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	专业（方向）	移动互联网
学号	15352344	姓名	吴文标
电话	13112312320	Email	wwb.bill@qq.com
开始日期	2017.12.26	完成日期	2017.12.27

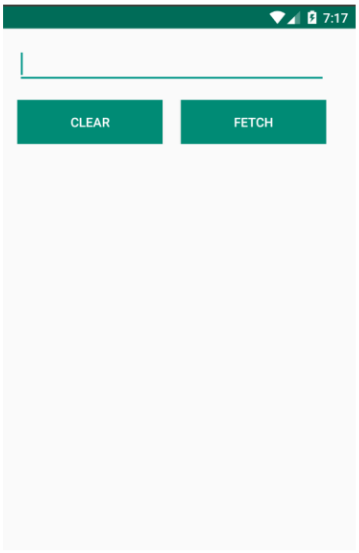
一、 实验题目

实验九 Retrofit+RxJava+OkHttp 实现网络请求

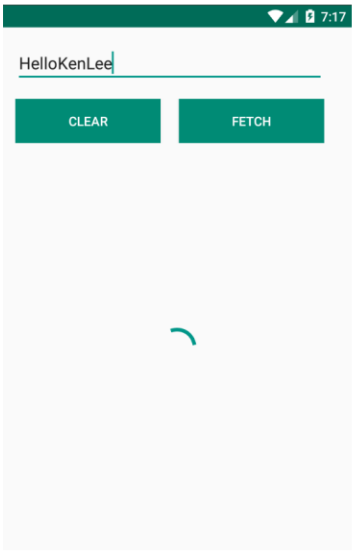
二、 实验目的

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。

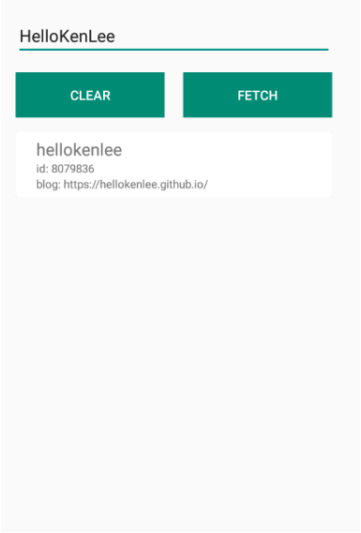
三、 实现内容



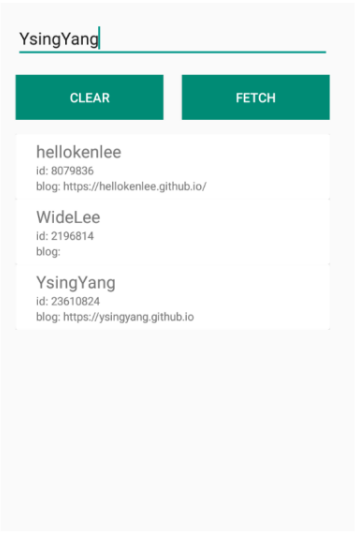
主界面



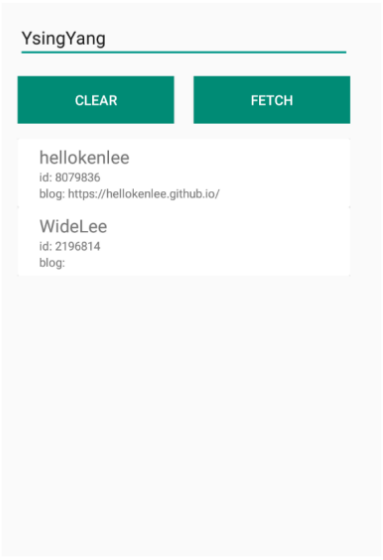
搜索用户



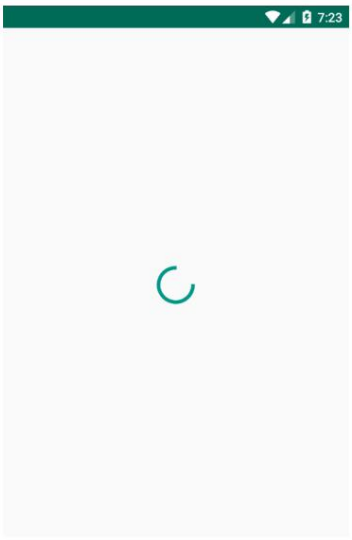
搜索结果



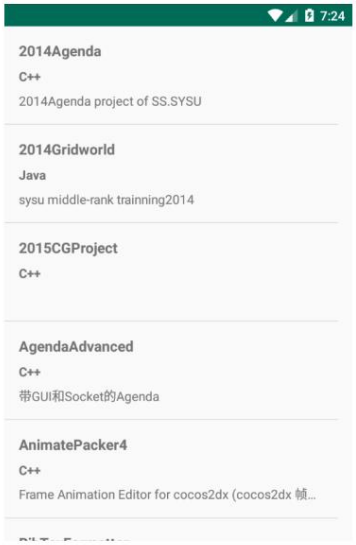
搜索结果



长按删除



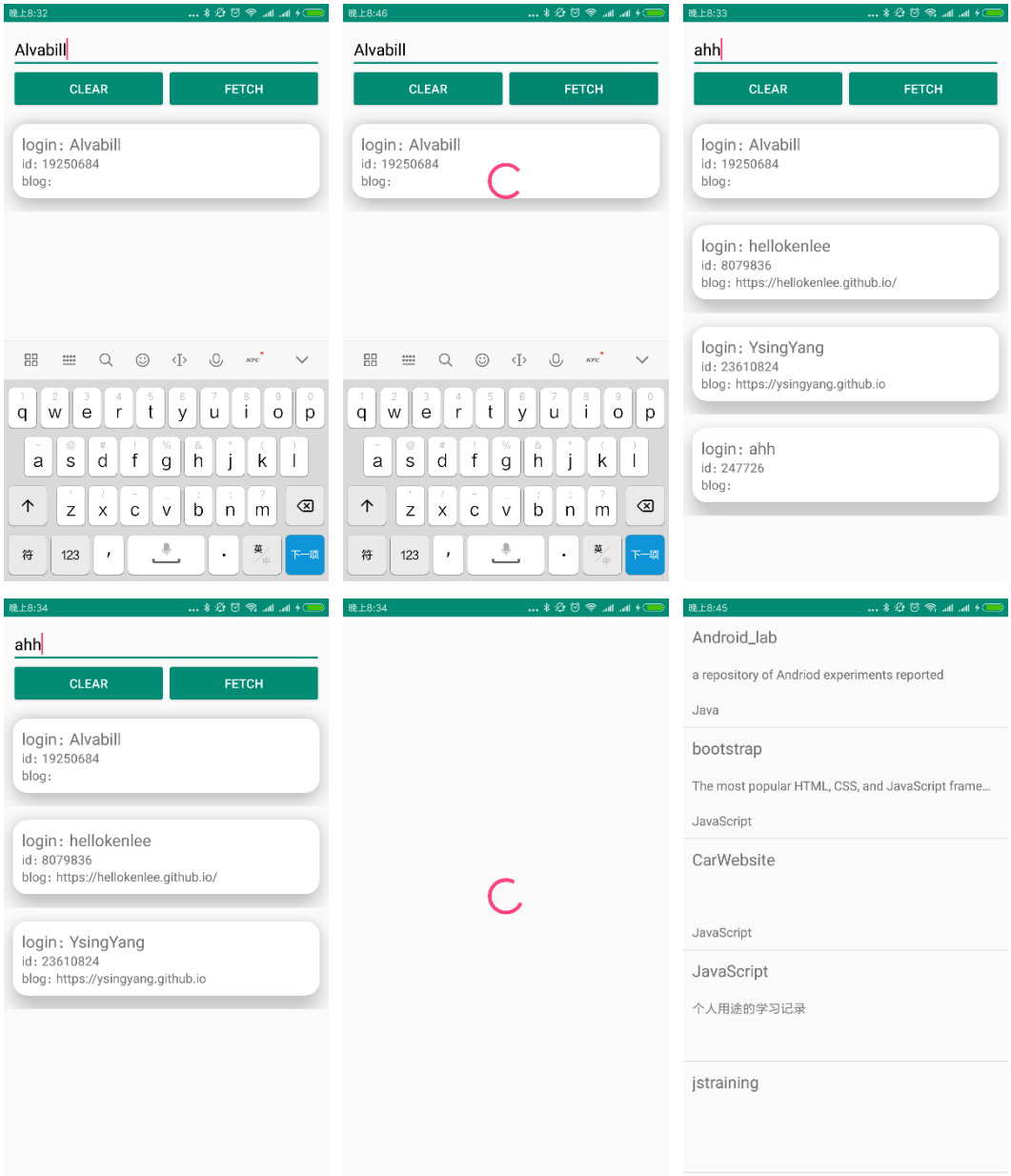
点击进入个人详情页面



详情信息获取显示

四、 课堂实验结果

（1） 实验截图

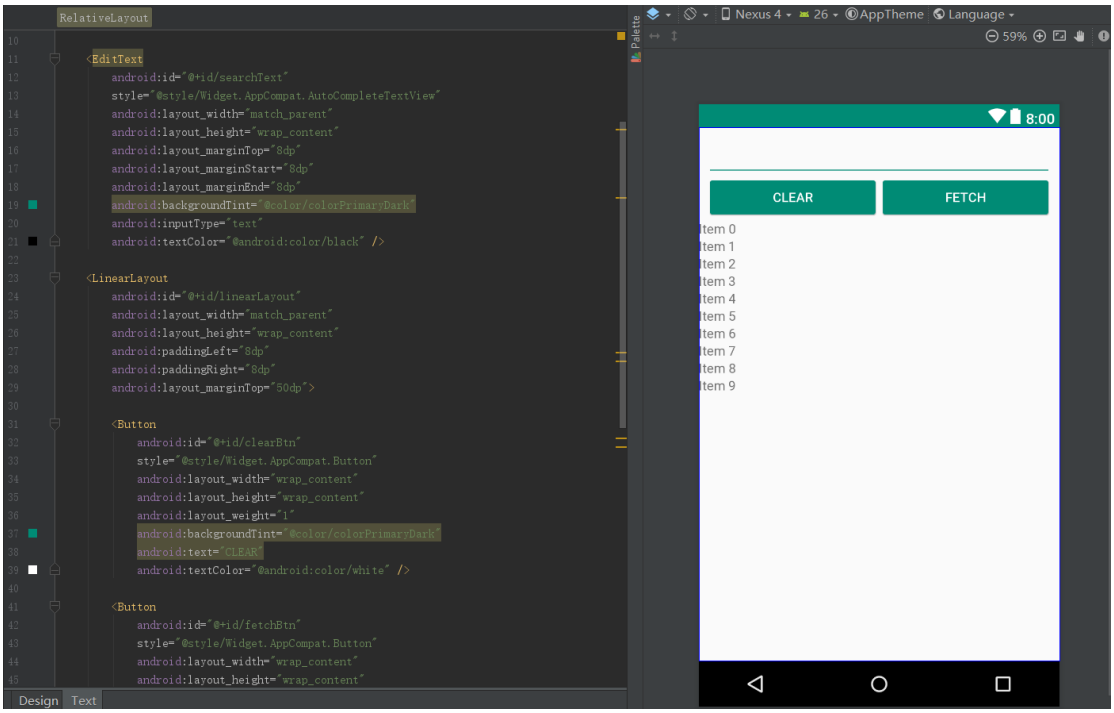


（2） 实验步骤以及关键代码

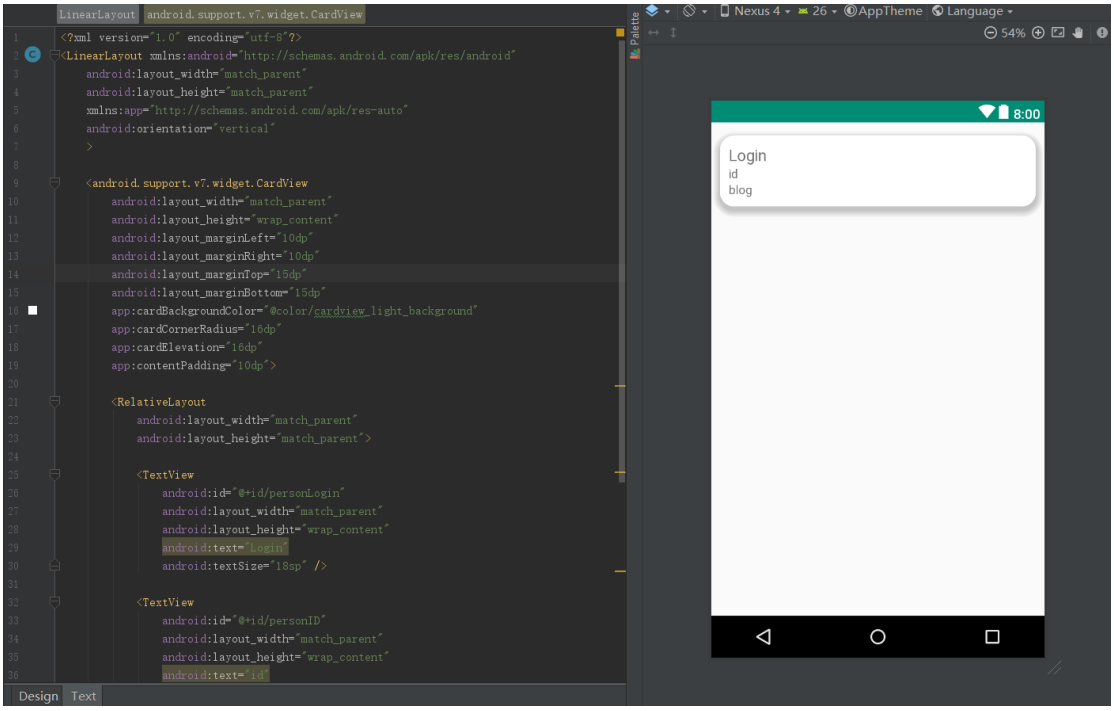
步骤 1：新建 project，布局界面实现。

1）页面布局及其部分代码

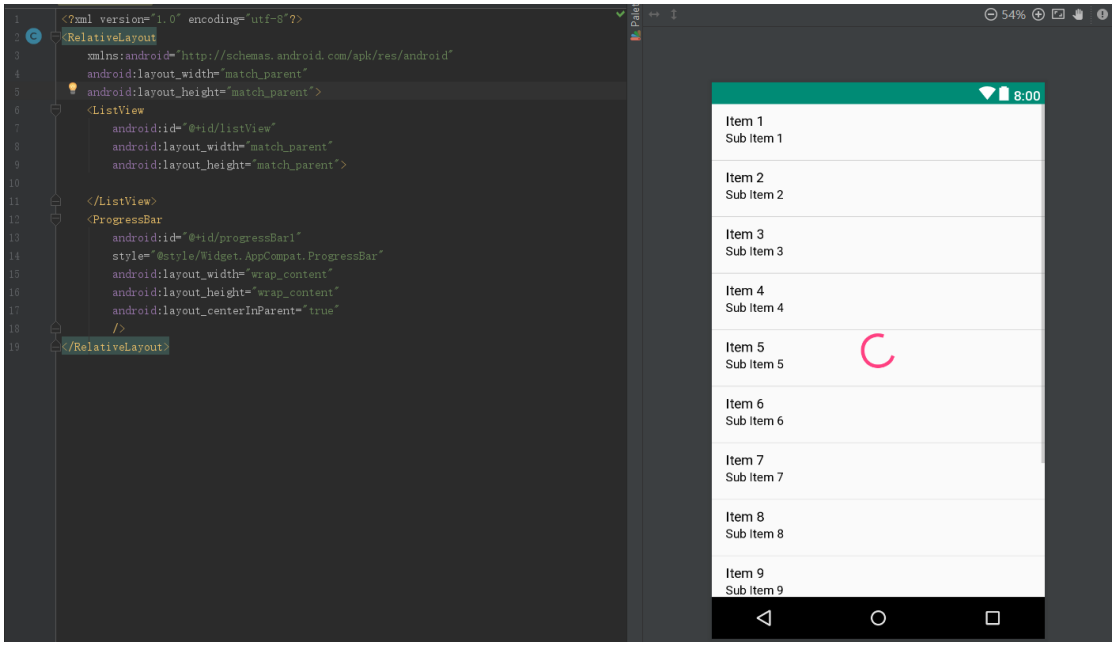
Activity_main.xml:



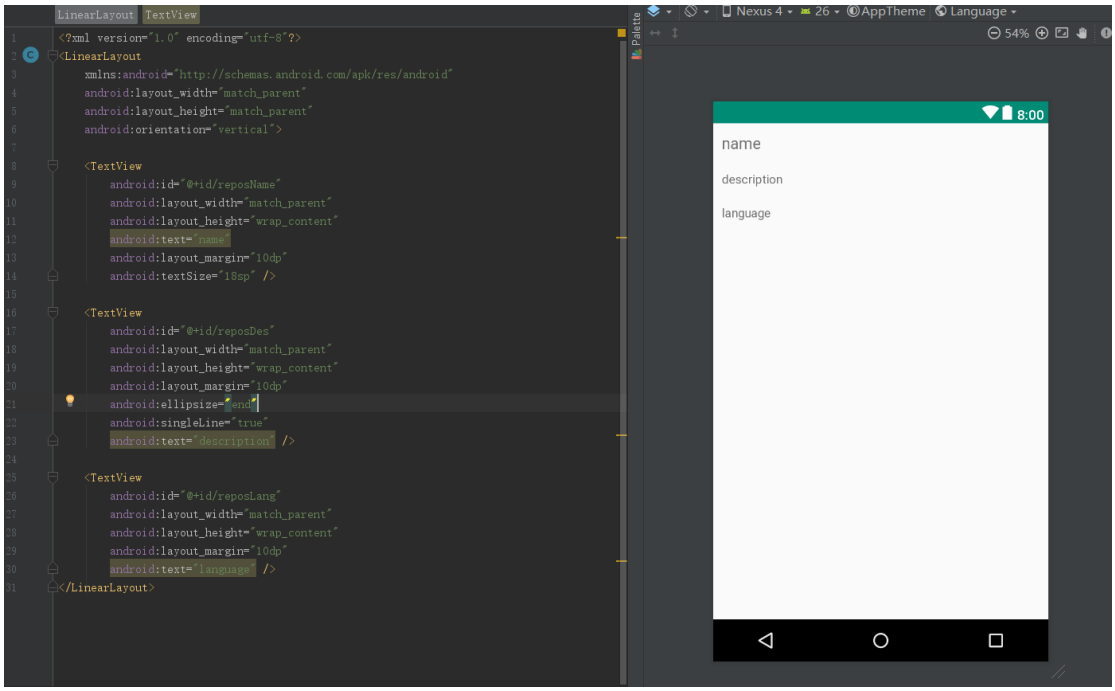
Item.xml:



Activity_repos.xml



Repos_item.xml：



颜色基调设置以及标题栏移除：

```
<resources>

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

</resources>
```

步骤 2：项目结构搭建。

1） 清单文件 AndroidMainfest.xml

manifest

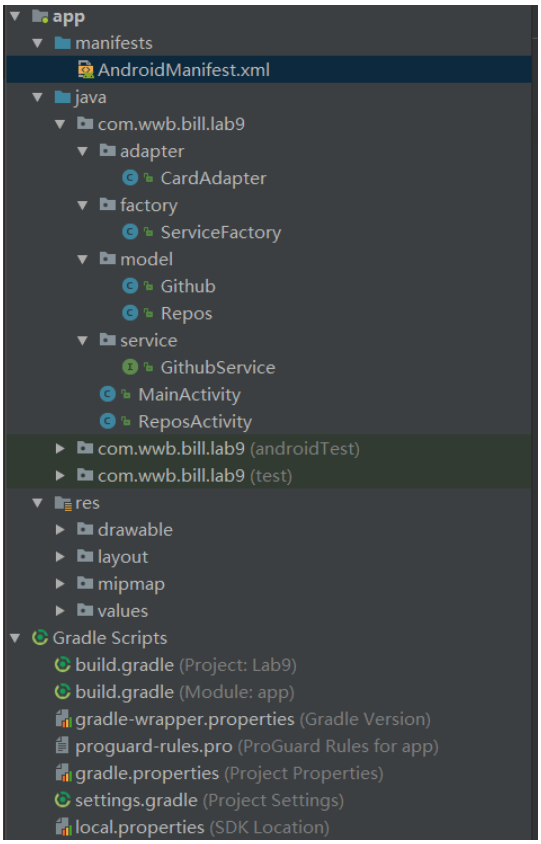
```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.wwb.bill.lab9">

    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Lab9"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".ReposActivity"/>
    </application>
</manifest>
```

框出来部分为需要创建的 activity 和 app 申请的权限。

2） 目录结构。



3） 添加依赖

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.1'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.1'
    compile 'com.android.support:cardview-v7:26.1.0'
    compile 'com.android.support:recyclerview-v7:26.1.0'
    compile 'com.squareup.retrofit2:retrofit:2.1.0'//转换器，请求结果转换成Model
    compile 'com.squareup.retrofit2:converter-gson:2.1.0'
    compile 'io.reactivex:rxjava:1.1.5'
    compile 'io.reactivex:rxandroid:1.1.0'
    compile 'com.squareup.retrofit2:adapter-rxjava:2.1.0'//配合Rxjava 使用
```

步骤 3：主界面获取 user 功能实现

1） 初始化 UI

相关常量、变量：

```
28      private RecyclerView recyclerview;
29      private LinearLayoutManager linearLayoutManager;
30      private List<Github> list;
31      private ServiceFactory service;
32      private CardAdapter cardAdapter;
33
34
35      private TextView searchText;
36      private Button clearBtn;
37      private Button fetchBtn;
38      private ProgressBar progressBar;
```

主体代码：

```
40      @Override
41      protected void onCreate(Bundle savedInstanceState) {
42          super.onCreate(savedInstanceState);
43          setContentView(R.layout.activity_main);
44
45          initView();//初始化布局
46          setListener();//设置监听事件
47
48      }
```

```
50      private void initView() {
51          recyclerview = (RecyclerView)findViewById(R.id.recycleLayout);
52          linearLayoutManager = new LinearLayoutManager( context: this);
53          recyclerview.setLayoutManager(linearLayoutManager);
54          list = new ArrayList<>();
55          cardAdapter = new CardAdapter( context: this,list);
56          recyclerview.setAdapter(cardAdapter);
57          recyclerview.setItemAnimator(new DefaultItemAnimator());
58
59          searchText = (TextView)findViewById(R.id.searchText);
60          clearBtn = (Button)findViewById(R.id.clearBtn);
61          fetchBtn = (Button)findViewById(R.id.fetchBtn);
62          progressBar = (ProgressBar)findViewById(R.id.progressBar);
63      }
64
65      private void setListener() {
66          fetchBtn.setOnClickListener(new clickFetch());
67          clearBtn.setOnClickListener(new clickClear());
68          cardAdapter.setOnItemClickListener(new clickItem());
69      }
```

2) RecycleView CardAdapter 实现

```
20  public class CardAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder>{
21      private List<Github> list;
22      private Context context;
23      private LayoutInflater inflater;
24
25      public CardAdapter(Context context, List<Github> list){
26          this.context = context;
27          this.list = list;
28          inflater = LayoutInflater.from(context);
29      }
30
31      @Override
32      public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
33          View view = inflater.inflate(R.layout.item,  root: null);
34          return new MyViewHolder(view);
35      }
```

```
37      public interface OnItemClickListener{
38          void onItemClick(View view, int position);
39          void onItemLongClick(View view, int position);
40      }
41      private OnItemClickListener mOnItemClickListener;
42      public void setOnItemClickListener(OnItemClickListener mOnItemClickListener) {
43          this.mOnItemClickListener = mOnItemClickListener;
44      }
```

```
101     private class MyViewHolder extends RecyclerView.ViewHolder{
102         private TextView loginText;
103         private TextView idText;
104         private TextView blogText;
105         public MyViewHolder(View itemView){
106             super(itemView);
107             loginText = (TextView)itemView.findViewById(R.id.personLogin);
108             idText = (TextView)itemView.findViewById(R.id.personID);
109             blogText = (TextView)itemView.findViewById(R.id.personBlog);
110         }
```

```
@Override
public void onBindViewHolder(final RecyclerView.ViewHolder holder, int position) {
    final MyViewHolder holder1 = (MyViewHolder) holder;
    Github itemEntity = list.get(position);
    holder1.loginText.setText(String.format("login: %s",itemEntity.getLogin()));
    holder1.idText.setText(String.format("id: %d",itemEntity.getID()));
    holder1.blogText.setText(String.format("blog: %s",itemEntity.getBlog()));
    if (mOnItemClickListener != null)
    {
        holder1.itemView.setOnClickListener((v) -> {
                int pos = holder1.getLayoutPosition();
                mOnItemClickListener.onItemClick(holder1.itemView, pos);
        });

        holder1.itemView.setOnLongClickListener((v) -> {
                int pos = holder1.getLayoutPosition();
                mOnItemClickListener.onItemLongClick(holder1.itemView, pos);
                return false;
        });
    }
}
```

```
@Override
public int getItemCount() {
    if(null == list || list.size() ==0 ){
        return 0;
    }
    else return list.size();
}

public void addData(Github github) { list.add(github); }
public Github getData(int pos){
    return list.get(pos);
}
public void removeData(int pos) { list.remove(pos); }
public void clear() { list.clear(); }
```

步骤 4：Retrofit 实现网络请求

1）定义 Model 类

```
package com.wwb.bill.lab9.model;

/**
 * Created by 15945 on 2017/12/22.
 */

public class Github {
    private String login;
    private int id;
    private String blog;

    public String getLogin() { return login; }

    public int getID() { return id; }

    public String getBlog() { return blog; }
}
```

```java
package com.wwb.bill.lab9.model;

/**
 * Created by 15945 on 2017/12/22.
 */

public class Repos {
    private String name;
    private String description;
    private String language;

    public String getDescription() { return description; }

    public String getLanguage() { return language; }

    public String getName() { return name; }
}
```

2）定义相应的访问接口(interface)

```java
package com.wwb.bill.lab9.service;

import com.wwb.bill.lab9.model.Github;
import com.wwb.bill.lab9.model.Repos;

import java.util.List;

import retrofit2.http.GET;
import retrofit2.http.Path;
import rx.Observable;

/**
 * Created by 15945 on 2017/12/22.
 */

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

    @GET("/users/{user}/repos")
    Observable<List<Repos>> getRepos(@Path("user") String user);
}
```

3） 构造 Retrofit+okHttp 工厂类

```java
public class ServiceFactory {
    private final String url = "https://api.github.com/";
    private static OkHttpClient createOkHttp(){
        OkHttpClient okHttpClient = new OkHttpClient.Builder()
                .connectTimeout( timeout: 10, TimeUnit.SECONDS) //连接超时
                .readTimeout( timeout: 30,TimeUnit.SECONDS) //读超时
                .readTimeout( timeout: 10,TimeUnit.SECONDS) //写超时
                .build();
        return okHttpClient;
    }

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

    public Observable<Github> getUser(String user) {
        GithubService service= createRetrofit(url).create(GithubService.class);
        return service.getUser(user);
    }

    public Observable<List<Repos>> getRepos(String user) {
        GithubService service= createRetrofit(url).create(GithubService.class);
        return service.getRepos(user);
    }
}
```

步骤 5：MainActivity 功能实现

1）Fetch 按钮功能实现

```java
private class clickFetch implements View.OnClickListener {
    @Override
    public void onClick(View view) {
        progressBar.setVisibility(View.VISIBLE);

        String user = searchText.getText().toString();
        service = new ServiceFactory();
        //Toast.makeText(getApplicationContext(),user,Toast.LENGTH_SHORT).show();
        service.getUser(user)
                .subscribeOn(Schedulers.newThread())
                .observeOn(AndroidSchedulers.mainThread())
                .subscribe(new Subscriber<Github>() {
                    @Override
                    public void onCompleted() {
                        System.out.println("完成传输");
                        removeWait();
                    }

                    @Override
                    public void onError(Throwable e) {
                        Toast.makeText( context: MainActivity.this, text: e.hashCode()+"请确认你搜索的用户存在", Toast.LENGTH_SHORT).show();
                        Log.e( tag: "Github-Demo",e.getMessage());
                        removeWait();
                    }

                    @Override
                    public void onNext(Github github) {
                        cardAdapter.addData(github);
                        cardAdapter.notifyDataSetChanged();
                    }
                });
    }
}
```

```java
    private void removeWait() { progressBar.setVisibility(View.GONE); }
```

2）Clear 按钮功能实现

```
    private class clickClear implements View.OnClickListener {
        @Override
        public void onClick(View view) {
            cardAdapter.clear();
            cardAdapter.notifyDataSetChanged();
        }
    }
```

3）RecycleView 中 item 点击长按功能

```
    private class clickItem implements CardAdapter.OnItemClickListener {
        @Override
        public void onItemClick(View view, int position) {
            Intent intent = new Intent( packageContext: MainActivity.this,ReposActivity.class);
            intent.putExtra( name: "name", cardAdapter.getData(position).getLogin());
            startActivity(intent);
        }

        @Override
        public void onItemLongClick(View view, int position) {
            cardAdapter.removeData(position);
            cardAdapter.notifyDataSetChanged();
        }
    }
}
```

步骤 6：ReposActivity 功能实现

```
public class ReposActivity extends AppCompatActivity {
    private ProgressBar progressBar;
    private ArrayList<Map<String,String>> mInfos= new ArrayList<Map<String,String>>();
    private ListView listView;
    private ServiceFactory service;
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_repos);
        listView = (ListView)findViewById(R.id.listView);
        progressBar = (ProgressBar)findViewById(R.id.progressBar1);
        progressBar.setVisibility(View.VISIBLE);
        final SimpleAdapter simpleAdapter = new SimpleAdapter( context: ReposActivity.this, mInfos, R.layout.repos_item,
                new String[]{"name","description","language"},
                new int[]{R.id.reposName,R.id.reposDes,R.id.reposLang});
        Intent intent = getIntent();
        String name = intent.getStringExtra( name: "name");
```

```
        service = new ServiceFactory();
        service.getRepos(name)
                .subscribeOn(Schedulers.newThread())
                .observeOn(AndroidSchedulers.mainThread())
                .subscribe(new Subscriber<List<Repos>>() {
                    @Override
                    public void onCompleted() {
                        System.out.println("完成传输");
                        removeWait();
                    }

                    @Override
                    public void onError(Throwable e) {
                        Toast.makeText( context: ReposActivity.this, text: e.hashCode()+"请确认你搜索的用户存在", Toast.LENGTH_SHORT).show();
                        Log.e( tag: "Github-Demo",e.getMessage());
                        removeWait();
                    }

                    @Override
                    public void onNext(List<Repos> repos) {
                        for (Repos temp:repos) addRepos(temp);
                        simpleAdapter.notifyDataSetChanged();
                    }
                });
        listView.setAdapter(simpleAdapter);
    }
```

```
74      private void removeWait() { progressBar.setVisibility(View.GONE); }
77      private void addRepos(Repos temp) {
78          Map<String,String> item=new HashMap<>();
79          item.put( k: "name", temp.getName());
80          item.put( k: "description", temp.getDescription());
81          item.put( k: "language", temp.getLanguage());
82          mInfos.add(item);
83      }
84  }
```

五、 实验遇到的问题及解决方法

（1） 主要遇到的问题是网络链接那一块，第一次是获取 user 时有显示联网不过却没有找到用户，一直以为是联网的问题，因为报错的信息一直是：

```
@Override
public void onError(Throwable e) {
    Toast.makeText( context: ReposActivity.this,  text: e.hashCode() + "请确认你搜索的用户存在", Toast.LENGTH_SHORT).show();
    Log.e( tag: "Github-Demo",e.getMessage());
    removeWait();
}
```

这一块，后来在 Android Studio 下的 Logcat 中查看了 log 信息才知道不只是联网出错时 onError 才会执行，RecycleView 写入失败也会触发这个函数，后来找到问题是 int id 读入失败，要转换为字符串才行。

（2） 第二次遇到的问题还是联网，这次 log 直接报 404 了，也就是真的找不着资源，循着执行路径反方向一步步 debug 回去后找到问题关键，intent 传输的问题，纠正一下也就解决了。

六、 实验思考及感想

本次实验 TA 说很简单，不过实在是很多坑啊，对于新手来说，没有实践过也没有理解透彻的话是很容易被坑到的，例如我前面的几个问题，都是一些小失误导致大问题，还容易被迷惑找不到解决方法，所以还是需要加深理解，同时基础也不能落下，以前遇到的小问题应该记录下来，不能下一次又踩坑。

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。