目录

1 引言

电子邮件作为人们沟通交流的主要工具，在网络中有着广泛的应用。邮件系统的架构可分为邮件传输代理 MTA、邮件投递代理 MDA 和邮件用户代理 MUA 。邮件用户代理是一个发信和收信的程序，负责将电子邮件发送到 SMTP 服务器或者从邮件服务器取回收到的邮件。常用的邮件用户代理有微软的 OUTLOOK、腾讯的 FOXMAIL 等，其可以从遵循 POP3 协议的邮件服务器中收取邮件。

UDP、TCP/IP 等相关网络协议，以及应用程序网络协议的设计。

本设计以计算机网络课程为背景，帮助学生熟悉邮件服务器服务端和客户端设计原理，掌握 SOCKET 网络编程以及应用层网络协议的设计方法，训练 PHP 和 Android 移动操作系统 APK 的开发能力。本说明书旨在介绍基于 POP3 的邮件服务端和移动客户端（安卓系统）的设计需求，设计方法和环境介绍。

2 项目概述

2.1 项目简介

项目是基于 POP3 和 SMTP 的邮件服务端和移动客户端（安卓系统）的设计，设计一个邮件

服务器和一个移动端（安卓系统）的邮件客户端，服务器端除了提供最基本的收发邮件功能之外，还应具有注册新用户、管理用户、群发邮件以及修改服务器相关参数、修改管理员密码、邮件和 IP 地址过滤等功能。客户端分为

普通用户端和管理员端。普通用户端可实现基本的注册、收发邮件，修改个人资料等功能；管理员端主要实现群发邮件功能，除此之外，它还可以实现浏览用户信息以及删除用户等操作。

2.2 项目任务

2.2.1 项目名称

基于 SMTP 和 POP3 协议的邮件服务端和移动客户端设计

2.2.2 项目内容

1）基于 SMTP 的邮件发送服务器设计与实现;

2）基于 POP3 的邮件接收服务器设计与实现;

3）移动 Android 客户端平台设计与实现。

2.2.3 项目周期

2.5 个月（2020 年 3 月 15 日——2020 年 5 月 31 日）

2.2.4 参与人数

4 人

2.3 实验环境介绍

1）python 编程语言

python 是一种用处广泛的脚本语言。Python 具有脚本语言中最丰富和强大的类库，足以支持绝大多数日常应用。 Python 语法简捷而清晰，具有丰富和强大的类库。

2) 移动端 APP 开发环境

Android Studio 是谷歌推出的一个 Android 集成开发工具，基于 IntelliJ IDEA. 类似 Eclipse ADT，Android Studio 提供了集成的 Android 开发工具用于开发和调试。

3) MySQL

MySQL 是一个关系型数据库管理系统，由瑞典 MySQL AB 公司开发，目前属于 Oracle 公司。Mysql 是最流行的关系型数据库管理系统，在 WEB 应用方面 MySQL 是最好的 RDBMS(Relational Database Management System：关系数据库管理系统)应用软件之一。MySQL 是一种关联数据库管理系统，关联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。搭配 PHP 和 Apache 可组成良好的开发环境。

3 项目设计

3.1 协议介绍

3.1.1 POP3

POP (Post Office Protocol)邮局通讯协议 POP 是互联网上的一种通讯协议，主要功能是用在传送电子邮件,当我们寄信给另外一个人时，对方当时多半不会在线上,所以邮件服务器必须为收信者保存这封信,直到收信者来检查这封信件。当收信人收信的时候，可以通过 POP 通讯协议取得邮件。目前主要是采用 POP3 协议。

3.1.2 SMTP

SMTP(Simple Mail Transfer Protocol)简易邮件传输通讯协议 SMTP 是互联网上的一种通讯协议，主要功能是用在邮件服务器之间传送电子邮件。

3.2 系统组成

3.2.1 服务器总体设计

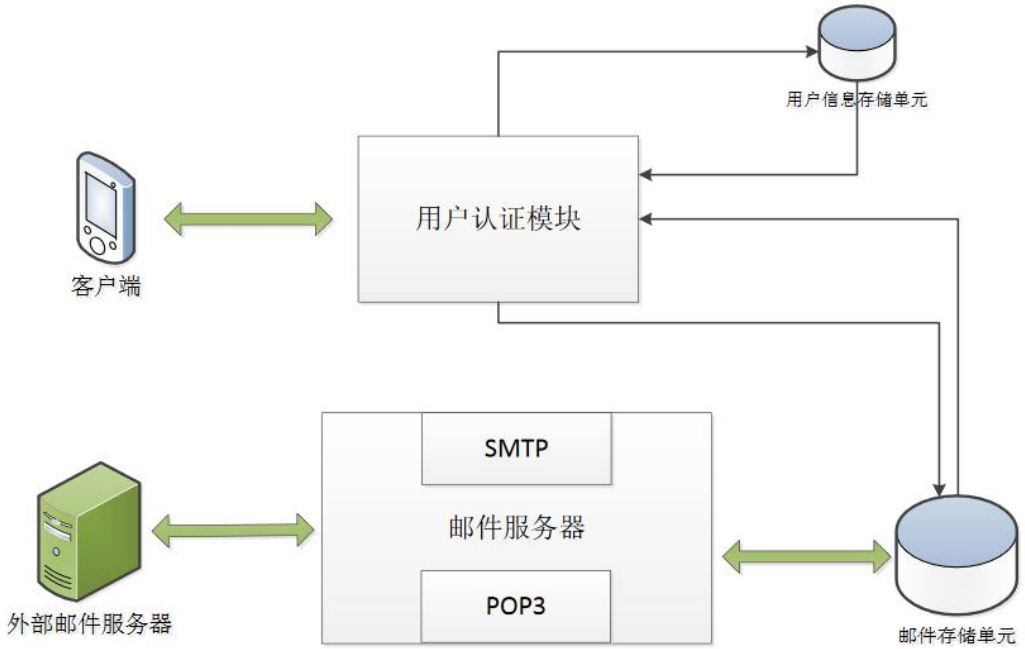


图 1 系统示意图

邮件收发服务器使用 python 语言中的邮件服务模块开发，邮件存储单元使用 MYSQL 数据库开发。移动端 Android 客户端模块使用 Android Studio 开发。

邮件收发系统根据功能划分，可分为三个子系统：

服务器管理模块：

a) 服务器参数设置；

b) Admin 对 User 的创建、授权和消权

c) Admin 和 User 对邮箱账号的创建、修改、管理及群发邮件

邮件传输模块：

a) 完成客户端与服务器、服务器与服务器之间的收发信操作

b） 使用 SMTP（发信）、POP3（收信）

移动端 Android 客户端模块：

a) 邮件的发送、接收和删除

b） 用户的基本信息管理

3.2.2 系统模块设计

1) 服务器端：

邮箱管理

主要是设置邮箱的大小。

客户的管理

主要实现在服务器端创建新的客户账号和密码，还包括对创建的新用户的权限的设置（既是否具有管理员的职能），是否对该用户禁用等，还实现删除客户账号等功能。

服务的起停

包括对 SMTP 服务、POP3 服务的起停，这是可选择的起停，通过它可以对客户端有选择的进行服务，包括对服务器的起停。

系统设置

包括设置服务器中 SMTP 端口（默认 25）、POP3 端口（默认 110）、服务器的域名设置（默认 test.com），管理员还可在这重新设置密码。还有对邮件的过滤，可实现账号的过滤和 IP 地址的过滤。

日志管理

对服务器 SMTP 日志、POP3 日志的查看和清除，及日志文件的存储位置、日志文件的大小的管理等。

日常管理

主要是邮件的群发功能，可方便发送通知。

帮助

对管理员提供服务器的使用帮助。

2)客户端：

主要由 JAVA MAIL 实现邮件的收发。

邮件的操作

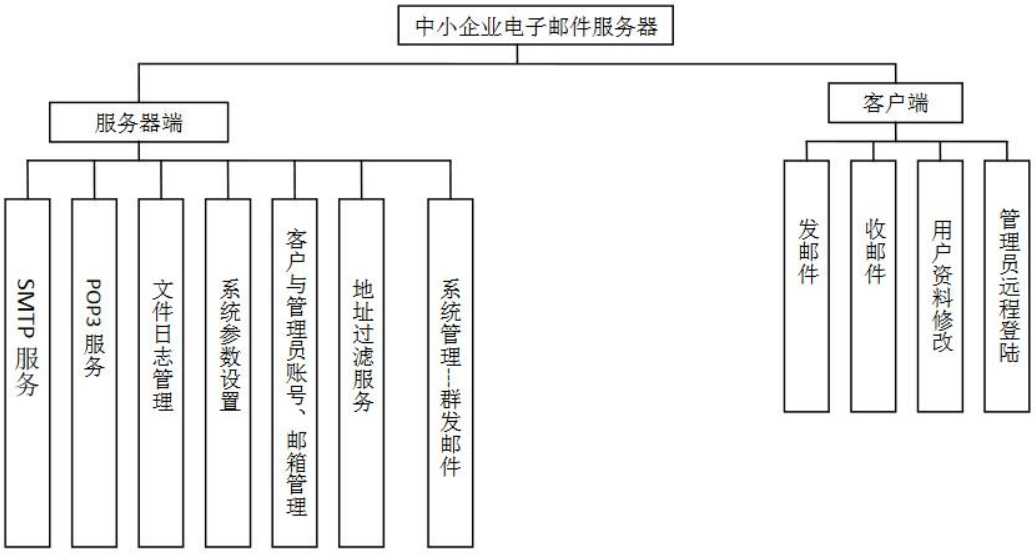
邮件的发送、接收和删除

用户管理

用户可以在此修改自己邮箱的账户密码，还提供新用户的注册功能。

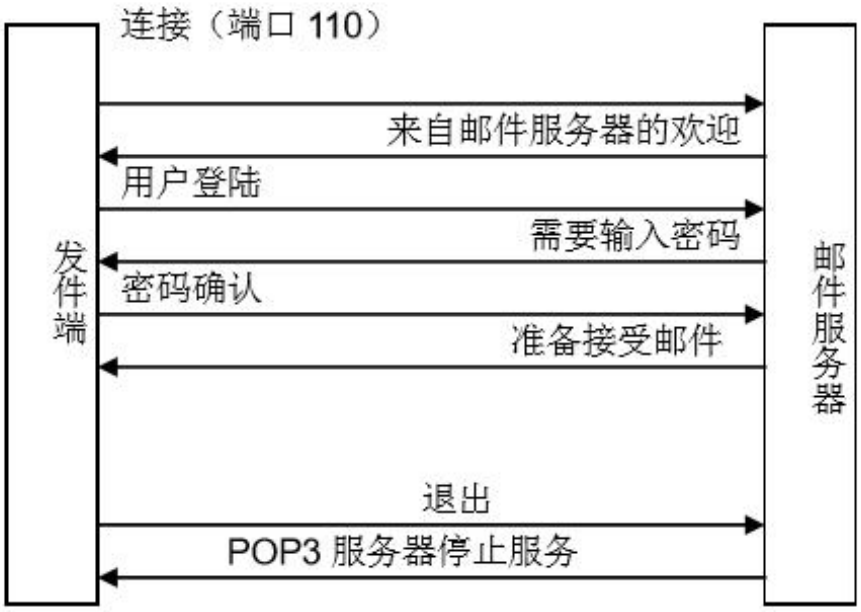
管理员管理

在客户端可以提供管理员远程登陆对服务器进行管理。管理功能同服务器端的对客户账号的创建、删除、授权、消权、禁用等。



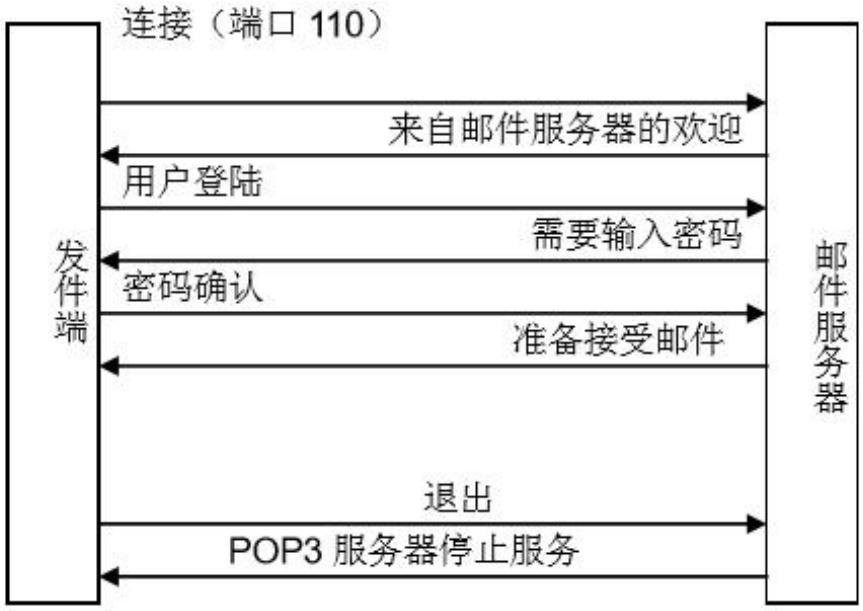
3.2.3 系统数据流图

（1）用户认证流程:

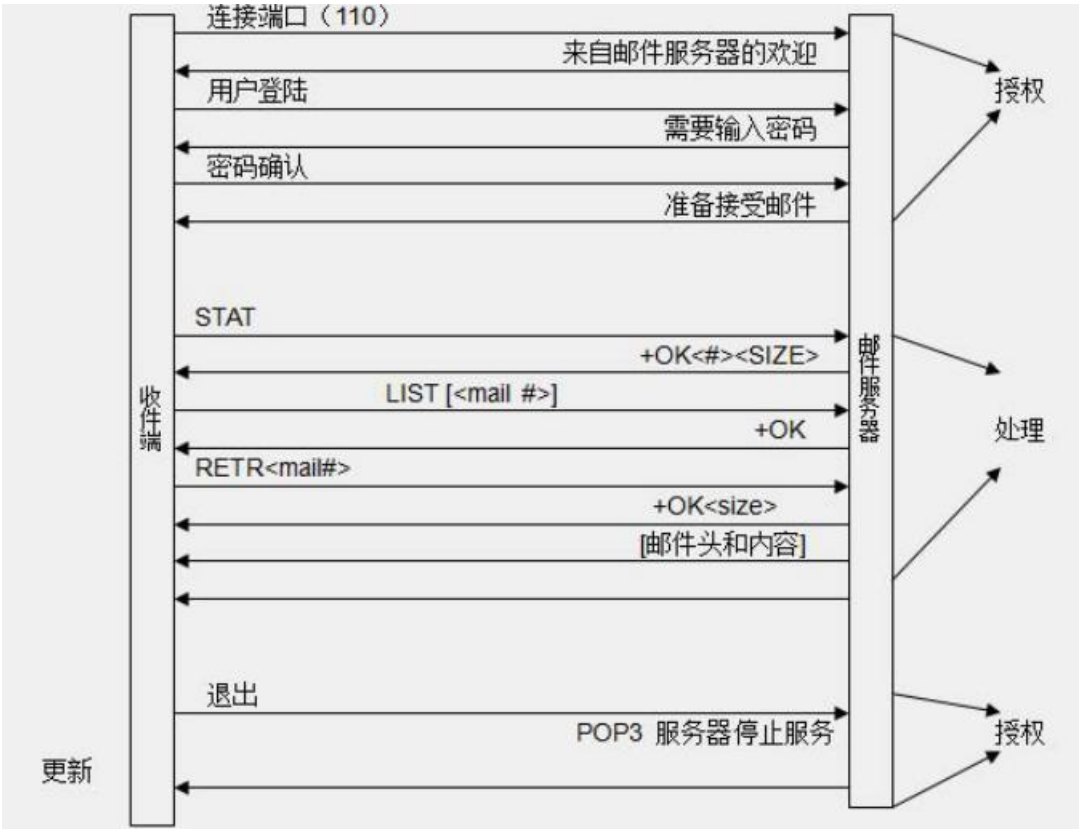


2 POP3 传送流程与用户认证流程

(1) 用户认证流程:



(2) POP3 传送流程:


连接端口（110）
来自邮件服务器的欢迎
用户登陆
需要输入密码
密码确认
准备接受邮件
STAT
+OK<#><SIZE>
LIST [<mail #>]
+OK
RETR<mail#>
+OK<size>
[邮件头和内容]
退出
POP3 服务器停止服务
更新
收件端
邮件服务器
授权
处理
授权

3.3 移动端 **Android** 客户端模块

3.3.1 用户注册

```
private void initCommit(){
    registerRegister.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String number=registerNumber.getText().toString().trim();
              if(!(number.endsWith("@163.com")||number.endsWith("@qq.com"))){
                  runOnUiThread(new Runnable() {
                      @Override
                      public void run() {
                          Toast.makeText(RegisterActivity.this, "邮箱格式错误", Toast.LENGTH_SHORT).show();
                      }
                  });
              }
            String str[]=number.split( regex: "@");
            Log.d( tag: "receive",  msg: "receiveMsg:" + str[0]+" "+str[1]);
            String password = registerPassword.getText().toString().trim();
            new Thread(new Runnable() {
                @Override
                public void run() {
                    try {
                        //步骤一
                        Socket socket=new Socket(HOST,PORT);
                        socket.setSoTimeout(60000);
                        OutputStreamWriter outputStreamWriter=new OutputStreamWriter(   //步骤二
                                socket.getOutputStream(),  charsetName: "UTF-8");
                        BufferedInputStream bufferedInputStream=new BufferedInputStream(socket.getInputStream());
                        String send="";
                        if(str[1].contains("163.com")) {
                            send="register "+str[0]+" "+password+" 59000";
                        }else if(str[1].contains("qq.com")){
                            send="register "+str[0]+" "+password+" 58000";
                        }else {
                            runOnUiThread(new Runnable() {
                                @Override
                                public void run() {
```

```
                                    Toast.makeText( context: RegisterActivity.this,  text: "账号输入错误", Toast.LENGTH_SHORT).show();
                                }
                            });
                        }
                        Log.d( tag: "receive",  msg: "receiveMsg:" + send);
                        outputStreamWriter.write(send);
                        outputStreamWriter.flush();
                        int ch = -1;
                        StringBuilder sb = new StringBuilder();
                        while (true) {
                            try {
                                ch = bufferedInputStream.read();
                                if (ch == '!'||ch == 10 || ch == -1 || ch == '\n') {
                                    break;
                                }
                                if (ch == 13)
                                    continue;
                                sb.append((char) ch);
                            } catch (Exception e) {
                                System.out.println("error");
                            }
                        }
                        String receiveMsg = sb.toString();
                        Log.d( tag: "receive",  msg: "receiveMsg:" +receiveMsg);
                        if(receiveMsg.contains("OK : register succeed")){
                            runOnUiThread(new Runnable() {
                                @Override
                                public void run() {
                                    Toast.makeText( context: RegisterActivity.this,  text: "账号注册成功，请返回进行登录", Toast.LENGTH_SHORT).show();
                                }
                            });
                        }else{
                            Toast.makeText( context: RegisterActivity.this,  text: "账号注册错误", Toast.LENGTH_SHORT).show();
```

```
                    }else{
                        Toast.makeText( context: RegisterActivity.this,  text: "账号注册错误", Toast.LENGTH_SHORT).show();
                    }

                    outputStreamWriter.close();
                    bufferedInputStream.close();
                    socket.close();
                } catch (Exception e) {
                    Log.e( tag: "connectService", ("connectService:" + e.getMessage()));   //如果Socket对象获取失败，即连接建立失败，会走到这段逻辑
                }
            }
        }).start();
    }
  });
 }
}
```

3.3.2 用户登录

```
private void doit(){
    newThread=new Thread(new Runnable(){
        public void run(){
            id = m_id.getText().toString().trim();
            pwd = m_pwd.getText().toString().trim();
            boolean flag = DBLogin.login(id,pwd);
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    if(flag==true){
                        Intent intent = new Intent(M_login.this,RecyclerViewActivity.class);
                        startActivity(intent);
                    }
                    else
                        Toast.makeText(M_login.this,"请填写正确的用户名和密码",Toast.LENGTH_SHORT).show();
                }
            });
        }
    });
}
```

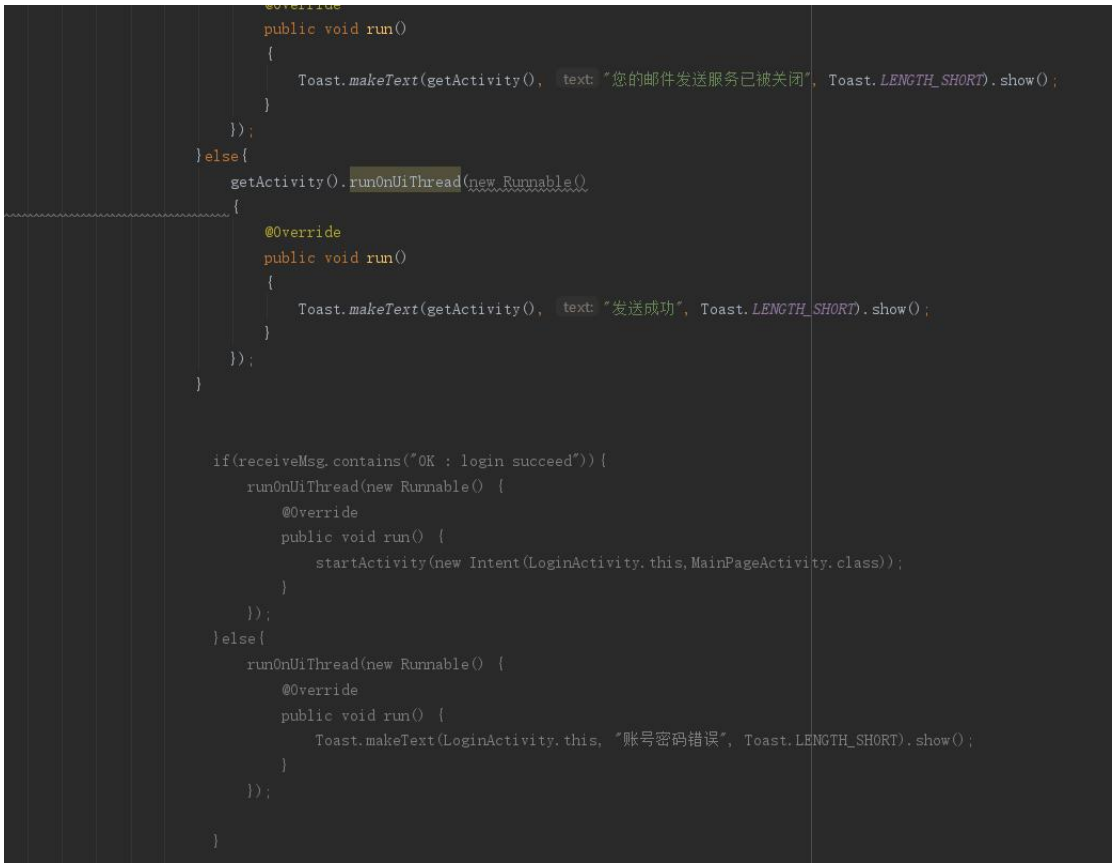
3.3.3 用户邮件发送接收（写邮件，接收邮件，邮件详情获取）

写邮件

```
private void initGet(){
    send.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String tos=to.getText().toString().trim();
            String subjects=subject.getText().toString().trim();
            String contenta=content.getText().toString().trim();
            String number = sharedPreferences.getString( key: "email", defValue: "");
            String password = sharedPreferences.getString( key: "password", defValue: "");
            String str[]=number.split( regex: "@");
            Log.d( tag: "receive",  msg: "receiveMsg:" + str[0]+" "+str[1]);
            int PORT;
            if(str[1].contains("qq.com")){
                PORT=PORT1;
            }else{
                PORT=PORT2;
            }
            new Thread(new Runnable() {
                @Override
                public void run() {
                    try {
                        //步骤一

                        Socket socket=new Socket(HOST,PORT);
                        socket.setSoTimeout(60000);
                        OutputStreamWriter outputStreamWriter=new OutputStreamWriter(   //步骤二
                                socket.getOutputStream(),  charsetName: "UTF-8");
                        BufferedInputStream bufferedInputStream=new BufferedInputStream(socket.getInputStream());
                        String receiveMsg="";
                        String[] send=new String[10];
                        send[0]="EHLO";
                        send[1]="AUTH LOGIN";
                        send[2]=str[0];
                        send[3]=password;
```

```
send[2]=str[0];
send[3]=password;
send[4]="MAIL FROM: "+number;
send[5]="RCPT TO: "+tos;
send[6]="SUBJECT: "+subjects;
send[7]="DATA";
send[8]=contenta;
send[9]=".";
for(int i=0;i<send.length;i++){
    Log.d( tag: "receive",  msg: "receiveMsg:" + send[i]);
    outputStreamWriter.write(send[i]);
    outputStreamWriter.flush();
    int ch = -1;
    StringBuilder sb = new StringBuilder();
    while (true) {
        try {
            ch = bufferedInputStream.read();
            if (ch == '!') {
                break;
            }
            if (ch == 13)
                continue;
            sb.append((char) ch);
        } catch (Exception e) {
            System.out.println("error");
        }
    }
    receiveMsg = sb.toString();
    Log.d( tag: "receive",  msg: "receiveMsg:" +receiveMsg);
}
if(receiveMsg.contains("502 Error: command not i")){
    getActivity().runOnUiThread(new Runnable()
    {
        @Override
        public void run()
```



```
                @Override
                public void run()
                {
                    Toast.makeText(getActivity(), text: "您的邮件发送服务已被关闭", Toast.LENGTH_SHORT).show();
                }
            });
        }else{
            getActivity().runOnUiThread(new Runnable()
            {
                @Override
                public void run()
                {
                    Toast.makeText(getActivity(), text: "发送成功", Toast.LENGTH_SHORT).show();
                }
            });
        }


        if(receiveMsg.contains("OK : login succeed")){
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    startActivity(new Intent(LoginActivity.this,MainPageActivity.class));
                }
            });
        }else{
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    Toast.makeText(LoginActivity.this, "账号密码错误", Toast.LENGTH_SHORT).show();
                }
            });

        }
```

接收邮件

```
private void initGet(){
            String number = sharedPreferences.getString( key: "email", defValue: "");
            String password = sharedPreferences.getString( key: "password", defValue: "");
            String str[]=number.split( regex: "@");
            Log.d( tag: "receive",  msg: "receiveMsg:" + str[0]+" "+str[1]);
            int PORT;
            if(str[1].contains("qq.com")){
                PORT=PORT1;
            }else{
                PORT=PORT2;
            }
            new Thread(new Runnable() {
                @Override
                public void run() {
                    try {
                        //步骤一

                        Socket socket=new Socket(HOST,PORT);
                        socket.setSoTimeout(60000);
                        OutputStreamWriter outputStreamWriter=new OutputStreamWriter(   //步骤二
                                socket.getOutputStream(),  charsetName: "UTF-8");
                        BufferedInputStream bufferedInputStream=new BufferedInputStream(socket.getInputStream());
                        String receiveMsg="";
                        String[] send=new String[3];
                        send[0]="USER "+str[0];
                        send[1]="PASS "+password;
                        send[2]="LIST";
                        for(int i=0;i<send.length;i++){
                            Log.d( tag: "receive",  msg: "receiveMsg:" + send[i]);
                            outputStreamWriter.write(send[i]);
                            outputStreamWriter.flush();
                            int ch = -1;
                            StringBuilder sb = new StringBuilder();
                            while (true) {
```

```
                    StringBuilder sb = new StringBuilder();
                    while (true) {
                        try {
                            ch = bufferedInputStream.read();
                            if (ch == '!') {
                                break;
                            }
                            if (ch == 13)
                                continue;
                            sb.append((char) ch);
                        } catch (Exception e) {
                            System.out.println("error");
                        }
                    }
                    receiveMsg = sb.toString();
                    Log.d( tag: "receive",  msg: "receiveMsg:" +receiveMsg);
                }
                List<Item> itemLi=new ArrayList<>();
                String[] recstr=receiveMsg.split( regex: "\n");
                for(int i=0;i<recstr.length-1;i++) {
                    Log.d( tag: "receive",  msg: "receiveMsg:" + recstr[i]);
                    itemLi.add(new Item( read: false, recstr[i]));
                    Log.d( tag: "receive",  msg: "receiveMsg:" + itemLi.get(i).content);
                }
                if(receiveMsg.contains("-ERR Unknown comman")){
                    getActivity().runOnUiThread(new Runnable()
{
                        @Override
                        public void run()
                        {
                            Toast.makeText(getActivity(),  text: "您的邮件接收服务已被关闭", Toast.LENGTH_SHORT).show();
                        }
                    });
                }else{
                    getActivity().runOnUiThread(new Runnable()
```

```
                        itemList.clear();
                        itemList.addAll(itemLi);
                    myAdapter.replaceData(itemList);
                      myAdapter.notifyDataSetChanged();
                    }
                });
            }

              if(receiveMsg.contains("OK : login succeed")){
                  runOnUiThread(new Runnable() {
                      @Override
                      public void run() {
                          startActivity(new Intent(LoginActivity.this,MainPageActivity.class));
                      }
                  });
              }else{
                  runOnUiThread(new Runnable() {
                      @Override
                      public void run() {
                          Toast.makeText(LoginActivity.this, "账号密码错误", Toast.LENGTH_SHORT).show();
                      }
                  });

              }

            outputStreamWriter.close();
            bufferedInputStream.close();
            socket.close();
        } catch (Exception e) {
            Log.e( tag: "connectService", ("connectService:" + e.getMessage()));    //如果Socket对象获取失败，即连接建立失败，会走到这段逻辑
        }
    }
}).start();
}
```

邮件详情获取

```
@Override
protected void init(){
    super.init();

    textView=findViewById(R.id.activity_detail_content);
    imageView=findViewById(R.id.activity_detail_return);
    button=findViewById(R.id.activity_detail_delete);

    sharedPreferences= MyApplication.getContext().getSharedPreferences( name: "config", Context.MODE_PRIVATE);

    imageView.setOnClickListener(v -> startActivity(new Intent( packageContext: this,MainPageActivity.class)));

    String number = sharedPreferences.getString( key: "email", defValue: "");
    String password = sharedPreferences.getString( key: "password", defValue: "");
    String str[]=number.split( regex: "@");
    Log.d( tag: "receive",  msg: "receiveMsg:" + str[0]+" "+str[1]);
    int PORT;
    if(str[1].contains("qq.com")){
        PORT=PORT1;
    }else{
        PORT=PORT2;
    }
    int id = getIntent().getIntExtra( name: "id", defaultValue: 0);
    Log.d( tag: "receive",  msg: "receiveMsg:" +id);
    new Thread(new Runnable() {
        @Override
        public void run() {
            try {
                //步骤一

                Socket socket=new Socket(HOST,PORT);
                socket.setSoTimeout(60000);
                OutputStreamWriter outputStreamWriter=new OutputStreamWriter(   //步骤二
```

```
                BufferedInputStream bufferedInputStream=new BufferedInputStream(socket.getInputStream());
                String receiveMsg="";
                String[] send=new String[3];
                send[0]="USER "+str[0];
                send[1]="PASS "+password;
                send[2]="RETR "+id;
                for(int i=0;i<send.length;i++){
                    Log.d( tag: "receive",  msg: "receiveMsg:" + send[i]);
                    outputStreamWriter.write(send[i]);
                    outputStreamWriter.flush();
                    int ch = -1;
                    StringBuilder sb = new StringBuilder();
                    while (true) {
                        try {
                            ch = bufferedInputStream.read();
                            if (ch == '!') {
                                break;
                            }
                            if (ch == 13)
                                continue;
                            sb.append((char) ch);
                        } catch (Exception e) {
                            System.out.println("error");
                        }
                    }
                    receiveMsg = sb.toString();
                    Log.d( tag: "receive",  msg: "receiveMsg:" +receiveMsg);
                }
                  List<Item> itemLi=new ArrayList<>();
                  String[] recstr=receiveMsg.split("\n");
                  for(int i=0;i<recstr.length-1;i++){
                      Log.d("receive", "receiveMsg:" +recstr[i]);
                      itemLi.add(new ClipData.Item(false,recstr[i]));
                      Log.d("receive", "receiveMsg:" +itemLi.get(i).content);
                  }
```

```
            }
            String finalReceiveMsg = receiveMsg;
            runOnUiThread(new Runnable()
            {
                @Override
                public void run()
                {
                    textView.setText(finalReceiveMsg);
                }
            });
                    if(receiveMsg.contains("OK : login succeed")){
                        runOnUiThread(new Runnable() {
                            @Override
                            public void run() {
                                startActivity(new Intent(LoginActivity.this,MainPageActivity.class));
                            }
                        });
                    }else{
                        runOnUiThread(new Runnable() {
                            @Override
                            public void run() {
                                Toast.makeText(LoginActivity.this, "账号密码错误", Toast.LENGTH_SHORT).show();
                            }
                        });

                    }

            outputStreamWriter.close();
            bufferedInputStream.close();
            socket.close();
        } catch (Exception e) {
            Log.e( tag: "connectService", ("connectService:" + e.getMessage()));    //如果Socket对象获取失败，即连接建立失败，会走到这段逻辑
        }
    }
```

3.3.4 邮件操作

```
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    //步骤一

                    Socket socket=new Socket(HOST,PORT);
                    socket.setSoTimeout(60000);
                    OutputStreamWriter outputStreamWriter=new OutputStreamWriter(   //步骤二
                            socket.getOutputStream(),  charsetName: "UTF-8");
                    BufferedInputStream bufferedInputStream=new BufferedInputStream(socket.getInputStream());
                    String receiveMsg="";
                    String[] send=new String[3];
                    send[0]="USER "+str[0];
                    send[1]="PASS "+password;
                    send[2]="DELE "+id;
                    for(int i=0;i<send.length;i++){
                        Log.d( tag: "receive",  msg: "receiveMsg:" + send[i]);
                        outputStreamWriter.write(send[i]);
                        outputStreamWriter.flush();
                        int ch = -1;
                        StringBuilder sb = new StringBuilder();
                        while (true) {
                            try {
                                ch = bufferedInputStream.read();
                                if (ch == '!') {
                                    break;
                                }
                                if (ch == 13)
```

```
                receiveMsg = sb.toString();
                Log.d( tag: "receive",  msg: "receiveMsg:" +receiveMsg);
            }
        List<Item> itemLi=new ArrayList<>();
        String[] recstr=receiveMsg.split("\n");
        for(int i=0;i<recstr.length-1;i++){
            Log.d("receive", "receiveMsg:" +recstr[i]);
            itemLi.add(new ClipData.Item(false,recstr[i]));
            Log.d("receive", "receiveMsg:" +itemLi.get(i).content);
        }
            String finalReceiveMsg = receiveMsg;
            runOnUiThread(new Runnable()
            {
                @Override
                public void run()
                {
                    startActivity(new Intent(getApplicationContext(),MainPageActivity.class));
                }
            });
              if(receiveMsg.contains("OK : login succeed")){
                  runOnUiThread(new Runnable() {
                      @Override
                      public void run() {
                          startActivity(new Intent(LoginActivity.this,MainPageActivity.class));
                      }
                  });
              }else{
                  runOnUiThread(new Runnable() {
                      @Override
                      public void run() {
                          Toast.makeText(LoginActivity.this, "账号密码错误", Toast.LENGTH_SHORT).show();
                      }
                  });
              }
            }
            outputStreamWriter.close();
            bufferedInputStream.close();
            socket.close();
        } catch (Exception e) {
            Log.e( tag: "connectService", ("connectService:" + e.getMessage()));   //如果Socket对象获取失败，即连接建立失败，会走到这段逻辑
        }
    }
}).start();
}
});
}
```

3.3.5 用户密码管理

修改密码

```
private void initCommit(){
    button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String number = sharedPreferences.getString( key: "email", defValue: "");
//            if(!(number.endsWith("@163.com")||number.endsWith("@qq.com"))){
//                runOnUiThread(new Runnable() {
//                    @Override
//                    public void run() {
//                        Toast.makeText(RegisterActivity.this, "邮箱格式错误", Toast.LENGTH_SHORT).show();
//                    }
//                });
//            }
            String str[]=number.split( regex: "@");
            Log.d( tag: "receive",  msg: "receiveMsg:" + str[0]+" "+str[1]);
            String password = editText.getText().toString().trim();
            new Thread(new Runnable() {
                @Override
                public void run() {
                    try {
                        //步骤一
                        Socket socket=new Socket(HOST,PORT);
                        socket.setSoTimeout(60000);
                        OutputStreamWriter outputStreamWriter=new OutputStreamWriter(   //步骤二
                                socket.getOutputStream(),  charsetName: "UTF-8");
                        BufferedInputStream bufferedInputStream=new BufferedInputStream(socket.getInputStream());
                        String send="update "+str[0]+" "+password;
                        Log.d( tag: "receive",  msg: "receiveMsg:" + send);
                        outputStreamWriter.write(send);
                        outputStreamWriter.flush();
                        int ch = -1;
                        StringBuilder sb = new StringBuilder();
                        while (true) {
                            try {
```

```
                        int ch = -1;
                        StringBuilder sb = new StringBuilder();
                        while (true) {
                            try {
                                ch = bufferedInputStream.read();
                                if (ch == '!'||ch == 10 || ch == -1 || ch == '\n') {
                                    break;
                                }
                                if (ch == 13)
                                    continue;
                                sb.append((char) ch);
                            } catch (Exception e) {
                                System.out.println("error");
                            }
                        }
                        String receiveMsg = sb.toString();
                        Log.d( tag: "receive",  msg: "receiveMsg:" +receiveMsg);
                        if(receiveMsg.contains("OK : update succeed")){
                            getActivity().runOnUiThread(new Runnable() {
                                @Override
                                public void run() {
                                    Toast.makeText(getActivity(),  text: "密码修改成功", Toast.LENGTH_SHORT).show();
                                }
                            });
                        }else{
                            Toast.makeText(getActivity(),  text: "密码修改失败", Toast.LENGTH_SHORT).show();
                        }

                        outputStreamWriter.close();
                        bufferedInputStream.close();
                        socket.close();
                    } catch (Exception e) {
                        Log.e( tag: "connectService", ("connectService:" + e.getMessage()));   //如果Socket对象获取失败，即连接建立失败，会走到这段逻辑
                    }
                }
```

3.3.6 管理员创建账号

```
public class RecyclerViewActivity extends AppCompatActivity {

    private RecyclerView recyclerview;
    private ArrayList<String> datas;
    private MyRecyclerViewAdapter adapter;
    private Thread newThread;
    private Button insert_user;
    private AlertDialog.Builder builder;
    private EditText et;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_recyclerview);
        recyclerview = findViewById(R.id.recyclerview);
        insert_user = findViewById(R.id.insert_user);
        builder = new AlertDialog.Builder( context: this);
        builder.setTitle("请输入用户名");
        et = new EditText( context: this);
        builder.setView(et);
        builder.setPositiveButton( text: "确定", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                newThread = new Thread(new Runnable() {
                    @Override
                    public void run() {
                        datas = new ArrayList<>(DBLogin.insertUser(et.getText().toString().trim()));
                    }
                });
```

```
    private void initRecyclerView() {
        //设置RecyclerView的适配器
        adapter = new MyRecyclerViewAdapter( context: RecyclerViewActivity.this,datas);
        recyclerview.setAdapter(adapter);

        //设置LayoutManager
        recyclerview.setLayoutManager(new LinearLayoutManager( context: RecyclerViewActivity.this, RecyclerView.VERTICAL,

        //设置某条的监听
        adapter.setOnItemClickListenner(new MyRecyclerViewAdapter.OnItemClickListenner() {
            @Override
            public void onItemClick(View view, String data) {
                Toast.makeText(RecyclerViewActivity.this,"用户"+data,Toast.LENGTH_SHORT).show();
            }
        });
    }

    private void initData() throws InterruptedException {
        //准备数据集合
        datas = new ArrayList<>();
        newThread = new Thread(new Runnable() {
            public void run() {
                datas = new ArrayList<>(DBLogin.getID());
            }
        });
        newThread.start();
        newThread.join();
    }
```

3.3.7 管理员管理用户权限

显示权限列表

```
itemView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
 //       Toast.makeText(context,"用户"+datas.get(getLayoutPosition()),Toast.LENGTH_SHORT).show();
        if(onItemClickListenner!=null){
            onItemClickListenner.onItemClick(v, datas.get(getLayoutPosition()));
        }
    }
});
```

删除用户

```
  delete.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
          newThread = new Thread(new Runnable() {
              @Override
              public void run() { DBLogin.delele_user(datas.get(getLayoutPosition())); }
          });
          newThread.start();
          try {
              newThread.join();
          } catch (InterruptedException e) {
              e.printStackTrace();
          }
          removeData(getLayoutPosition());
      }
  });
```

开启 SMTP 服务

```
on.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        newThread = new Thread(new Runnable() {
            @Override
            public void run() { DBLogin.on(datas.get(getLayoutPosition())); }
        });
        newThread.start();
        try {
            newThread.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        Toast.makeText(context,"已开启smtp服务",Toast.LENGTH_SHORT).show();
    }
});
```

关闭 SMTP 服务

```
off.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        newThread = new Thread(new Runnable() {
            @Override
            public void run() { DBLogin.off(datas.get(getLayoutPosition())); }
        });
        newThread.start();
        try {
            newThread.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        Toast.makeText(context,"已关闭smtp服务",Toast.LENGTH_SHORT).show();
    }
```

3.4 邮件传输模块

3.4.1 邮件发送流程

服务器开启之后，使用 SMTP 协议的服务器可以接受手机客户端发送的邮.件内容，并且根据 SMTP 协议获取邮件的发送人，接收人、主题、内容等保存到数据库。

由于我们只是在局域网内使用该邮件收发系统，所以 SMTP 协议的服务器只能实现存储到本地的功能。

使用 POP3 协议的服务器可以在用户上线之后，根据用户发送的命令，作出响应，比如提醒用户邮件总数，列出所有邮件详细信息等功能。

3.4.2 SMTP 协议的实现

```
#客户端到服务器的SMTP服务
def handle_client_smtp(self,client_socket,client_address):
    welcome_state = 0 #欢迎状态
    login_state = 0 #记录用户认证状态
    source_state = 0 #记录是否填写邮件双方
    subject_state = 0 #记录是否填写邮件主题
    des_state = 0 #记录是否填写邮件双方
    mail_cont = ""#记录邮件内容
    mail_subject = ""#记录邮件主题
    mail_source = "" #记录信的来源
    mail_des = "" #记录信的去处
```

sstr[]中存放的是来自客户端的请求

```
if sstr[0] == "EHLO":
    response = "250-smtp.qq.com\n" \
               "250-PIPELINING\n" \
               "250-STARTTLS\n" \
               "250-AUTH LOGIN PLAIN\n" \
               "250-AUTH=LOGIN\n"\
               "250-MAILCOMPRESS\n"\
               "250 8BITMIME!"
    welcome_state = 1    #可以进行用户认证
    client_socket.send(response.encode("utf-8"))
```

Ehlo

用 ehlo 申明，表示自己需要身份验证，如果身份有效，则服务器进入等待认证状态，主动推送自身支持的所有 SMTP 认证方式：表示其支持 LOGIN、PLAIN 两种认证方式

客户端向服务器请求认证

发送“AUTH LOGIN”，如果认证请求合理，服务器将进入等待用户输入状态，客户端向服务器发送转码后的用户名，发送经过 Base64 转码后的用户名，客户端向服务器发送转码后的密码，发送经过 Base64 转码后的密码

如果用户名或者密码出错，服务器将返回 530 错误，发送“535 auth failed”，表示认证失败。否则将返回 235，发送“235 auth successfully”，表示用户认证成功。

```
elif sstr[0] + sstr[1] == "AUTHLOGIN" and welcome_state == 1 and len(sstr) == 2:
    #用户名认证
    response = "username=!"
    # response = base64.b64encode(str.encode("utf-8"))
    client_socket.send(response.encode("utf-8"))
    recv_data = client_socket.recv(1024)
    # username = base64.b64decode(recv_data).decode("utf-8")
    username = recv_data.decode("utf-8")
    print("From:[%s,%s]" % client_address, end=" ")
    print(username)

    #用户密码认证
    response = "Password=!"
    # response = base64.b64encode(str.encode("utf-8"))
    client_socket.send(response.encode("utf-8"))
    recv_data = client_socket.recv(1024)
    # password = base64.b64decode(recv_data).decode("utf-8")
    password = recv_data.decode("utf-8")
    print("From:[%s,%s]" % client_address, end=" ")
    print(password)

    #服务校验
    state = conmysql.state_port(username)
    print(state)
    smtp_state = state[0]
    pop_state = state[1]

    #用户名密码数据库验证
    if conmysql.user_identified(username,password) == 1 and smtp_state=='1':
        response = "235 auth successfully!"#认证成功
        login_state = 1
    else:
        if conmysql.user_identified(username,password) == 1:

            response = "535 auth failed!" #认证失败
     else:
            response = "536 smtp failed!" #smtp服务被禁用
 client_socket.send(response.encode("utf-8"))
```

```
elif sstr[0] + sstr[1] == "AUTHLOGIN" and welcome_state == 1 and len(sstr) == 2:
    #用户名认证
    response = "username=!"
    # response = base64.b64encode(str.encode("utf-8"))
    client_socket.send(response.encode("utf-8"))
    recv_data = client_socket.recv(1024)
    # username = base64.b64decode(recv_data).decode("utf-8")
    username = recv_data.decode("utf-8")
    print("From:[%s,%s]" % client_address, end=" ")
    print(username)

    #用户密码认证
    response = "Password=!"
    # response = base64.b64encode(str.encode("utf-8"))
    client_socket.send(response.encode("utf-8"))
    recv_data = client_socket.recv(1024)
    # password = base64.b64decode(recv_data).decode("utf-8")
    password = recv_data.decode("utf-8")
    print("From:[%s,%s]" % client_address, end=" ")
    print(password)

    #服务校验
    state = conmysql.state_port(username)
    print(state)
    smtp_state = state[0]
    pop_state = state[1]

    #用户名密码数据库验证
    if conmysql.user_identified(username,password) == 1 and smtp_state=='1':
        response = "235 auth successfully!"#认证成功
        login_state = 1
    else:
        if conmysql.user_identified(username,password) == 1:
```

发送“DATA”:客户端告诉服务器自己准备发送邮件正文，服务器返回 354，表示自己已经作好接受邮件的准备，提醒客户端开始发送邮件并以“.”结束

```
elif sstr[0] == "DATA" and len(sstr) == 1 and des_state == 1 and source_state == 1 and subject_state == 1:
    response = "354 Enter mail,end with '.' on a line by itself!"
    client_socket.send(response.encode("utf-8"))
    while True:
        recv_data = client_socket.recv(1024)
        strr = recv_data.decode("utf-8")
        print(strr)
        if strr == '.':
            break
        else:
            reply = "continue!"
            client_socket.send(reply.encode("utf-8"))
        mail_cont += strr
        mail_cont += "$$"
```

拼接信内容

```
# 拼接信的内容
mail = {}
mail['From'] = mail_source
mail['To'] = mail_des
mail['Subject'] = mail_subject
mail['Cont'] = mail_cont
mail_cont = ' ' #刷新mail_cout 避免连续发信内容叠加
self.smtpMailDeliver(mail)
```

服务校验

```
#服务校验
state = conmysql.state_port(username)
print(state)
smtp_state = state[0]
pop_state = state[1]
```

用户名密码数据库验证

```
#用户名密码数据库验证
if conmysql.user_identified(username,password) == 1 and smtp_state=='1':
    response = "235 auth successfully!"#认证成功
    login_state = 1
else:
    if conmysql.user_identified(username,password) == 1:
        response = "535 auth failed!" #认证失败
    else:
        response = "536 smtp failed!" #smtp服务被禁用
client_socket.send(response.encode("utf-8"))
```

客户端告诉服务器邮件来自何方，发送“MAIL FROM:

<xxx@xxx.com>

```
elif sstr[0] + sstr[1] == "MAILFROM:" and len(sstr) == 3 and login_state ==1:
    mail_source = sstr[2]
    source_state = 1
    response  =  "250 ok!"
    client_socket.send(response.encode("utf-8"))
```

客户端告诉服务器邮件去往何地，发送“RCPT TO: <

xxx@xxx.com >

```
elif sstr[0] + sstr[1] == "RCPTTO:" and len(sstr) == 3 and login_state ==1:
    mail_des = sstr[2]
    des_state = 1
    response = "250 ok!"
    client_socket.send(response.encode("utf-8"))
```

关闭客户端连接

```
#关闭客户端连接
client_socket.close()
```

3.4.3 POP3 协议的实现

```
login_state = 0 #记录用户登录状态
user_name = "" #记录用户名
pass_word = "" #记录密码
```

user 命令是 POP3 客户端程序与 POP3 邮件服务器建立连接后通常发送的第一条命令，参数 username 表示收件人的帐户名称。

```
if sstr[0] == "USER" and len(sstr) == 2:
    user_name= sstr[1]
    response = "+OK"
    client_socket.send(response.encode("utf-8"))
```

pass 命令是在 user 命令成功通过后，POP3 客户端程序接着发送的命令，它用于传递帐户的密码，参数 password 表示帐户的密码。

```
elif sstr[0] == "PASS" and len(sstr) == 2:
    pass_word = sstr[1]
    #服务校验
    state = conmysql.state_port(user_name)
    print(state)
    smtp_state = state[0]
    pop_state = state[1]
    if conmysql.user_identified(user_name,pass_word) == 1 and pop_state == '1':
        #认证成功
        response = "+OK user successfully logged on"
        login_state = 1
    else:
        if conmysql.user_identified(user_name,pass_word) != 1:
        #认证失败
            response = "-ERR user identify failed"
        else :
            #pop被禁用
            response = "-ERR user pop failed"
    client_socket.send(response.encode("utf-8"))
```

// list<SP>[MSG#]<CRLF> list 命令用于列出邮箱中的邮件信息，参数 msg#是一个可选参数，表示邮件的序号。当不指定参数时，POP3 服务器列出邮箱中所有的邮件信息；当指定参数 msg#时，POP3 服务器只返回序号对应的邮件信息。

```
elif sstr[0] == "LIST" and len(sstr) == 1 and login_state == 1:
    info = conmysql.list_email(user_name+"@qq.com")
    response = ""
    for i in info:
        response += str(i[0])
        response += " "
        response += str(i[1])
        response += "\n"
    response += "."
    client_socket.send(response.encode("utf-8"))
```

retr<SP>msg#<CRLF> retr 命令用于获取某封邮件的内容，参数 msg#表示邮件的序号。

```
elif sstr[0] == "RETR" and len(sstr) == 2 and login_state == 1:
    li_id = int(sstr[1])
    cont = conmysql.cont_email(user_name+"@qq.com",li_id)
    if cont == -1:
        response = "-ERR input email id doesn't exist"
    else:
        response = cont
    client_socket.send(response.encode("utf-8"))
```

dele<SP>msg#<CRLF> dele 命令用于在某封邮件上设置删除标记，参数 msg#表示邮件的序号。POP3 服务器执行 dele 命令时，只是为邮件设置了删除标记，并没有真正把邮件删除掉，只有 POP3 客户端发出 quit 命令后，POP3 服务器才会真正删除所有设置了删除标记的邮件。

```
elif sstr[0] == "DELE" and len(sstr) == 2 and login_state == 1:
    del_id = int(sstr[1])
    conmysql.dele_mail(user_name+"@qq.com",del_id)
    response = "+OK"
    client_socket.send(response.encode("utf-8"))
```

quit<CRLF> quit 命令表示要结束邮件接收过程，POP3 服务器接收到此命令后，将删除所有设置了删除标记的邮件，并关闭与 POP3 客户端程序的网络连接。

```
elif sstr[0] == "QUIT" and len(sstr) == 1:
    response = "+OK POP3 server signing off"
    client_socket.send(response.encode("utf-8"))
    break
```

3.4.4 模拟邮件发送过程

服务器发送邮件到服务器

```
#服务器发送邮件到服务器
def smtpMailDeliver(self,mail):
    print(mail)
    mail_des = mail['To']
    hostip = "127.0.0.1"
    port = conmysql.des_port(mail_des)  # 另一个服务器程序的端口号
    port = port + 2
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    flag = 0
    while flag != 1:
        try:
            cli_port = random.randint(50000, 59999)
            client_socket.bind(("", cli_port))
            flag = 1
            print(cli_port)
        except socket.error:
            print("端口号被占用")
            flag = 0
    i = 0
    while i != 1:
        try:
            client_socket.connect((hostip, port))
            i = 1
        except socket.error:
            print("目的服务器程序未启动")
            time.sleep(1800)
    client_socket.send(str(mail).encode("utf-8"))
    client_socket.close()
```

服务器接收从服务器发送而来的邮件

```
#服务器接收从服务器发送而来的邮件
def smtpMailRecv(self,client_socket,client_address):
    mail_data = client_socket.recv(1024)
    print("From:[%s,%s]" % client_address, end=" ")
    print(mail_data.decode("utf-8"))
    mail = eval(mail_data.decode("utf-8"))
    response = "250 ok"
    client_socket.send(response.encode("utf-8"))
    #将邮件存入数据库中
    conmysql.save_mail(mail)
```

3.5 服务器管理模块

3.5.1 服务的起停

```
elif sstr[0] == "manage" and len(sstr) == 4:
    try:
        start_stop = sstr[1]
        smtp_pop = sstr[2]
        user_name = sstr[3]
        if login_state==0:
            msg = "Error : authority error!"
        else:
            msg= "OK : manage succeed!"
            if start_stop == "start":
                if smtp_pop == "smtp":
                    conmysql.start_smtp(user_name)
                else:
                    conmysql.start_pop(user_name)
            else :
                if smtp_pop == "smtp":
                    conmysql.stop_smtp(user_name)
                else:
                    conmysql.stop_pop(user_name)
    except IOError:
        msg= "Error : input error!"
```

```
#管理用户的smtp和pop3的状态码
def stop_smtp(user):
    con = connectdb()
    cursor = con.cursor()
    sql = "UPDATE user SET smtp_state = '%s' where username = '%s'"
    data = ('0',user,)
    cursor.execute(sql%data)
    con.commit()
    cursor.close()
    con.close()
    return user + " smtp服务停止"

def stop_pop(user):
    con = connectdb()
    cursor = con.cursor()
    sql = "UPDATE user SET pop_state = '%s' where username = '%s'"
    data = ('0', user,)
    cursor.execute(sql % data)
    con.commit()
    cursor.close()
    con.close()
    return user + " pop服务停止"
```

```
def start_smtp(user):
    con = connectdb()
    cursor = con.cursor()
    sql = "UPDATE user SET smtp_state = '%s' where username = '%s'"
    data = ('1', user,)
    cursor.execute(sql % data)
    con.commit()
    cursor.close()
    con.close()
    return user + " smtp服务开启"


def start_pop(user):
    con = connectdb()
    cursor = con.cursor()
    sql = "UPDATE user SET pop_state = '%s' where username = '%s'"
    data = ('1', user,)
    cursor.execute(sql % data)
    con.commit()
    cursor.close()
    con.close()
    return user + " pop服务开启"

#通过端口号查询smtp和pop3服务的启停
def state_port(user):
    con = connectdb()
    cursor = con.cursor()
    sql = "SELECT smtp_state, pop_state FROM user where username = '%s'"
    data = (user,)
    cursor.execute(sql%data)
    state = []
    for row in cursor.fetchall():
        state.append(row[0])
        state.append(row[1])
    #print(state)
    return state
```

3.5.2 管理列表

```
elif sstr[0] == "list" and len(sstr) == 1:
    try:
        if login_state==0:
            msg = "Error : authority error!"
        else:
            info = conmysql.info_userandport()
            response = ""
            for i in info:
                response += str(i[0])
                response += " "
                response += str(i[1])
                response += " "
                response += str(i[2])
                response += "\n"
            response += "."
            msg =response
    except IOError:
        msg= "Error : input error!"
client_socket.send(msg.encode("utf-8"))
```

```
#列举出用户名和服务状态
def info_userandport():
    con = connectdb()
    cursor = con.cursor()
    sql = "SELECT username,smtp_state,pop_state from user"
    cursor.execute(sql)
    info = []
    for row in cursor.fetchall():
        info.append(row)
    return info
```

3.5.3 收/发邮件

```
def connect(self,hostip):
    port = -1
    while port == -1:
        user_name = input("请输入用户名:\n")
        port = conmysql.port_find(user_name)
        if port == -1:
            print("用户名不存在，请重新输入")
    print("请选择发信还是收信服务(1.发信    2.收信)")
    choose = input()
    while choose != '1' and choose != '2':
        print("输入错误，请重新选择")
        choose = input()
    if choose == '1':
        port = port
    else:
        port = port + 1
    self.client_socket.connect((hostip,port))
    print("连接服务器成功")
```

```
#将邮件存入数据库
def save_mail(mail):
    con = connectdb()
    cursor = con.cursor()
    sql = "SELECT COUNT(*) FROM email where email_to = '%s'"
    data = (mail['To'],)
    cursor.execute(sql%data)
    num = -1
    try:
        for row in cursor.fetchall():
            num = row[0] + 1
    except IndexError:
        num = 1
    # print(num)
    mail_size = len(mail['Cont'])
    sql = "INSERT INTO email(email_no,email_from,email_to,email_subject,email_cont,time,email_size) VALUES('%d','%s','%s','%s','%s','%s','%d')"
    recv_time = time.strftime('%Y-%m-%d %H:%M:%S',time.localtime(time.time()))
    content = mail['Cont'].split("$$")
    mail_cont = ""
    for i in range(len(content)):
        mail_cont += content[i]
        if i != len(content)-1:
            mail_cont += "\n"
    data = (num,mail['From'],mail['To'],mail['Subject'],mail_cont,recv_time,mail_size)
    cursor.execute(sql%data)
    con.commit()
    cursor.close()
    con.close()
```

```python
#将邮箱中对应用户的信件提取出来
def list_email(user_name):
    info = []
    con = connectdb()
    cursor = con.cursor()
    sql = "SELECT email_no,email_subject,time FROM email where email_to = '%s'"
    data = (user_name ,)
    cursor.execute(sql%data)
    for row in cursor.fetchall():
        info.append(row)
    #print(info)
    cursor.close()
    con.close()
    return info

#读取对应参数的邮件
def cont_email(user,id):
    useremailaddr = user
    con = connectdb()
    cursor = con.cursor()
    sql = "SELECT email_cont from email where email_to = '%s' and email_no = '%d'"
    data = (useremailaddr,id)
    cursor.execute(sql%data)
    cont = ""
    try:
        for row in cursor.fetchall():
            cont = row[0]
    except IndexError:
        return -1
    print(cont)
    cursor.close()
    con.close()
    return cont
```

3.5.4 用户注册

```
if sstr[0] == "register" and len(sstr) == 4:
    try:
        user_name = sstr[1]
        user_code = sstr[2]
        port = sstr[3]
        if port == "58000" :
            user_email = user_name + "@qq.com"
        else:
            user_email = user_name + "@163.com"
        #将用户的信息存入数据库
        conmysql.register(user_name,user_code,port,user_email)
        mailadd = user_name
        return "OK : register succeed!\n mailaddr ="+mailadd
    except IOError :
        return "Error : input error!"
```

```
#用户注册
def register(user_name,user_code,port,user_email):
    con = connectdb()
    cursor = con.cursor()
    port2 =int(port)
    sql = "INSERT INTO user(username,usercode,useremail,smtp_state,pop_state,port) VALUES('%s','%s','%s','%s','%s','%d')"
    data = (user_name,user_code,user_email,"1","1",port2)
    cursor.execute(sql%data)
    con.commit()
    cursor.close()
    con.close()
```

3.5.5 修改用户密码

```
elif sstr[0] == "update" and len(sstr) == 3:
    try:
        user_name = sstr[1]
        user_code = sstr[2]
        # 进行用户信息校验
        if conmysql.update_password(user_name, user_code) == 1:
            return "OK : update succeed!"
        else :
            return "Error : update error!"
    except IOError:
        return "Error : input error!"
else:
    return "Error : input error!"
```

```
def update_password(user,password):
    con = connectdb()
    cursor = con.cursor()
    sql = "UPDATE user SET usercode = '%s' where username = '%s'"
    data = (password, user,)
    cursor.execute(sql % data)
    con.commit()
    cursor.close()
    con.close()
    return 1
```

3.5.8 界面介绍

普通用户:

1.进入安卓端

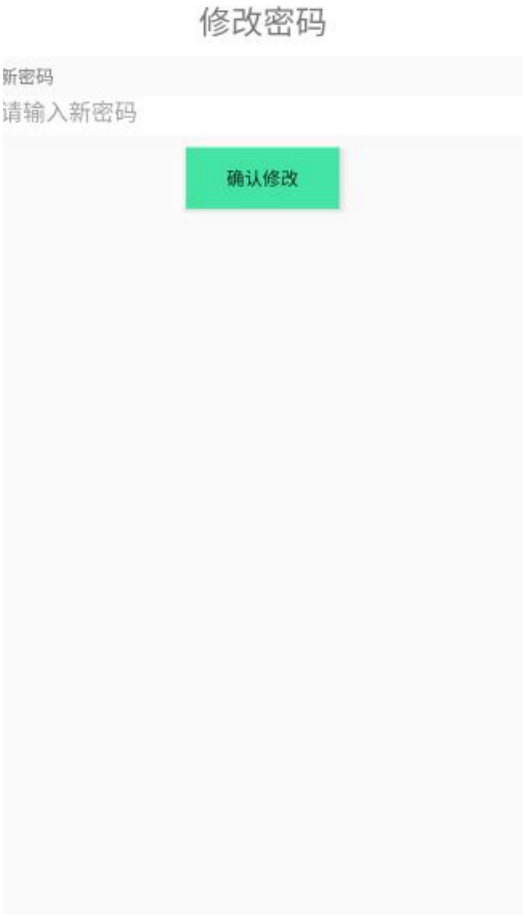
进入 Android 端，可以选择身份登录，用户登录或者管理员登录

2. **登录注册**

登录注册是可以相互跳转的在登录页面可以选择注册新用户, 新用户注册成功后, 进入登录界面进行登录。

3.**修改密码**

用户在当前页面进行密码修改

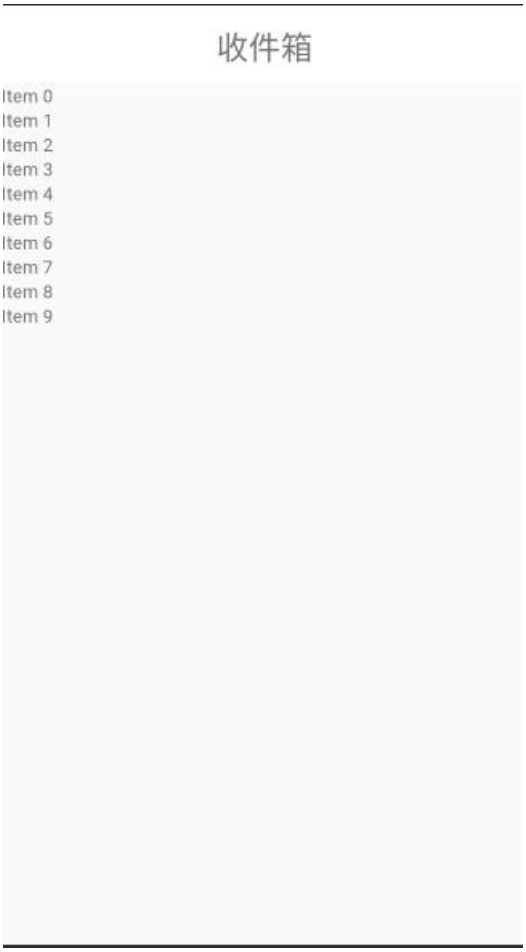


4. **发送邮件**

用户当前页面进行邮件的编写：输入收件人、邮件主题、具体内容。进行发送。

5. **查看收到的邮件**

用户当前页面进行邮件的查看，点击查看具体邮件的详情，也可以对已浏览的邮件，进行删除。



管理员用户：

管理员功能：新增用户，删除用户，管理用户的 SMTP 权限

1. **管理员登录**

管理员在当前页面进行账号密码登录

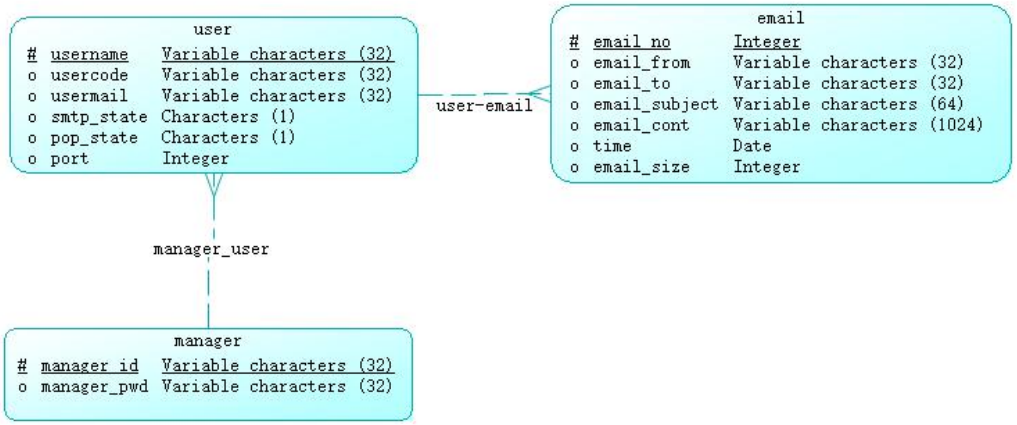
2. **用户管理**

管理员在当前页面进行新增用户，删除用户，管理用户的 SMTP 权限。

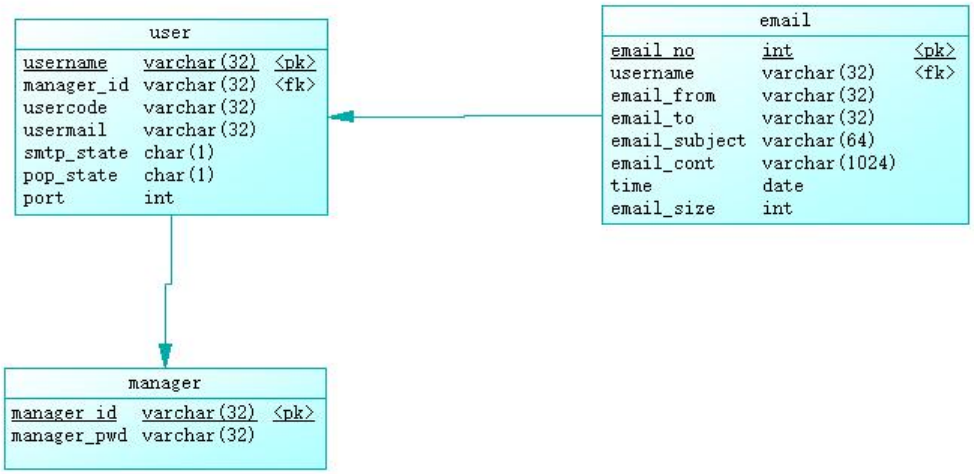

Mail
用户管理
用户列表
权限
删除
zhangsan
开启
禁用
删除
lisi
开启
禁用
删除
wangwu
开启
禁用
删除
Tom
开启
禁用
删除
创建用户

3.6 数据库

3.6.1 概念数据模型



3.6.2 物理数据模型



3.6.3 USER 表

对象 email @mailsystem (localho... user @mailsystem (localhos...

开始事务 文本 筛选 排序 导入 导出

username	usercode	useremail	smtp_state	pop_state	port
jishengyi	123456	jishengyi@qq.com	1	1	58000
jsy	1234	jsy@163.com	1	1	59000
jjj	123456	jjj@qq.com	0	1	58000
jiji	123456	jiji@163.com	1	1	59000
adfasf	dfadsfasdf	adfasf@qq.com	1	1	58000
jijiji	123456	jijiji@163.com	1	1	59000
jissy	12345	jissy@qq.com	1	1	58000
jissy	12345	jissy@qq.com	1	1	58000
test	12345	test@qq.com	1	1	58000

3.6.4 email 表

对象 email @mailsystem (localho...

开始事务 文本 筛选 排序 导入 导出

email_no	email_from	email_to	email_subject	email_cont	time	email_size
2	jsy@163.com	jishengyi@qq.com	adsfasd	asdfsad	2020-06-01 16:13:34	9
3	jsy@163.com	jishengyi@qq.com	qwerqwer	wqerqwer	2020-06-01 16:52:39	10
1	jsy@163.com	test@qq.com	hello	hello	2020-06-03 16:42:30	7

3.6.4 manager 表

对象 manager @mail (local) - 表

开始事务 备注 筛选

manager_id	manager_pwd
jishengyi	123qwe
wangshuhan	123qwe

4.项目总结

4.1 任务难点

完成客户端与服务器、服务器与服务器之间的收发信操作，使用 SMTP（发信）、POP3（收信）。

4.2 项目感想

经过将近两个多月的奋战，基于 SMTP 和 POP3 协议的邮箱服务器以及客户端的开发已经基本结束了。这次的课程设计使自己的动手能力得到了很大的提高，更深入的理解了课程知识。

在我们的小组中，我们分工明确具体。任何事情在刚开始时总会让人觉得茫然不知所措。不过，慢慢的就了解了基本的方法，并一步步的完成任务。在我们课程设计的这段时间里，很感谢老师和助教的帮助，也很感谢队友们的配合。

实战是提高能力的有效方式，以后我们一定要多多进行实战，才能更好地掌握知识。

5 参考文献

[1] Matt Zandstra. 深入 PHP:面向对象、模式与实践:第 3 版[M]. 人民邮电出版社

[2] [美]库罗斯. 计算机网络：自顶向下方法（原书第 4 版）[M]. 机械工业出版社

[3]W.Richard Stevens. TCP/IP 详解卷 1-卷 4[M]. 机械工业出版社

[4] W. Jason Gilmore. PHP 与 MySQL 程序设计[M]. 人民邮电出版社

[5] Armando Padilla, T im Hawkins. 高性能 PHP 应用开发[M]. 人民邮电出版社

[6] 黄隽实. Android+PHP 最佳实践[M]. 人民邮电出版社

[7] Ian G. Clifton. Android 用户界面设计[M]. 电子工业出版社

[8]软件开发标准：国家标准(GB8567——88)

《项目开发计划规范》(GB856T——88)

《软件需求说明书规范》(GB856T——88)

《数据库设计说明书规范》(GB8567——88)

《数据要求说明书规范》(GB856T——88)

《详细设计说明书规范》(GB8567——88)

《用户手册规范》(GB8567——88)

《测试计划、测试分析报告规范》(GB8567——88)