中山大学数据科学与计算机学院本科生实验报告

(2017 年秋季学期)

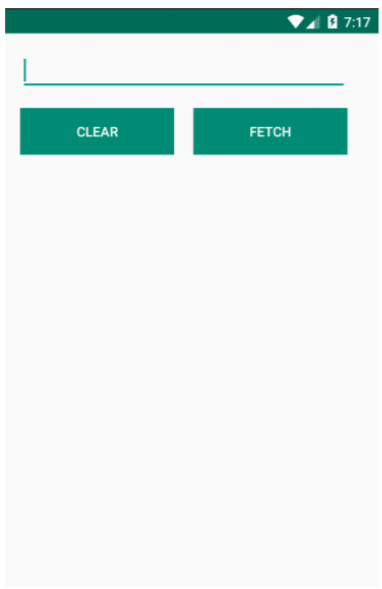
课程名称：手机应用开发　　任课教师： 刘宁

年级	2015	专业（方向）	计应
学号	14970011	姓名	宋思亭
电话	18826073511	Email	1056284551@qq.com
开始日期	2017.12.11	完成日期	2017.12.15

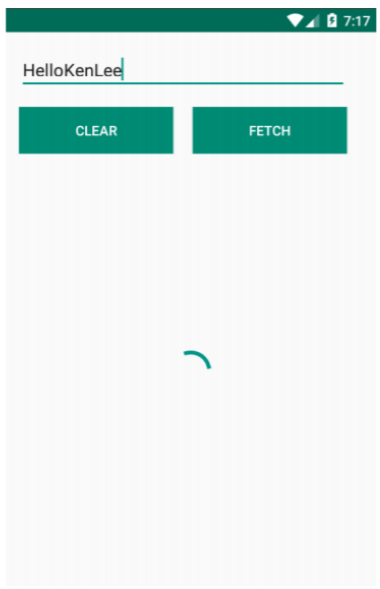
一、 实验目的

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

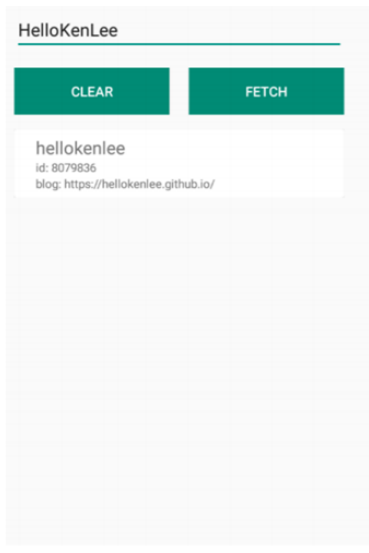
二、 实现内容



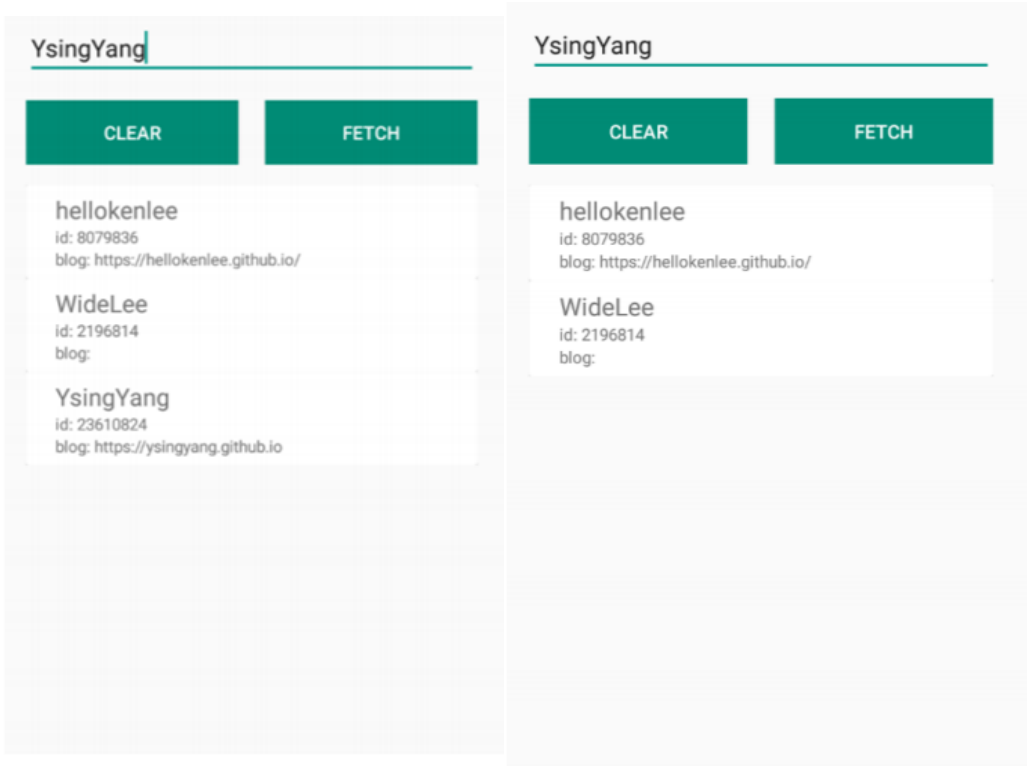
主界面



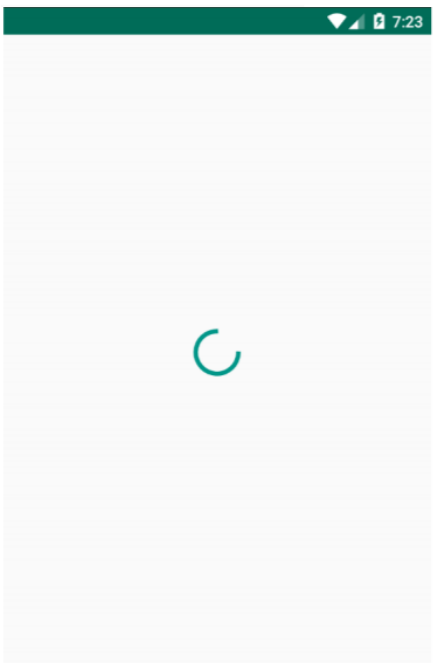
搜索用户



搜索结果



搜索结果



点击进入个人详情页面

长按删除

详情信息获取显示

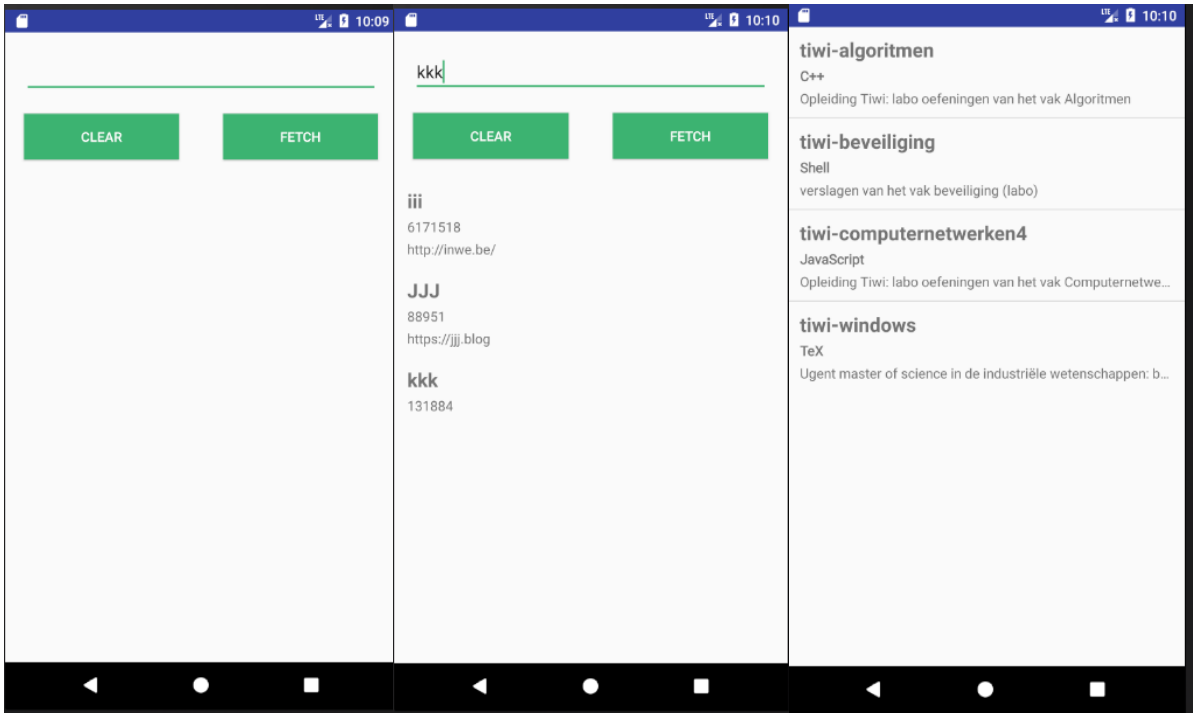
对于 User Model, 显示 id, login, blog

对于 Repository Model, 显示 name, description,language

(特别注意， 如果 description 对于 1 行要用省略号代替)

三、 课堂实验结果

（1） 实验截图



（2） 实验步骤以及关键代码

1. 写 activity_main.xml，设置外观样式

MainActivity.java × activity_main.xml ×

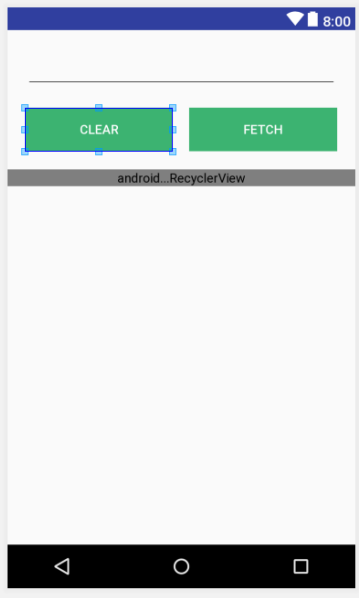
RelativeLayout Button

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <EditText
        android:id="@+id/searchET"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"/>

    <Button
        android:id="@+id/clearBT"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignStart="@+id/searchET"
        android:layout_below="@+id/searchET"
        android:background="@drawable/shape"
        android:text="CLEAR"
        android:textColor="#fff"
        android:layout_marginBottom="20dp"/>

    <Button
        android:id="@+id/fetchBT"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignEnd="@+id/searchET"
```

Preview

Palette

Nexus 4 26 AppTheme Language 49%

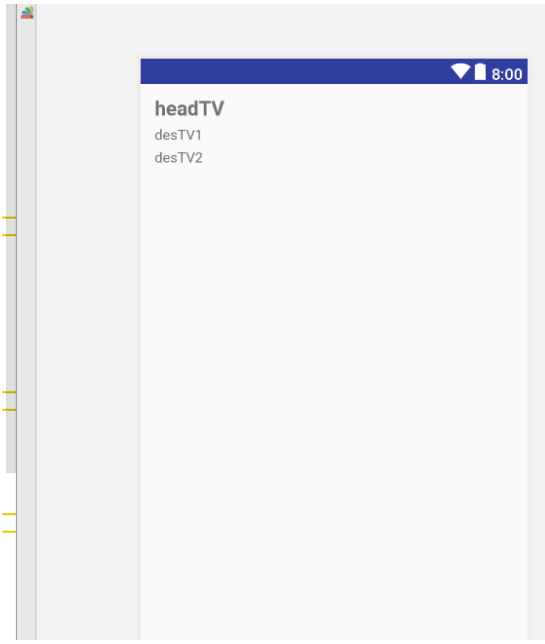


写 activity_repos.xml，设置外观样式

```
<ListView
    android:id="@+id/reposLV"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:visibility="invisible"/>

<ProgressBar
    android:id="@+id/waitPB"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="175dp"
    android:visibility="visible"/>
```

写 list.xml，设置列表样式

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:paddingLeft="12dp"
    android:paddingRight="12dp"
    android:paddingTop="8dp"
    android:paddingBottom="8dp">

    <TextView
        android:id="@+id/headTV"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="headTV"
        android:textSize="20dp"
        android:textStyle="bold"
        android:padding="2dp"/>

    <TextView
        android:id="@+id/desTV1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="2dp"
        android:text="desTV1"
        android:textSize="14dp"/>
```



写 shape.xml，设置按钮样式

```
<shape xmlns:android="http://schemas
    android:shape="rectangle" >
    <solid android:color="#3CB371" />
    <padding
        android:bottom="5dp"
        android:left="60dp"
        android:right="60dp"
        android:top="5dp" />
</shape>
```

2. 写 model 中的 User 类和 Repos 类

```java
public class User {
    private String login;
    private String blog;
    private int id;

    public User(String login, String blog, int id) {
        this.login = login;
        this.blog = blog;
        this.id = id;
    }

    public String getLogin() { return login; }

    public String getBlog() { return blog; }

    public int getId() { return id; }

    public void setId(int id) { this.id = id; }
}
```

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

    public String getName() { return name; }

    public String getDescription() {
        return description;
    }

    public String getLanguage() { return language; }
}
```

3. 写 adapter 中的 CardAdapter 和 ListAdapter 分别适配 RecyclerView 和 ListView

```java
public class CardAdapter extends RecyclerView.Adapter<CardAdapter.CardViewHolder> {
    public interface OnItemClickListener {
        void onClick(int position);
        void onLongClick(int position);
    }

    private OnItemClickListener mOnItemClickListener = null;
    private List<User> users;
    private Context context;

    public CardAdapter(List<User> users, Context context) {
        this.users = users;
        this.context = context;
    }

    @Override
    public CardViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        return new CardViewHolder(LayoutInflater.from(context).inflate(R.layout.list, parent,
    }

    @Override
    public void onBindViewHolder(final CardViewHolder holder, int position) {
        holder.login.setText(users.get(position).getLogin());
        holder.id.setText("" + users.get(position).getId());
        holder.blog.setText(users.get(position).getBlog());
        if (mOnItemClickListener != null) {
            holder.itemView.setOnClickListener((v) -> {
                    mOnItemClickListener.onClick(holder.getAdapterPosition());
            });
```

```
public class ListAdapter extends BaseAdapter {
    private List<Repos> repos;

    public ListAdapter(List<Repos> repos) { this.repos = repos; }

    @Override
    public int getCount() { return repos != null ? repos.size() : 0; }

    @Override
    public Object getItem(int i) { return repos != null ? repos.get(i) : null; }

    @Override
    public long getItemId(int i) { return i; }

    @Override
    public View getView(int position, View view, ViewGroup viewGroup) {
        View convertView;
        ViewHolder holder;
        if (view == null) {
            convertView = LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.list,
            holder = new ViewHolder();
            holder.name = (TextView)convertView.findViewById(R.id.headTV);
            holder.language = (TextView)convertView.findViewById(R.id.desTV1);
            holder.description = (TextView)convertView.findViewById(R.id.desTV2);
            convertView.setTag(holder);
        } else {
```

4. 写 ServiceFactory 类，封装 Retrofit 和 OkHttpClient

```
public static GithubService createGithubService(){
    return createRetrofit(BASE_URL).create(GithubService.class);
}

public static ReposService createReposService() {
    return createRetrofit(BASE_URL).create(ReposService.class);
}

// 构造Retrofit对象实现网络访问，使用GsonConverterFactory解析返回的数据
private static Retrofit createRetrofit(String baseUrl) {
    return new Retrofit.Builder()
            .baseUrl(baseUrl)
            .addConverterFactory(GsonConverterFactory.create())
            .addCallAdapterFactory(RxJavaCallAdapterFactory.create())
            .client(createOkHttp())
            .build();
}

// 配置相应的OKHttp对象
private static OkHttpClient createOkHttp() {
    OkHttpClient okHttpClient = new OkHttpClient().newBuilder()
            .connectTimeout(DEFAULT_TIME_OUT, TimeUnit.SECONDS)
            .readTimeout(DEFAULT_READ_TIME_OUT, TimeUnit.SECONDS)
            .writeTimeout(DEFAULT_TIME_OUT, TimeUnit.SECONDS)
            .build();
    return okHttpClient;
}
```

5. 写 service 中的两个请求服务

```
public interface GithubService {
    @GET("/users/{user}")
    Observable<User> getUser(@Path("user") String user);
}
public interface ReposService {
    @GET("/users/{user}/repos")
    Observable<List<Repos>> getRepos(@Path("user") String user);
}
```

6. 写 MainActivity 类

初始化控件、列表和网络请求服务

```
// 初始化控件
searchET = (EditText)findViewById(R.id.searchET);
clearBT = (Button)findViewById(R.id.clearBT);
fetchBT = (Button)findViewById(R.id.fetchBT);
userRV = (RecyclerView)findViewById(R.id.userRV);
waitPB = (ProgressBar)findViewById(R.id.waitPB);

// 初始化列表，设置RecyclerView
users = new ArrayList<User>();
userRV.setLayoutManager(new LinearLayoutManager( context: this));
myAdapter = new CardAdapter(users,  context: MainActivity.this);
userRV.setAdapter(myAdapter);

// 初始化网络请求服务包
mServiceFactory = new ServiceFactory();
mGithubService = mServiceFactory.createGithubService();
```

设置 clear 按钮点击监听，清空 EditTextview 和列表中所有数据

```
clearBT.setOnClickListener((v) -> {
        searchET.setText("");
        users.clear();
        myAdapter.notifyDataSetChanged();
});
```

设置网络请求权限

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

设置 fetch 按钮点击监听，包含使用封装好的服务进行异步请求，gson 解析 json 文件得到 User 可以方便地直接添加到列表中

```
// 获取搜索框字符串，检查不为空
String searchS = searchET.getText().toString();
if (searchS.equals("")) {
    Toast.makeText( context: MainActivity.this,  text: "查询栏为空", Toast.LENGTH_SHORT).show();
    return;
}

// 两次请求时间间隔不小于1s
Time2 = Time1;
Time1 = (new Date()).getTime();
if (Time1 - Time2 < 1000) {
    Toast.makeText( context: MainActivity.this,  text: "点击速度过快", Toast.LENGTH_SHORT).show();
    return;
}
// 请求前打开progressBar
waitPB.setVisibility(View.VISIBLE);
// 异步处理网络请求
mGithubService.getUser(searchS)
        .subscribeOn(Schedulers.newThread())
        .observeOn(AndroidSchedulers.mainThread())
        .subscribe(new Subscriber<User>() {
            // 请求完成后关闭progressBar
            @Override
            public void onCompleted() { waitPB.setVisibility(View.INVISIBLE); }

            // 请求失败弹出提示
            @Override
            public void onError(Throwable e) {
                Toast.makeText( context: MainActivity.this,  text: "数据请求失败: "
                        + e.getMessage(), Toast.LENGTH_LONG).show();
                waitPB.setVisibility(View.INVISIBLE);
            }

            // 请求得到数据添加到列表中
            @Override
            public void onNext(User user) {
                users.add(user);
                myAdapter.notifyDataSetChanged();
            }
        });
```

设置列表监听事件

```
myAdapter.setOnItemClickListener(new CardAdapter.OnItemClickListener() {
    // 点击使用intent将login内容发送到ReposActivity
    @Override
    public void onClick(int position) {
        Intent intent = new Intent( packageContext: MainActivity.this, ReposActivity.class);
        intent.putExtra( name: "login", users.get(position).getLogin());
        startActivityForResult(intent,  requestCode: 0);
    }

    // 长按从列表删除选择的项
    @Override
    public void onLongClick(int position) {
        Toast.makeText(getApplicationContext(),  text: "从列表删除" +
                users.get(position).getLogin(), Toast.LENGTH_SHORT).show();
        users.remove(position);
        myAdapter.notifyDataSetChanged();
    }
});
```

7. 写 ReposActivity 类

初始化控件和网络请求包

```
// 初始化控件
reposLV = (ListView)findViewById(R.id.reposLV);
waitPB = (ProgressBar)findViewById(R.id.waitPB);

// 初始化网络请求服务包
mServiceFactory = new ServiceFactory();
mReposService = mServiceFactory.createReposService();
```

获取 MainActivity 通过 intent 传递的 login 数据，使用同样的方法执行异步的网络请求，将返回的 Repos 列表直接添加到 ListAdapter 中

```
Intent intent = getIntent();
final String login = intent.getStringExtra( name: "login");
// 异步处理网络请求
mReposService.getRepos(login)
        .subscribeOn(Schedulers.newThread())
        .observeOn(AndroidSchedulers.mainThread())
        .subscribe(new Subscriber<List<Repos>>() {
            // 请求完成后关闭progressBar，显示列表
            @Override
            public void onCompleted() {
                waitPB.setVisibility(View.INVISIBLE);
                reposLV.setVisibility(View.VISIBLE);
            }

            // 请求失败弹出提示
            @Override
            public void onError(Throwable e) {
                Toast.makeText( context: ReposActivity.this, text: "数据请求失败: "
                        + e.getMessage(), Toast.LENGTH_LONG).show();
                waitPB.setVisibility(View.INVISIBLE);
            }

            // 初始化列表，设置ListView，将请求得到数据添加到列表中
            @Override
            public void onNext(List<Repos> repos) {
                mListAdapter = new ListAdapter(repos);
                reposLV.setAdapter(mListAdapter);
            }
        });
```

（3） 实验遇到困难以及解决思路

1. mGithubService 请求数据出错，检查发现 setText 参数为 getId ()，返回 int 类型从而类型不符，修改为 holder.id.setText("" + users.get(position).getId())解决
2. 点击列表跳出，检查发现 ReposActivity 未注册，在 manifest 中注册<activity android:name=".ReposActivity"></activity>后解决

3. 长按列表跳出，检查发现执行 users.remove(position)移除数据后再执行 users.get(position).getLogin()导致空指针，调整两个语句顺序后解决
4. 由于两个列表适用同一个 list.xml，repos 页面需要设置 language 字体加粗，在 ListAdapter 中执行

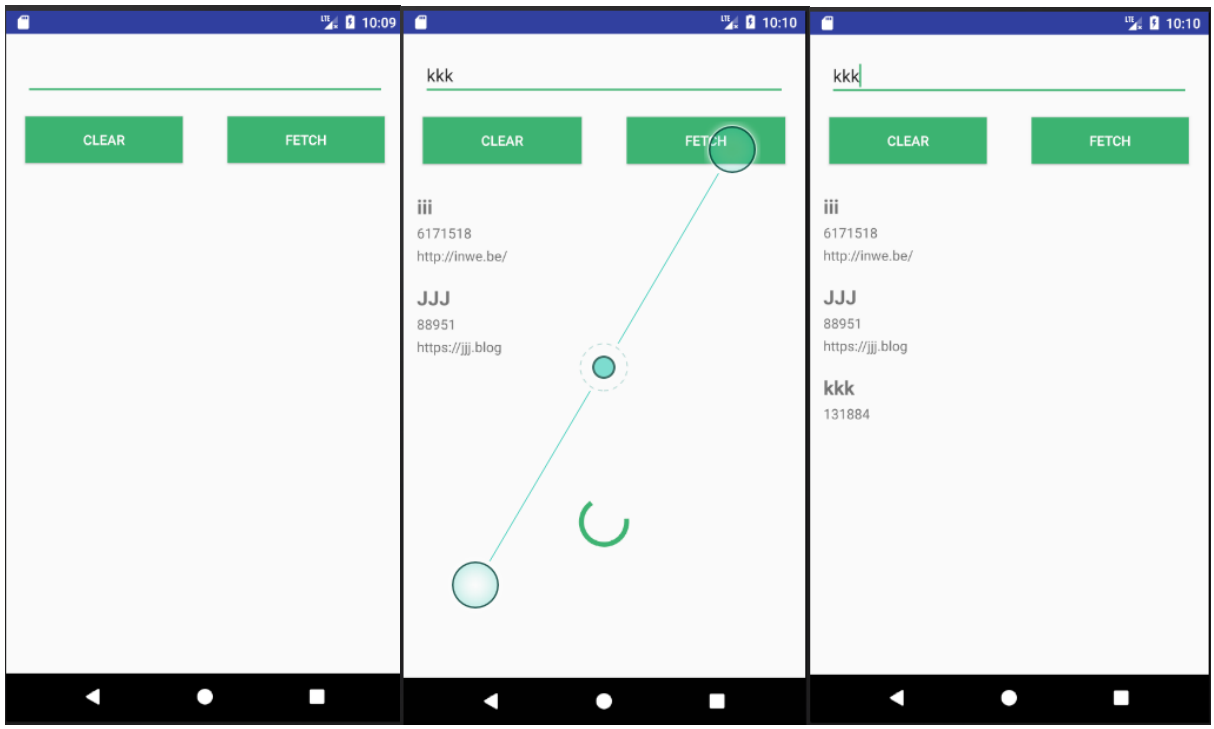
   TextPaint paint = holder.language.getPaint();

   paint.setFakeBoldText(true);

   即可
5. 绿色主题修改 color.xml 中的<color name="colorAccent">#3CB371</color>实现
6. ProgressBar 是否显示通过设置 visibility 实现
7. description 单行显示在 xml 文件中设置 ellipsize="end"和 singleLine="true"即可，若内容超过一行显示为省略号

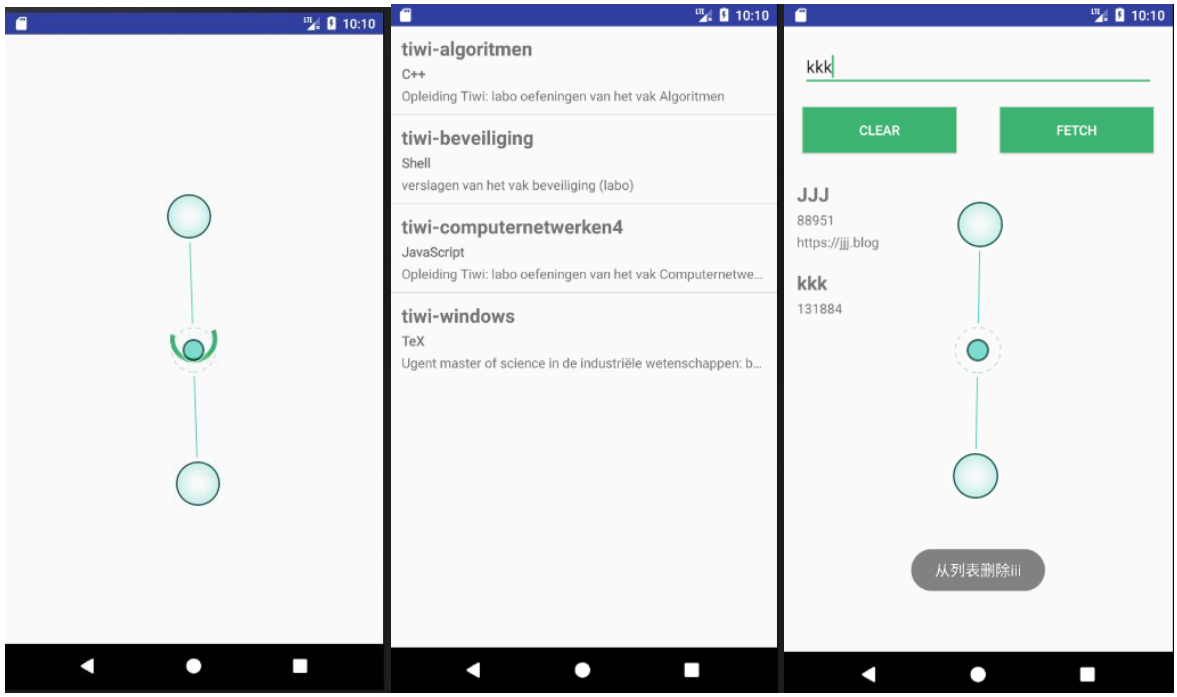
四、 课后实验结果

1. 打开应用，输入内容点击 FETCH 按钮进行网络请求，过程中显示 ProgressBar，请求结果加入列表中

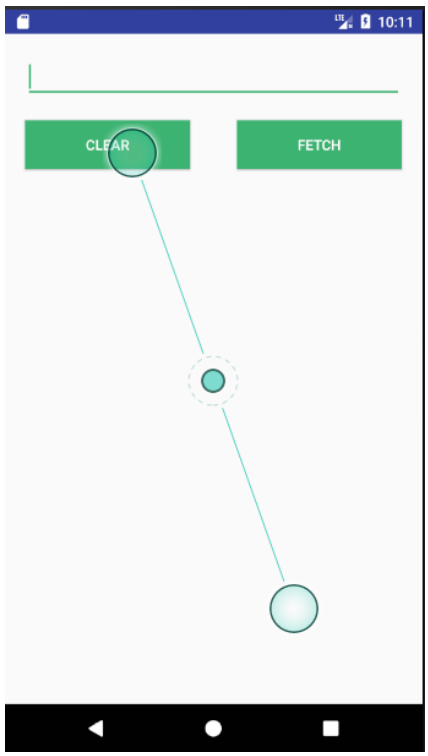


2. 点击列表跳转到 repos 页面，发送网络请求过程中显示 ProgressBar，返回数据加入列表，description 单行显示，多余的内容显示为省略号

   长按列表删除项目



3. 点击 CLEAR 清空 EditTextview 和列表中所有数据



五、 实验思考及感想

1. 安卓开发过程中有很多坑等我们去跳，需要多用搜索引擎多看文档
2. 遇到奇怪的问题不知道怎么解决时，可以重启试试