程序设计说明

引言

编写目的

为程序的整体架构，模块组成和数据结构设计提供说明，对项目所采用的技术，提供的功能及实现的方法作出解释

范围

包括程序运行的环境，完整的运行流程，以及数据库设计和系统的部署等

背景说明

主题与设计

本项目实现系一集登陆注册，收发邮件，联系信息，邮件检索为一体且具有配套后台的基于安卓系统的移动邮件客户端。能够在连接网络的情况下实现完整的邮件操作，具有很强的实用性
项目的设计采用 Material Desigh，风格简洁明快，易于使用。结合下拉通知和振动传感器等带来完整方便的使用体验

技术要点

必选内容：

1.界面框架设计
2.事件处理的实现
3.SQLite，SharedPreference 等数据存储方式

自选内容：
1.Notification的显示
2.Broadcast
3.Service
4.网络访问功能
5.后台交互功能
6.传感器使用
7.2D动画应用

拓展内容：
采用阿里云云服务器实现系统后台及邮件代理

系统环境

开发环境

客户端：
MI NOTE LTE (Android 6.0.1 MMB29M MIUI 8.1.4.0)
服务端：
Aliyun CentOS 7 64bit

数据库

SQLite

开发语言

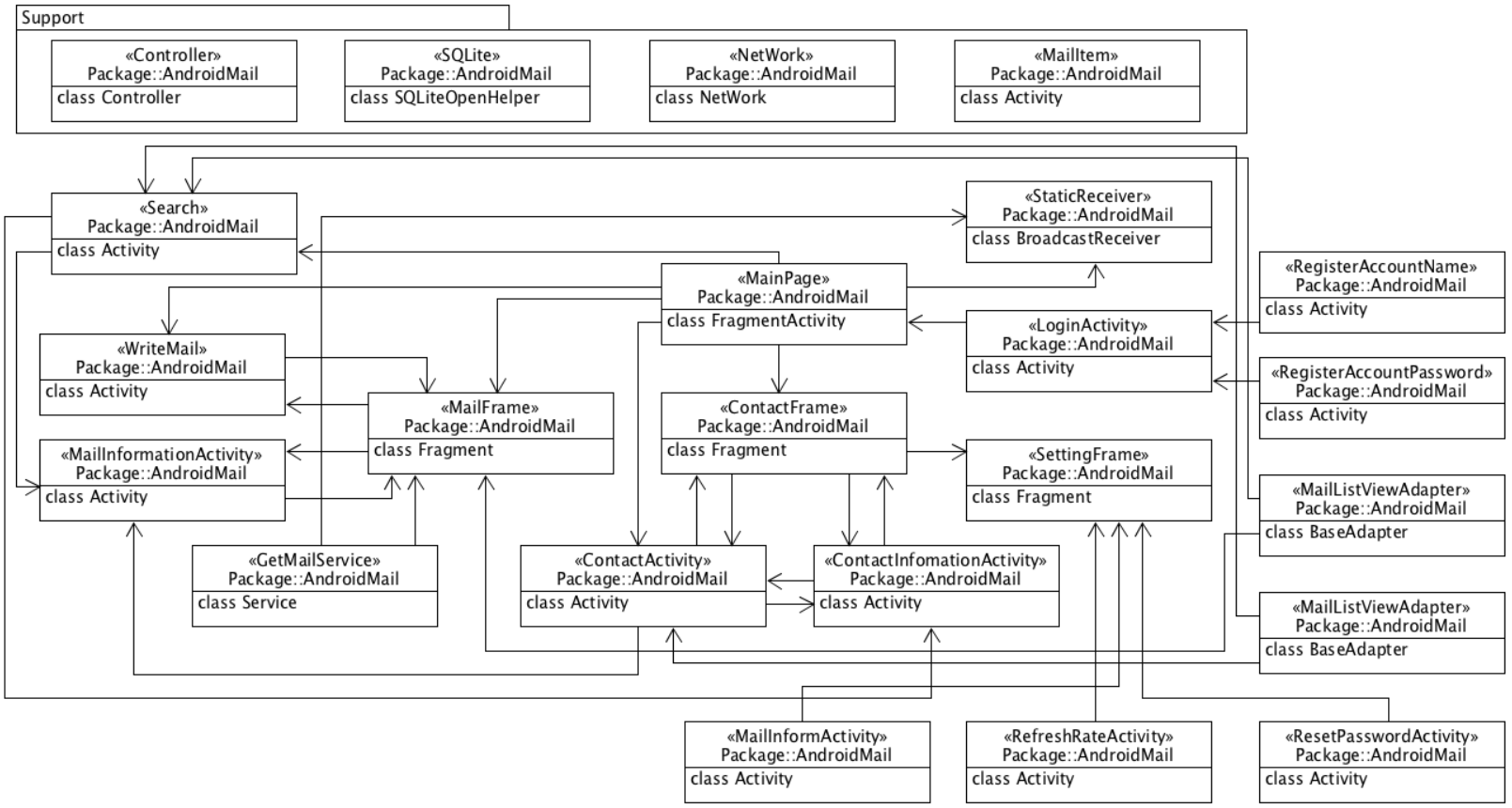
客户端：
JAVA
服务端：
Python

网络及硬件设备

客户端：
具有 Android 6.0 以上版本系统且满足以下要求的移动终端：最低512MB内存，最小8GB ROM，有网络访问
服务端：
具有Pentium Ⅲ处理器以上且满足以下要求的计算机：最低256MB内存，最小20GB外置存储，有网络访问

总体概述

系统架构



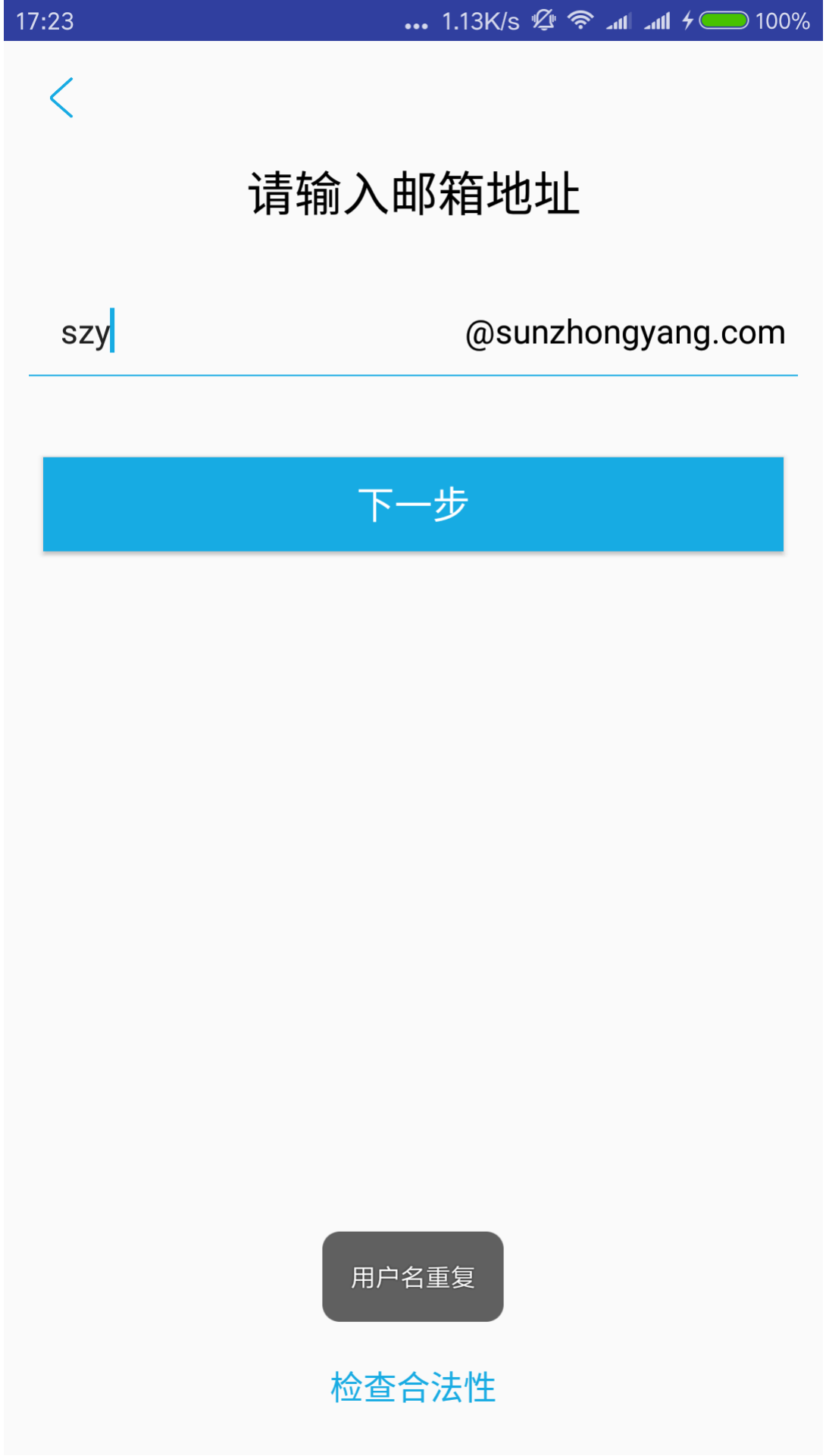
系统目标

用户登录

在之前没有成功登录的情况下，用户打开应用会首先进入到用户登陆页面提示用户登陆，输入正确的用户名密码即可跳转到主页面，如果之前没有注册或者忘记邮箱密码，也可点击相应按钮跳转到指定页面获取合法的账号

用户注册

注册过程中，首先需要输入想要注册的邮箱地址，点击下一步或者检查合法性程序会联系服务器自动判断邮箱地址是否合法

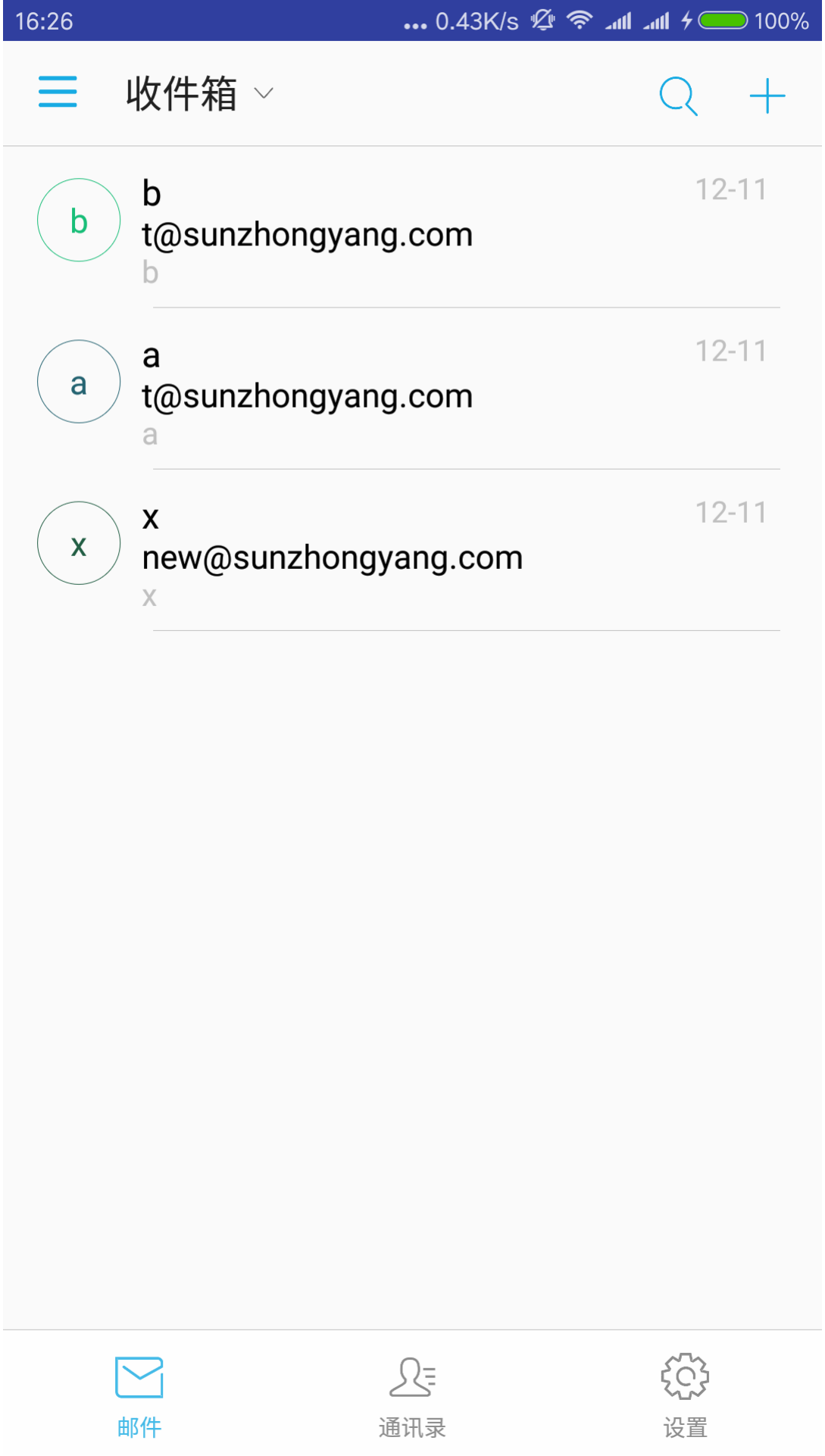


如果地址合法，点击下一步，后面的界面会显示完整的邮箱，如下图所示

重复输入密码，如果密码格式合法，服务器认可，那么注册成功，并跳转到主界面

主界面

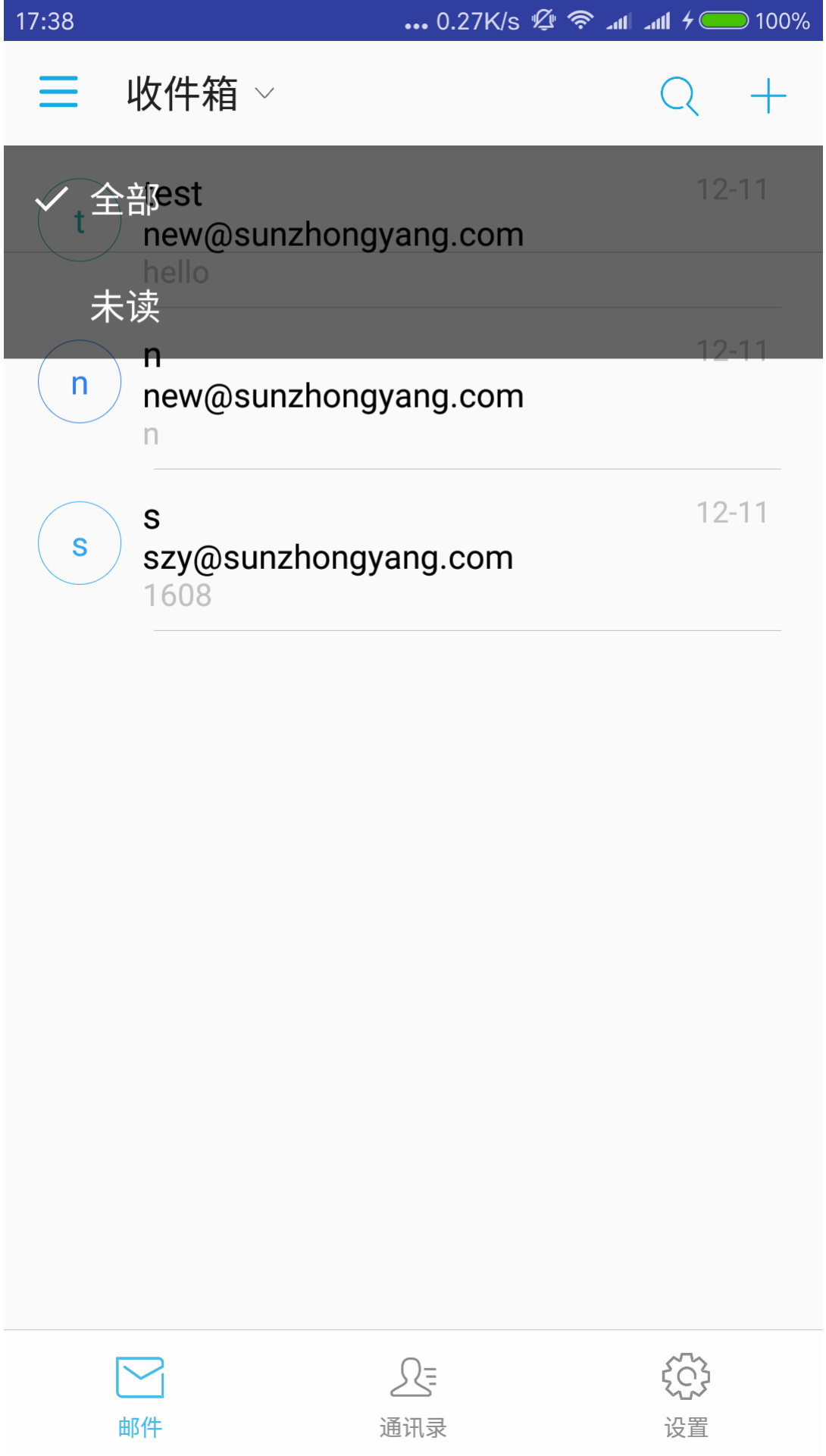
在主界面中可切换标签（邮件、联系人、设置），并在对应的标签执行不同的功能



三个标签为别为邮件标签，联系人标签和设置标签，在邮件页面中，长按邮件可以将其移入垃圾箱或者完全删除

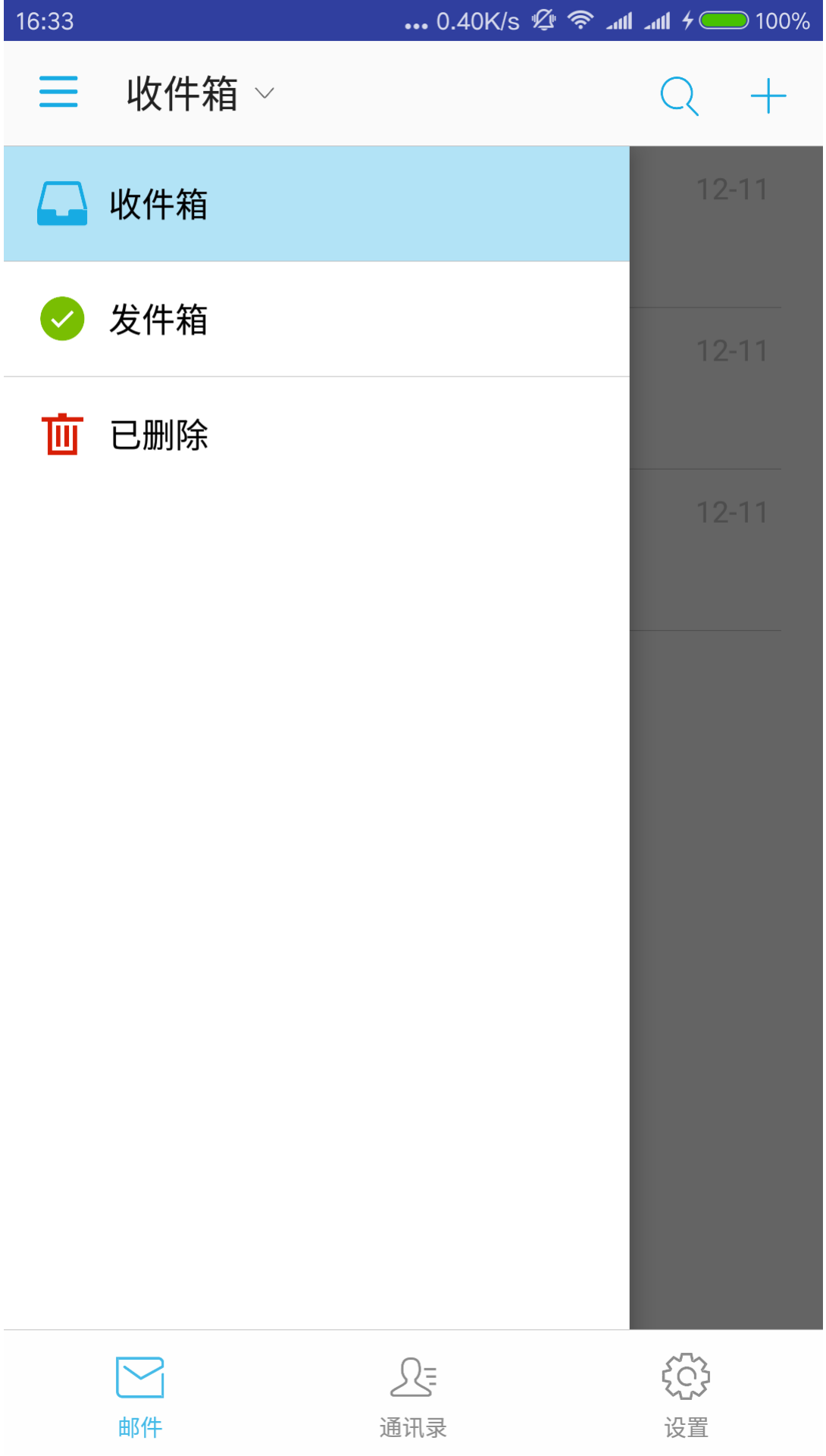
切换已读或者未读邮件

主页面默认会显示收件箱中的邮件，收件箱中的邮件有已经浏览的，也有没有浏览过的，通过点击收件箱或者收件箱右侧的箭头，可以切换已读邮件和未读邮件



查看已发送或已删除的邮件

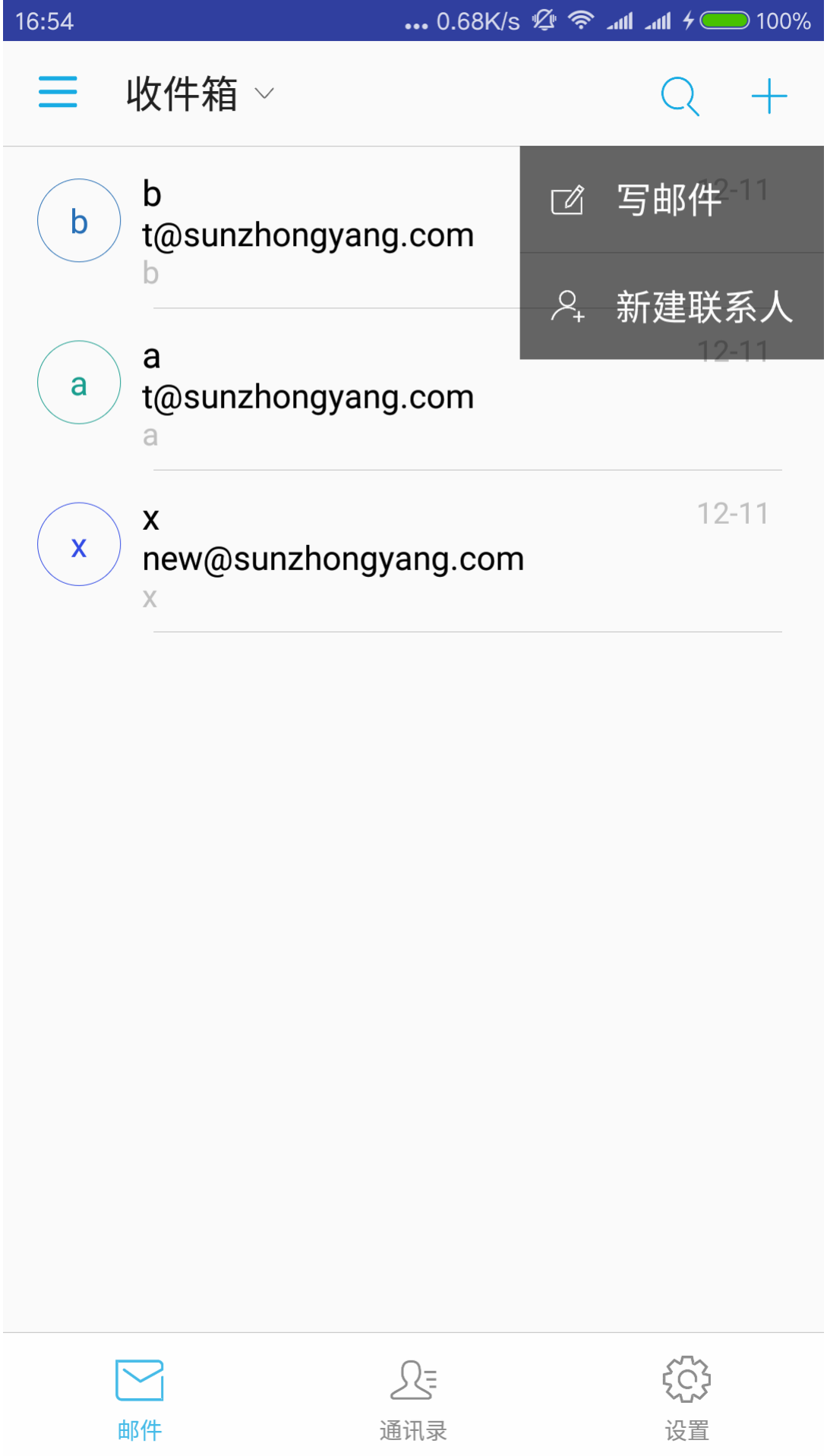
左划或者点击左上方按键可以切换信箱，左划部分的布局属于一个 `DrawerLayout`



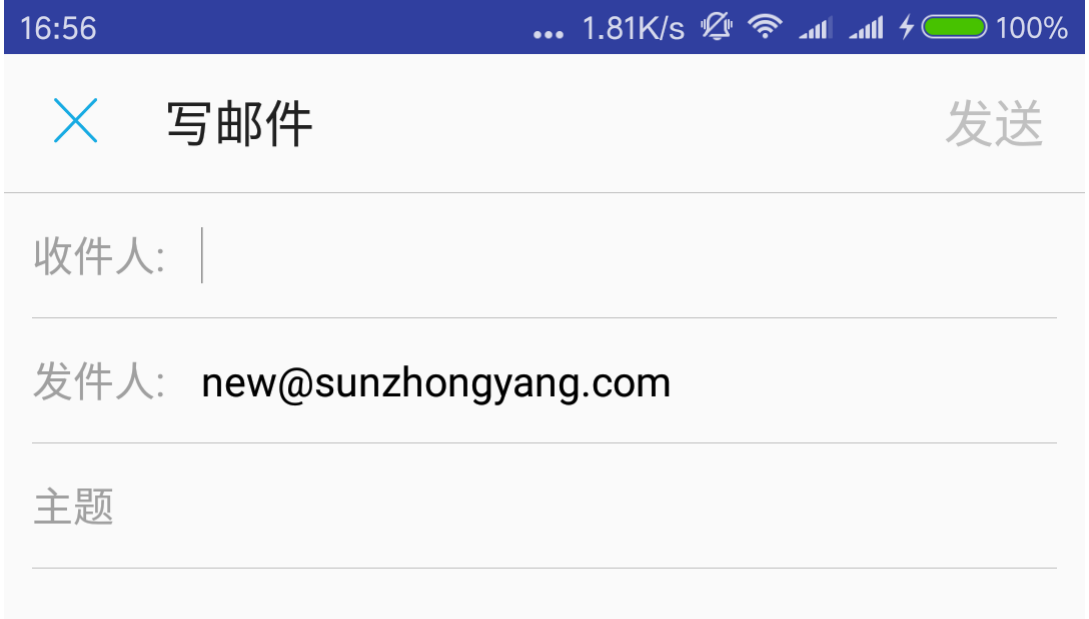
点击不同的信箱可以使主页面显示不同信箱的邮件

发送邮件

发送邮件有多方式，这里看一下通过主界面发送邮件
点击右上角可以打开一个 `popwindow` ，点击 `"写邮件"` 可以跳转到新建邮件的界面


16:54
0.68K/s
100%
收件箱
b
t@sunzhongyang.com
b
12-11
写邮件
新建联系人
a
t@sunzhongyang.com
a
12-11
x
new@sunzhongyang.com
x
12-11
邮件
通讯录
设置

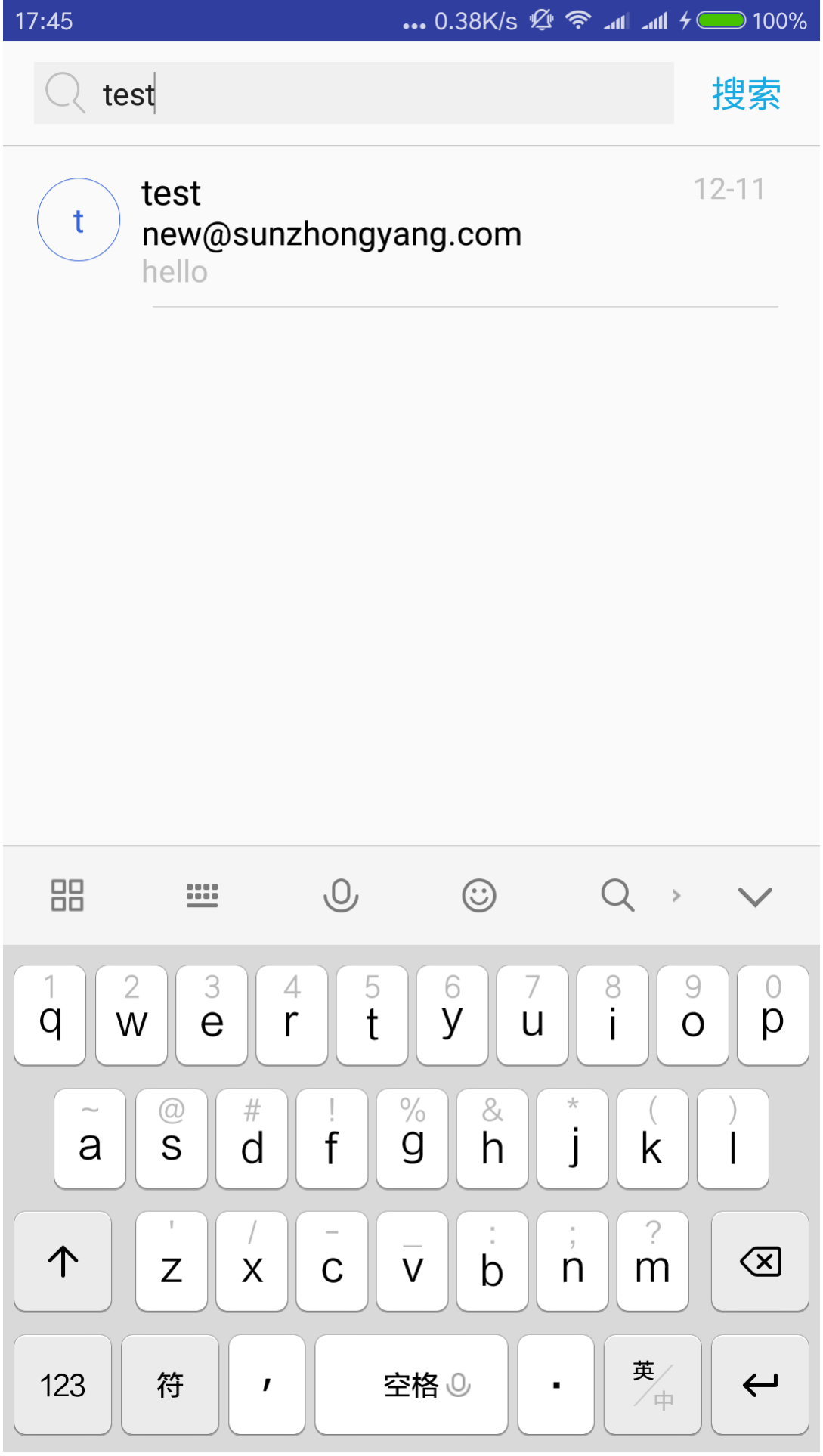
下图为新建邮件界面



发件人会自动根据本地数据保存的用户名填写，之后可手工填写发件人和具体邮件内容，点击发送键发送邮件，发送成功后会有提示

查找邮件

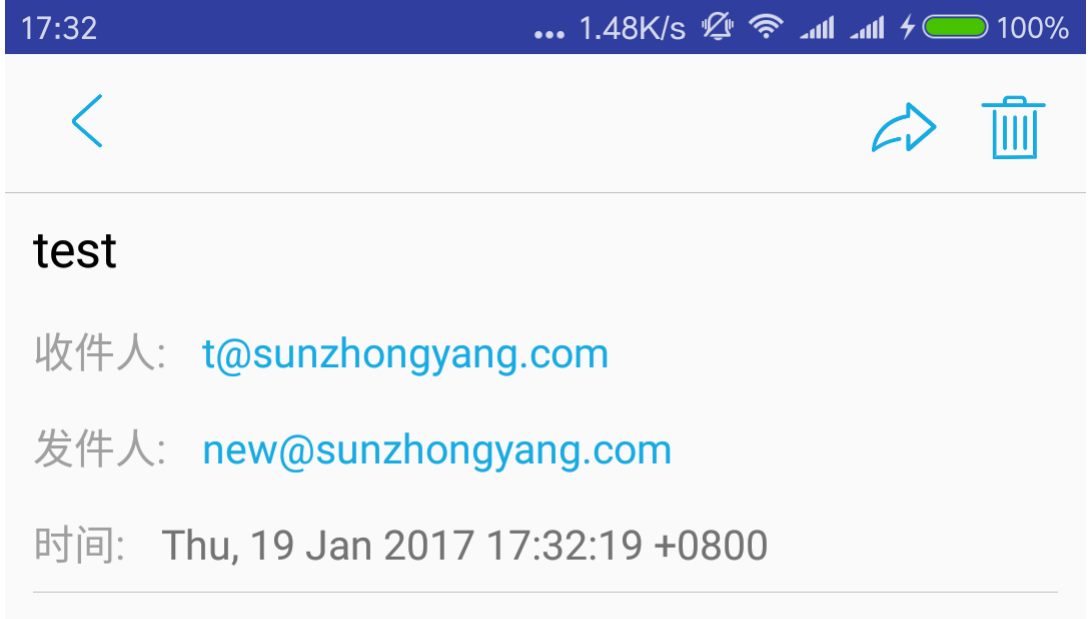
在主页面中点击右上方的搜索按键，可以根据标题和发件人搜索邮件



点击邮件同样可以查看邮件详情

查看邮件详情

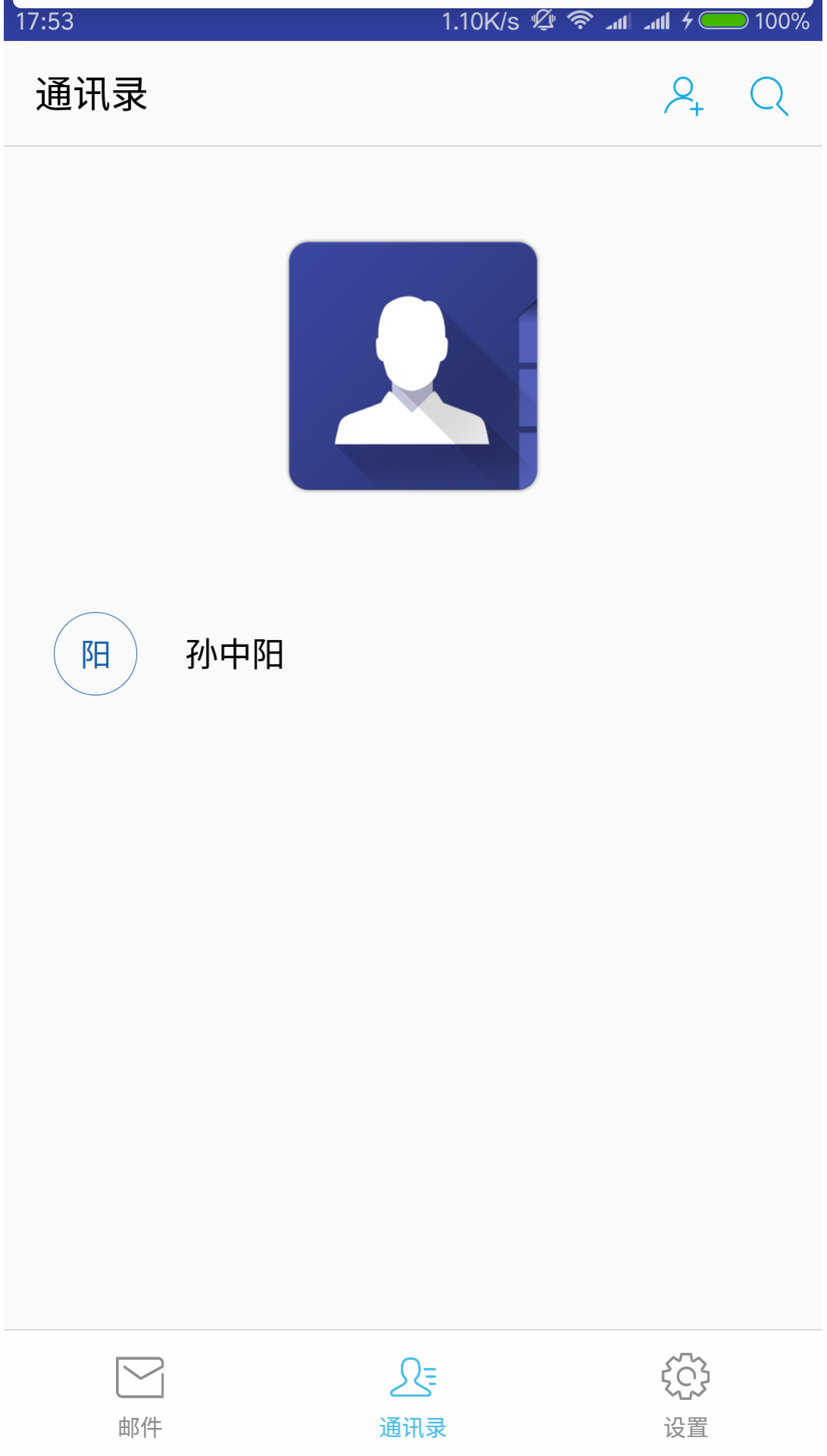
在所有有邮件的界面中，短按一个邮件可以查看邮件的详情，如下图所示



可以看到邮件的收件人，发件人，主题，创建时间和正文。点击右上角的回复键，可以直接以发件人为收件对象创建新邮件，也可以直接删除邮件
收件箱和发件箱中的邮件被删除后会进入垃圾箱，垃圾箱中的邮件被删除后会被直接从数据库中移除

浏览联系人

在主界面下方点击 `"通讯录"` 选项可切换到联系人标签



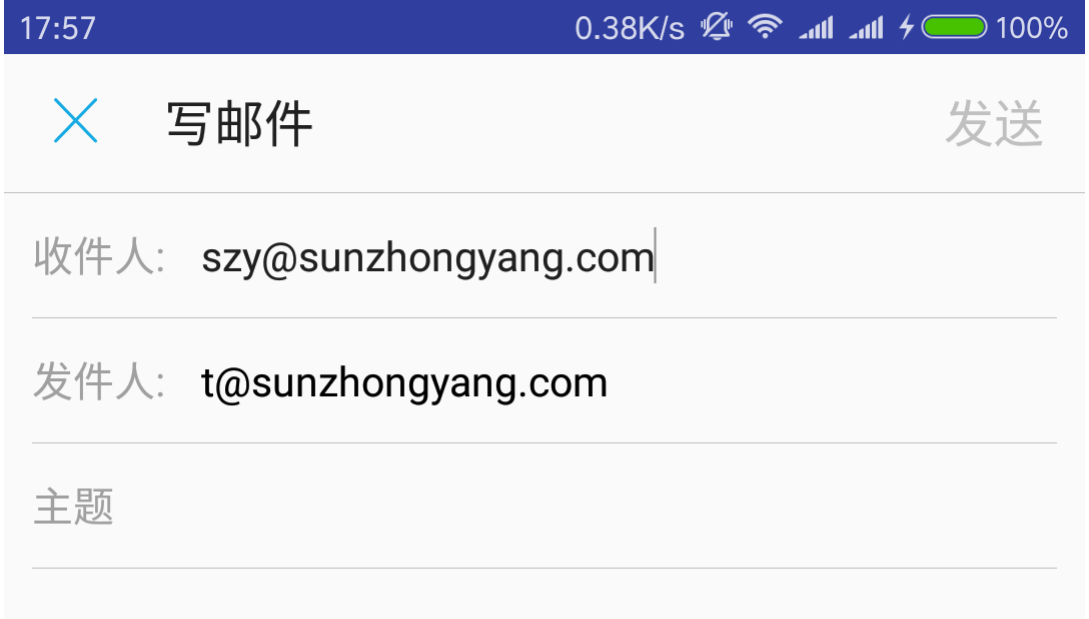
点击右上方的新建联系人按钮可进入编辑联系人页面新建联系人，长按联系人可以删除联系人

短按联系人则可查看联系人详情

查看联系人详情

在联系人详情页面可查看联系人的姓名，单位和邮件地址三个信息

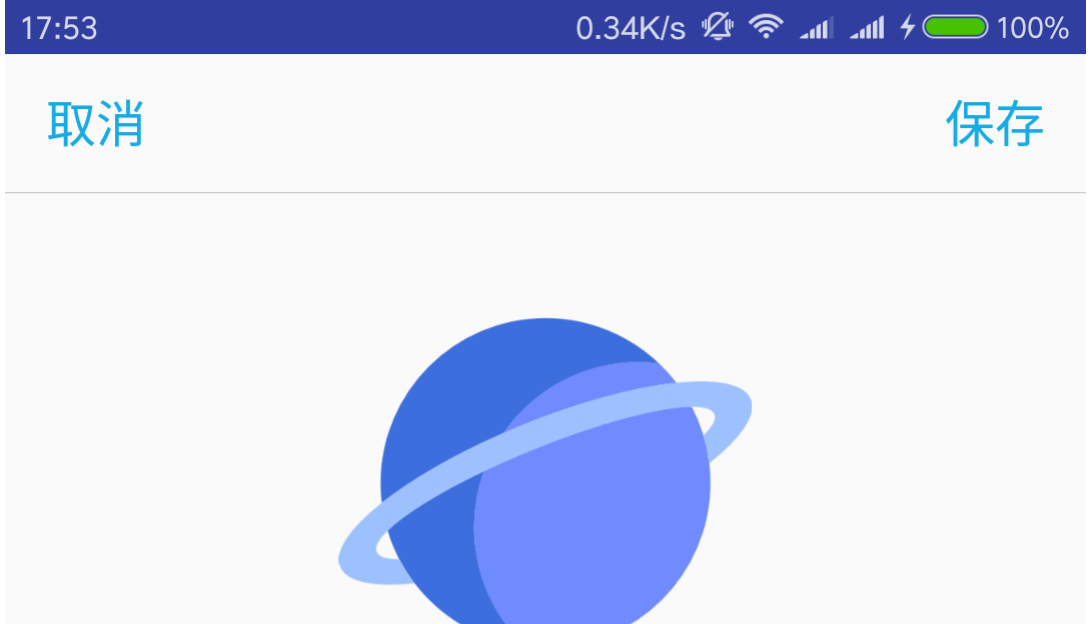
可以点击发送按键向联系人的邮件地址发送邮件



或者点击编辑按键编辑联系人

编辑联系人

如果不是新建联系人，则会自动填写已经保存的联系人信息



点击保存完成更改

查找联系人

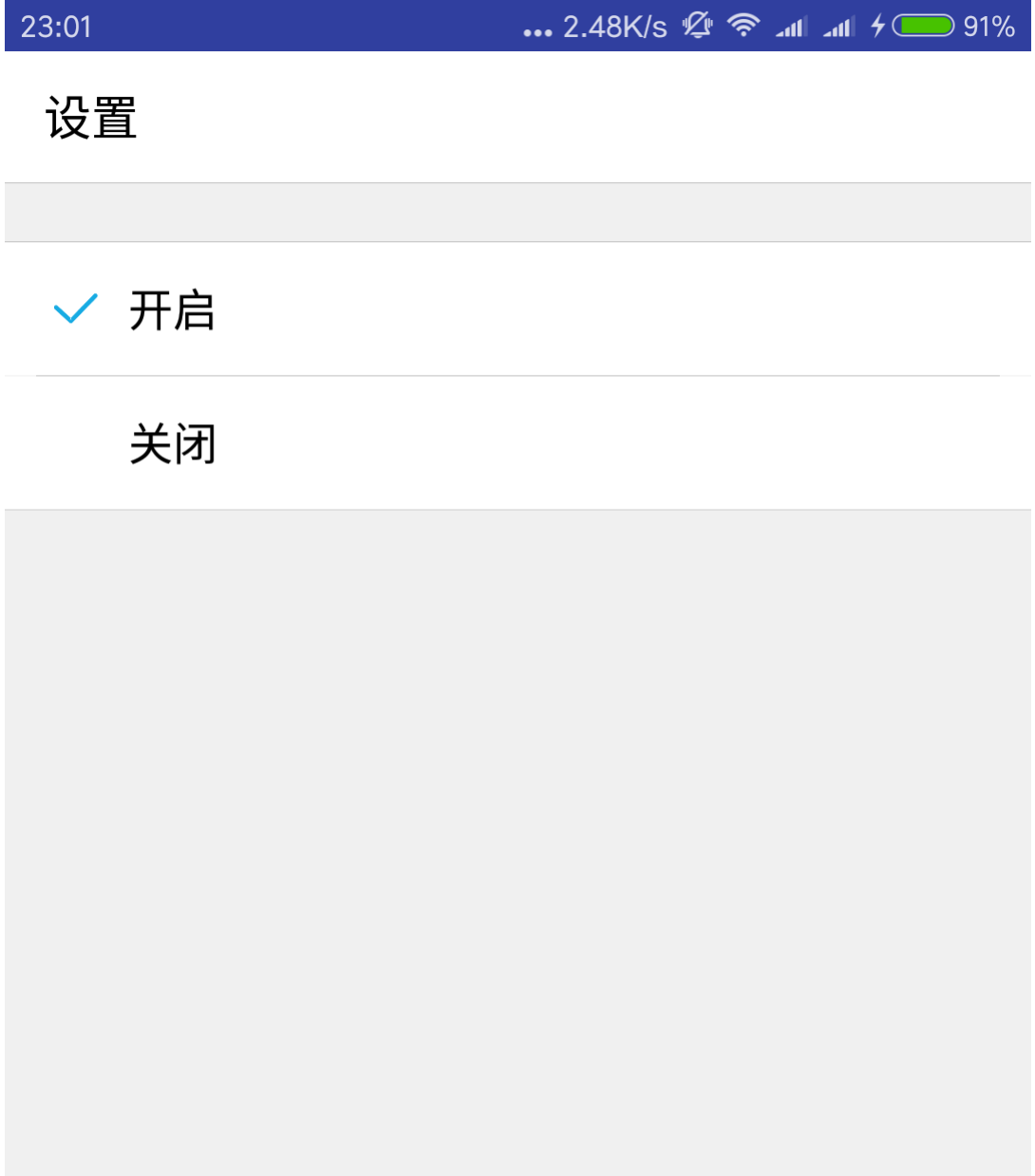
在联系人标签，同样可以点击查找按键，不过这一次查找的是联系人而不是邮件

同样可以点击联系人查看详情以及编辑，但是不能删除

有关设置

主页面点击设置，可以切换到设置标签

在这里可以设置是否提醒新邮件



如果不提醒新邮件则新邮件到来时不会发送下拉通知，也不会振动

另一方面，也可以设置更新间隔，单位为毫秒

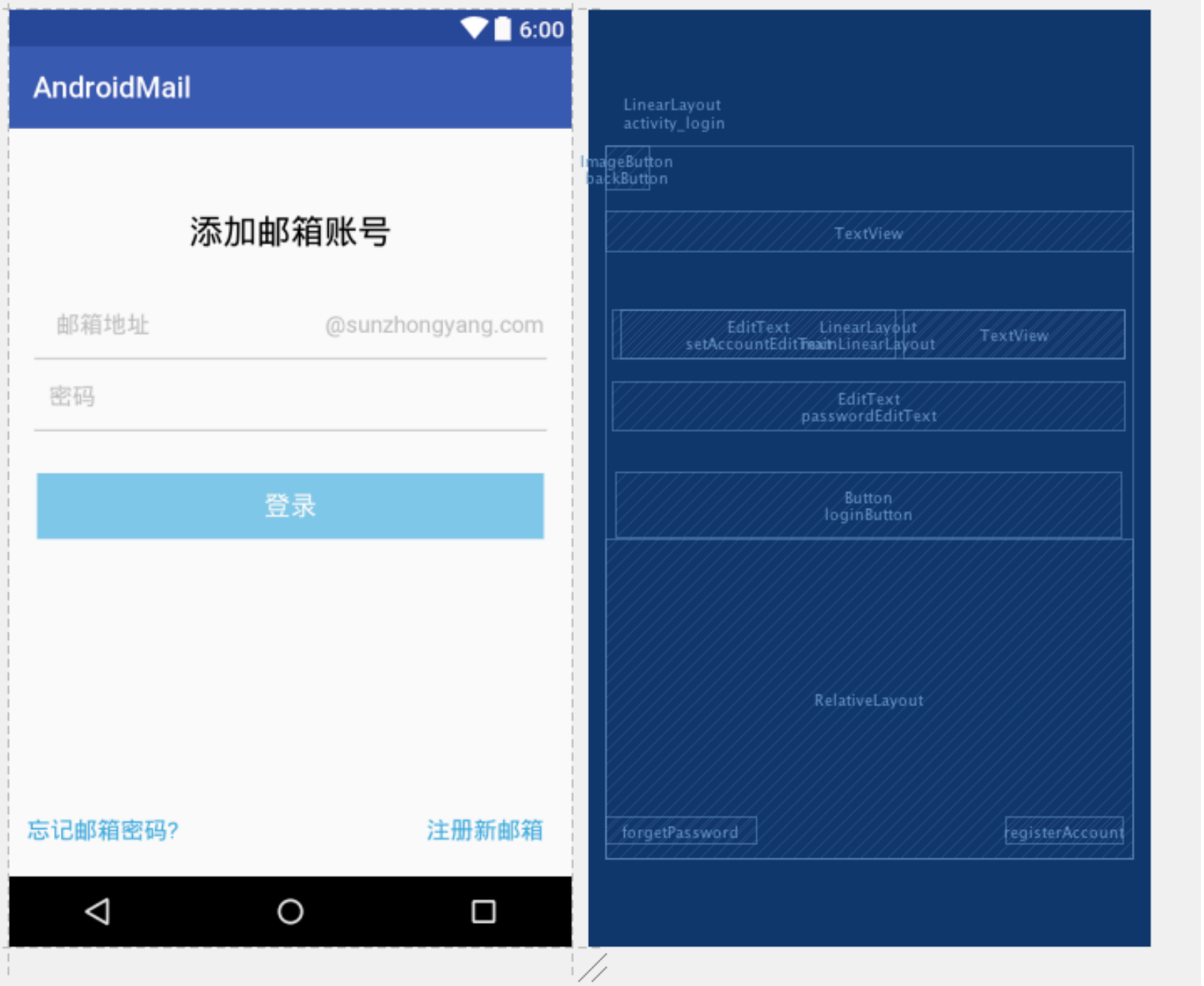
客户端会自动以此为间隔向服务器检查新邮件
另外也可以更改密码

也可以退出登录，退出登录后会跳转到登录界面，不可返回上一页面

UI设计

登陆界面

页面布局:



这里的特点是登陆过程中输入用户名密码的过程会发现光标颜色和下划线颜色，以及下划线宽度和系统默认是不同的，而且下划线上方还可显示邮箱后缀，这里用了一系列自定义的页面设计

这一设计主要通过采用一个带背景的 `LinearLayout` 和一个不带背景的 `EditText` 以及 `TextView` 实现

```
<LinearLayout
    android:id="@+id/mainLinearLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginStart="5dp"
    android:layout_marginEnd="5dp"
    android:background="@drawable/edt_bg_selector">

    <EditText
        android:id="@+id/setAccountEditText"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="5"
        android:hint="@string/mail_address"
        android:textColorHint="@color/grey"
        android:textSize="16sp"
        android:inputType="text"
        android:maxLines="1"
        android:paddingBottom="15dp"
        android:paddingStart="10dp"
        android:layout_marginStart="5dp"
        android:layout_marginEnd="5dp"
        android:background="@null"
        />

    <TextView
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="4"
        android:textSize="16sp"
        android:textColor="@color/grey"
        android:text="@string/mail_end"/>

</LinearLayout>
```

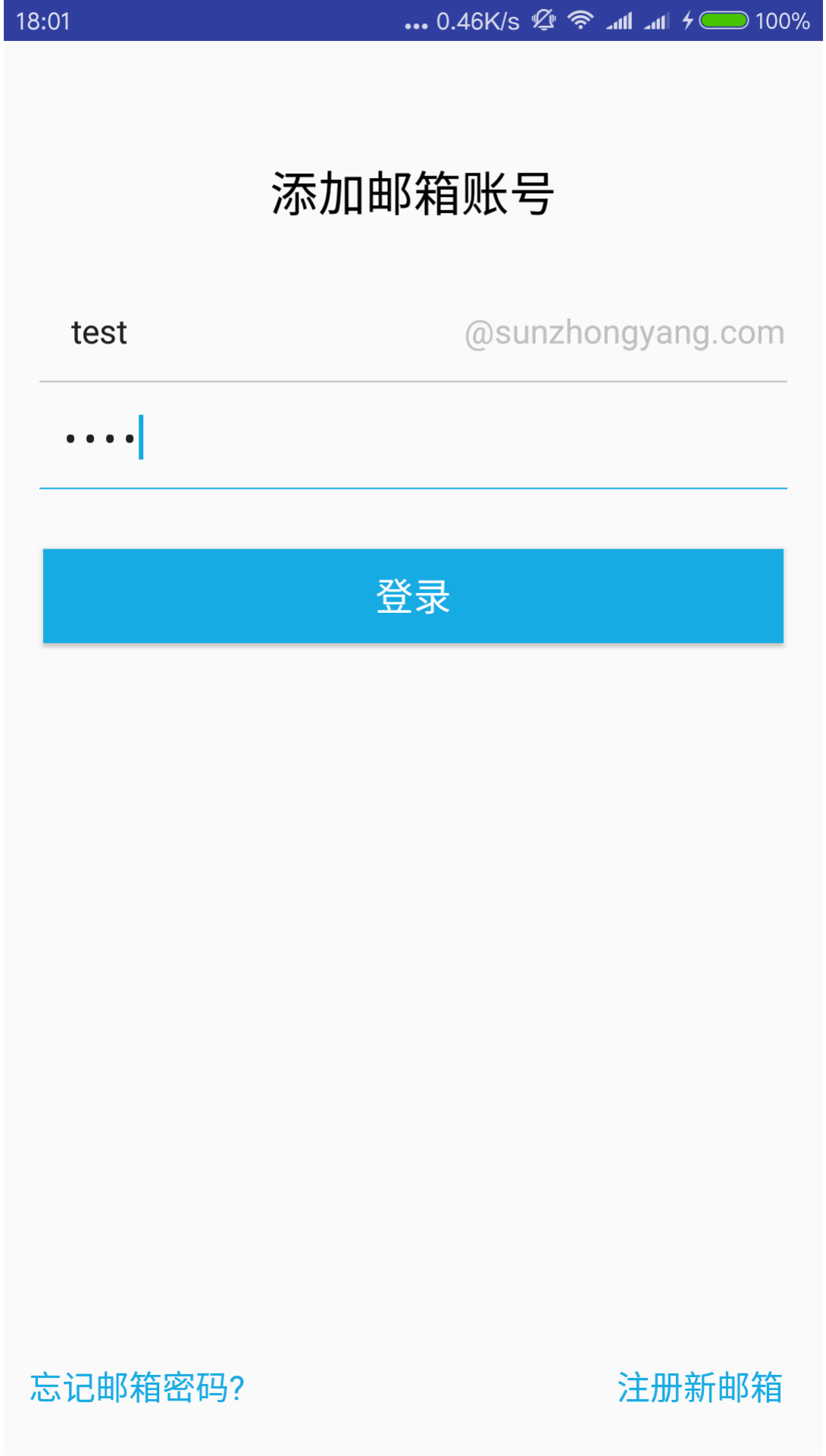
`LinearLayout` 的背景是一个选择器 `edt_bg_selector`

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/edt_bg_selected" android:state_focused
="true"/>
    <item android:drawable="@drawable/edt_bg_normal" android:state_focused="
false"/>
</selector>
```

`edt_bg_selector` 使得其关联的控件在被选择和没有被选择时呈现不同的状态，在被选择时使用 `edt_bg_selected` 作为背景， `LinearLayout` 需要设置监听器，程序中已给出实现

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:bottom="1dp"
        android:left="-2dp"
        android:right="-2dp"
        android:top="-2dp">
        <shape android:shape="rectangle" >
            <stroke
                android:width="2px"
                android:color="@color/colorAccent" />

            <solid android:color="#00FFFFFF" />
        </shape>
    </item>
</layer-list>
```

这里讨巧的使用一个蓝色的矩形作为背景，矩形上方及左右的三条边的边距被设置为负值从而不会被实际显示出来，营造出只有一个下划线的效果。由此，我们可以控制选中后下划线的宽度，以及下划线的颜色，后面会多次使用到这一设计



另外有趣的一个设计是当用户名和密码的输入都不为空时登录按钮的颜色会变得更鲜艳，这通过监听 `EditText` 变化，并根据变化改变按钮的透明度来实现

```
setAccountEditText.addTextChangedListener(textWatcher);
passwordEditText.addTextChangedListener(textWatcher);
```

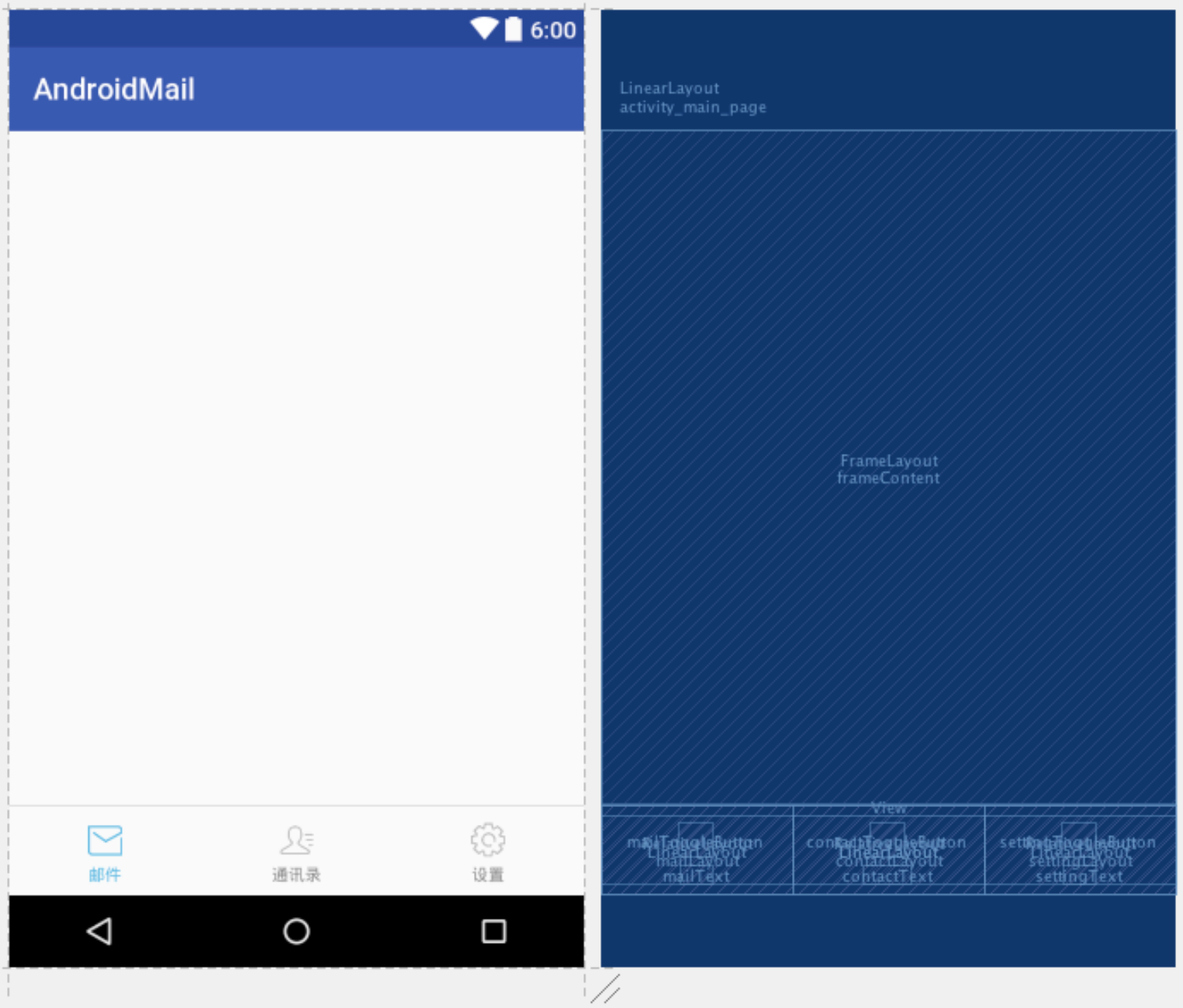
增加监听器

```
private TextWatcher textWatcher = new TextWatcher()
{

    @Override
    public void afterTextChanged(Editable s)
    {

    }

    @Override
    public void beforeTextChanged(CharSequence s, int start, int count,
                                  int after)
    {

    }

    @Override
    public void onTextChanged(CharSequence s, int start, int before,
                              int count)
    {
        //根据输入内容是否为空设置按钮透明度
        String account = setAccountEditText.getText().toString();
        String password = passwordEditText.getText().toString();
        if(account.equals("") || password.equals(""))
        {
            loginButton.setAlpha((float) 0.6);
        }
        else
        {
            loginButton.setAlpha((float) 1.0);
        }
    }
};
```

每次输入内容有改变就触发一次 `onTextChanged()` ，然后会检查用户名和密码是否为空，如果都为非空则更改按钮透明度为不透明，否则更改透明度为60%，这一设计在后面也会用到

主界面

主界面本质上由一个上方的 `FrameLayout` 和一个位于底部用于切换标签的 `LinearLayout` 组成，点击 `LinearLayout` 不同选项即可在 `FrameLayout` 中载入不同的 `Frame` ，默认载入的是 `MailFrame` ，如下图所示



一个非常简单的 `FrameLayout`

```
<FrameLayout
    android:id="@+id/frameContent"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    >

</FrameLayout>
```

设置不同标签页的过程为

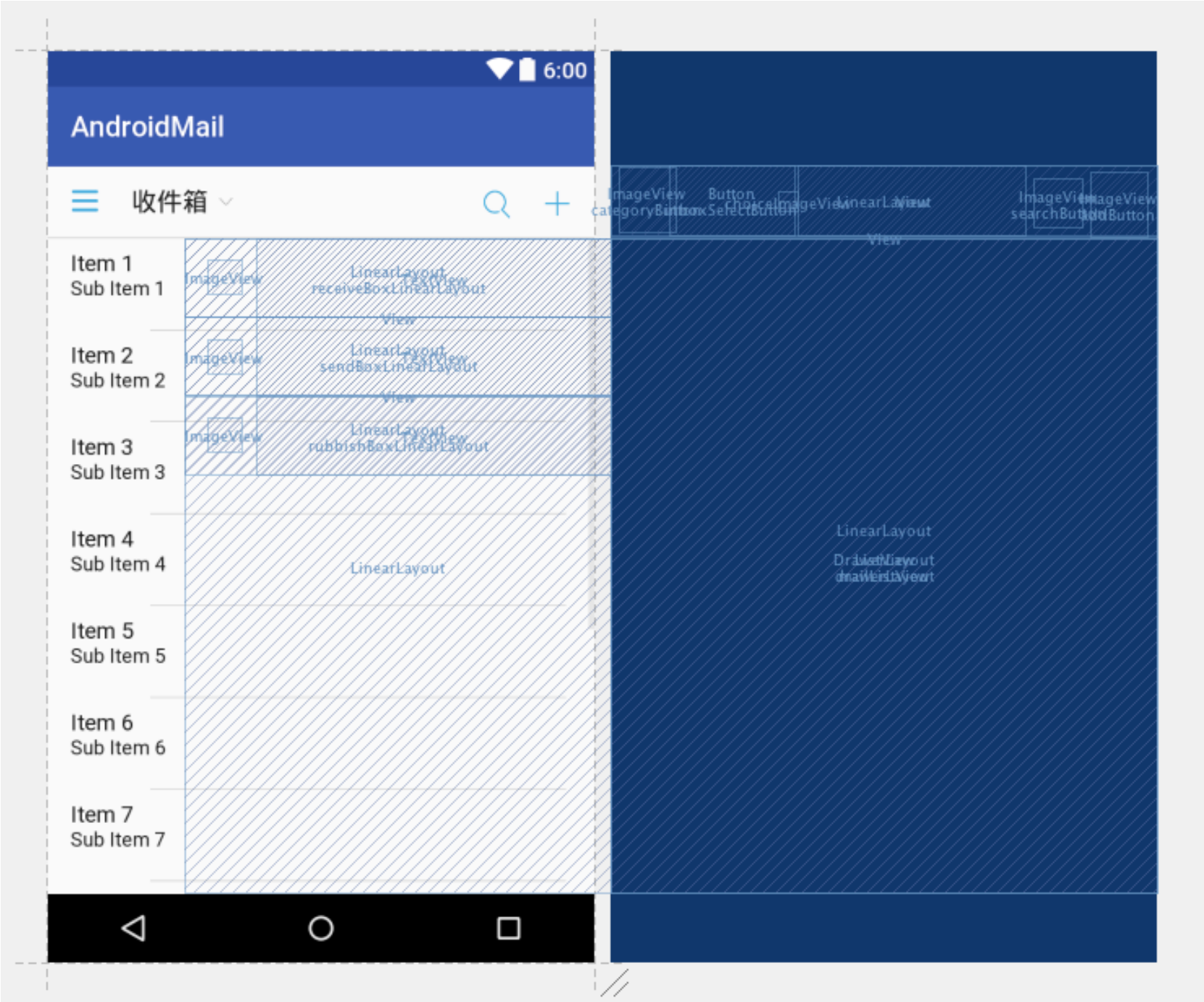
```
private void setFrame(int frameNumber)
{
    FragmentTransaction fragmentTransaction = fragmentManager.beginTransacti
on();
    //重设底部栏为初始状态, 也就是恢复到均为被点击的样式
    clearBottomBar();
    //隐藏正在显示的标签
    closeFragments(fragmentTransaction);
```

```
switch(frameNumber)
{
    case 0:
        //更换按钮样式为点击后的效果
        mailToggleButton.setImageResource(R.drawable.mail);
        //设置说明文字为点击后的效果
        resetBottomBarText(mailText);

        if(mailFrame == null)
        {
            //如果为空，则建一个新的并将它显示出来
            mailFrame = new MailFrame();
            fragmentTransaction.add(R.id.frameContent, mailFrame);
        }
        else
        {
            //如果不为空，则直接将它显示出来
            fragmentTransaction.show(mailFrame);
        }
        break;
    case 1:
        //更换按钮样式为点击后的效果
        contactToggleButton.setImageResource(R.drawable.contact);
        //设置说明文字为点击后的效果
        resetBottomBarText(contactText);

        if(contactFrame == null)
        {
            //如果为空，则建一个新的并将它显示出来
            contactFrame = new ContactFrame();
            fragmentTransaction.add(R.id.frameContent, contactFrame);
        }
        else
        {
            // 如果不为空，则直接将它显示出来
            fragmentTransaction.show(contactFrame);
        }
        break;
    case 2:
        //更换按钮样式为点击后的效果
        settingToggleButton.setImageResource(R.drawable.set);
        //设置说明文字为点击后的效果
        resetBottomBarText(settingText);

        if(settingFrame == null)
        {
            //如果为空，则建一个新的并将它显示出来
            settingFrame = new SettingFrame();
            fragmentTransaction.add(R.id.frameContent, settingFrame);
        }
        else
```

```
            {
                // 如果不为空，则直接将它显示出来
                fragmentTransaction.show(settingFrame);
            }
            break;
    }

    //执行更改
    fragmentTransaction.commit();
}
```

DrawerLayout

左划呼出的切换信箱的呼出菜单效果很不错，这里用到的其实是一个位于 `mailFrame` 中的 `DrawerLayout` ，布局如下图所示



`DrawerLayout` 本身就像一个 `frame` 一样，有自己的内部结构，其中 `layout_gravity` 设置为 `left` 的部分是左划会呼出的部分

```
<android.support.v4.widget.DrawerLayout xmlns:android="http://schemas.androi
d.com/apk/res/android"
```

```
android:id="@+id/drawerLayout"
android:layout_width="match_parent"
android:layout_height="match_parent"
>

<ListView
    android:id="@+id/mailListView"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:divider="@drawable/list_item_divider"
    android:dividerHeight="1px"
    />
<LinearLayout
    android:layout_width="300dp"
    android:layout_height="match_parent"
    android:layout_gravity="left"
    android:background="@color/white"
    android:orientation="vertical"
    >

    <LinearLayout
        android:id="@+id/receiveBoxLinearLayout"
        android:layout_width="match_parent"
        android:layout_height="55dp"
        android:orientation="horizontal"
        android:gravity="center"
        android:background="@color/colorAccentAlpha">

        <ImageView
            android:layout_width="24dp"
            android:layout_height="24dp"
            android:layout_marginStart="16dp"
            android:src="@drawable/icon_inbox"
            />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:gravity="center_vertical"
            android:textSize="16sp"
            android:text="@string/inbox"
            android:textColor="@color/realBlack"
            android:layout_marginStart="10dp"
            />

    </LinearLayout>

    <View
        android:layout_height="1px"
        android:layout_width="match_parent"
```

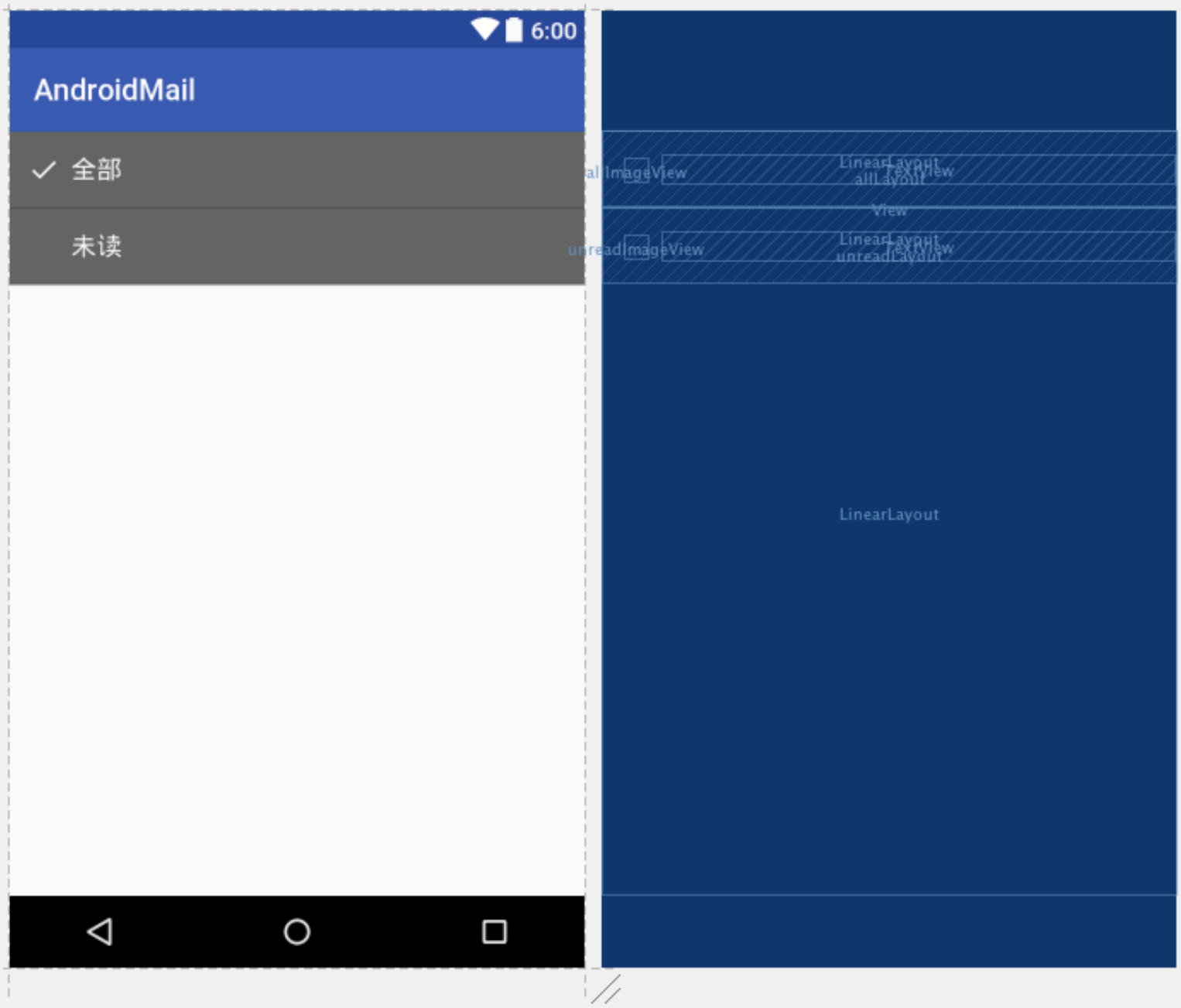
```
        android:background="@color/grey"
        />

    <LinearLayout
        android:id="@+id/sendBoxLinearLayout"
        android:layout_width="match_parent"
        android:layout_height="55dp"
        android:orientation="horizontal"
        android:gravity="center"
        android:background="@color/white"
        android:alpha="1">

        <ImageView
            android:layout_width="24dp"
            android:layout_height="24dp"
            android:layout_marginStart="16dp"
            android:src="@drawable/check"
            />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:gravity="center_vertical"
            android:textSize="16sp"
            android:text="@string/sendbox"
            android:textColor="@color/realBlack"
            android:layout_marginStart="10dp"
            />

    </LinearLayout>

    <View
        android:layout_height="1px"
        android:layout_width="match_parent"
        android:background="@color/grey"
        />

    <LinearLayout
        android:id="@+id/rubbishBoxLinearLayout"
        android:layout_width="match_parent"
        android:layout_height="55dp"
        android:orientation="horizontal"
        android:gravity="center"
        android:background="@color/white"
        android:alpha="1">

        <ImageView
            android:layout_width="24dp"
            android:layout_height="24dp"
            android:layout_marginStart="16dp"
            android:src="@drawable/rubbish"
```

```
                />

            <TextView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:gravity="center_vertical"
                android:textSize="16sp"
                android:text="@string/deleted"
                android:textColor="@color/realBlack"
                android:layout_marginStart="10dp"
                />

        </LinearLayout>

    </LinearLayout>

</android.support.v4.widget.DrawerLayout>
```

也就是说，左划会呼出的是其中的一个 `LinearLayout`

popwindow

在主页面切换已读未读邮件或者新建邮件联系人会弹出 `popwindow` ，切换已读未读 `popwindow` 的布局如下所示



从视频中可以看出，`popwindow` 的弹出是从上到下的，这是有意控制的结果，用到了一个 2D 动画，将原来的由中心向外扩展改为由上到下，设置动画的代码如下

```
//创建 PopupWindow 实例
popupwindow = new PopupWindow(customView, getActivity().getWindowManager().g
etDefaultDisplay().getWidth(), 300);
popupwindow.setAnimationStyle(R.style.AnimationFade);
```

其中 `AnimationFade` 定义了两个动画模式，分别为展开和收回时的动作，`AnimationFade` 在 `style.xml` 中

```
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
        <item name="colorControlNormal">@color/grey</item>
    </style>

    <style name="AnimationFade">

        <!-- PopupWindow上下弹出的效果 -->
        <item name="android:windowEnterAnimation">@anim/popwindowin</item>
        <item name="android:windowExitAnimation">@anim/popwindowout</item>
    </style>

</resources>
```

需要注意的是二维动画里的从起点按比例缩放和绝对值缩放是不同的， `popwindowin` 的详细定义如下

```
<?xml version="1.0" encoding="UTF-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android" >

    <scale
        android:fromXScale="1.0"
        android:toXScale="1.0"
        android:fromYScale="0.0"
        android:toYScale="1.0"
        android:pivotX="0"
        android:pivotY="0"
        android:duration="100" />

</set>
```

是Y轴上的按比例变化， `popwindoout` 与之类似，不过是反过来

```
<?xml version="1.0" encoding="UTF-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android" >

    <scale
    android:fromXScale="1.0"
    android:toXScale="1.0"
    android:fromYScale="1.0"
    android:toYScale="0.0"
    android:pivotX="0"
    android:pivotY="0"
    android:duration="100" />

</set>
```

其他

每个 `ListViewItem` 之间的分隔线通过 `list_item_divider.xml` 构造，这使得分隔线离左右两边屏幕边缘有一定的距离

```
<?xml version="1.0" encoding="UTF-8"?>
<inset xmlns:android="http://schemas.android.com/apk/res/android"
    android:insetLeft="72dp"
    android:insetRight="20dp"
    android:drawable="@color/grey">
</inset>
```

所有 `ListViewItem` 左边的圆形通过 `circle_bg.xml` 以一个十分简便的方法绘制

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="oval">

    <corners
        android:radius="30dp"/>

    <stroke
        android:width="1dp"
        android:color="@color/colorAccent" />

    <solid android:color="@color/transplant" />
</shape>
```

除此之外还更改了应用图标，这通过更改 `AndroidManifest.xml` 中 `application` 目录下的 `android:icon="@mipmap/radar"` 实现

所有矢量图标来自 [阿里巴巴矢量图标库](#)，非常不错的一个网站

模块设计

SharedPreferences 本地数据存取

本地数据的主要作用在于用户信息的保存，如为了获取可能的之前成功登录的账号，程序会首先通过一个 `SharedPreferences` 对象获取存储于本地的用户名密码，并使用此账户信息访问服务器，如果用户名密码正确则跳转

由于无存储用户名及密码的情况下 `SharedPreferences` 对象会默认返回一个空字符串，而注册账号密码时不允许注册空字符串为用户名或者密码，因此如果之前没有保存任何账户信息，则会使用不可能存在的账户密码登陆，登陆也就不会成功

```
//获取保存有登陆用户信息的 SharedPreferences 对象
loginStatus = getSharedPreferences("loginStatus", Context.MODE_PRIVATE);
//获取可编辑 loginStatus 的编辑器
sharedPreferencesEditor = loginStatus.edit();

//获取存储在本次的用户名和密码,如果没有则为空(注册时不能为空)
username = loginStatus.getString("username", "");
final String password = loginStatus.getString("password", "");

//向服务器检查用户名密码是否正确
netWork.checkAuthentication(username, password, checkHandler);
```

`checkAuthentication()` 的实现

```
//向服务器检查用户名密码是否正确
public void checkAuthentication(String username, String password, Handler lo
ginHandler)
{
    JSONObject jsonObj = new JSONObject();

    try
    {
        jsonObj.put("username", username);
        jsonObj.put("password", password);
    }
    catch(JSONException e)
    {

    }

    sendContent("http://sunzhongyang.com:7000/login", jsonObj.toString(), lo
ginHandler);
}
```

另外也有一些其他的作用，如下表所示，基本是为提供值而设计

value	note
username	已登录用户的名称，用于登陆时的校验，记录新邮件的所属以及自动填写发件人收件人，向服务器请求指定用户的邮件
password	已登录用户的密码，用于登陆时的校验
readstatus	值为 all 或者 unread，用于初始化 popwindow 时指示当前显示的是全部收件箱邮件还是未读的收件箱邮件
mailInform	值为 true 或者 false，用于指示是否进行新邮件的通知

网络访问

很多活动中的 `netWork` 对象是集成了一系列网络访问功能，集中处理网络方面任务的 `NetWork` 类的实例，其核心部分是 `sendContent()` 函数，能够向指定 URL 发送指定字符串，并指派必要的 `Handler` 去处理 UI 变化

```
public void sendContent(final String url, final String content, final Handle
r handler)
{
    //新建一个线程执行网络访问
    new Thread(){
        @Override
        public void run()
        {
            //建立并设置连接
            HttpURLConnection connection = null;
            try
            {
                connection = (HttpURLConnection) ((new URL(url).openConnecti
on()));
                connection.setRequestMethod("POST");
                connection.setReadTimeout(8000);
                connection.setConnectTimeout(8000);

                //向服务器发送信息
                DataOutputStream out = new DataOutputStream(connection.getOu
tputStream());

                out.writeBytes(content);

                //读取服务器返回的信息
                InputStream in = connection.getInputStream();
                BufferedReader reader = new BufferedReader(new InputStreamRe
ader(in));
                StringBuilder response = new StringBuilder();

                String line;
```

```
            while((line = reader.readLine()) != null)
            {
                response.append(line);
            }

            //向handler对象发送消息以传递数据
            Message message = new Message();
            message.what = 0;
            message.obj = response.toString();
            handler.sendMessage(message);
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
        //断开连接
        finally
        {
            if(connection != null)
            {
                connection.disconnect();
            }
        }
    }
}.start();
}
```

另外几个主要的函数和其用途如下

function	**note**
checkAuthentication(String username, String password, Handler loginHandler)	检查指定用户名密码是否合法
public void checkDuplicate(String name, Handler checkHandler)	检查指定用户名是否已经被注册
public void registerUser(String username, String password, Handler registerHandler)	为一个已经注册的用户更改密码
public void changePassword(String username, String oldPassword, String newPassword, Handler loginHandler)	值为 true 或者 false，用于指示是否进行新邮件的通知
public boolean isConnected(Activity activity)	视目前是否有网络连接返回 true 或者 false

需要与服务器通讯的函数以类似如下结构调用 `sendContent()`

```
//向服务器检查用户名密码是否正确
public void checkAuthentication(String username, String password, Handler lo
ginHandler)
{
    JSONObject jsonObj = new JSONObject();

    try
    {
        jsonObj.put("username", username);
        jsonObj.put("password", password);
    }
    catch(JSONException e)
    {

    }

    sendContent("http://sunzhongyang.com:7000/login", jsonObj.toString(), lo
ginHandler);
}
```

这里用了一个 `JSONObject` 构造 `JSON` 字符串，所有传向服务器的字符串均是 `JSON` 格式

Service

获取新邮件是通过一个 `Service` 实现的，用户登录成功后第一次进入主界面时，主界面会发送广播启动 `GetMailService` ，这一广播是静态广播

```
//发送广播启动 GetMailService, 开始接收邮件
Intent intent = new Intent(MainPage.this, GetMailService.class);
startService(intent);
```

启动广播后， `GetMailService` 会启动一个新线程，默认每2500毫秒检查一次新邮件，第一次检查是必须要设置的，否则检查不会开始

```
Runnable runnable = new Runnable()
{
    @Override
    public void run()
    {
        //通过网络控制器向服务器发送一个检查新邮件的请求
        netWork.sendContent(controller.receiveMailURL, loginStatus.getString
("username", "none") + "@sunzhongyang.com", handler);
        //netWork.sendContent(controller.receiveMailURL, "checkunseenmail",
handler);
        //每2.5秒检查一次新邮件
        handler.postDelayed(this, loginStatus.getInt("frequency", 2500));
    }
};

//开始第一次检查
handler.postDelayed(runnable, loginStatus.getInt("frequency", 2500))
```

`handler` 处理服务器的返回，如发送一个广播通知 `mailFrame` 更改 UI 等

```
//发送一个广播给 MailFrame, 更新含有所有邮件的 ListView
Intent intent = new Intent("android.intent.action.refreshUI");
sendBroadcast(intent);
```

传感器

刚才提到的更新 `mailFrame` UI 的广播是动态注册的

```
IntentFilter intentFilter = new IntentFilter("android.intent.action.refreshU
I");
getActivity().registerReceiver(mReceiver, intentFilter);
```

`onDestroy()` 中会自动取消

```
@Override
public void onDestroy()
{
    super.onDestroy();
    getActivity().unregisterReceiver(mReceiver);
}
```

`Receiver` 如下所示

```
mReceiver = new BroadcastReceiver()
{
    public void onReceive(Context context, Intent intent)
    {

        if(loginStatus.getString("mailinform", "true").equals("true"))
        {
            //振动一次
            Vibrator vibrator = (Vibrator) getActivity().getSystemService(Co
ntext.VIBRATOR_SERVICE);
            long [] pattern = {100, 400, 100, 400};
            vibrator.vibrate(pattern, -1);
        }

        //刷新主Activity界面
        onResume();
    }
};
```

这里会调用振动传感器振动一次，相关权限已经申请

Notification

`GetmailService` 同样会发送一个通知启动下拉菜单

```
//发送一个下拉之后可以看到的通知,包含邮件的发送者和邮件主题
Intent notificationIntent = new Intent("com.sunzhongyang.sjd.AndroidMail.sta
ticcreceiver");
```

下拉菜单如下所示

```
public class StaticReceiver extends BroadcastReceiver
{
    private static final String STATICACTION = "com.sunzhongyang.sjd.Android
Mail.staticcreceiver";

    public StaticReceiver()
    {

    }

    @Override
    public void onReceive(Context context, Intent intent)
    {
        //如果获取到的广播是某个准备响应的广播
        if(intent.getAction().equals(STATICACTION))
        {
            //获取bundle及其中的信息
            Bundle bundle = intent.getExtras();
```

```
            String sender = bundle.getString("sender");
            String subject = bundle.getString("subject");

            //获取保存有登陆用户信息的 SharedPreferences 对象
            SharedPreferences loginStatus = context.getSharedPreferences("lo
ginStatus", Context.MODE_PRIVATE);

            if(loginStatus.getString("mailinform", "true").equals("true"))
            {
                //创建一个通知
                Notification.Builder builder = new Notification.Builder(cont
ext);

                builder.setContentTitle(sender)
                        .setContentText(subject)
                        .setTicker("您有一封新邮件")
                        .setLargeIcon(BitmapFactory.decodeResource(context.g
etResources(), R.drawable.radar_1))
                        .setSmallIcon(R.drawable.radar_1)
                        .setAutoCancel(true);

                NotificationManager manager = (NotificationManager) context.
getSystemService(Context.NOTIFICATION_SERVICE);

                //设置点击通知为返回主界面
                Intent mintent = new Intent(context, MainPage.class);
                PendingIntent pendingIntent = PendingIntent.getActivity(cont
ext, 0, mintent, 0);
                builder.setContentIntent(pendingIntent);

                manager.notify(0, builder.build());
            }
        }
    }
}
```

SQLite

数据库的支持是必不可少的，一些数据库的功能被集中于 `SQLite` 类中，除了基本的建表，获取数据库实例外，还提供以下功能

将邮件保存至某个表

```
public void insertMail(MailItem mail, SQLiteDatabase db, String table)
{
    //通过ContentValues在数据库中增加新内容
    ContentValues cv = new ContentValues();
    cv.put("user", loginStatus.getString("username", "none"));
    cv.put("sender", mail.sender);
    cv.put("receiver", mail.receiver);
    cv.put("subject", mail.topic);
    cv.put("content", mail.mailContent);
    cv.put("time", mail.time);

    //写入数据库
    db.insert(table, null, cv);
}
```

`MailItem` 是一个包含邮件所有信息的类

根据 `ID` 从某个表获取邮件

```
public MailItem getMail(int id, Cursor full_cursor)
{
    MailItem mail = new MailItem();
    while(full_cursor.moveToNext())
    {
        if(full_cursor.getInt(0) == id)
        {
            mail.sender = full_cursor.getString(1);
            mail.receiver = full_cursor.getString(2);
            mail.topic = full_cursor.getString(3);
            mail.mailContent = full_cursor.getString(4);
            mail.time = full_cursor.getString(5);
        }
    }

    return mail;
}
```

返回一个 `MailItem` 对象

以及根据 `ID` 从某个表中删除数据

```
public void deleteMail(int id, SQLiteDatabase db, String table)
{
    db.execSQL("delete from " + table + " where id=" + id);
    db.close();
}
```

数据库设计

数据库采用用 `sqlite` 方式， `sqlite` 是一个本地的轻型的关系型数据库,支持一一般的 `sql` 操作，在本程序中数据库主要用于在本地存储用户收发，删除的邮件以及联系人信息，数据库一共有四个表，数据的具体存储格式如下

contact

field	**type**
id	INTEGER
name	TEXT
work	TEXT
address	TEXT
user	TEXT

inbox

field	**type**
id	INTEGER
sender	TEXT
receiver	TEXT
subject	TEXT
content	TEXT
time	TEXT
user	TEXT
status	TEXT

outbox

field	type
id	INTEGER
sender	TEXT
receiver	TEXT
subject	TEXT
content	TEXT
time	TEXT
user	TEXT

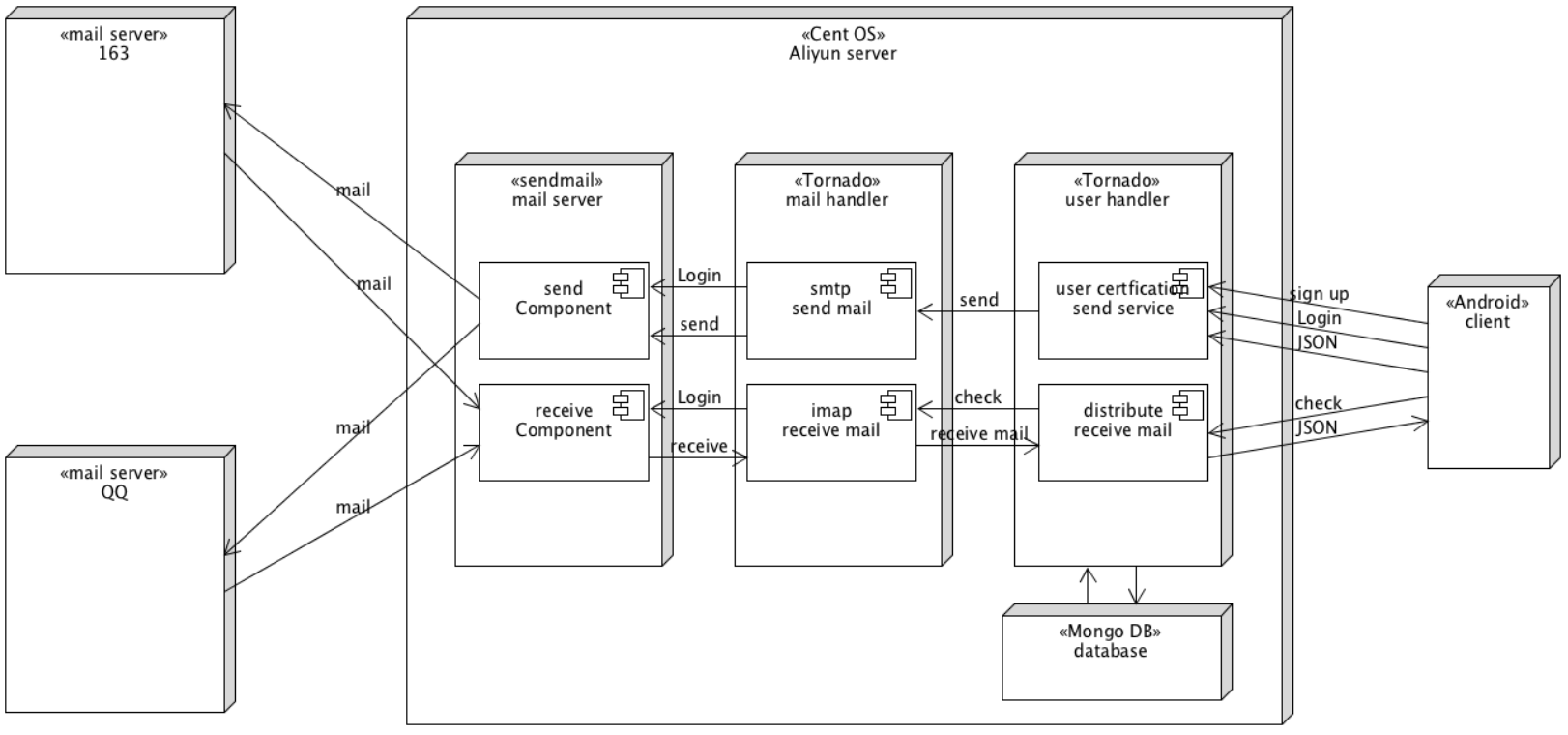
trashbox

field	type
id	INTEGER
sender	TEXT
receiver	TEXT
subject	TEXT
content	TEXT
time	TEXT
user	TEXT

其中 `user` 是邮件所属的用户， `inbox` 中的 `status` 标记邮件是已读或者未读

部署分析

部署图如下



服务端内部有两个组件，邮件服务器以及负责和客户端沟通的邮件处理服务器。邮件服务器采用 `Linux` 常用的 `sendmail` ，可以向其他邮件服务器发送邮件。比如图中的 163、QQ等。其内部的发件和收件模块分别有各自的用户名密码。邮件处理服务器采用 `Tornado` 架构，通过内部的两个模块（使用 `python` 库）与其通信

这一设计的具体优势在PPT中已经有所描述，简单来说是用 `tornado` 作一个中间件每3秒从邮件服务器获取所有未读邮件，在客户端请求时分发合适的邮件给用户，避免大量客户端直接访问邮件服务器造成对邮件服务器的高频次访问（ `tornado` 擅长处理高频词访问），另外也可以使用一些 `python` 的优势，这里不多赘述

后台设计

很明显，后台分为两个部分， `mail_handler` 和 `user_handler` ，其中 `mail_handler` 处理邮件， `user_handler` 处理用户信息（登陆、注册等）

例如处理注册请求，将用户名密码加入字典

```
#处理注册请求的类
class registerHandler(tornado.web.RequestHandler):
    def post(self):
        result = ""
        text = self.request.body
        text = str(text)

        userInformation = json.loads(text)
        print userInformation

        if user_password.has_key(userInformation['username']):
            result = "User exist"
        else:
            user_password[userInformation['username']] = userInformation['pa
ssword']
            result = "OK"
        self.write(result)
```

后台代码均有注释，实现上没有很大的区别

主要的问题

大部分的问题都被解决了，如 `tornado` 和邮件服务器的连接以及数据库要随取随用，不能设置全局变量，否则会超时失去连接等，目前遗留的问题主要在于使用除了 `szy` 之外的没有在 `linux` 系统中注册的用户不能接收其他邮件服务器发来的邮件，也就是说不能用163或者qq邮箱给 `szy` 之外的用户发送邮件，但是 `szy` 可以和其他用户互发邮件，其他用户也可以发送邮件到163，qq等。这主要是由于邮件服务器 `sendmail` 只接受 `linux` 系统账户名为用户名的邮件输入，`szy` ，`root` 等都是系统账户，`szy@sunzhongyang.com` 以及 `root@sunzhongyang.com` 就可以接受外部邮件服务器发来的邮件，而对应的 `test@sunzhongyang.com` 就没有这一能力。内部邮件互发已经通过一个小技巧在后台解决

另外一个问题是时间有限，受开发策略的限制一直没有能够充分的测试，缺乏足够的测试可能导致出现一些意想不到的问题，不过基本的使用已经能够满足了

总结与收获

总的来说，完成本次项目还是非常有收获的，邮件客户端这一主题相对较综合，涉及大量知识点，是对所学知识是一个很好地总结，经过这一次的练习，基本能够对安卓开发的基本方法有大致的把握，对于移动端程序的特点有比较充分的了解。这一次开发的规模比较大，充分巩固了知识，一段时间内应该是不会忘记了

最终的成品还是能够实际使用的，这一点符合我最初对于课程和程序的要求，学以致用的同时要求做出来的东西必须能用，不仅能够锻炼和检验开发的综合能力，也是所掌握知识的意义所在

比较遗憾的是完成时间远远超出预计，很多功能是在开发过程中不断构思出来的，这一方面导致开发时间被拉长，测试及撰写文档的时间被缩减，另一方面导致程序的复杂性大大增长，一些小的细节上

的改动会牵涉较大规模的更改，稍有不慎就会出现难以发现或者解决的bug。同时一些最初的设计比如数据库或者活动之间传递消息的接口并不能适应新的需求：或者太复杂以至于难于使用，或者太简单不能满足需求。不得不说最初的需求分析非常重要，这样后面迭代开发的部分需要少很多。迭代开发对软件设计的技巧以及开发流程开发顺序的掌握都有很高的要求，目前阶段还不应该过多涉及

时间关系有两个对实用性影响比较大但是和安卓开发关联较少的功能还没有加进去，一个是登录注册过程中的加密（RSA实现），数据库及密码MD5加盐以及邮件传输加密（DES实现）没有来得及做，没有加密那么服务器很可能会被欺骗。不过从实现角度来讲，JAVA以及Python的库都是充分的，如果要实现不会花费很多时间，后续的更新会放到我的GitHub上

一学期的课程下来学到不少内容，很感谢老师能够每周来上早课，大学城离市区有很长的路程。也感谢TA编写指导，并且亲自讲解以及检查，印象中每次检查的人都非常多，需要花费很多时间。除了每次的指导之外，完成作业的主要参考书主要是《第一行代码》，讲解循序渐进，由浅入深，是一本不错的入门书。经过这学期的课程，以后使用移动端程序也会从可能的内部原理在遇到问题时更好的解决使用中的问题，这一点我已经能够感受到了。