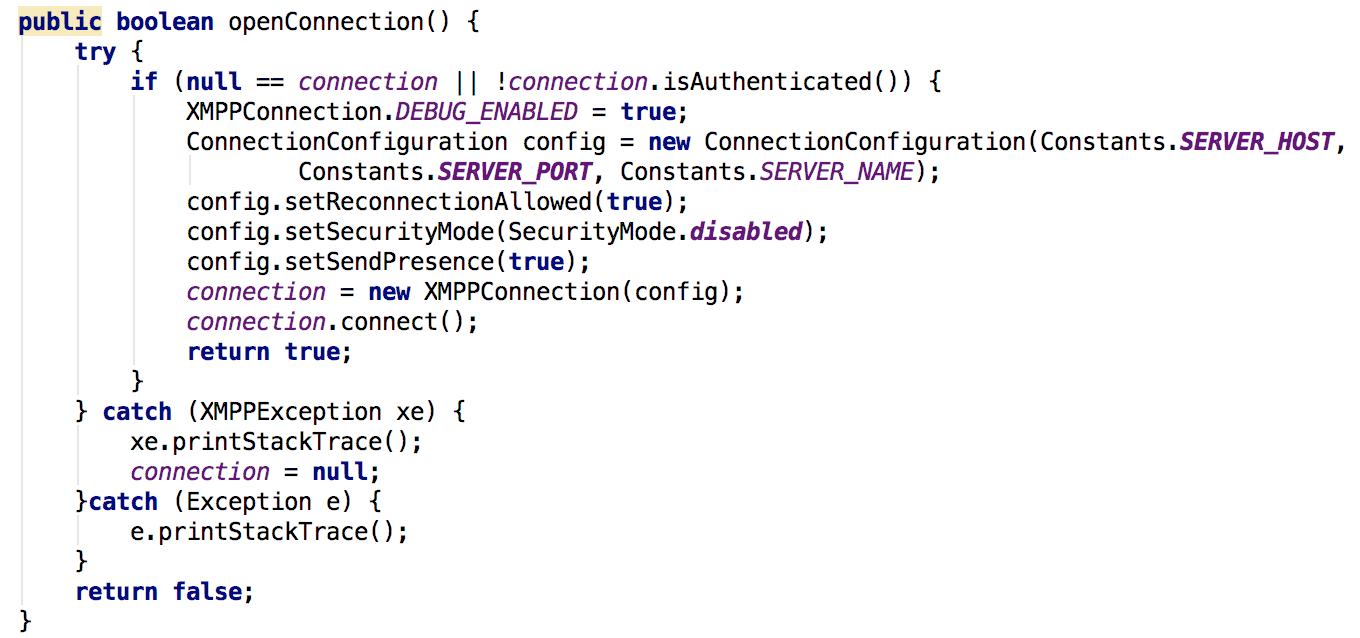


图4.4连接服务器的代码实现

### 4.2.2用户注册

在注册页面，用户需要按照要求填写信息，包括账号，邮箱和密码，用户填写的邮箱必须符合邮箱的规范，而且密码要输入两次进行确认，防止出错。单击“注册”按钮，客户端会先检查是否还有信息没填写，两次密码是否正确等。如果通过会给服务端发送一个请求，请求注册，否则提示“信息有误，重新填写”。

接着，服务端会在用户表中校验账号是否重复，如果没有重复，会完成注册工作，然后返回成功响应，客户端会进入首页；否则，提示“注册失败”，让用户再次注册。图4.5展示了用户发送给服务端的注册请求。

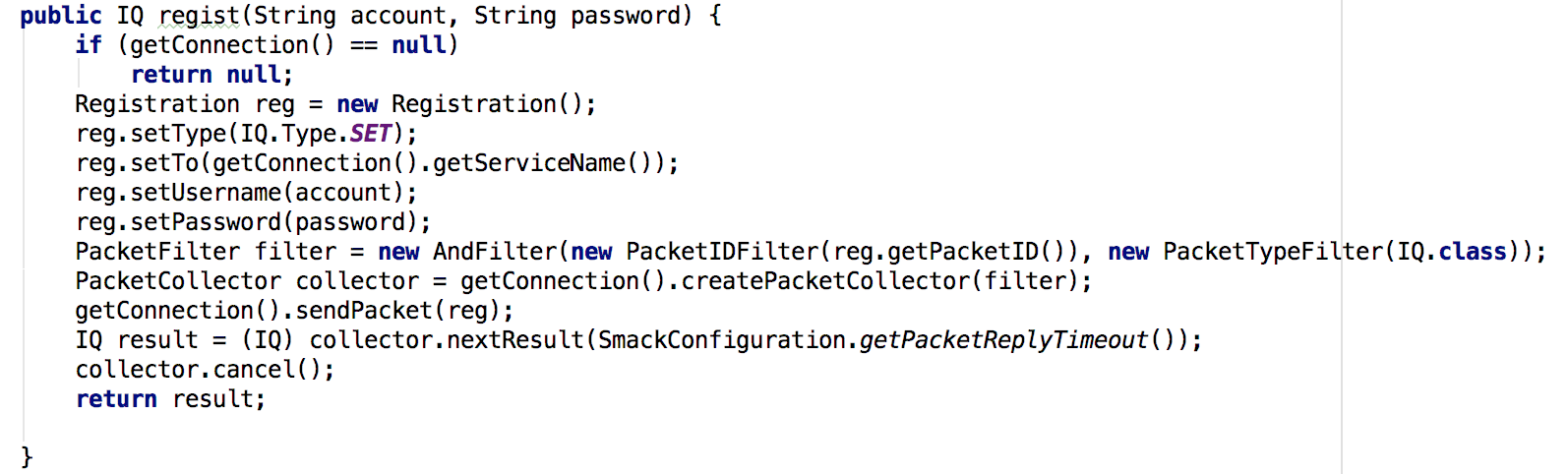


图4.5用户注册的代码实现

### 4.2.3 用户登录

在注册完后，就会有一个账号，在登录页输入此账号密码，然后点击下方的“登录”按钮。如果账号或密码为空，客户端会先判断一下，让用户填写完整，这个过程不会和服务端进行交互。全部填写后，账号密码会被发送给服务端，服务器会到对应的表中查找是否有该用户，然后返回响应。客户端根据响应判断有没有登录成功。如果成功，进入首页，否则提示“用户和密码不存在”。图4.6是用户登录的代码：

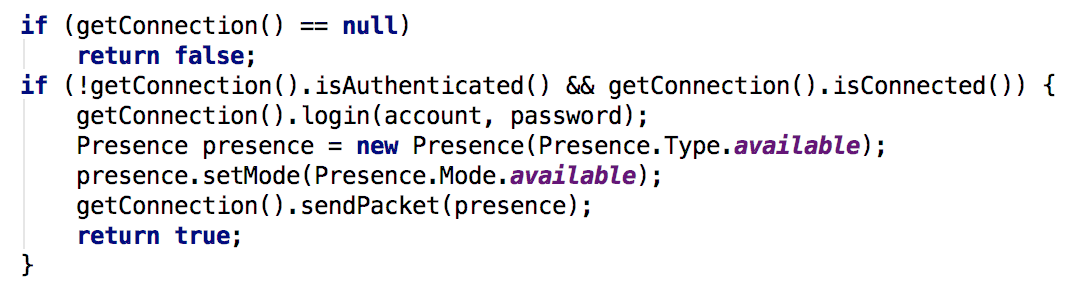


图4.6用户登录的代码实现

### 4.2.4 好友管理

好友管理总共有3个模块：好友列表，添加好友和删除好友。用户登录后，点击底部的通信录菜单，会获取到自己所有的好友并展示出来。想添加其他人成为好友，只需要点击“添加好友”按钮，进入搜索界面，输入用户名，会出现相关的好友列表，点击自己想要添加的人，发送成为好友请求，等待对方同意时，两个用户才能成为好友。删除好友时，只需要点击“移除通信录”按钮就可以删除对方。好友管理在Openfire中是通过带着不同信息的Presence包来实现的，Presence包的包含7种信息：好友请求、同意加好友、取消好友、拒绝好友请求、下线、探测、错误。通过发送这些不同的信息，就能达到不同效果。

（1）好友列表：用户在登录后，可查看自己的好友列表。这主要采用了getRoster()这个方法,通过这个方法，可以获得一个用户的所有好友以及申请加好友的用户列表。

（2）添加好友：用户首先要通过搜索获取好友列表，在输入关键字后，会向服务端发送请求，安装了search插件的服务端，会根据该请求返回列表。



图4.7搜索好友的代码实现

Asmack中有一个Roster对象，这个对象是用来保存好友的名册。用户将想要添加好友的信息保存在这个对象中，然后发送给服务器。服务器根据Roster对象中的信息，向想要添加的好友发送好友申请请求，并会返回一个响应。

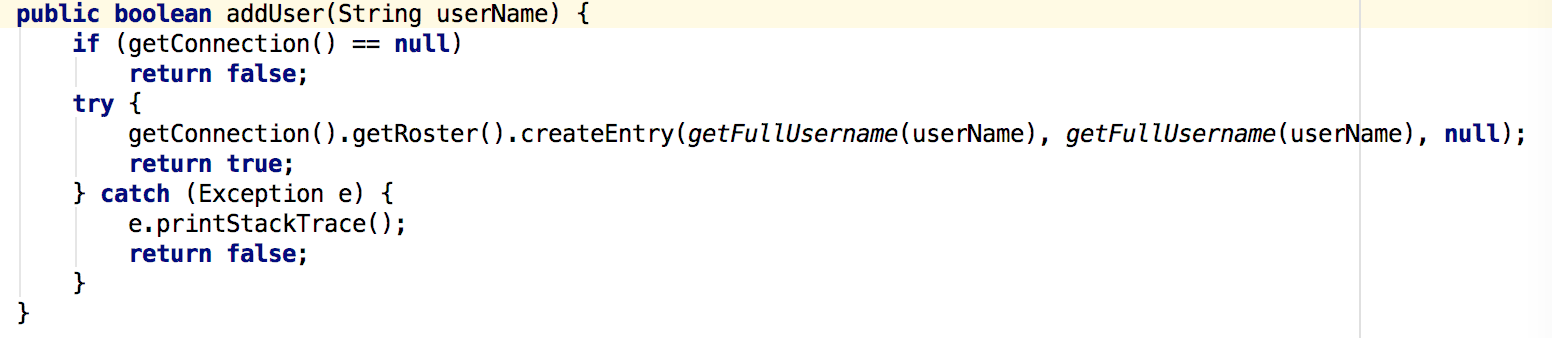


图4.8添加好友的代码实现

（3）删除好友：点击“移除通信录”这个按钮时，客户端会向服务端发送一个Presence包，里面存放着unsubscribe取消订阅的状态消息。然后服务端收到这个包时，会解析这个包中的相关信息，在相应的表中删除相关消息，返回IQ响应。

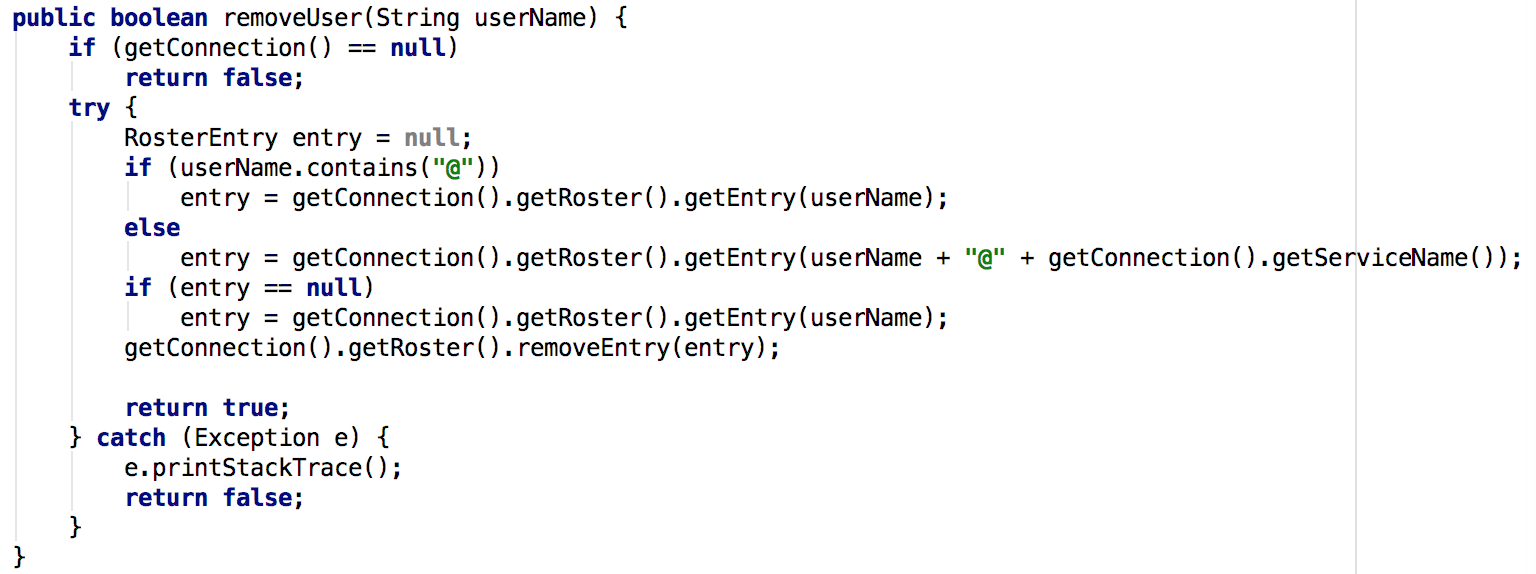


图4.9删除好友的代码实现

### 4.2.5消息传递

消息的发送主要是通过sendMessage()方法实现的。客户端将信息保存在chat对象中，然后通过sendMessage方法发送给服务器，服务器从中获得要发送的好友地址，发送给这个好友。图4.10展示了发送消息的代码实现



图4.10发送消息的代码实现

接收消息的实现主要是通过Asmack中自带的事件监听器来监听服务器，一旦发现服务器传来的关于好友消息的信息，触发器则会被触发。

# 第五章 功能测试

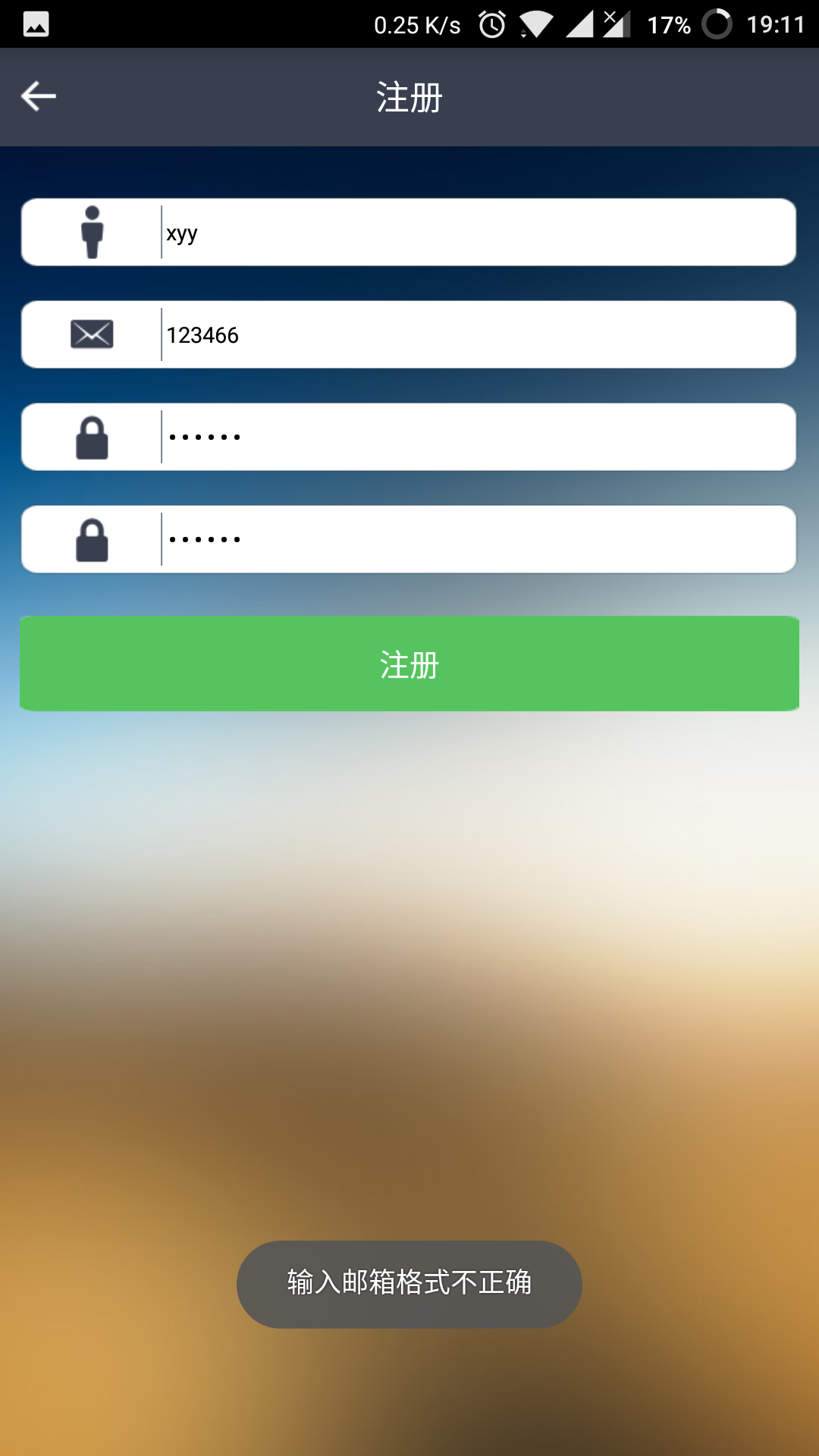
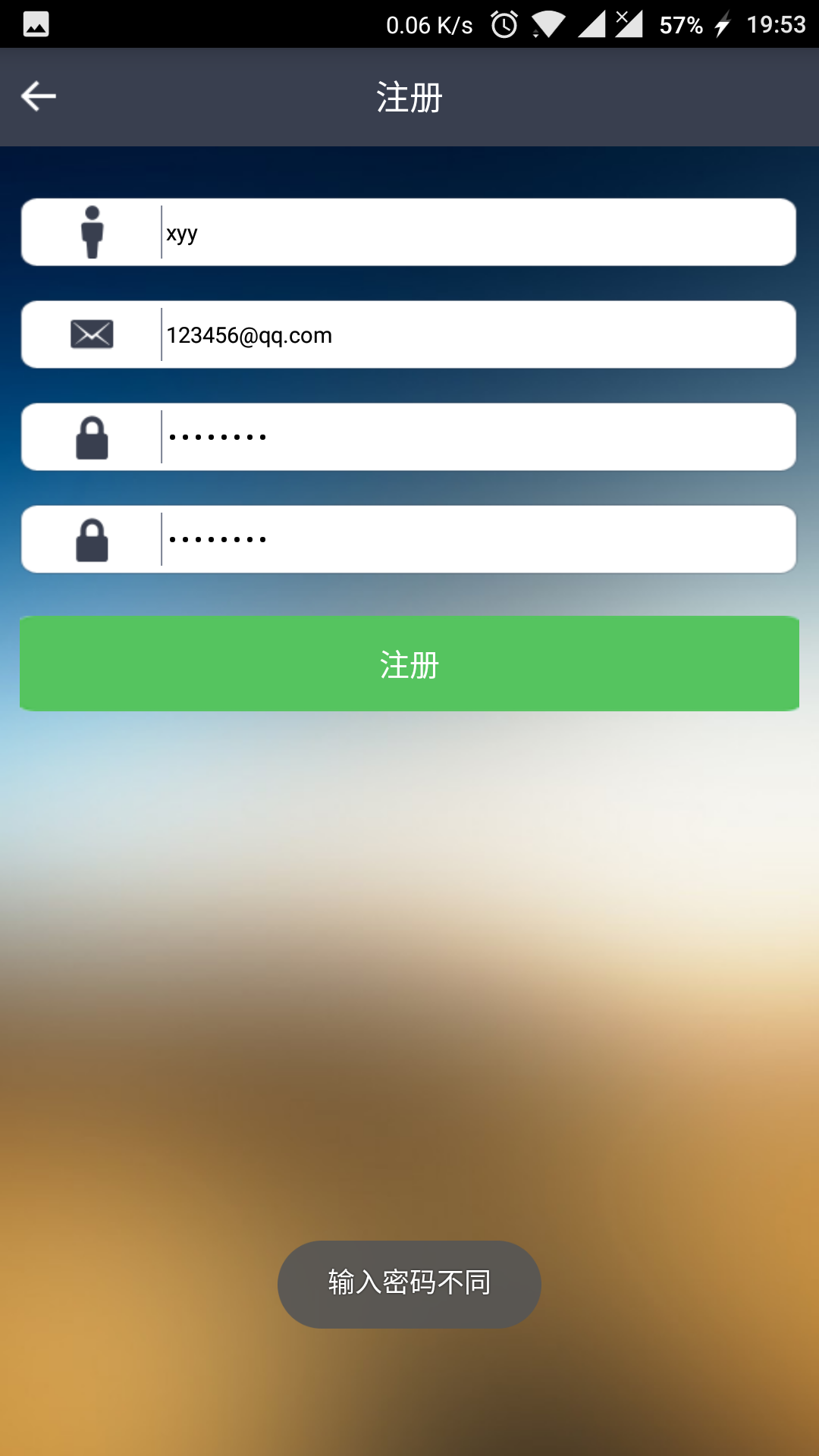
在开发过程中，使用java语言进行开发，运用Android Studio进行开发，客户端使用基于XMPP实现的Asmack包来实现XMPP的各项功能，服务器则选择了开源的Openfire服务器。最后，实现了可以进行用户登录注册，好友管理，消息发送与接收的Android客户端。下面进行了相关测试：

## 5.1用户注册测试

用户需要填写账号，邮箱，2次密码进行注册账号。客户端会判断字段是否为空，邮箱格式，以及两次密码是否一致。这个功能有四个测试用例：测试用例a是注册界面；测试用例b是当有字段为空时，会提醒输入该字段；测试用例c是会测试邮箱的格式是否正确；测试用例d是当两次密码不同时会报错。如果上述问题都不存在，则会注册成功。

a b

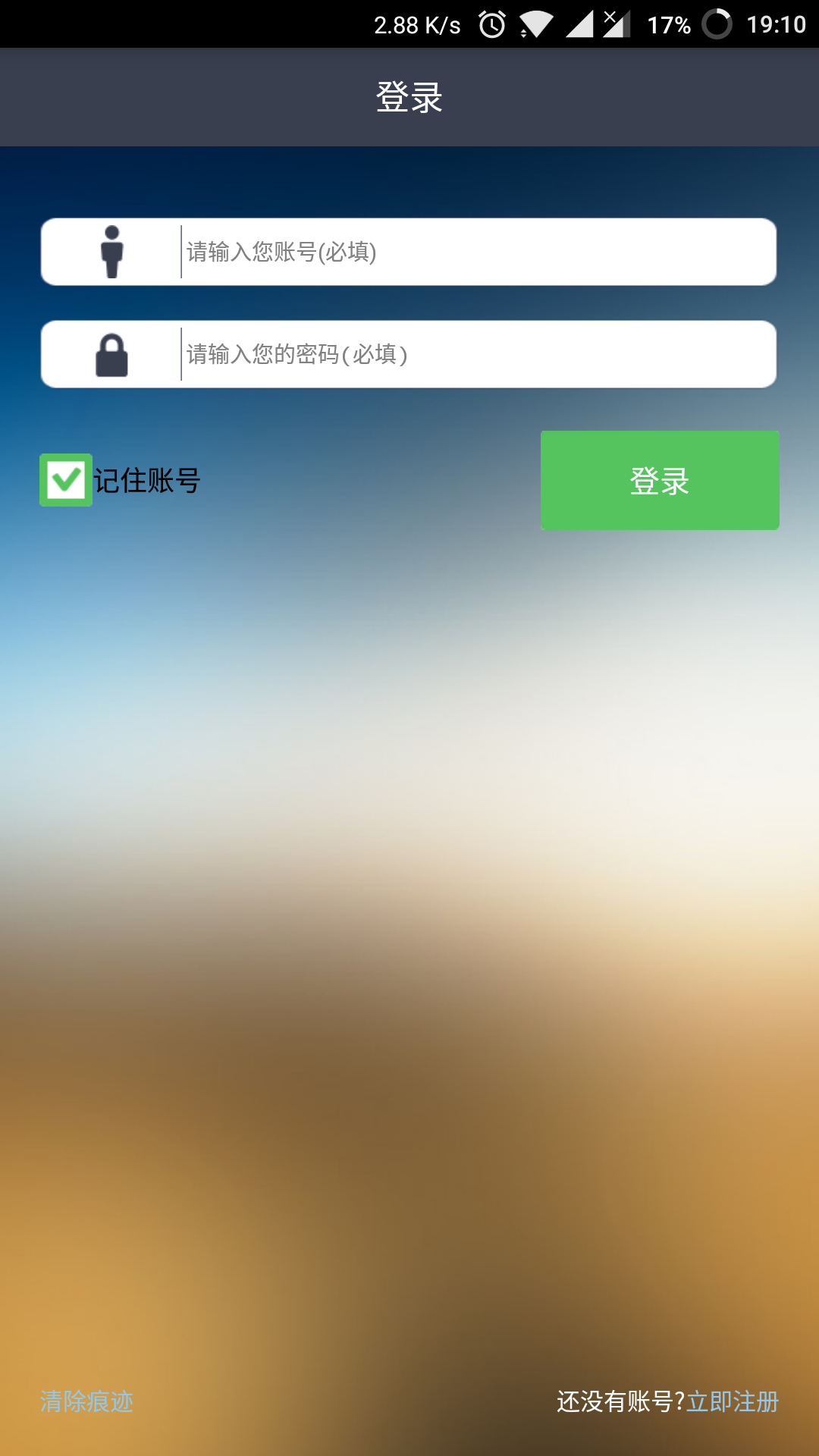
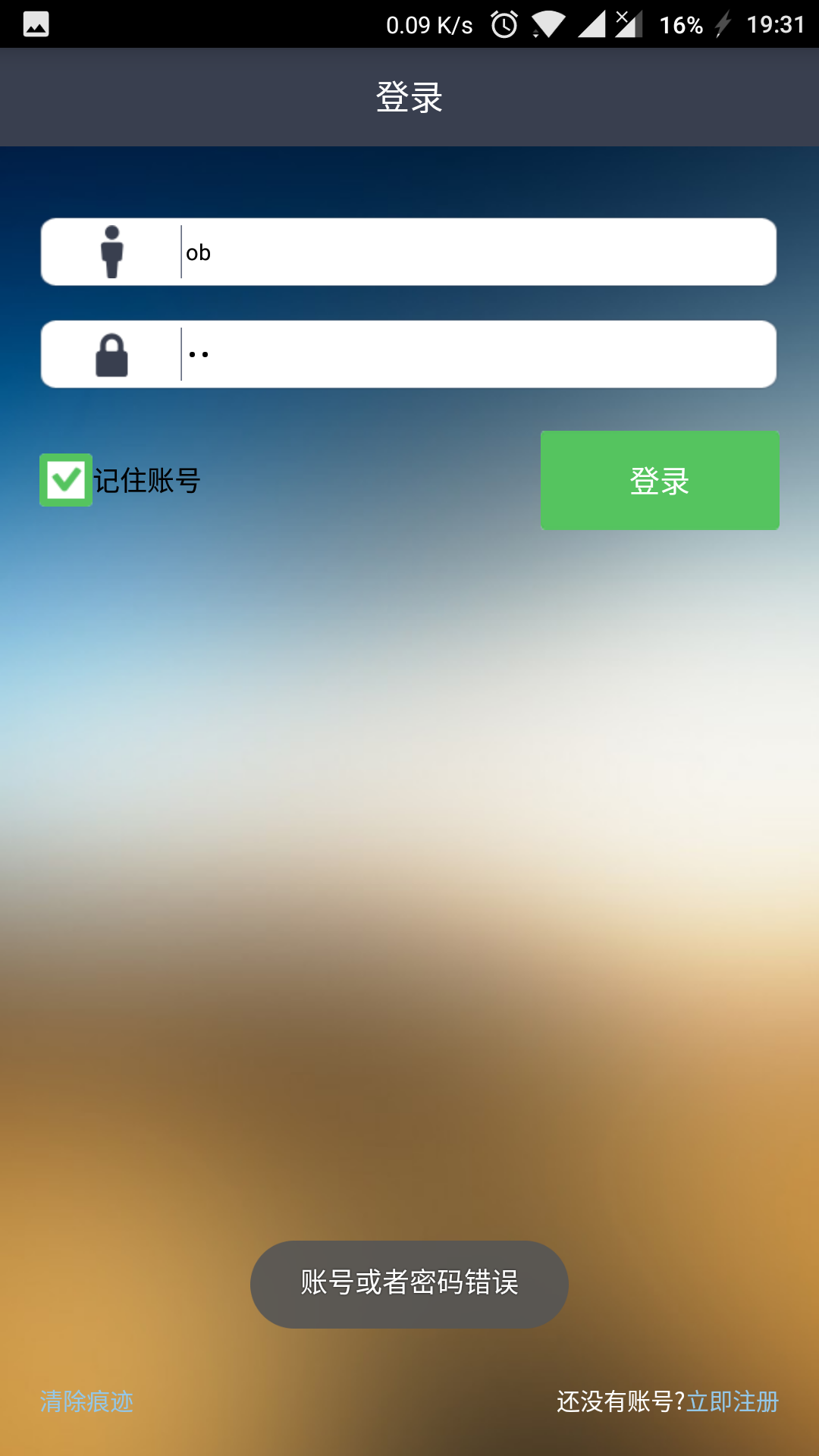
 

c d

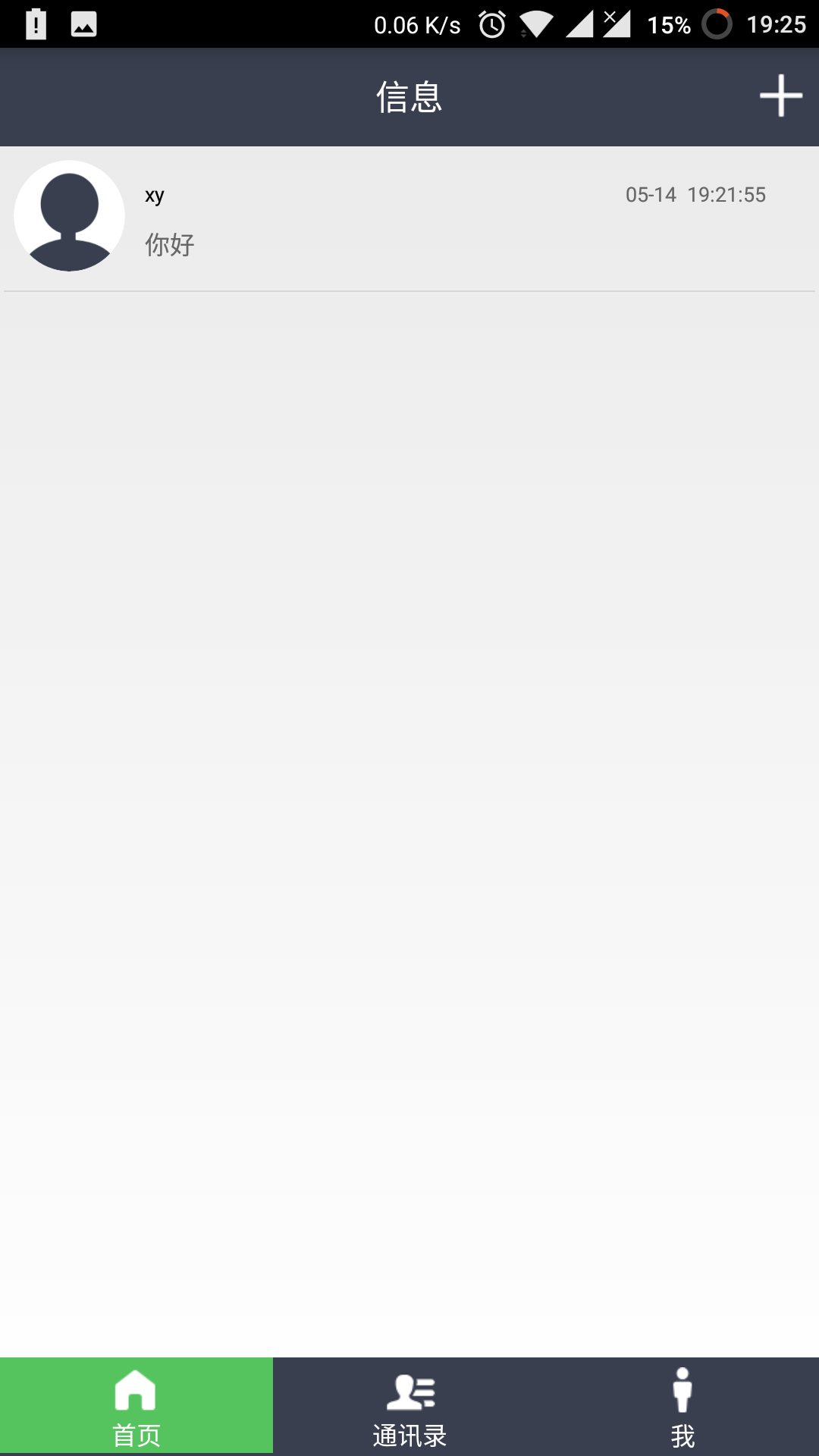
图5.1注册功能演示

## 5.2用户登录测试

用户拥有账号后，可以在登录页中，输入账号和密码。如果正确，则会进入系统，否则提示“账号和密码错误”。该部分有三个测试用例：测试用例a是进入到的登录界面；测试用例b是当账号和密码输错时，会提示；测试用例c是登录成功后，进到的界面。

a b

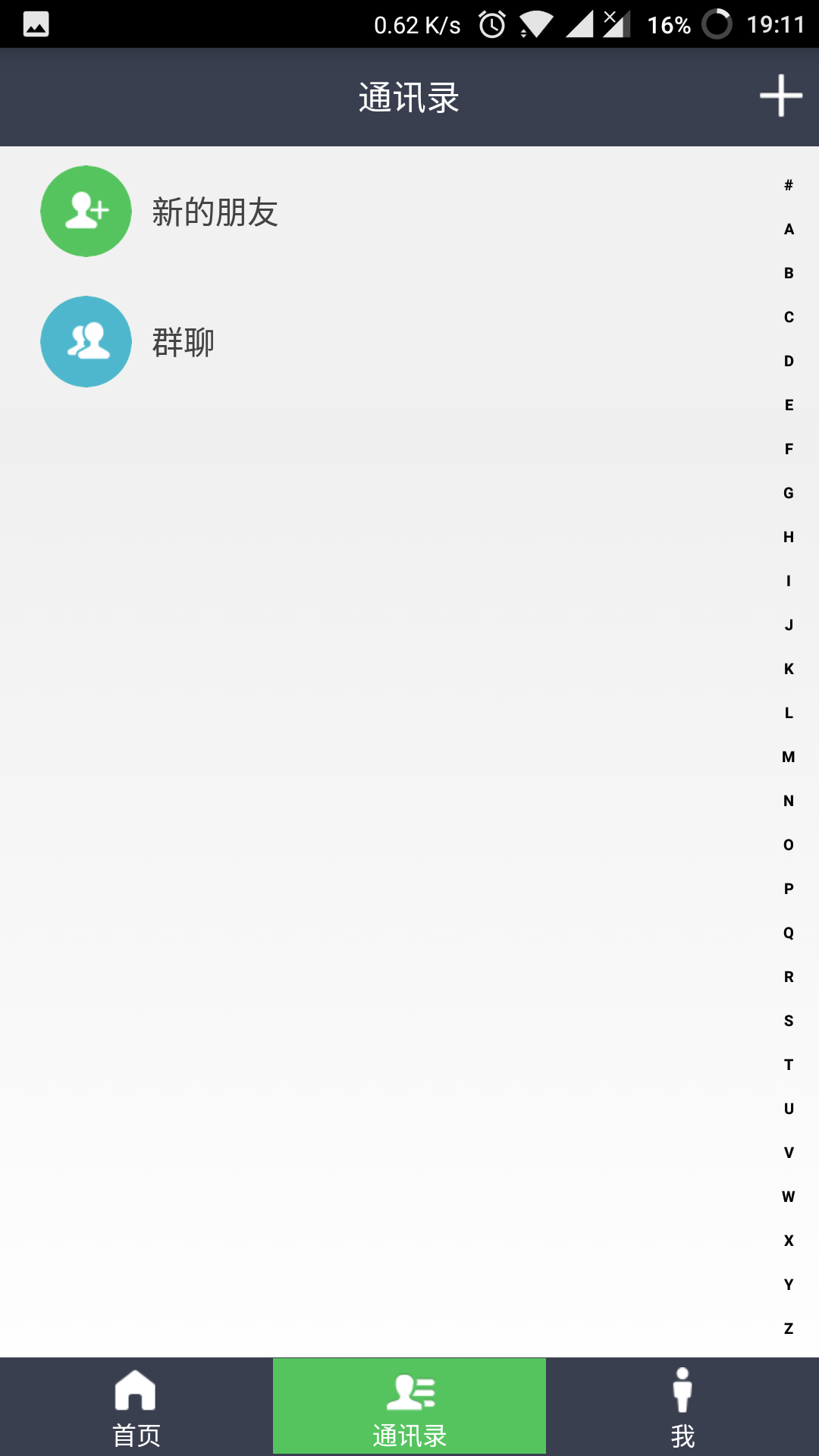
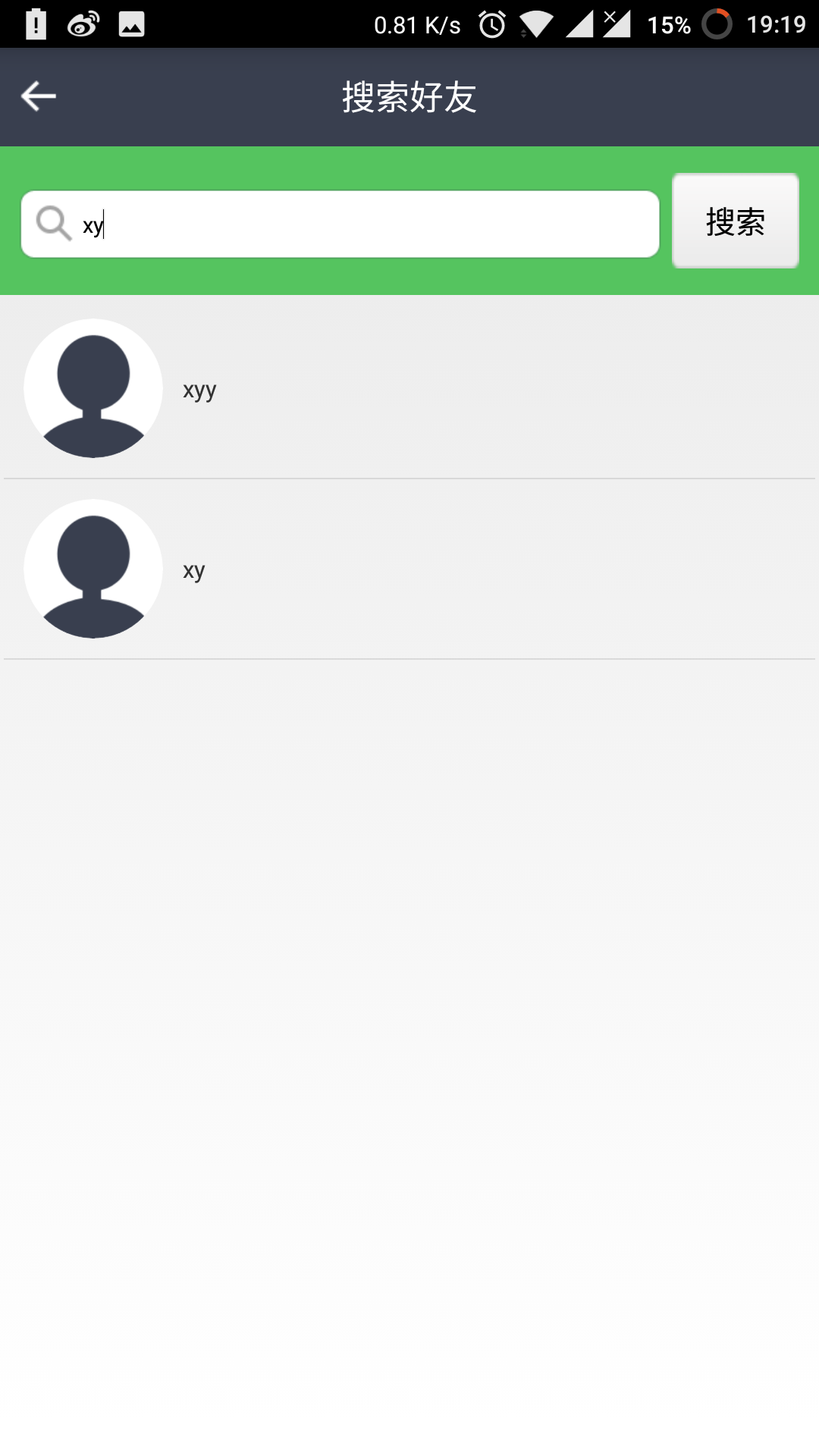


c

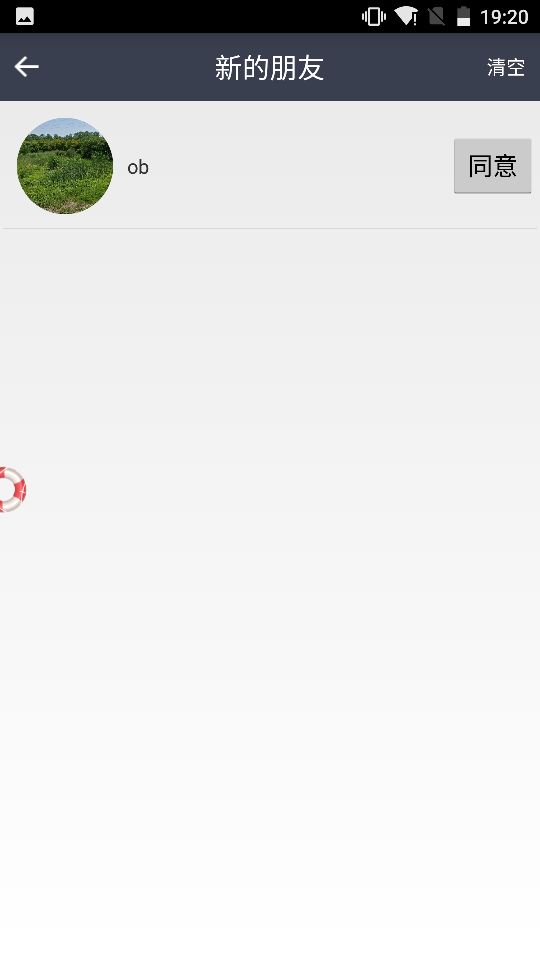
图5.2登录功能演示

## 5.3好友测试

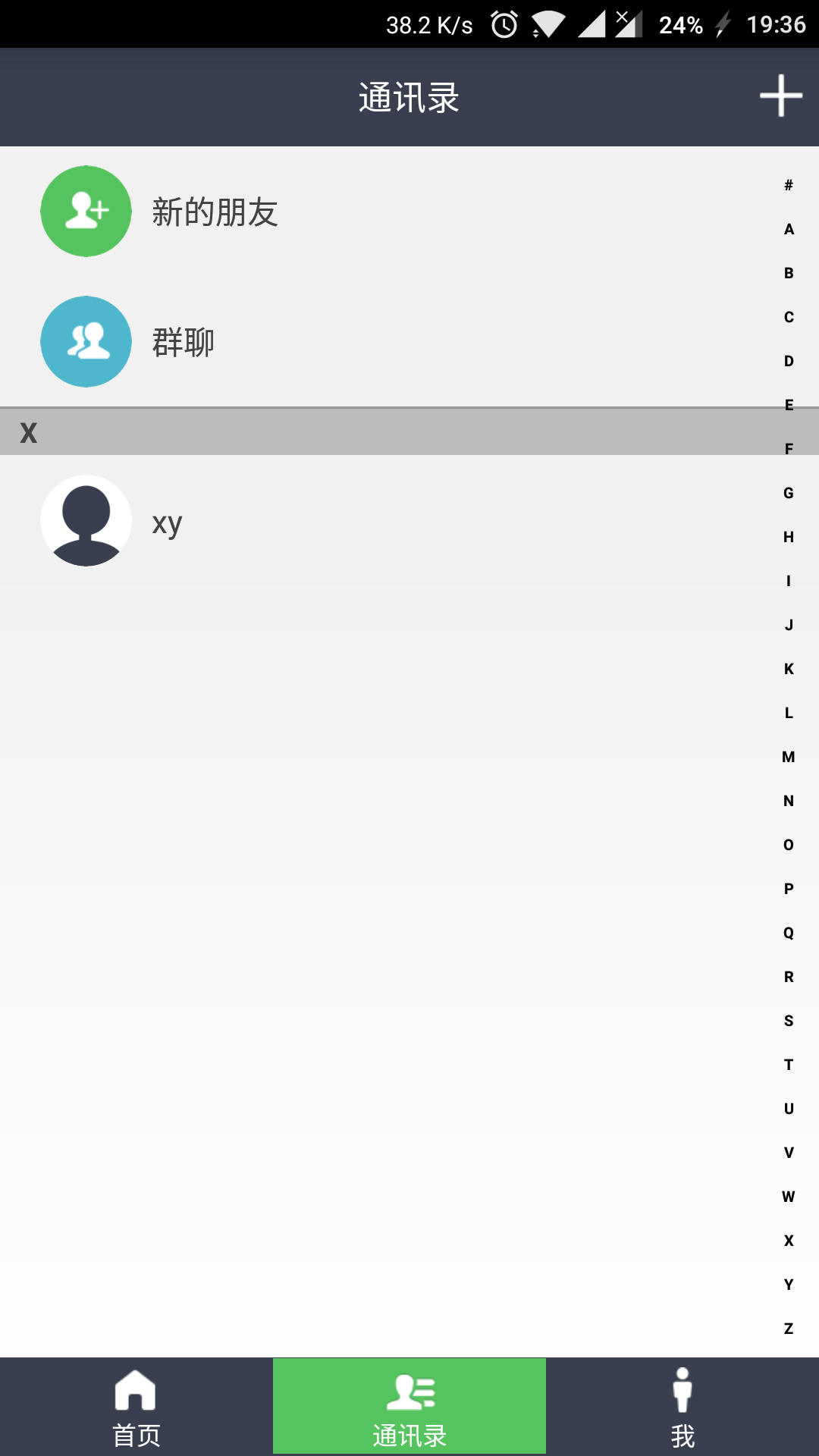
好友功能有三大功能：好友添加，好友列表和删除好友功能。用户可以在通信录页，通过输入要加的好友名，点击添加到通信录。单方面的添加并没有作用，只有当自己想加的一方接受了你的请求，这两个人才能成为好友。总共有六个测试用例a,b,c,d,e,f，分别是搜索好友界面，添加好友界面，同意添加好友界面，通信录列表界面和删除界面。

a b

c d

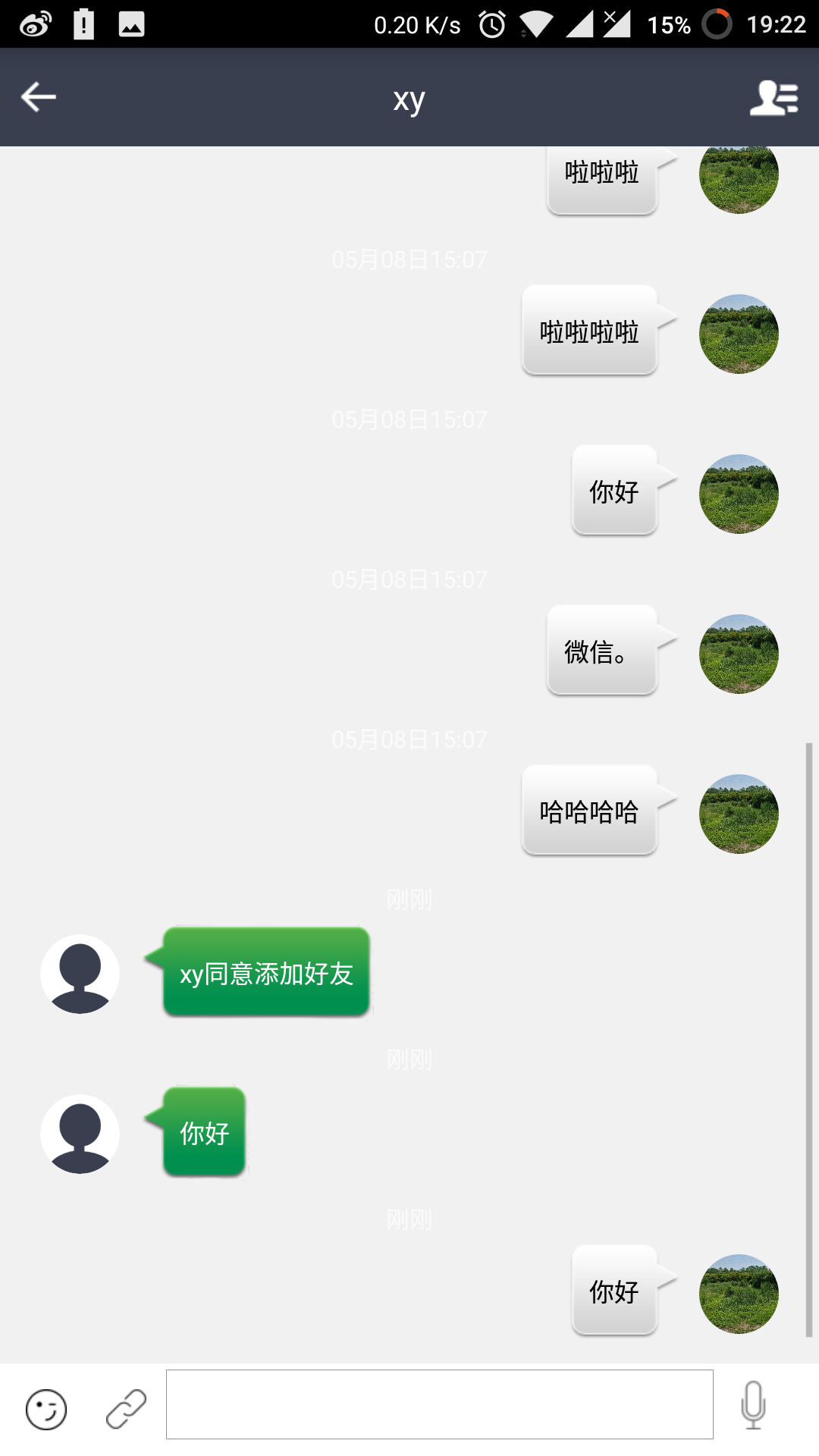
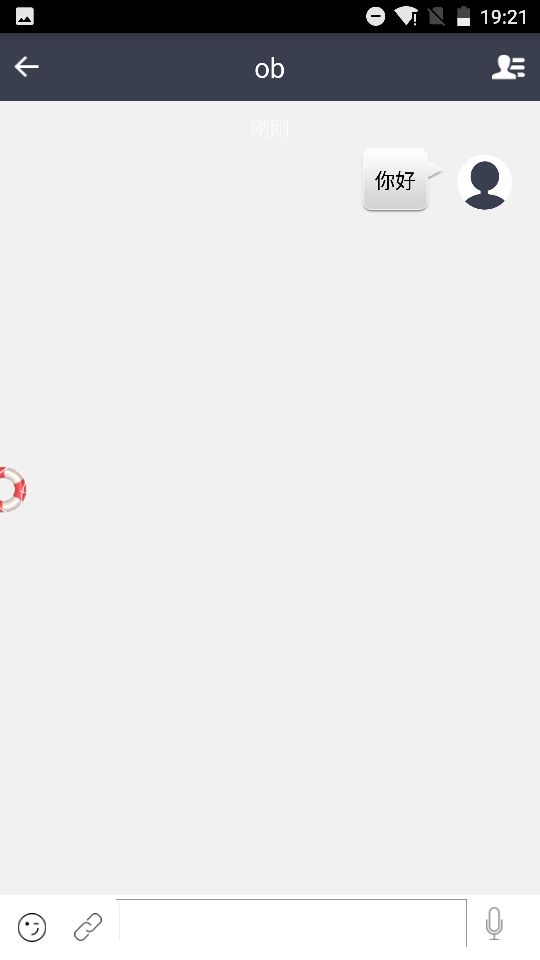
 

e f

图5.3好友管理功能演示

## 5.4聊天测试

两人成为好友后，就可以对话了，可以发送文字，图片和语音等。有两个测试用例a,b：分别是两个人的聊天。

a b

图5.4聊天功能演示

# 第六章 总结与展望

## 6.1总结

本论文主要描述了如何在现在流行的移动安卓平台上，运用XMPP的知识原理，完成一个允许用户进行实时聊天的手机应用。

该软件是运行在Android平台上的，采用开源的Openfire作为服务器的，手机端采用Asmack开源库实现与其的信息交互，完成相应的功能。

我进行了如下工作：

（1）分析了如今即时通信的意义背景和优势，并对XMPP的内容和格式进行了深入了解。

（2）选择开源的openfire作为自己的服务器，通过配置安装搭建起来，并了解了openfire的通讯方式，为以后客户端与服务器通讯打下基础。

（3）完成客户端的设计。界面用Android自带的控件和布局来实现，后台采用XMPP作为即时通讯协议，运用基于XMPP的开源库Asmack包来实现用户注册，用户登录，好友管理和消息转发等各项功能。

（4）进行该软件的测试工作，测试相关的功能模块，找出不足和bug。

## 6.2展望

如今，随着移动网络的快速发展，基于Android的即时通信软件有越来越大的前景，在中国这个人口众多的国家，更应如此。本文讲述了自己研究和实现这个手机应用的全过程。但由于时间有限，以及自己的水平问题，该系统还存在着一些问题和缺陷，需要深入分析探索，下面列举了一些还存在的问题或不足。

（1）由于数据都是经过服务器进行转发和管理，一但同一时间请求过多，服务器的负载会有很大压力。之后可以利用集群方式搭建服务器，降低服务器压力，来进行改善这个问题。

（2）该系统的通信功能还不完善，仅有一些基础功能，后期可添加实时视频通话等功能来完善。

（3）添加黑名单功能，虽然能删除好友，但对方还能在删除好友后发送添加好友请求。可以添加黑名单功能防止骚扰。

# 结束语

本文着重描述了如何在现在流行的移动安卓平台上，运用XMPP的知识原理，完成一个允许用户进行实时聊天的手机应用。在进行这些操作之前，我对即时通信的意义和作用进行了深入了解，对当今通信软件的现状进行了分析，了解到了如今存在的共性问题。XMPP协议的出现，有助于生成统一的标准，实现不同软件间的互联互通。在此基础上，我开始对完成该软件所需的知识进行了深入了解，并对这个软件进行了需求分析工作。借助开源的Openfire服务器，利用基于XMPP的开源库Asmack实现了客户端与服务器之间的联通。然后，实现了用户注册登录，聊天等各种功能。最后，进行了测试工作，找出了不足和Bug。如今，系统已经大体完成，虽还有着一些不足和改进，但我已从中体验到了成功的喜悦。希望，之后的自己也能再接再厉，一步一步稳稳当当地向前进发。

# 致 谢

该论文是在我的导师李养群的悉心指导下完成的。从选择论文题目，到如今论文的完成，李老师起到了不可磨灭的作用。不仅在课题选择后帮助我了解了完成该软件所需的知识，而且还时不时地询问我毕设的进度，让我清晰地了解了下一步所需做的工作。李老师那细致的思想，广博的知识面，平易近人的人生态度，深深影响着我。

此外，还要感谢敬业的老师们。谢谢你们不求回报地教导我们，让我们在以后的工作中懂得了怎么面对自己的任务。感谢大学四年一直带领我的萌导，在生活，学习，活动等多个方面帮助着我。在此，我想对你说声辛苦了。

最后，感谢陪伴了我四年的大学同学。在我犯错时，包容着我；在我失落时，鼓励着我；在我困境时，帮助着我……感谢你们在我的大学生活中画下了浓墨重彩的一划，让我的大学五彩缤纷，永远成为了我的美好回忆。

时光荏苒，大学时光已到了尽头。在这四年的时光里，无论是老师，还是同学，都给了我巨大的帮助。在这里我只想对你们说：谢谢你们!

# 参考文献

[1] 李昊. 基于XMPP协议的Android即时通信软件的设计与实现[D].东华理工大学,2016.

[2] 朱姝. 基于XMPP协议的Android手机即时通讯软件研究与实现[D].南京邮电大学,2015.

[3] 周士雄. 基于XMPP协议的移动平台即时通讯系统的设计与实现[D].哈尔滨工业大学,2013.

[4] 袁宾奇. 基于XMPP的跨平台即时通讯软件库的设计与实现[D].南京大学,2013.

[5] 高明鹏. 基于XMPP协议的Android手机即时通信应用研究与实现[D].南昌大学,2012.

[6] 黄伟敏. 基于XMPP协议的Android即时通信系统设计[J]. 电子设计工程,2011,(08):57-59.

[7] 周文琼,王乐球,周桐,周春光. 基于XMPP的企业即时通信系统研究与应用[J]. 吉林大学学报(信息科学版),2010,(01):106-111.

[8] 付莎. 基于XMPP协议企业级IM的研究与实现[D].西南交通大学,2009.

[9] 施济瑜,苗放,王华军,李刚. 基于XMPP协议文件传输的研究与实现[J]. 计算机测量与控制,2009,(04):732-733+741.

[10]陈武. 基于XMPP的企业级即时通信系统的研究与实现[D].武汉理工大学,2009.

[11]剧忻,苗放. 基于MINA开发高性能网络应用程序——以实现XMPP协议Openfire3.3.3为例[J]. 重庆工学院学报(自然科学版),2008,(10):121-125+160.

[12] J. N. G. K. Kumar, M. Madasamy, M. M. Bathusha and V. Sivakumar, "Efficient file sharing system using XMPP (Let's Share)," 2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, India, 2016, pp. 1589-1593.

[13] A. Stanik and O. Kao, "A proposal for REST with XMPP as base protocol for intercloud communication," 2016 7th International Conference on Information, Intelligence, Systems & Applications (IISA), Chalkidiki, Greece, 2016, pp. 1-6.

[14] A. Veichtlbauer, M. Parfant, O. Langthaler, F. P. Andrén and T. Strasser, "Evaluating XMPP communication in IEC 61499-based distributed energy applications," 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), Berlin, 2016, pp. 1-8.

[15] P. Membrey and Y. Demchenko, "Intercloud Control and Management Plane with XMPP," 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC), Limassol, 2015, pp. 471-476.