**2016年全国高校移动互联网**

**应用开发创新大赛**

**[作品名称]**

**设计文档**

所属组别： 本科组□ 高职高专组□

所属赛区：

所属学校：

团队名称：

指导老师：

队 长：

队 员： （至多三人）

目　　录

[一、概述](#_Toc398910541)

[1.1 选题背景](#_Toc398910542)

[1.2 项目意义](#_Toc398910543)

[二、可行性分析和目标群体](#_Toc398910544)

[2.1 可行性分析](#_Toc398910545)

[2.2 目标群体](#_Toc398910546)

[三、作品功能和原型设计](#_Toc398910547)

[3.1 总体功能结构](#_Toc398910548)

[3.2 具体功能模块设计](#_Toc398910549)

[3.3 界面设计](#_Toc398910554)

[四、作品实现、特色和难点](#_Toc398910557)

[4.1 作品实现](#_Toc398910558)

[4.2 特色分析](#_Toc398910559)

[4.3 难点和解决方案](#_Toc398910560)

[五、团队介绍和人员分工](#_Toc398910561)

[5.1 团队介绍](#_Toc398910562)

[5.2 人员分工](#_Toc398910563)

[六、其他](#_Toc398910564)

[6.1 开发结束后的感触](#_Toc398910565)

[6.2 不足之处和今后设想](#_Toc398910566)

[6.3 用到的第三方代码或组件](#_Toc398910567)

[七、致谢](#_Toc398910568)

1. 概述（一级标题，黑体，四号，段前1行，段后0行，单倍行距）

1.1选题背景（二级标题，黑体，四号，段前0行，段后0行，单倍行距）

（正文，宋体，小四，首行缩进2字符，段前0行，段后0行，行距固定值20磅；英文，Times New Roman，五号，段前0行，段后0行，单倍行距；图表中，中文，宋体，五号，英文，Calibri，五号）

1.2项目意义

二、可行性分析和目标群体

2.1 可行性分析

2.2 目标群体

三、作品功能和原型设计

3.1 总体功能结构

乐安全主要功能包括：安全功能：设置求救信息及联系人；常用功能：天气预报，穿衣提醒，新闻；娱乐功能：笑话。可以通过此应用预设求救信息，并快捷发送，同时可以获取不同类别的信息。既可确保用户独自出行遇到危险时能够及时发送求救信息，也能满足用户日常的休闲娱乐需求。

**安全功能：**可以预先设置好求救信息和信息的接收者，此功能为一项后台服务，退出应用后能正常使用。使用方式：连续点击电源键四下，手机震动表示消息发送成功，消息发送的同时获取当前的位置信息并发送给信息接收者。

**位置数据：**通过高德开放平台提供的Android定位SDK实现实时定位功能。

**电话拦截功能：**当陌生电话打入时提醒用户此电话为陌生号码，用户可选择是否拉入黑名单，拉入黑名单的电话再次打入时将自动挂断。

**天气预报功能：**自动获取用户所在地区，提供当前地区未来三天的天气情况，并根据当天的温度情况，给用户提供穿衣参考。

**新闻功能：**该模块以列表的形式提供实时新闻信息，点击可查看每条新闻的详细内容，下拉刷新列表内容，向下滑动可以自动加载更多。

**娱乐功能：**本模块通过开放接口获取网络搞笑段子，以列表形式显示，下滑列表可以自动加载数据，实现无限列表效果。当列表数据多于15条后，出现悬浮按钮可以快速返回到顶部。

页面主体使用了蓝色，辅色使用了红黄绿三色，整体色调的选择参考了Google logo的配色方案。

系统结构功能图 如图3-1所示：

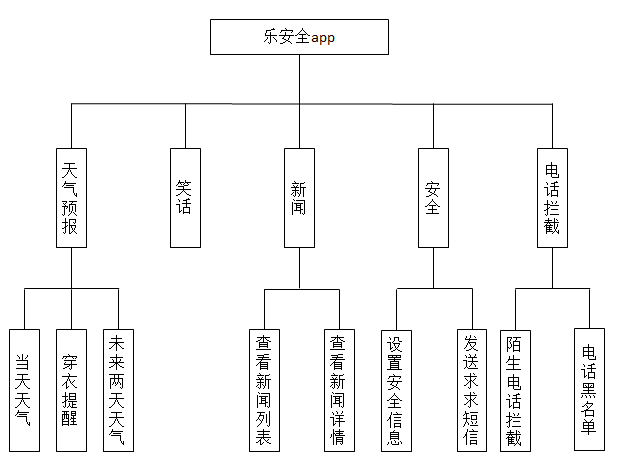


图3-1 系统结构功能图

3.2 界面设计

3.2.1 首页天气预报

首页的设计颜色风格使用的Google的logo的色彩搭配，首页的头部显示的是当前的定位信息，上半部分显示的是天气信息，根据不同的天气显示不同的天气图片，有一个可爱的卡通人物会根据不同的气温给用户穿衣提醒。卡通人物也会穿出不同的衣服。除了当天的天气外，还能预告未来两天的天气情况。首页下方还有其他功能的入口。如图4-1所示:



图4-1 首页

4.1.2 安全模块

安全模块使用了Service 功能，此页面用户填入信息和电话号码后点击设置求救服务按钮后程序获取用户填入的信息，使用SharedPreferences将数据储存到本地，每次重新设置后本地信息会得到更新，在设置完成同时打开服务，同时也可以使用下面的按钮关闭或开启服务，当服务开启后 ，只要连续按电源键就能触发此功能，为了避免误操作只有两次按键时间小于2秒才触发事件，就可以将设定好的信息加位置信息发送给设置好的收信人。定位功能主要集成来了高德地图的定位功能。效果如图4-2所示：

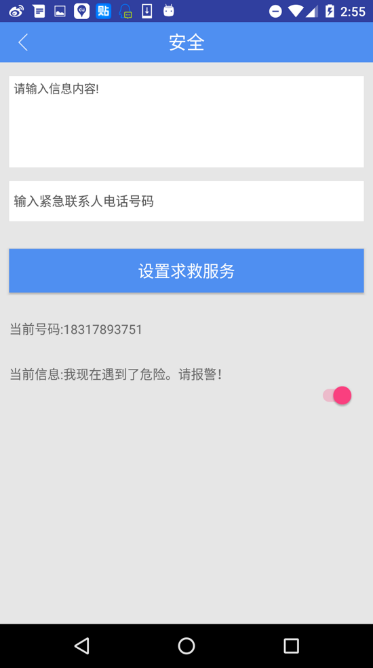


图4-2安全设置

4.1.3 电话拦截

此功能是为了防止电话骚扰，当有电话打进事，电话拦截服务就会获取打进来的电话的电话号码，然后会和手机的中储存的手机号码做匹配，如果没有此号码会提示用户是否拉入黑名单，电话一旦拉入黑名单后，如果再次打来此电话拦截功能会自动挂断电话。还可以将手机中已经存在的电话号码拉入黑名单，一样可以拦截，主要实现方法是监听电话状态，当有电话打来时，判断此号码是否在黑名单中，如果存在就调用系统挂断电话的功能来实现拦截。

如图4-3电话拦截所示:



图4-3电话拦截

4.1.5 新闻模块

新闻模块同样使用列RecyclerView 以列表的形式来展示接口返回的数据，新闻接口返回的信息主要包括，名称，收录时间，来源，详情的地址。刷新功能和加载更多功能和笑话模块相同。点击每条新闻可以查看新闻的详情，功能是跳转到一个activity中，再次activity中使用webview加载返回的详情地址。在加载的同时显了加载的进度，给用户一个提示，在网络较差的时候不至于让用户以为卡顿。如图4-5所示:

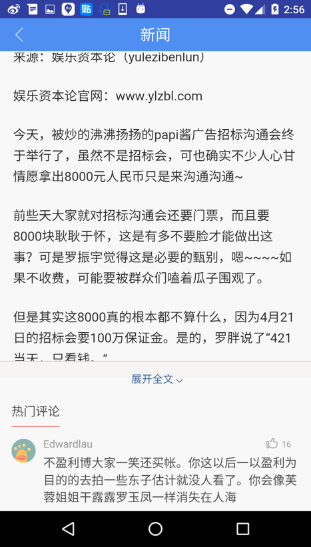


图4-5新闻

4.1.4 笑话模块

笑话模块主要使用列表展示了接口返回的笑话数据，数据返回内容包括笑话内容和作者信息，当列表处最上端时，下拉列表可以刷新数据，默认每次请求的数据为10条，向下滑动列表时，当屏幕最下方的item处于列表的倒数第三条是列表会自动加载更多数据，可以让用户感觉不到有加载更多的动作，能够增加用户体验。但下拉数据过多时，会出现一键返回顶部按钮，能快速的回到顶部，列表使用了RecyclerView，取代了listView，RecyclerView相比listView来说具备跟大的可塑性，并且RecyclerView自带的Adapter适配器能够自动的复用ContentView，能够起到优化内存的效果。笑话内容有的存在图片，使用的网络图片加载库picasso。如图4-4所示:

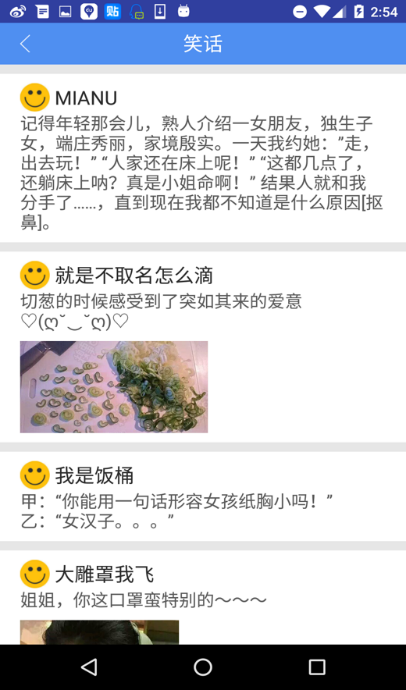


图4-4笑话

四、作品实现、特色和难点

4.1 作品实现

4.1.1 首页天气预报

首页的设计颜色风格使用的Google的logo的色彩搭配，首页的头部显示的是当前的定位信息，上半部分显示的是天气信息，根据不同的天气显示不同的天气图片，有一个可爱的卡通人物会根据不同的气温给用户穿衣提醒。卡通人物也会穿出不同的衣服。除了当天的天气外，还能预告未来两天的天气情况。首页下方还有其他功能的入口。如图4-1所示:



图4-1 首页

首页核心代码展示如下：

public void getData(String city) {

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");

String date = sdf.format(new java.util.Date());

NetService.getWhtear(city, date, "2").subscribe(new Action1<String>() {

@Override

public void call(String s) {

Logger.json(s);

Weather weather = NetUtil.gson.fromJson(s, Weather.class);

if (weather.status.equals("000000")) {

weatherList = weather.detail;

Logger.v(weatherList.size() + "");

updateUI();

} else {

MyToast.Short(HomeActivity.this, "请求出错!");

}

}

}, new Action1<Throwable>() {

@Override

public void call(Throwable throwable) {

Logger.v(throwable.toString());

MyToast.Short(HomeActivity.this, "网络出错!请检查网络!");

}

});

}

}

实现代码主要包括HomeActivity .java activity\_home .xml 文件

4.1.2 安全模块

安全模块使用了Service 功能，此页面用户填入信息和电话号码后点击设置求救服务按钮后程序获取用户填入的信息，使用SharedPreferences将数据储存到本地，每次重新设置后本地信息会得到更新，在设置完成同时打开服务，同时也可以使用下面的按钮关闭或开启服务，当服务开启后 ，只要连续按电源键就能触发此功能，为了避免误操作只有两次按键时间小于2秒才触发事件，就可以将设定好的信息加位置信息发送给设置好的收信人。定位功能主要集成来了高德地图的定位功能。效果如图4-2所示：

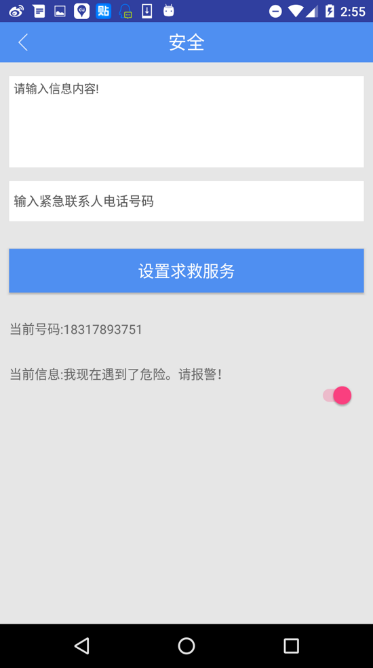


图4-2安全设置

高德地图集成主要步骤：首先导入AMap\_Location\_V2.4.0\_20160308.jar包

// onCreate()方法中注册

IntentFilter filter = new IntentFilter(Intent.ACTION\_SCREEN\_OFF);

registerReceiver(mBatInfoReceiver, filter);

locationClient = new AMapLocationClient(this.getApplicationContext());

locationOption = new AMapLocationClientOption();

// 设置定位模式为低功耗模式

locationOption.setLocationMode(AMapLocationClientOption.AMapLocationMode.Battery\_Saving);

// 设置定位监听

locationClient.setLocationListener(this);

locationClient.setLocationOption(locationOption);

// 启动定位

locationOption.setOnceLocation(true);

求救功能主要代码如下（服务类）

private final BroadcastReceiver mBatInfoReceiver = new BroadcastReceiver() {

@Override

public void onReceive(final Context context, final Intent intent) {

mContext = context;

final String action = intent.getAction();

if (Intent.ACTION\_SCREEN\_OFF.equals(action)) {

if (System.currentTimeMillis() - doExitTime > 2000) {

doExitTime = System.currentTimeMillis();

} else {

locationClient.startLocation();

Logger.v("测试成功!");

}

}

}

};

//定位完成

case Utils.MSG\_LOCATION\_FINISH:

AMapLocation loc = (AMapLocation) msg.obj;

String result = Utils.getLocationStr(loc);

Logger.v(result);

locationClient.stopLocation();

smsManager = SmsManager.getDefault();

paIntent = PendingIntent.getService(mContext, 0, new Intent(mContext, SafeService.class), 0);

VibratorUtil.Vibrate(getApplication(), 1000);

if (loc.getErrorCode() == 0) {

smsManager.sendTextMessage(SafeManager.getSafeNum(mContext).phone, null, SafeManager.getSafeNum(mContext).msg + " 位置: " + loc.getAddress(), paIntent, null);

} else {

smsManager.sendTextMessage(SafeManager.getSafeNum(mContext).phone, null, SafeManager.getSafeNum(mContext).msg + " 位置: " + loc.getAddress(), paIntent, null);

}

break;

实现代码主要包括：SafeService.java SafeActivity.java activity\_safe.xml

4.1.3 电话拦截

此功能是为了防止电话骚扰，当有电话打进事，电话拦截服务就会获取打进来的电话的电话号码，然后会和手机的中储存的手机号码做匹配，如果没有此号码会提示用户是否拉入黑名单，电话一旦拉入黑名单后，如果再次打来此电话拦截功能会自动挂断电话。还可以将手机中已经存在的电话号码拉入黑名单，一样可以拦截，主要实现方法是监听电话状态，当有电话打来时，判断此号码是否在黑名单中，如果存在就调用系统挂断电话的功能来实现拦截。

如图4-3电话拦截所示:



图4-3电话拦截

主要实现代码：

private ArrayList<String> getPhoneNum(Context context) {

ArrayList<String> numList = new ArrayList<String>();

//得到ContentResolver对象

ContentResolver cr = context.getContentResolver();

//取得电话本中开始一项的光标

Cursor cursor = cr.query(ContactsContract.Contacts.CONTENT\_URI, null, null, null, null);

while (cursor.moveToNext())

{

// 取得联系人ID

String contactId = cursor.getString(cursor.getColumnIndex(ContactsContract.Contacts.\_ID));

Cursor phone = cr.query(ContactsContract.CommonDataKinds.Phone.CONTENT\_URI, null,

ContactsContract.CommonDataKinds.Phone.CONTACT\_ID + " = " + contactId, null, null);

// 取得电话号码(可能存在多个号码)

while (phone.moveToNext())

{

String strPhoneNumber = phone.getString(phone.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER));

numList.add(strPhoneNumber);

Log.v("tag","strPhoneNumber:"+strPhoneNumber);

}

phone.close();

}

cursor.close();

return numList;

}

4.1.4 笑话模块

笑话模块主要使用列表展示了接口返回的笑话数据，数据返回内容包括笑话内容和作者信息，当列表处最上端时，下拉列表可以刷新数据，默认每次请求的数据为10条，向下滑动列表时，当屏幕最下方的item处于列表的倒数第三条是列表会自动加载更多数据，可以让用户感觉不到有加载更多的动作，能够增加用户体验。但下拉数据过多时，会出现一键返回顶部按钮，能快速的回到顶部，列表使用了RecyclerView，取代了listView，RecyclerView相比listView来说具备跟大的可塑性，并且RecyclerView自带的Adapter适配器能够自动的复用ContentView，能够起到优化内存的效果。笑话内容有的存在图片，使用的网络图片加载库picasso。如图4-4所示:

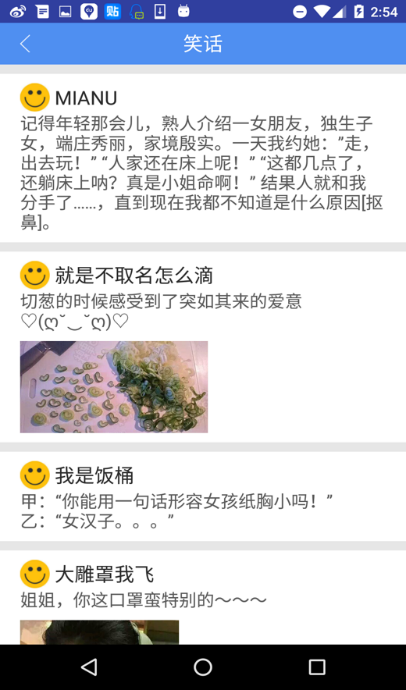


图4-4笑话

笑话模块核心代码：

protected void getData(final boolean b) {

NetService.getJokes(String.valueOf(page), String.valueOf(size)).subscribe(new Action1<String>() {

@Override

public void call(String s) {

Logger.json(s);

Jock jock = NetUtil.gson.fromJson(s, Jock.class);

if (jock.status.equals("000000")) {

if (b)

jockBeans.clear();

jockBeans.addAll(jock.detail);

page++;

isLoadMore = false;

adapter.notifyDataSetChanged();

} else {

MyToast.Short(JockActivity.this, "请求出错!");

}

mPush.setRefreshing(false);

}

}, new Action1<Throwable>() {

@Override

public void call(Throwable throwable) {

MyToast.Short(JockActivity.this, "网络错误!请检查网络!");

Logger.v(throwable.toString());

mPush.setRefreshing(false);

}

});

}

Adapter中关键代码

public void onBindViewHolder(MyViewHolder holder, int position) {

JockBean jockBean = list.get(position);

holder.author.setText(jockBean.author);

holder.text.setText(jockBean.content);

if (jockBean.picUrl != "") {

Picasso.with(context).load(jockBean.picUrl).into(holder.imageView);

holder.imageView.setVisibility(View.VISIBLE);

} else {

holder.imageView.setVisibility(View.GONE);

}

}

实现代码主要包括：JockActivity.java JockAdapter.java item\_joke.xml

4.1.5 新闻模块

新闻模块同样使用列RecyclerView 以列表的形式来展示接口返回的数据，新闻接口返回的信息主要包括，名称，收录时间，来源，详情的地址。刷新功能和加载更多功能和笑话模块相同。点击每条新闻可以查看新闻的详情，功能是跳转到一个activity中，再次activity中使用webview加载返回的详情地址。在加载的同时显了加载的进度，给用户一个提示，在网络较差的时候不至于让用户以为卡顿。如图4-5所示:

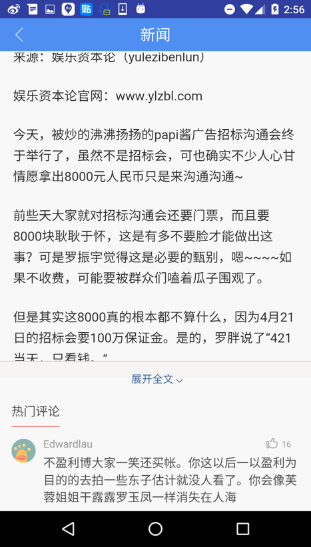


图4-5新闻

新闻模块核心代码：

protected void getData(final boolean b) {

NetService.getNews(String.valueOf(time), String.valueOf(size)).subscribe(new Action1<String>() {

@Override

public void call(String s) {

Logger.json(s);

New news = NetUtil.gson.fromJson(s, New.class);

if (news.status.equals("000000")) {

if (b)

newBeans.clear();

newBeans.addAll(news.detail);

isLoadMore = false;

adapter.notifyDataSetChanged();

time = Long.parseLong(newBeans.get(newBeans.size() - 1).behot\_time);

} else {

MyToast.Short(NewActivity.this, "请求出错!");

}

mPush.setRefreshing(false);

}

}, new Action1<Throwable>() {

@Override

public void call(Throwable throwable) {

Logger.v(throwable.toString());

MyToast.Short(NewActivity.this, "网络错误!请检查网络!");

mPush.setRefreshing(false);

}

});

}

Adapter中关键代码

public void onBindViewHolder(MyViewHolder holder, final int position) {

NewBean bean = list.get(position);

holder.title.setText(bean.title);

holder.from.setText(bean.source);

holder.time.setText(TimeUtil.getYear(bean.behot\_time));

holder.itemView.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

if (onItemClick != null) {

onItemClick.onClick(position);

}

}

});

}

实现代码主要功能：CommonHtmlActivity.java NewActivity .java NewAdapter.java

activity\_news.xml

4.3 难点和解决方案

### 4.3.1 数据的获取

本项目的数据获取使用的是第三方的提供的数据接口，新闻，天气，笑话。三个接口都是<http://ajita.iteye.com/blog/2188914>此网站的一个网友提供，数据格式返回为json格式，数据请求使用的是retrofit网络请求库。然后使用Gson解析返回的json串封装成实体类。

定位服务使用的是高德地图提供的定位服务，项目中集成高德定位SDK。请求高德定位接口获取准确的定位信息。

### 4.3.2 数据字典

表4-1 天气表

| 字段名 | 字段含义 | 类型 | 备注 |
| --- | --- | --- | --- |
| id | 天气id | String |  |
| city | 城市 | String |  |
| county | 县区 | String |  |
| date | 日期 | String |  |
| day\_temperature | 最高气温 | String |  |
| night\_temperature | 最低气温 | String |  |
| day\_wind | 风况 | String |  |

表4-2 新闻表

| 字段名 | 字段含义 | 类型 | 备注 |
| --- | --- | --- | --- |
| id | 新闻id | String |  |
| title | 标题 | String |  |
| article\_url | 新闻url地址 | String |  |
| source | 来源 | String |  |
| behot\_time | 收录时间 | String |  |

表4-3 笑话表

| 字段名 | 字段含义 | 类型 | 备注 |
| --- | --- | --- | --- |
| id | 笑话id | String |  |
| author | 作者 | String |  |
| content | 内容 | String |  |
| picUrl | 图片url | String |  |

### 4.3.3 数据的解析

请求接口后返回数据为json格式的字符串，本项目选用Google提供的Gson库来解析json。

首先在项目中添加Gson库的依赖：

compile 'com.squareup.retrofit2:converter-gson:2.0.0-beta4'

其次在工具类中实例化Gson 对象；

public static Gson gson = new GsonBuilder().serializeNulls().create();

最后直接将json串 解析成所对应的对象类Jock jock = NetUtil.gson.fromJson(s, Jock.class);Gson库自动将解析过后的数据赋值给实体类。

五、团队介绍和人员分工

5.1 团队介绍

团队名称GKATE，是geek and technique的缩写，极客、技术的含义，也正是团队的精神所在，时刻保持一个极客的心态，以创新、技术和时尚为工作意义，在不断掌握新的技术的同时，将所学知识运用于实践。

团队由河南大学软件学院王龙葛老师为学习指导核心，成员分别来自河南大学软件学院计算机科学与技术、网络工程两个不同的专业。目前团队在前端设计、网站实现、软件开发等方面都具有一定的项目经验，团队根据不同成员的专业技能，有针对性的培养，以达到团队全面发展的目标。

5.2 人员分工

|  |  |  |
| --- | --- | --- |
| 成员 | 角色 | 分工 |
| 张一飞 | 组长 | 项目管理，项目相关文档，项目测试 |
| 林佳辰 | 成员 | 数据库设计，代码优化 |
| 李嘉哲 | 成员 | 项目UI界面设计，代码优化 |

六、其他

6.1 开发结束后的感触

6.2 不足之处和今后设想

6.3 用到的第三方代码或组件

### 6.3.1 RxJava(RxAndroid) 的介绍和使用

#### 6.3.1.1 Rxjava(RxAndroid)的介绍

Rxjava 其实是一个实现异步操作的类。但是实现异步功能AsyncTask和Handler都能实现为什么又要用RxJava呢！原因就是RxJava简洁，随着逻辑变得越来越复杂，他依然能够保持简洁，举个简单的例子：例如自定义一个视图imageViews ，它可以用来同时显示多个图片，需要程序将一个现有的的目录数组File[] images 中每个目录下的图片加载出来显示，由于加载图片是个比较耗时的过程，因此需要在后台来完成，但是在android中显示图片又必须在主线程中执行。如果使用一般的方法来实现，代码中一定会出现很多的循环，使得代码逻辑繁琐，可读性降低。如果使用rxjava代码逻辑会变得更加的清晰，从上到下没有任的嵌套代码，而是一条链式的调用，这在逻辑上的简洁性具有优势，当需求变得复杂的时候这种优势将变得更加明显。

### 6.3.2 RxJava的观察者模式

RxJava有四个基本的概念：Observable（可观察者，也叫被观察者），Observe（观察者），subscribe（订阅，既事件）。Observable和Observe通过subscribe方法实现订阅的关系，就好比设置了监听事件一样。这样在一定条件下Observable就可以发出事件来通知Observe。

和一般的观察这模式不同的时，Rxjava的事件回调不仅仅只有事件onNext方法，onNext方法相当于onClick。它还定义了两个比较特别的事件：onCompleted

和onErro。

onCompleted():事件列队执行完调用此方法，Rxjava不仅仅把每个事件单独处理，还会把它们看做一个列队。在Rxjava中规定当不会再有新的onNext发出时。需要触发此方法作为标志。

onError():当事件队列中出现异常时调用此方法，同时事件队列会终止，不允许再有事件发生。

以上两个方法是互斥的二者只能其一被执行。

### 6.3.3 RxJava在本项目中的使用

Rxjava 在本项目中主要结合了retrofit网络请求库使用，核心代码如下：（以其中一个接口为例）

定义网络接口

/\*\*

\* Created by shen on 2016/3/21.

\*/

public interface NetInterface {

@GET("xiaohua/list.do ")

Observable<String> getJoke(@QueryMap Params params);

@GET("news/list.do ")

Observable<String> getNews(@QueryMap Params params);

@GET("weather/list.do")

Observable<String> getWeather(@QueryMap Params params);

}

/\*\*

\* Created by shen on 2016/3/21.

\* 网络请求工具类 实例Retrofit

\*/

public class NetUtil {

public static Gson gson = new GsonBuilder().serializeNulls().create();

public static Retrofit retrofit = new Retrofit.Builder().baseUrl("http://api.1-blog.com/biz/bizserver/")

.addCallAdapterFactory(RxJavaCallAdapterFactory.create())

.addConverterFactory(new StringConvertor())

.build();

public static NetInterface service = retrofit.create(NetInterface.class);

}

实例被观察者对象

public static Observable<String> getJokes(String page, String size) {

Params params = new Params();

params.put("size", size);

params.put("page", page);

return NetUtil.service.getJoke(params).compose(applySchedulers());

}

在activity中使用subscribe方法订阅被观察者

NetService.getJokes(String.valueOf(page), String.valueOf(size)).subscribe(new Action1<String>() {

@Override

public void call(String s) {

Logger.json(s);

Jock jock = NetUtil.gson.fromJson(s, Jock.class);

if (jock.status.equals("000000")) {

if (b)

jockBeans.clear();

jockBeans.addAll(jock.detail);

page++;

isLoadMore = false;

adapter.notifyDataSetChanged();

} else {

MyToast.Short(JockActivity.this, "请求出错!");

}

mPush.setRefreshing(false);

}

}, new Action1<Throwable>() {

@Override

public void call(Throwable throwable) {

MyToast.Short(JockActivity.this, "网络错误!请检查网络!");

Logger.v(throwable.toString());

mPush.setRefreshing(false);

}

});

### 6.3.4 ButterKnife的使用和介绍

ButterKnife 是一个开源库，通过注解的方法来取代了findViewById()方法，不需要声明过控件后再通过findViewById()方法来获得控件的实例，只需要在声明的时候加上注解，就可以获得控件的实例。解放控件对象实例化，也就是省掉findViewById（）；一直以来的做法都是一个一个的定义，然后再使用findViewById（）进行实例化，而使用ButterKnife，只需要在代码中使用注解方式进行对象申明，然后在 setContentView() 或 inflate() 之后调用一句话ButterKnife.bind(this);，那么申明的所有对象自动创建出来。Fragment的和adapter里也可以用，不过调用时要多加一个 view参数。解放监听事件，可以直接为R.id.submit这个控件添加OnClickListener为submit函数。

七、致谢

（若有三级、四级标题，黑体，小四，段前0.5行，段后0.5行，单倍行距）