浙江大学

实验报告

课程名称：智能终端软件开发技术

APP 名称：秋学期智能终端 APP

专业：信息安全

成员 1 姓名：许多 学号：3160103958

成员 2 姓名：吕慕凡 学号：3160102463

电子邮件地址：3160103958@zju.edu.cn

手机：17342019969

任课老师：张 寅

目录

一．系统简介

这是一个基于 Android 的用于对纯文本进行命名实体及其关系标注的原生 APP。整体适配于 Android 6.0 版本以上的机型，在 Android5.1 上仅能实现关系标注功能。具体能实现的功能有登录/注册/登出等基本账户操作功能，从服务器端获取实体及其关系标注，上传本地编辑的实体标注和关系标注等。

二．总体架构及功能划分

2.1 相关 activity 及功能

（1）LoginActivity

主要实现登录功能，将用户输入的用户名和密码发送至服务端，在服务端进行登录。发送请求后接收服务器返回登录相关信息并将信息显示在屏幕上。登录的有效时长为 30min，若超过有效时长并为进行任何操作则自动登出。登录成功后跳转至 MainMenuActivity。点击注册按钮则跳转至 SignupActivity。

（2）SignupActivity

主要实现注册功能，将用户输入的用户名、邮箱和密码发送至服务端，在服务端进行注册。发送请求后接收服务器返回的注册相关信息并将信息显示在屏幕上。点击取消按钮跳转回 LoginActivity。

（3）MainMenuAvtivity

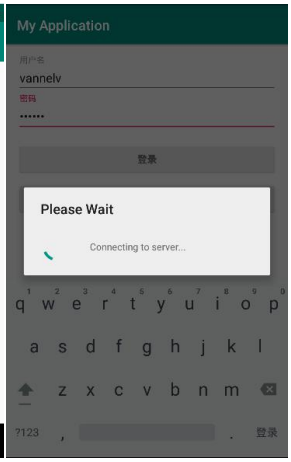
进入时为欢迎界面，用户可点击左侧任务菜单栏查看用户个人信息以及选择进行实体标注或段落标注，点击下方退出按钮实现登出功能。

当用户选择实体标注时，界面获取的将会是网站上的实体相关信息，点击“下一页”会刷新为另一篇实体标注相关文本段落。长按段落中的文字并在长按后弹出的菜单选择将字体标注为红色作为人名还是标注为绿色作为职位。标注结束后可点击“提交”按键将信息提交至服务器，若提交成功，将有浮动窗口显示提交成功字样。

当用户选择关系标注时，界面获取的将会是网站上的关系标注相关信息，点击“下一页”会刷新为另一篇关系标注相关文本段落。文本段落下方为标注了的关系信息，分别为“左实体”，“右实体”和“关系”，点击选中“关系”文本框，通过多次点击，每次点击可以修改“关系”显示的内容，如果修改后的“关系”内容与原本的不一致，则取消 CheckBox 的勾选状态。点击“提交”按键可将修改后的信息提交至服务器，若提交成功，将有浮动窗口显示提交成功字样。

2.2 相关界面及功能

（1）activity_login

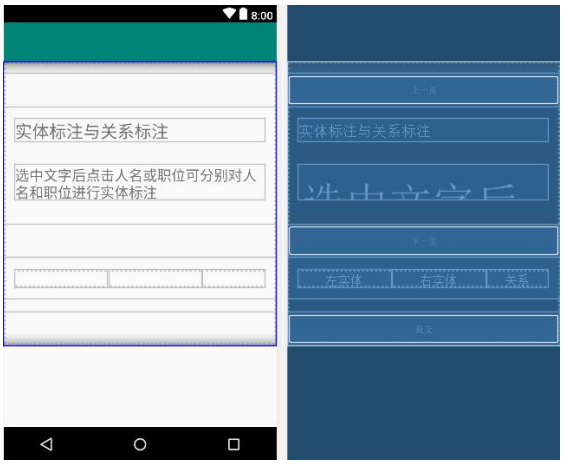


用户在用户名处输入用户名，在密码处输入密码，点击登录可进行登录，点击注册跳转至注册界面。

（2）activity_signup

用户在三个输入框中分别输入对应信息，点击注册可进行注册，已有账户的用户可点击取消返回登录界面。

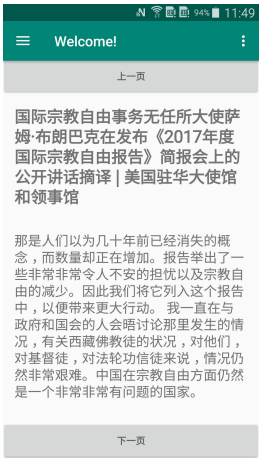
（3）content_main_menu



主界面，显示的内容由下面的菜单栏决定。当前状态下不必要的显示内容设置可见属性为隐藏。

（4）activity_main_menu

点击命名实体标注可在 content_main_menu 看到已有实体信息



点击关系标注可在 content_main_menu 看到已有关系标注信息。

（更改前）


（更改后）

点击退出可以登出服务器，返回至登录界面

2.3 相关类及功能

（1）User

存储用户的登录、注册等相关个人信息。用以实现用户登录成功后的 token 暂时存储以及后续的 token 校验问题。

（2）MyMessageTools

存储部分在 APP 各功能中重复使用的函数。

（3）EntitiesStructure

存储实体标注相关信息，为后续上传做准备。

（4）TriplesStructure

存储关系标注相关信息，为后续上传做准备。

2.4 附加控件及功能

（1）selection_action_menu（此功能在 6.0 及以上版本通过测试）

其功能为将长按文字后系统弹出的菜单自定义为我们想要的菜单名称，在本 APP 中应用在实体标注的部分，使长按文字后弹出的菜单为“人名（红）”和“职位（绿）”，用户选择对应的选项即可完成对实体的标注。

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/name"
        android:title="人名（红）" />
    <item
        android:id="@+id/job"
        android:title="职位（绿）" />
</menu>
```

（2）TextRatingBar

在页面中增加的自定义的调节字号大小的插件。

三．关键数据结构/算法

3.1 向指定服务器传输、返回数据

在整个 APP 中的数据传输都使用了 okhttp3 的网络架构，用 formbody 的形式向服务器 post 数据并接收返回的 json 数据，在此仅用登录部分代码进行详细解释。其作用是向指定服务器以 formbody 形式 post 用户名和密码，并反馈登录信息。

```
OkHttpClient login_client = new OkHttpClient();//用 okhttp 的网络架构进行登录
RequestBody postBody = new FormBody.Builder()//用 formbody 的形式向服务器传输用户名和密码
        .add("username", user.getUsername())
        .add("password", user.getPassword())
        .build();
Request request = new Request.Builder()
        .url("http://10.15.82.223:9090/app_get_data/app_signincheck")//指定访问的服务器地址
        .post(postBody)
        .build();
Call call = login_client.newCall(request);
setProgressDialouge();//显示联网状态
call.enqueue(new Callback() {
    @Override
    public void onFailure(Call call, IOException e) {
        Log.d("CONNECTION", "请求失败 !!") ;
    }//对服务器请求失败
    @Override
    public void onResponse(Call call, final Response response) throws IOException {
        progressDialog.dismiss();
        Log.d("CONNECTION", "请求成功");
        final String responseData = response.body().string() ;
        parseJSONWithJSONObject(responseData);//调用 parseJSONWithJSONObject()方法来解析数据
    }//对服务器请求成功
```

3.2 解析服务器传回的 json 数据

服务器将 json 格式的数据返回后，要输出到用户界面中，必须要对其进行解析和 Unicode

转换。我们构造了一个函数来实现这个过程。

```
private void parseJSONWithJSONObject(String responseData){
    String msg;
    try{
          JSONObject jsonObject = new JSONObject(responseData.substring(responseData.indexOf("{"),
responseData.lastIndexOf("}") + 1)) ;
          String message = jsonObject.getString("msg");
          msg = Message.unicodeToUtf8(message);//对数据进行 Unicode 转码为中文字符
    }catch (Exception e){
       e.printStackTrace();
    }
}
```

3.3 Unicode 转码为中文字符

由于服务器返回的数据为 Unicode 编码，因此要正常显示到用户界面之前要对其进行转码。

```
static String unicodeToUtf8(String theString) {
    char aChar;
    int len = theString.length();
    StringBuffer outBuffer = new StringBuffer(len);
    for (int x = 0; x < len;) {
       aChar = theString.charAt(x++);
       if (aChar == '\\') {
          aChar = theString.charAt(x++);
          if (aChar == 'u') {
             int value = 0;
             for (int i = 0; i < 4; i++) {
                aChar = theString.charAt(x++);
                switch (aChar) {
                   case '0':
                   case '1':
                   case '2':
                   case '3':
                   case '4':
                   case '5':
                   case '6':
                   case '7':
                   case '8':
                   case '9':
                      value = (value << 4) + aChar - '0';
                      break;
                   case 'a':
                   case 'b':
```

```
                    case 'c':
                    case 'd':
                    case 'e':
                    case 'f':
                        value = (value << 4) + 10 + aChar - 'a';
                        break;
                    case 'A':
                    case 'B':
                    case 'C':
                    case 'D':
                    case 'E':
                    case 'F':
                        value = (value << 4) + 10 + aChar - 'A';
                        break;
                    default:
                        throw new IllegalArgumentException(
                                "Malformed   \\uxxxx   encoding.");
                    }
                }
                outBuffer.append((char) value);
            } else {
                if (aChar == 't')
                    aChar = '\t';
                else if (aChar == 'r')
                    aChar = '\r';
                else if (aChar == 'n')
                    aChar = '\n';
                else if (aChar == 'f')
                    aChar = '\f';
                outBuffer.append(aChar);
            }
        } else
            outBuffer.append(aChar);
    }
    return outBuffer.toString();
}
```

3.4 与服务器交互状态的显示

在传递数据到服务器时，如果有一个对话框能显示等待、正在连接诸如此类的信息，会让联网状态更加直观。

```
private void setProgressDialog(){//联网状态的显示
    progressDialog = new ProgressDialog(LoginActivity.this) ;
```

```
    progressDialog.setTitle("Please Wait");
    progressDialog.setMessage("Connecting to server...");
    progressDialog.setIndeterminate(true);
    progressDialog.setCancelable(false);
    progressDialog.show();
}
```

3.5 长按文字后出现的菜单的功能改写

在实体标注的时候，我们希望用户的标注方法可以是长按文字并选中后，点击文字上方浮现的菜单栏对应的标注信息来对实体进行标注，而此时则需要对菜单的选项和对应功能进行改写。

(1) 重新定义一个 callback 类

```
private ActionMode.Callback callback2= new ActionMode.Callback()
```

(2) 重新定义长按后的菜单

重新定义长按后的菜单为我们单独编写的“*selection_action_menu*”，并且使用 getSelectionStart()、getSelectionEnd()两个函数对长按选中文字的位置信息进行标记，为后续的操作做准备。

```
    public boolean onCreateActionMode(ActionMode actionMode, Menu menu) {
        MenuInflater menuInflater = actionMode.getMenuInflater();
        menuInflater.inflate(R.menu.selection_action_menu,menu);
        int start = responseText.getSelectionStart();//获取选择部分的起始信息
        int end = responseText.getSelectionEnd();
        System.out.print(start);
        System.out.print(end);
        return true;//返回 false 则不会显示弹窗
    }
```

(3) 清除系统自带的“复制”、“全选”等菜单内容

```
    public boolean onPrepareActionMode(ActionMode actionMode, Menu menu) {
        MenuInflater menuInflater = actionMode.getMenuInflater();
        menu.clear();
        menuInflater.inflate(R.menu.selection_action_menu,menu);
        return true;
    }
```

(4) 处理我们自定义的点击事件

本部分功能为当用户点击了对应想标注的类型，如人名时，界面上会浮现弹窗“人名”，对应选中文字变为红色，并且相关实体标注信息将被存储在实体结构中，为后续上传做准备。

```
    public boolean onActionItemClicked(ActionMode actionMode, MenuItem menuItem) {
```

```
        //根据 item 的 ID 处理点击事件
        int start = responseText.getSelectionStart();//获取选择部分的起始信息
        int end = responseText.getSelectionEnd();
        switch (menuItem.getItemId()){
            case R.id.name:
                Toast.makeText(MainMenuActivity.this, "人名", Toast.LENGTH_SHORT).show();
                SpannableStringBuilder styled1 = new SpannableStringBuilder(responseText.getText());
                styled1.setSpan(new ForegroundColorSpan(Color.RED), start,end,
                Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
                String EntityName =  responseText.getText().toString().substring(start,end);
                Entities.getEntityName().add(EntityName);
                Entities.getStart().add(start);
                Entities.getEnd().add(end);
                Entities.getNerTag().add("PERSON");
                responseText.setText(styled1);//更改选中部分的颜色
                actionMode.finish();//收起操作菜单
                break;
            case R.id.job:
                Toast.makeText(MainMenuActivity.this, "职位", Toast.LENGTH_SHORT).show();
                SpannableStringBuilder styled2 = new SpannableStringBuilder(responseText.getText());
                styled2.setSpan(new ForegroundColorSpan(Color.GREEN), start,end,
                Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
                String EntityName1 =  responseText.getText().toString().substring(start,end);
                Entities.getEntityName().add(EntityName1);
                Entities.getStart().add(start);
                Entities.getEnd().add(end);
                Entities.getNerTag().add("TITLE");
                responseText.setText(styled2);
                actionMode.finish();
                break;
        }
        return true;//返回 true 则系统的"复制"、"搜索"之类的 item 将无效，只有自定义 item 有响应
    }
```

3.6 token，triples，entities 数据的本地存储

本功能采用 sharedPrefernces 方法本地暂存。以存储 token 数据为例，从服务器端得到的用户 token 数据，保存在本地一个名为“loginToken”的 xml 文件中，该数据可以在“/data/data/<包名>”目录下查看。当需要使用 token 发送请求时，再从该文件中取得并发送。

本地保存：

```
user.setToken(token);
SharedPreferences sp = getSharedPreferences("loginToken", MODE_MULTI_PROCESS);
SharedPreferences.Editor editor = sp.edit();
```

```
editor.putString("username", user.getUsername());//对 username 进行存储，在主界面个人信息中显示
editor.putString("token", token);//存储 token 验证登录情况
editor.apply();
```

本地获取：

```
SharedPreferences sp = getSharedPreferences("loginToken", MODE_MULTI_PROCESS);
user.setUsername(sp.getString("username", "user"));
user.setToken(sp.getString("token"," "));
```

3.7 关系标注动态生成显示的表格并设计 click 事件

由于每次获得的关系标注数据不一致，每次需要显示的内容的行数也不一致，因此需要动态生成显示的表格格式。我们采用了在 activity 中用代码创建控件的方法，在创建的同时，给控件添加 onclick 事件，判断是否对文本输出内容做出修改操作。而且每次在生成新的表格时，需要把以前已经创建好的行删除。

```
//清空上次操作
int len = tableLayout.getChildCount();
if (len > 1) { //这里的判断我是为了实现动态更新数据...保留标题
    //必须从后面减去子元素
    for (int i = len + 1; i > 0; i--) {
        tableLayout.removeView(tableLayout.getChildAt(i));
    }
}

for(int i = 0; i < triples.getSize(); i++){
    TableRow row = new TableRow(getApplicationContext());
    TableLayout.LayoutParams row_style = new
TableLayout.LayoutParams(TableLayout.LayoutParams.MATCH_PARENT,TableLayout.LayoutParams.WRAP
_CONTENT);
    row_style.setMargins(16,16,16,16);
    row.setLayoutParams(row_style);
    row.setGravity(View.TEXT_ALIGNMENT_CENTER);

    TextView left_entity = new TextView(getApplicationContext());
    MyMessageTools.setRelationViewTextStyle(left_entity,true);
    left_entity.setTextColor(Color.BLUE);
    left_entity.setText(response_leftEntity[i]);
    row.addView(left_entity);

    TextView right_entity = new TextView(getApplicationContext());
    MyMessageTools.setRelationViewTextStyle(right_entity,true);
    right_entity.setTextColor(Color.RED);
    right_entity.setText(response_rightEntity[i]);
```

```
    row.addView(right_entity);

    final TextView relation = new TextView(getApplicationContext());
    MyMessageTools.setRelationViewTextStyle(relation,false);
    int relation_id = Integer.parseInt(response_relationId[i]);
    String relation_text = null;
    if(relation_id == 0){
        relation_text = MyMessageTools.unicodeToUtf8("任职");
    }
    else if(relation_id == 1){
        relation_text = MyMessageTools.unicodeToUtf8("亲属");
    }
    relation.setText(relation_text);
    relation.setTextColor(Color.BLACK);
    row.addView(relation);

    //checkbox
    CheckBox checkbox = new CheckBox(getApplicationContext());
    checkbox.setChecked(true);
    row.addView(checkbox);

    tableLayout.addView(row);
    final int finalI = i;
    relation.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    int temp = (Integer) triples.getChange().get(finalI);
                    temp++;
                    triples.getChange().set(finalI,temp);
                    if(temp%2==0){
                        relation.setText(MyMessageTools.unicodeToUtf8("任职"));
                    }
                    else if(temp%2==1){
                        relation.setText(MyMessageTools.unicodeToUtf8("亲属"));
                    }
                }
            });
        }
    });
}
```

四．开发困难及解决方案

4.1 登录成功以后的界面无法跳转

解决方案：界面跳转是在 UI 线程之外的，在之前的输出信息部分，我们采用把 Toast 放在 Looper 之间在进行成功输出的方法，但是在 Looper 之外的界面跳转功能仍然无法实现，最后我们将 Looper 换成 runonUIthread 的方法，跳转成功。

4.2 实体标注采用输入框输入标注文段中所有相同字符串

解决方案：放弃输入框输入标注的做法，采用改写 callback 的方式，让用户可以采用长按选中一段文字的方式并点击长按后菜单中对应选项对这段文字进行标注，使用户的标注方式更加自由化，并且提高用户标注的准确性和交互体验。

4.3 sharedPreferences 本地暂存数据不生效

解决方案：将 getSharedPreferences()的模式设置成 MODE_MULTI_PROCESS。由于 Android Studio 版本的问题，在现在版本上操作时，sharedPreference 的 MODE_MULTI_PROCESS 属性不是默认开启的，需要手动开启。否则将不能在多个进程中访问这个文件。因此会读取不到数据。

4.4 关系标注上传后再次点击操作发生程序闪退

解决方案：由于在动态生成关系标注的显示表格时，同时添加了 TextView 的 onclick 事件，可能产生冲突等问题。因此为了避免发生此类问题，在上传完毕数据后，强制刷新一遍当前界面，获取下一段关系标注的文本。

五．分工

姓名	学号	完成任务
许多	3160103958	登录、注册、实体标注、屏幕滚动
吕慕凡	3160102463	内容获取、结果上传、关系标注、登出、字体大小

六．总结

本次实验中，小组两人配合默契，查阅大量资料，从零做起，完成了实验要求内容。但由于时间关系，部分可提升的功能依然待完成。比如登录状态下在菜单栏中显示用户名，我们尝试了多种方法，但是依然没有找到一个更加合适的解决办法，还有字体大小的调控问题，我们使用自定义的 TextbarTool 的回调来对字体进行调整，但遗憾的是字体仍然没有任何反应。时间关系我们最后还是选择完成代码的主体功能，但这些小缺憾在之后仍可以继续完善。

通过本次实验，我们学习到了更多开发 APP 的技巧，获得了更多安卓编程经验，对 java 的使用更加熟练，对 Android studio 这个平台也更加熟悉和了解。在调试时我们能更加准确的解读出调试日志带给我们的讯息，对各个控件的使用和功能也更加熟练，很多地方也让我们增进了对线程这一重要概念的了解。

本次实验和平时作业不同的是，由于这是一个比平时实验更加庞大的系统，我们的编程技术和模块的构建以及代码的可读性和规划都显得特别重要，当我们把 MainMenuActivity 的代码扩展得越来越多的时候，我们感受到在一个文件中的代码的长度对编程效率也是有非常大的影响；当我们建立的 Java 文件越来越多，界面相关 xml 文件也多起来的时候，一个一看就让两人同时理解功能的文件名也可以为我们节省大量的时间。此外，在小组两人的相互配合中，我们也更能了解到一个团队在合力编写一个程序的时候会有特别多比单人编程需要注意的地方，比如接口和命名方式等，合理的规划才能让团队更好的配合。经过了这一次的 APP 开发挑战，相信我们在期末可以拿出更好的作品。