# 中山大学移动信息工程学院本科生实验报告

课程名称：移动应用开发　　任课教师：郑贵锋

## 一、 实验题目

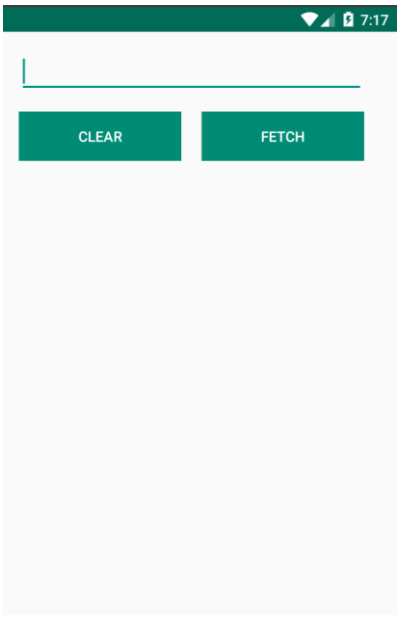
Retrofit+RxJava+OkHttp 实现网络请求

## 二、 实验目的

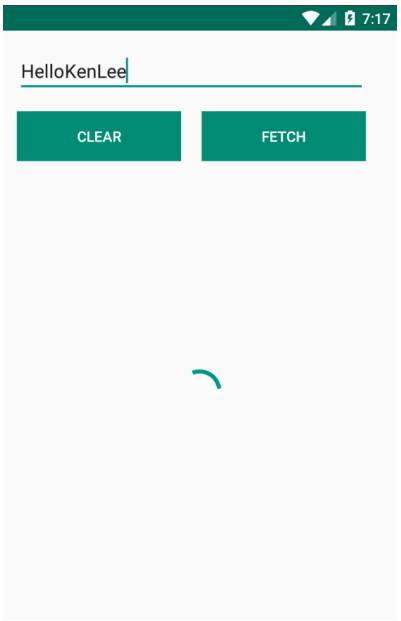
1. 学习使用 Retrofit 实现网络请求
2. 学习 RxJava 中 Observable 的使用
3. 复习同步异步概念

## 三、 实验内容

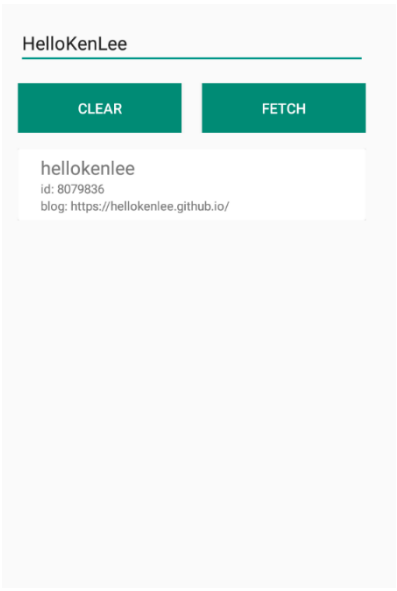
实现一个简单的 GitHub 界面如下：



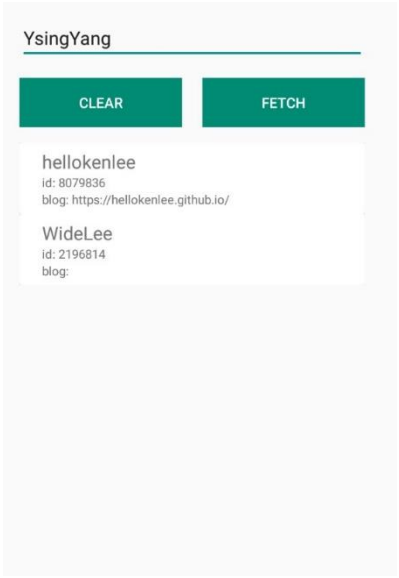

初始查询界面



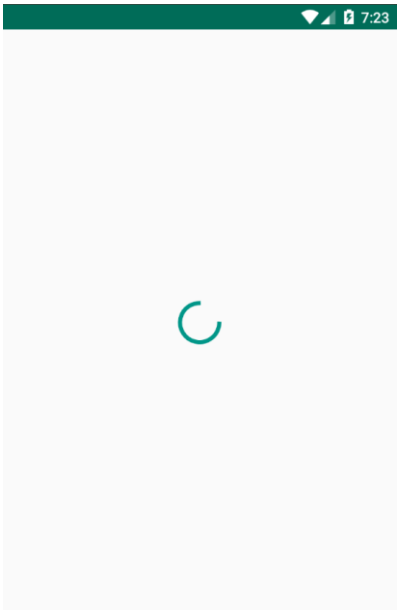

查询时界面



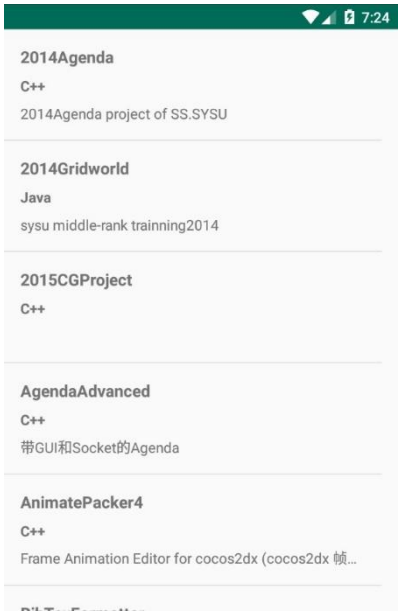

查询结果




长按删除




点击进入个人详情页面




详情信息获取显示

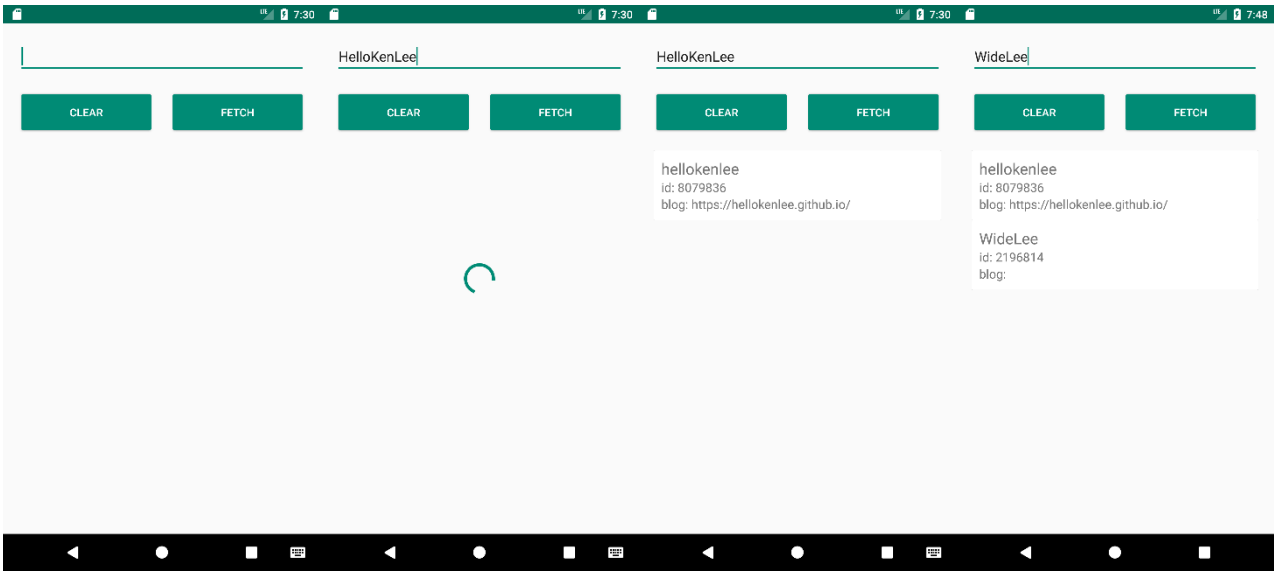
**具体要求如下：**

- 在查询界面时显示 id, login, blog；
- 在个人信息界面时显示 name, description, language(如果 description 超过 1 行要用省略号代替)。

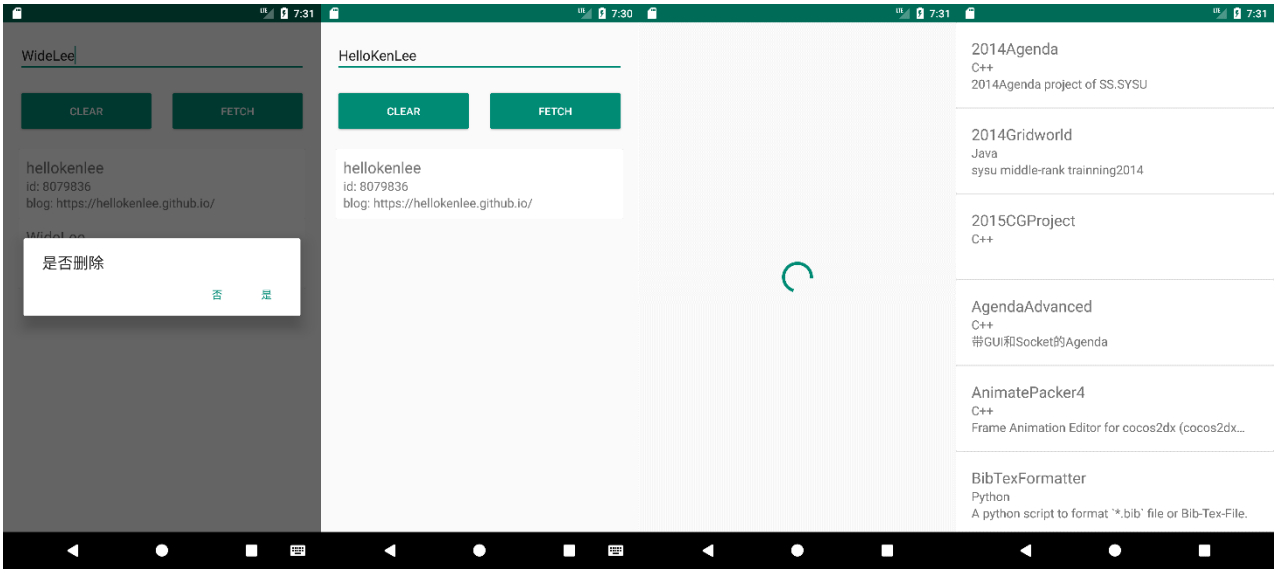
# 四、 课堂实验结果

## （一） 实验截图：

1. 虚拟机上效果如下：




初始界面　　查询时界面　　收索结果界面　　增加一个搜索结果

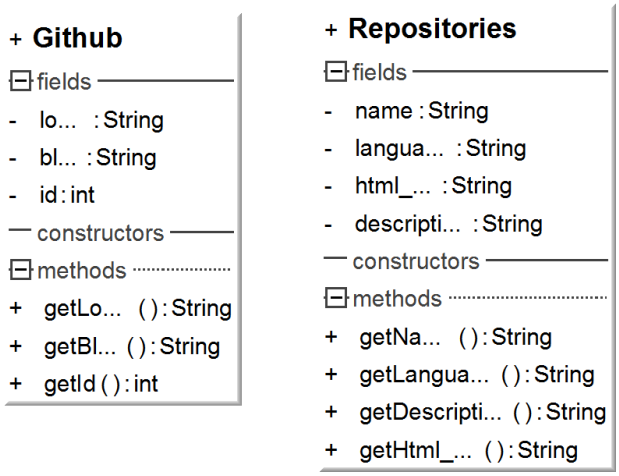



长按删除　　删除结果　　Description 进入界面　　Description 界面

## （二）　关键步骤：

### 1. Retrofit+RxJava+OkHttp 的使用

i. 根据我们的要求我们需要两个类一个用来存储 GitHub 的用户信息，一个用于存储用户的 Repository，我们先通过 API 查看获取到的数据格式，此时抛弃掉一些不需要的信息，所以我们定义两个类 GitHub 和 Repositories 类图如下：




- Github 类

```
public class Github {
    private String login; //登录名
    private String blog;//blog 地址
    private int id; //id
    public String getLogin(){return login;}
    public String getBlog(){return blog;}
    public int getId(){return id;}
}
```

- Repositories 类

```
public class Repositories {
    private String name;//repos 名字
    private String language;//语言
    private String html_url;//具体地址
    private String description;//描述
    public  String getName(){return name;}
    public  String getLanguage(){return  language;}
    public String getDescription(){return description;}
    public String getHtml_url(){return html_url;}
}
```

ii. 然后我们为这两个类定义对应的访问接口，retrofit 对象提供相应的 Interface，只需要提供相应的 URL，返回类型与参数即可。

- GitHub 的访问接口只需要直接从 GitHub 提供的 API 网址后加上/user/name，即可。

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

- Repos 的内容需要在 GitHub 提供的 API 网址后加上/user/name/repos,即可获取，此处注意一点返回的 Repos 不止 1 个，所以使用了 ArrayList 存储。

```
public interface ReposService {
    @GET("/users/{user}/repos")
    Observable<ArrayList<Repositories>> getRepos(@Path("user") String user);
}
```

iii. 然后我们为这两个类定义对应的访问接口，retrofit 对象提供相应的 Interface，只需要提供相应的 URL，返回类型与参数即可。

GitHub API 获取到的是一个 JSON 格式的数据，需要使用到 GsonConverter 将其转换为所需要的 model，使用 GsonConverter 需要添加相应的依赖。

```
compile 'com.squareup.retrofit2:converter-gson:2.1.0'
```

Retrofit 是基于 OkHttp 的封装，所以可以自己配置相应的 OKHttp 对象

```
private static OkHttpClient createOkHttp(){
    OkHttpClient okHttpClient = new OkHttpClient.Builder()
            .connectTimeout(10, TimeUnit.SECONDS)
            .readTimeout(30,TimeUnit.SECONDS)
            .writeTimeout(10,TimeUnit.SECONDS)
            .build();
    return okHttpClient;
}
```

然后构造 Retrofit 对象实现网络访问

```
public static Retrofit createRetrofit(String baseUrl) {
    return new Retrofit.Builder()
            .baseUrl(baseUrl)
            .addConverterFactory(GsonConverterFactory.create())
            .addCallAdapterFactory(RxJavaCallAdapterFactory.create())
            .client(createOkHttp())
            .build();
}
```

iv. 此处我们需要访问网络所以加上权限

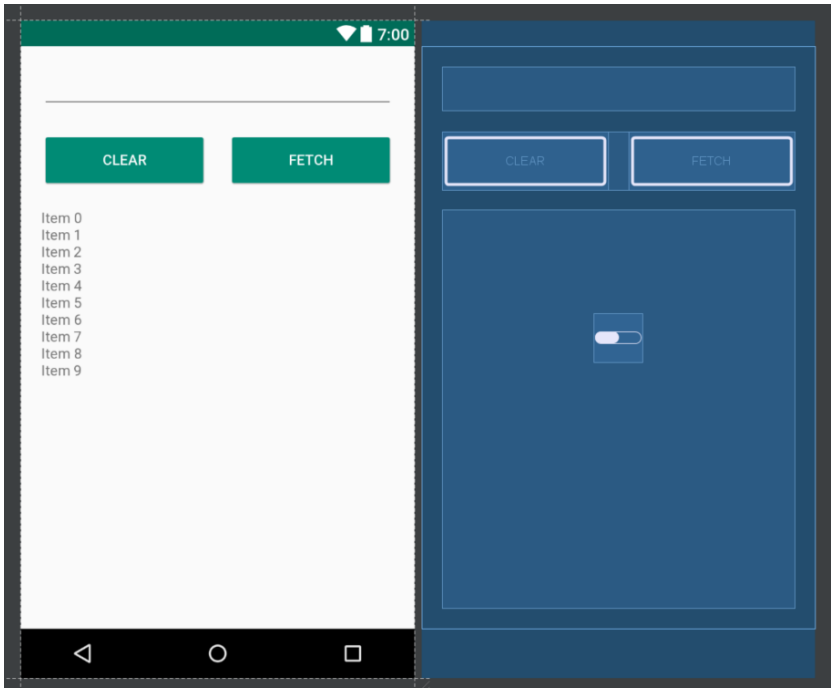
```
<uses-permission android:name="android.permission.INTERNET"/>
```

所有准备工作完成

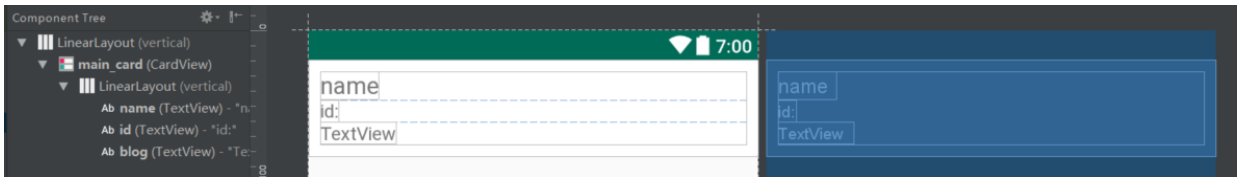
**2. 搜索界面**

i. 搜索界面由一个 EditText 两个 Button 和一个 RecycleView 构成，RecycleView 的 listitem 为使用 CardView 内部嵌套 3 个 TextView 构成，同时在加载时需要使用 progressBar。

界面布局如下：



主界面 listitem 布局如下：




此处不需要使用 CardView 的一些其他属性比如阴影圆角什么的，所以就没有加上去，直接套上 CardView

ii. 主界面使用 RecycleView，适配器使用之前 LAB3 中的适配器，修改填充方式即可。

```
public void convert(ViewHolder viewHolder, Github github){
    TextView name = viewHolder.getView(R.id.name);
    name.setText(github.getLogin());
    TextView id = viewHolder.getView(R.id.id);
    id.setText("id: "+String.valueOf(github.getId()));
    TextView blog = viewHolder.getView(R.id.blog);
    blog.setText("blog: "+github.getBlog());
}
```

iii. 点击 Clear 按钮清除所有数据，这里我们直接移除所有数据即可

```
clear.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {//清空
        for(int i=githubArrayList.size()-1;i>=0;i--){
            githubArrayList.remove(i);
            adapter.notifyDataSetChanged();
        }
    }
});
```

iv. 主界面的数据获取，当我们输入名字点击 Fetch 时候开始搜索，这里使用的是 Retrofit 加入 RxJava 的 Observable 对象。需要重载 3 个函数 onCompleted 函数为请求结束时调用的回调函数，onNext 表示收到每一次数据时调用的函数，onError 表示请求出现错误时调用的函数。搜索结果加入到数据 ArrayList 中，但是需要先判断是不是已经存在，所以此处多加一个 map

```
fetch.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String info;
        info=text.getText().toString();
        if(info.length()==0){//判断为空
            Toast.makeText(MainActivity.this,"输入为空",Toast.LENGTH_SHORT).show();
        }
        else{
            progressBar.setVisibility(View.VISIBLE); //切换显示
            recyclerView.setVisibility(View.GONE);
            GithubService githubService=
ServiceFactory.createRetrofit(GITHUB_API_URL).create(GithubService.class);
            githubService.getUser(info)
                    .subscribeOn(Schedulers.newThread())//请求在新的线程中执行
                    .observeOn(AndroidSchedulers.mainThread())
                    .subscribe(new Subscriber<Github>() {
                        @Override
                        public void onCompleted() { //数据获取完成
                            progressBar.setVisibility(View.GONE);
                            recyclerView.setVisibility(View.VISIBLE);
                        }

                        @Override
                        public void onError(Throwable e) { //出错
                            Toast.makeText(MainActivity.this,e.hashCode()+"请确认你搜索的用户存在
",Toast.LENGTH_SHORT).show();
                            progressBar.setVisibility(View.GONE);
                            recyclerView.setVisibility(View.VISIBLE);
                        }

                        @Override
                        public void onNext(Github github) {//获取到数据
                            if(!map.containsKey(github.getLogin())){//如果没有加入到数据列表中
                                githubArrayList.add(github);
                                map.put(github.getLogin(),1);
```

```
                                adapter.notifyDataSetChanged();
                            }
                        }
                    });
            }
        }
});
```

v. 点击 RecycleView 的 item 进入详细信息界面，长按删除，此处只要对两个对应的点击事件进行处理即可。

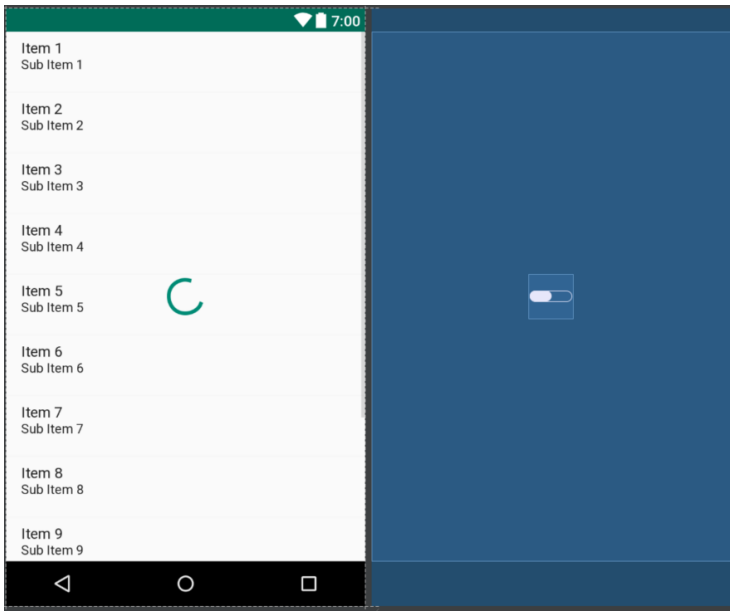
```
adapter.setOnItemClickListener(new RecyclerViewAdapter.OnItemClickListener() {
    @Override
    public void onClick(int position) { //跳转
        Intent intent=new Intent(MainActivity.this,RepositoryActivity.class);
        Bundle bundle = new Bundle(); //传递用户名
        bundle.putString("name",githubArrayList.get(position).getLogin());
        intent.putExtra("main",bundle);
        startActivity(intent);
    }

    @Override
    public void onLongClick(final int position) {//长按弹出删除
        AlertDialog.Builder builder=new AlertDialog.Builder(MainActivity.this);
        builder.setTitle("是否删除");
        builder.setNegativeButton("否", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {

            }
        });
        builder.setPositiveButton("是", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {//删除数据
                map.remove(githubArrayList.get(position).getLogin());
                githubArrayList.remove(position);
                adapter.notifyDataSetChanged();
            }
        });
        builder.show();
    }
});
```

### 3. Repos 列表界面

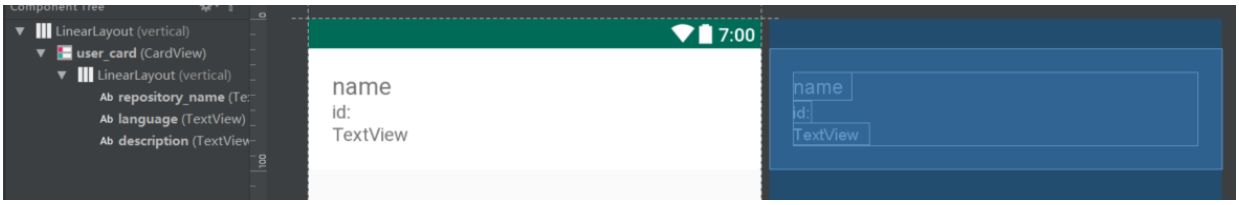
i. 界面设计只需要使用一个 ListView，即可。Listitem 也只需使用 3 个 TextView，外层嵌套 CardView。注意的一点就是 description 需要显示在一行之内，所以此处只需要额外设置 TextView 的属性。

```
android:ellipsize="end"
android:singleLine="true"
```

同样此处在加载时需要使用 processbar 过渡，界面设计如下：



Listitem 设计如下：




ii. 此处只需要获取到主界面传过来的用户名，然后类似的调用之前准备的 ReposService 接口。同样使用 Retrofit+RxJava+OkHttp 获取所有的 repos 数据即可。

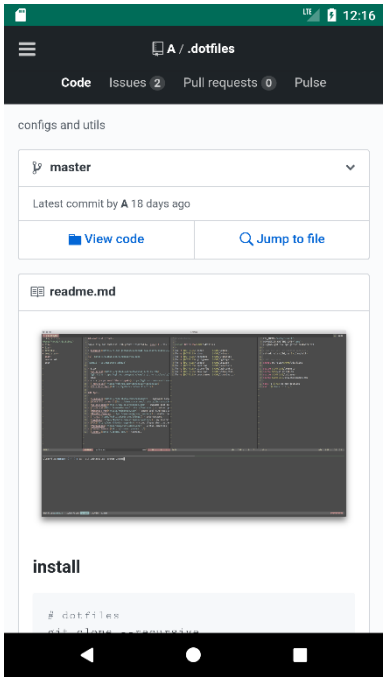
```
Bundle extras = getIntent().getBundleExtra("main");
if (extras != null) {
    name=extras.getString("name");
    progressBar.setVisibility(View.VISIBLE);
    listView.setVisibility(View.GONE);
    ReposService reposService= ServiceFactory.createRetrofit(GITHUB_API_URL).create(ReposService.class);
    reposService.getRepos(name)
            .subscribeOn(Schedulers.newThread())//请求在新的线程中执行
            .observeOn(AndroidSchedulers.mainThread())
            .subscribe(new Subscriber<ArrayList<Repositories>>() {
                @Override
                public void onCompleted() {//完成
                    progressBar.setVisibility(View.GONE);
                    listView.setVisibility(View.VISIBLE);
                    //设置适配器
                    adapter=new ListViewAdapter(RepositoryActivity.this,arrayList);
                    listView.setAdapter(adapter);
                    Listener();
                }
                @Override
                public void onError(Throwable e) {//出错
                    Toast.makeText(RepositoryActivity.this, e.hashCode() + "空", Toast.LENGTH_SHORT).show();
                    progressBar.setVisibility(View.GONE);
                    listView.setVisibility(View.VISIBLE);
                }

                @Override
                public void onNext(ArrayList<Repositories> repositories) {//获取全部的 repos
                    arrayList=repositories;
                }
            });
}
```

实验中遇到的困难和解决思路：

1. 接口的时候因为 import rx.Observable;这个其自己变成了数据库的 Observable，然后出现了报错，由于代码分开，所以没有注意到在接口定义的时候的 import 出错了，之后重新检查代码之后才发现了提示报错是在接口定义处。
2. onNext 返回的问题，一开始直接看说明以为是如果有多个 repos 的时候一次返回一个，然后使用发现做法不对，之后认真看了说明，然后看了一下 github api repos 的结构才发现这是一个 json 数组类的，所以一次返回的应该是一个数组，所以在 repos 获取中的接口要被定义为返回的是一个数组，Observable<ArrayList<Repositories>> getRepos。之后处理的时候也要用数组的方式处理。

# 五、 课后实验结果

1. 增加一个点击 repos 会自动转跳到那个 repos 页面的功能，实现起来比较简单，就是增加一个 webView，点击后使用我们之前获取到的 repos 的网址作为参数，设置 webView 的载入网址。效果如下：




# 六、 实验思考及感想

这次实验是进行网络访问，不过不用自己实现 http 访问，直接使用 Retrofit、RxJava、OkHttp，不用自己新建子线程实现异步获取，实验要求的网络方式实现起来友好很多。但是主要问题还是文档写得不是很清楚，要用什么东西都写了就是，但是突然跳出一段没有上下文的代码，也没有什么关键注释真的让人难以理解，整个文档除了用来看看要用什么之外，其他东西的使用完全要依靠自己去自学。

实验的话就是学习如何使用这些，使用方式自己去找一下博客或者看一下官方的文档很简单，不过要注意的就是使用的时候要根据需要的 api 进行代码的设置，不同的 api 可能返回的东西不一样，需要我们根据 api 修改代码，不然的话获取不到数据，其他并不困难。