中山大学移动信息工程学院本科生实验报告

（2017 年秋季学期）

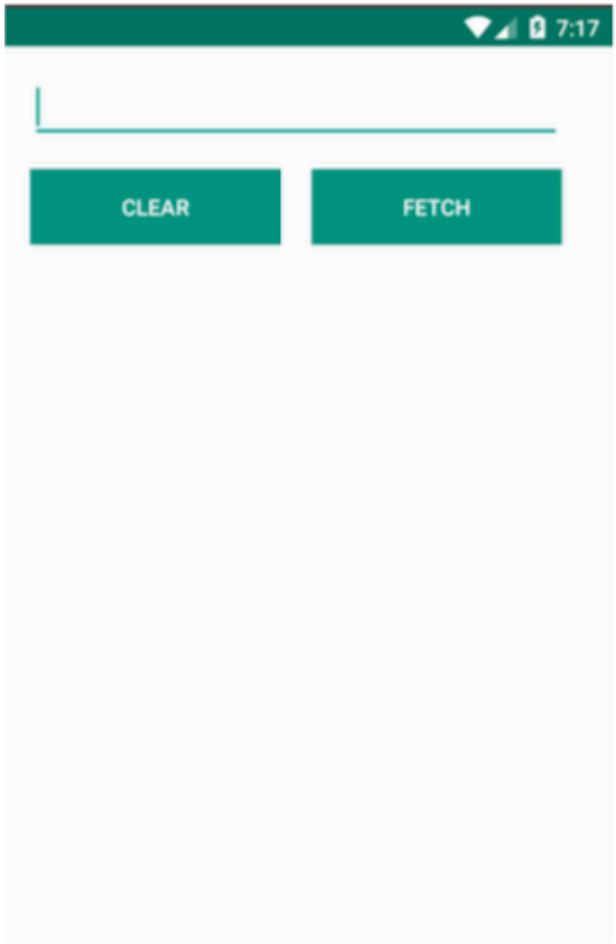
课程名称：移动应用开发　　任课教师： 郑贵锋

年级	2015 级	专业（方向）	移动信息工程（互联网方向）
学号	1535251	姓名	柯博
电话	13671412922	Email	kebo@mail2.sysu.edu.cn
开始日期		完成日期	

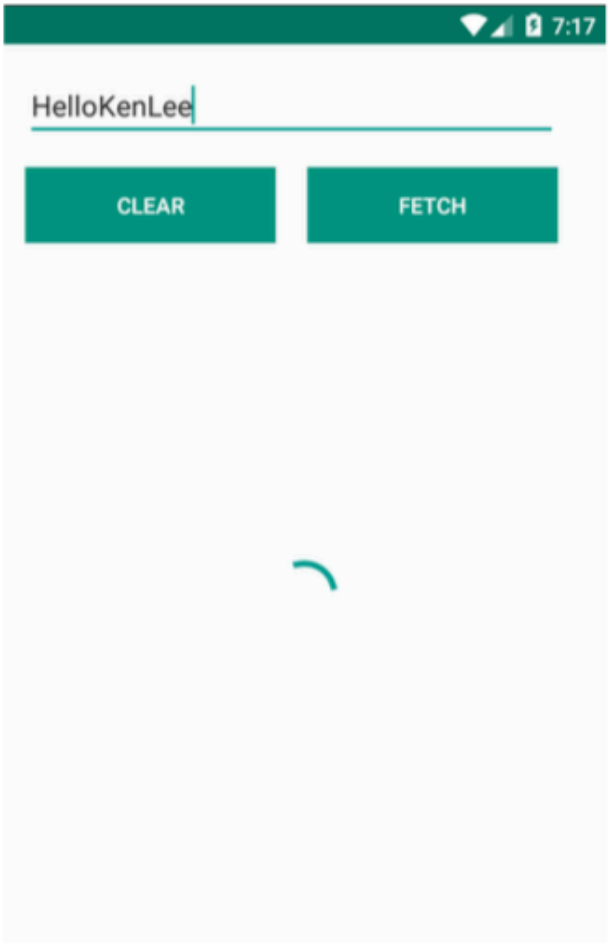
一、 实验题目

lab9 Retrofit+RxJava+OkHttp 实现网络请求

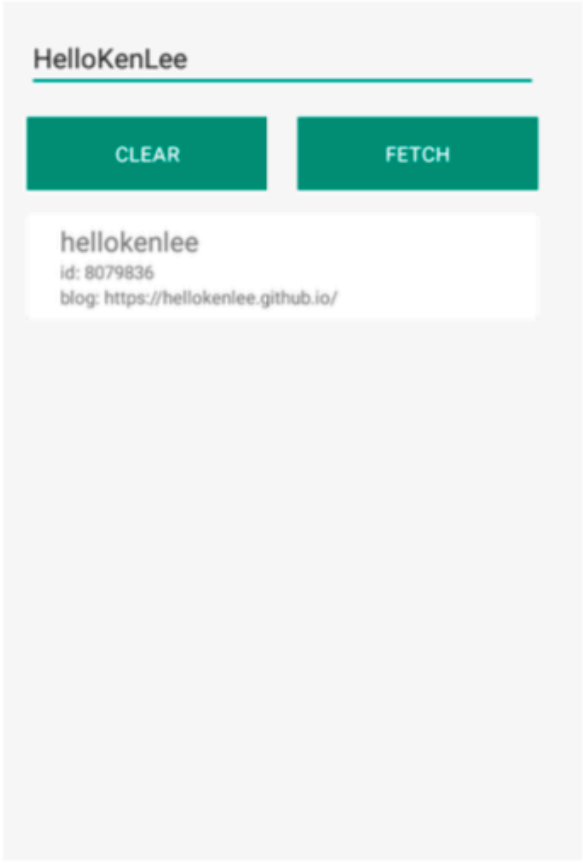
二、 实现内容



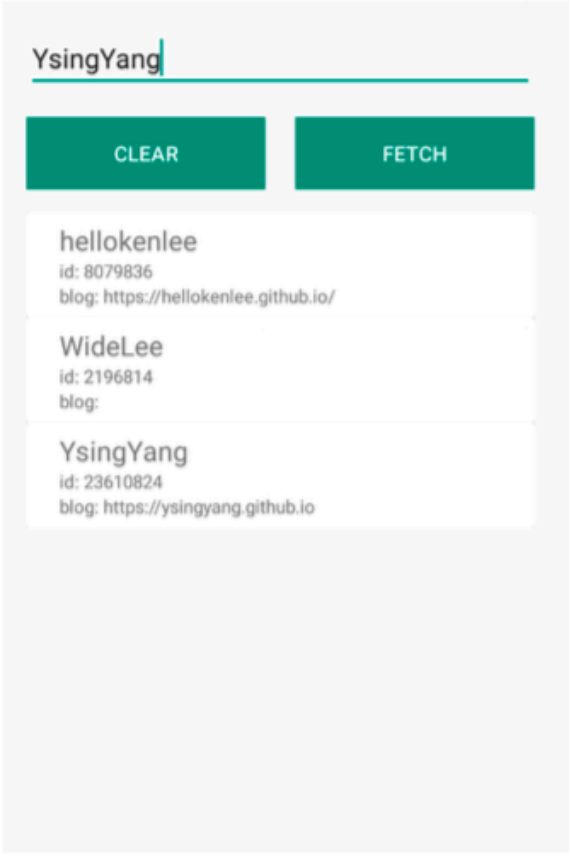
主界面



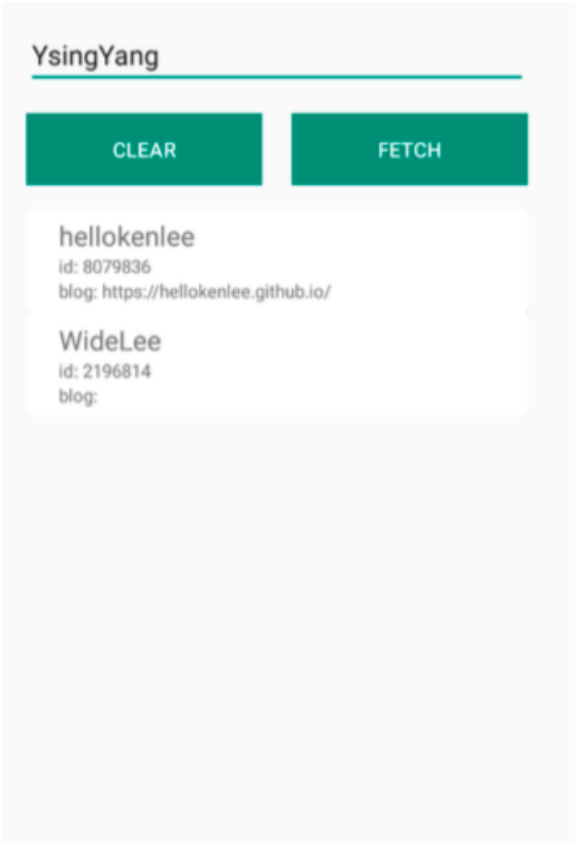
搜索用户



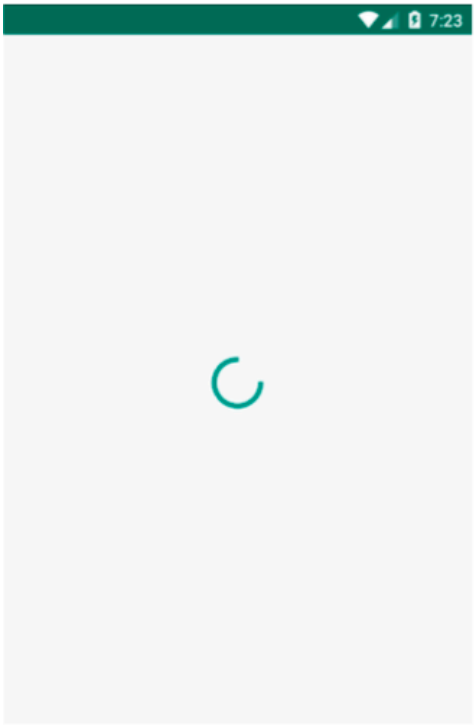
搜索结果

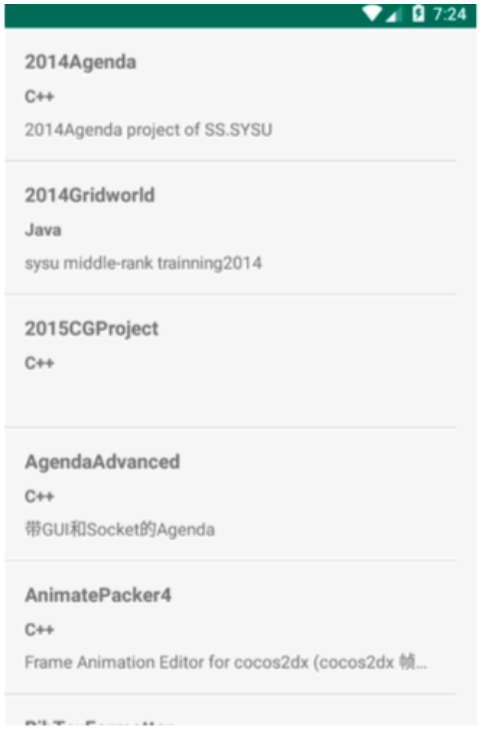


搜索结果



长按删除





点击进入个人详情页面　　详情信息获取显示

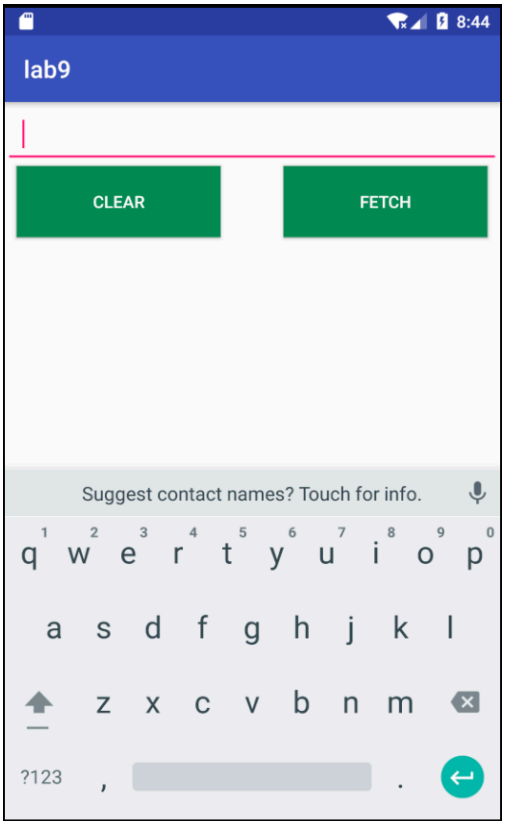
对于 User Model， 显示 id, login, blog

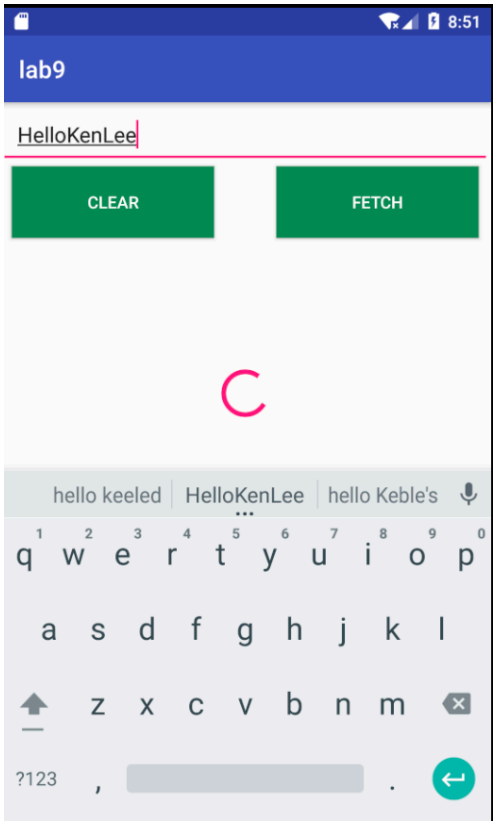
对于 Repository Model，显示 name, description，language

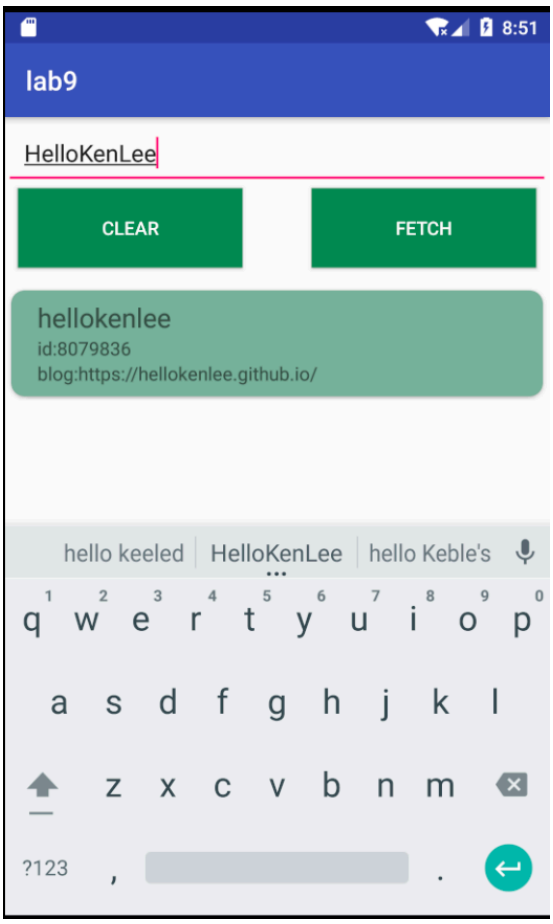
（特别注意，如果 description 多于一行要用省略号代替）

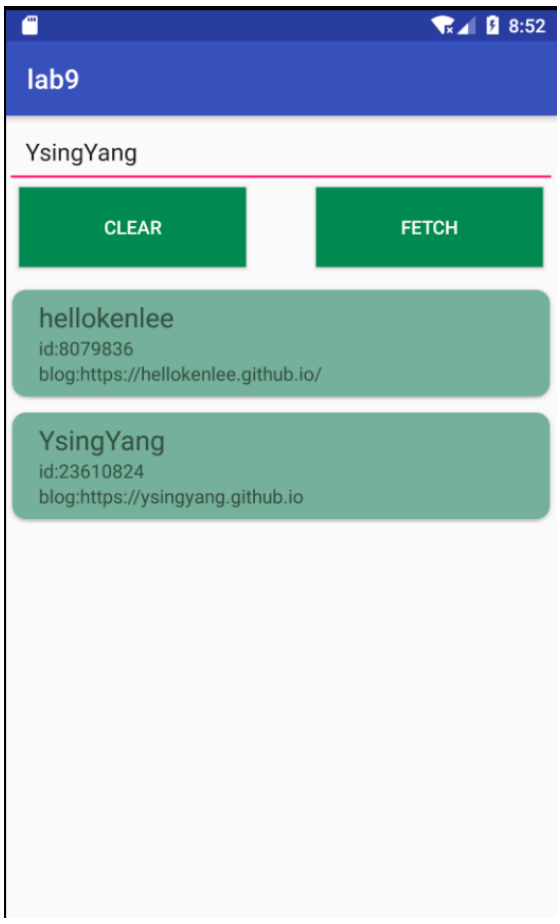
三、 课堂实验结果

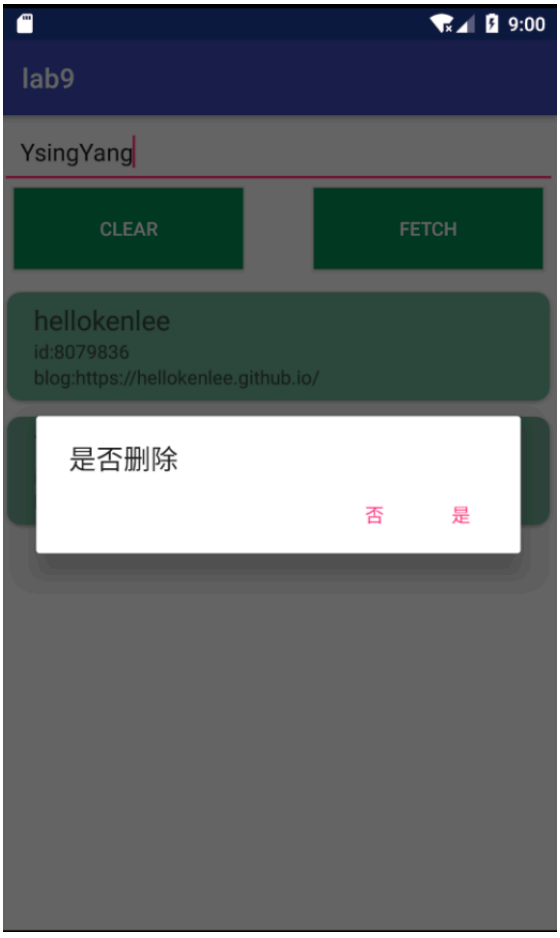
（1） 实验截图

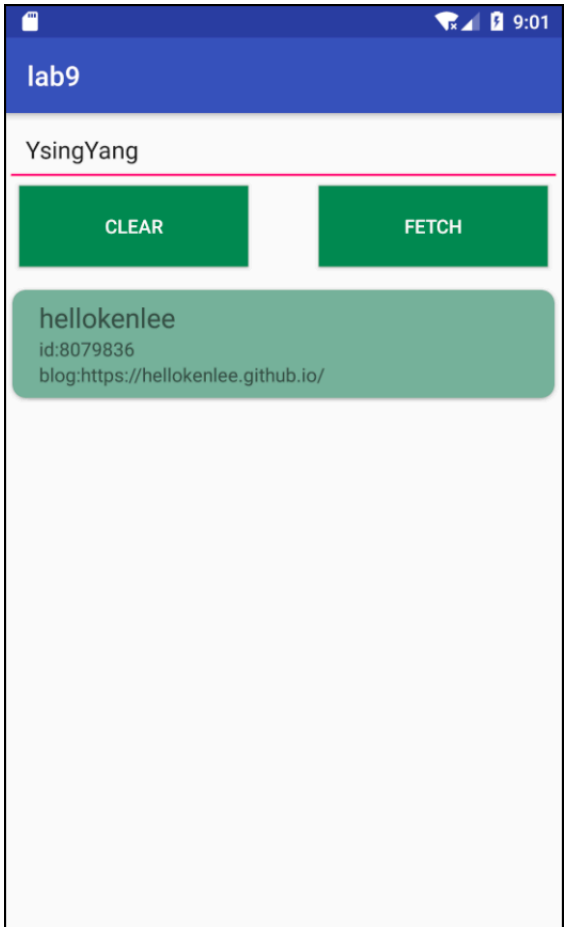


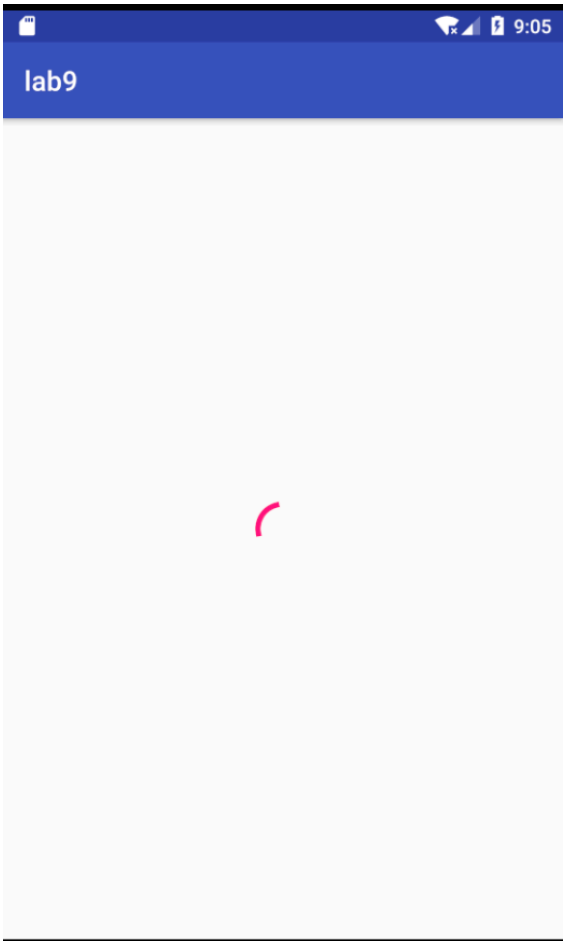



8:51
lab9
HelloKenLee
CLEAR
FETCH
hellokenlee
id:8079836
blog:https://hellokenlee.github.io/
hello keeled
HelloKenLee
hello Keble's


8:52
lab9
YsingYang
CLEAR
FETCH
hellokenlee
id:8079836
blog:https://hellokenlee.github.io/
YsingYang
id:23610824
blog:https://ysingyang.github.io


9:00
lab9
YsingYang
CLEAR
FETCH
hellokenlee
id:8079836
blog:https://hellokenlee.github.io/
是否删除
否
是


9:01
lab9
YsingYang
CLEAR
FETCH
hellokenlee
id:8079836
blog:https://hellokenlee.github.io/



（2） 实验步骤以及关键代码

发送 HTTP 请求给 Github 服务器之后，根据 Github API，会返回一个 JSON 格式的数据包，该数据包将会解包成 Java 实体类。因此先声明一个存放接收数据的类：

```
public class Github {
    private String login;//用户
    private String blog;//用户
    private int id;//用户
    private String name;//仓库
    private String language;//仓库
    private String description;//仓库
    public String getLogin(){return login;}
    public String getBlog(){return blog;}
    public int getId(){return id;}
    public String getName(){return name;}
    public String getLanguage(){return language;}
    public String getDescription(){return description;}
}
```

利用 RxJava 进行线程控制，进行网络请求访问。创建接口：

```
public interface GithubInterface{
    @GET("/users/{user}")
    Observable<Github> getUser(@Path("user") String user);
}
```

Observer 观察者设置：

```
ReposInterface reposInterface=ServiceGenerator.createService(ReposInterface.class);
reposInterface.getUser(name)
        .subscribeOn(Schedulers.newThread())
        .observeOn(AndroidSchedulers.mainThread())
        .subscribe(new Subscriber<List<Github>>() {
```

完整的网络请求部分代码如下：

```
@Override
public void onClick(View v) {
    if(v.getId()==R.id.fetch){
        EditText editText=(EditText) findViewById(R.id.edit);
        String name=editText.getText().toString();
        findViewById(R.id.user_mode_view).setVisibility(View.GONE);//View消失
        findViewById(R.id.user_progressbar).setVisibility(View.VISIBLE);//进度条出现，开始网络请求
        GithubInterface githubInterface=ServiceGenerator.createService(GithubInterface.class);
        githubInterface.getUser(name)
                .subscribeOn(Schedulers.newThread())
                .observeOn(AndroidSchedulers.mainThread())
                .subscribe(new Subscriber<Github>() {
                    @Override
                    public void onCompleted() {
                        findViewById(R.id.user_mode_view).setVisibility(View.VISIBLE);//View显示
                        findViewById(R.id.user_progressbar).setVisibility(View.GONE);//进度条消失
                    }

                    @Override
                    public void onError(Throwable e) {

                    }
                    @Override
                    public void onNext(Github github) {
                        //Toast.makeText(MainActivity.this,github.getId()+"",Toast.LENGTH_SHORT)
                        Map<String,String> tmp=new HashMap<>();
                        tmp.put("name",github.getLogin());
                        tmp.put("id_or_language","id:"+github.getId());
                        tmp.put("blog_or_description","blog:"+github.getBlog());
                        //Toast.makeText(MainActivity.this,github.getBlog(),Toast.LENGTH_SHORT).
                        list.add(tmp);
                        recyclerAdapter.notifyDataSetChanged();
                    }

                });
    }
    else if(v.getId()==R.id.clear){
        list.clear();
        recyclerAdapter.notifyDataSetChanged();
    }
}
```

由于本次实验需要 fetch 的数据有两部分：用户和仓库，于是采用了模版类来创建 Retrofit 对象实现网络访问服务：

```
public class ServiceGenerator {
    public static final String API_BASE_URL = "https://api.github.com";

    private static OkHttpClient.Builder httpClient = new OkHttpClient.Builder();

    private static Retrofit.Builder builder =
            new Retrofit.Builder()
                    .baseUrl(API_BASE_URL)
                    .addConverterFactory(GsonConverterFactory.create())
                    .addCallAdapterFactory(RxJavaCallAdapterFactory.create());

    public static <S> S createService(Class<S> serviceClass) {
        Retrofit retrofit = builder.client(httpClient.build()).build();
        return retrofit.create(serviceClass);
    }
}
```

同时更改获取的内容：

```
public interface ReposInterface{
    @GET("/users/{user}/repos")
    Observable<List<Github>> getUser(@Path("user") String user);
}
```

（3） 实验遇到困难以及解决思路

observer 方法不仅在 RxJava 包中出现，也出现在另外一个 SQL 语言包中，一开始没有注意到导致无法实例化订阅者方法，Debug 了好久，才在群友的帮助下解决。

四、 课后实验结果

五、 实验思考及感想

在 APP 上实现网络访问，说实话是头一遭。众所周知，网络已经成为现代人生活中不可或缺的一部分，无论初始设计目的是什么，开发者都会在应用中加入网络连接功能，就算是一款单机游戏，也会有需要联网的用户校验或者广告推送功能。与网络通信使用的是 RxJava 服务，根据资料，RxJava 有 1 和 2 两个版本，都可以使用，具体有什么区别尚待确认。还有就是 JSON 格式，这是一种数据交换格式，大多数互联网通信中使用的均是 JSON，之前在写爬虫和写 AI 实验时有稍微接触过，通过这次实验了解到 JSON 作为一种数据载体，其承载的内容是很丰富多样的，不只限于数组或者字符串等。

作业要求：

1. 命名要求: 学号_姓名_实验编号，例如 15330000_林 XX_lab1。
2. 实验报告提交格式为 pdf。
3. 实验内容不允许抄袭，我们要进行代码相似度对比。如发现抄袭，按 0 分处理。