

**计算机科学与技术系**

**实 验 报 告**

**2018-2019学年第一学期**

课程（模块） Android嵌入式软件开发

实验名称 天气预报

班 级 15计科2班

姓名(学号) 刘飞鸿（1501011026）

同组成员 于飞宇（1504012002）

评阅教师 孙欣欣

成 绩

**2018 年 11 月 16 日**

**一、实验目的：**

（本次实验要求达到的目的）

完成天气预报综合项目开发，掌握UI设计与开发、事件处理、Activity、网络编程、XML/JSON数据解析相关技术和原理。

**二、实验要求**

（本次实验的实验要求）

1. 创建项目，自行设计项目功能模块与UI界面
2. 项目至少包含6个界面
3. 项目需包括Web+Android端联网登录、注册功能，Web服 务器端代码请自行实现
4. 项目需包括天气预报功能，详细要求请参见实验过程
5. 2人一组完成项目开发
6. 完成综合实验设计报告，每组提交一份电子版+纸质版实验报告
7. 11月16日之前完成项目开发，并提交完电子版和纸质档实验报告

**三、任务分组说明**

（项目组成员工作内容介绍）

刘飞鸿：主要编写安卓客户端的注册，登录和天气预报的功能。

于飞宇：编写web端的注册，登录。以及计时器，定时器代码的编写

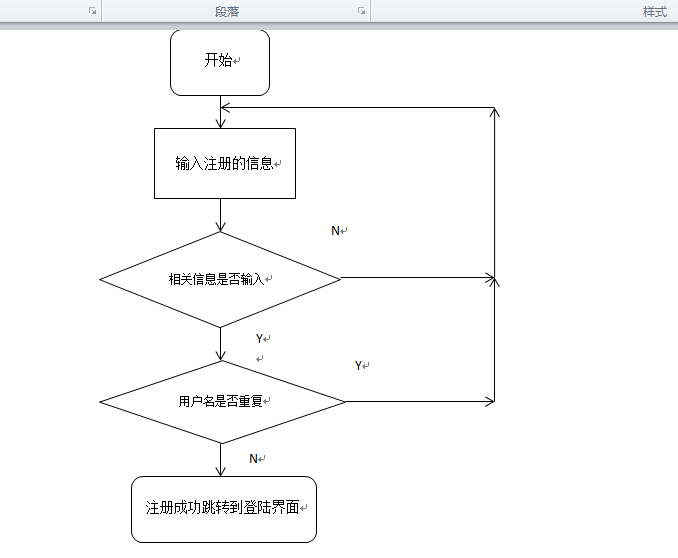
**四、系统设计与关键技术**

（系统包含功能模块，流程图，涉及关键技术）

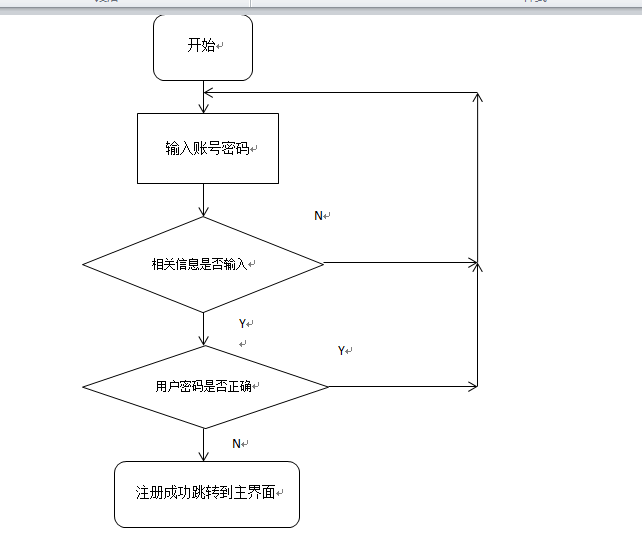
1、功能模块：

注册模块，登录模块，主界面模块，计算器器模块，计时器模块。天气预报模块。

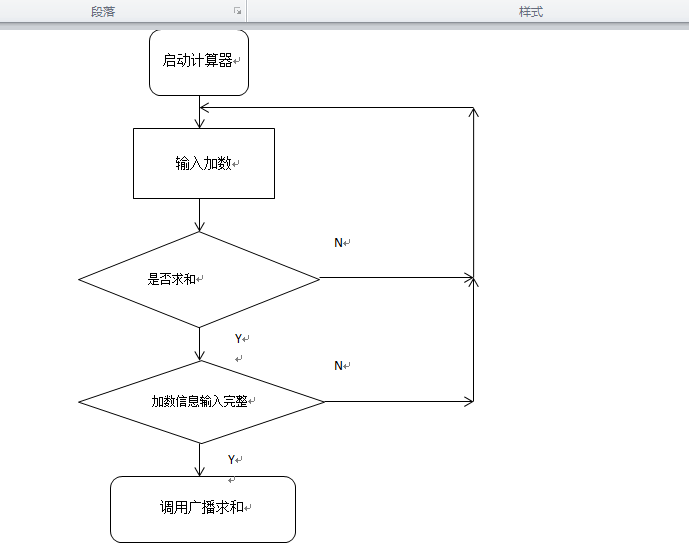
2、注册模块的实现流程图：



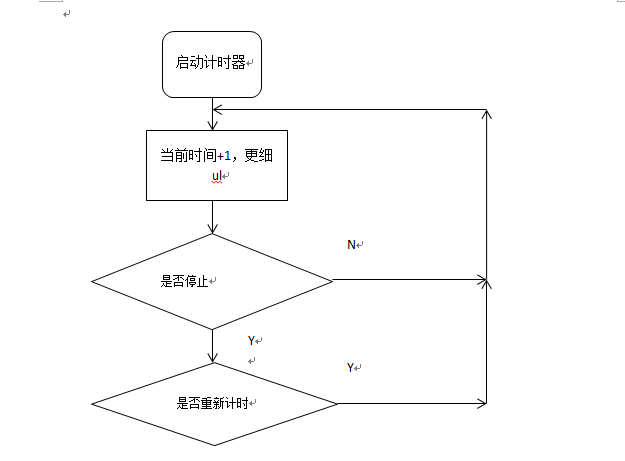
3、登录模块的实现流程图：



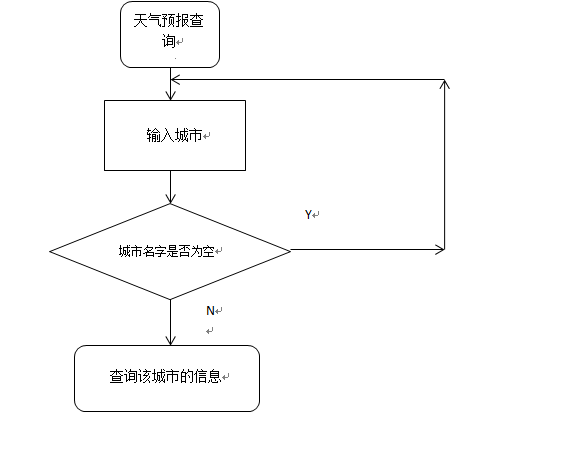
4、计算器模块的实现流程图：



5、计时器模块的实现流程图：



6、天气预报模块的实现流程图：



7、关键技术：

1.网络编程关于.HttpURLConnection的使用

2.json数据的解析

3.Hander的使用

4.多线程的使用

5.UI事件的处理

6.广播与activity之间的相互通信。

7.Web服务端

**五、系统实现**

（项目具体实现，截图，关键代码）

**Web服务端:**

1.采用了的是ssm框架。编写相关的bean，dao，xml文件，以及service和serciceImpl的实现。

用户类的信息：

public class User {  
  
 private Integer id;  
 private String userName;  
 private String password;  
  
 public Integer getId() {  
 return id;  
 }  
  
 public void setId(Integer id) {  
 this.id = id;  
 }  
  
 public String getUserName() {  
 return userName;  
 }  
  
 public void setUserName(String userName) {  
 this.userName = userName;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
 public User(Integer id, String userName, String password) {  
 this.id = id;  
 this.userName =userName ;  
 this.password = password;  
 }  
 public User(String userName, String password) {  
  
 this.userName = userName;  
 this.password = password;  
 }  
 public User() {  
  
 }  
}

**插入注册用户的信息**  
<insert id="insertSelective" parameterType="com.jin.bean.User">  
 INSERT into tbl\_user(user\_name,password) VALUES (#{userName,jdbcType=VARCHAR},#{password,jdbcType=VARCHAR})  
</insert>

查询用户名是否存在  
<select id="countUsername" parameterType="string" resultType="long">  
 SELECT count(\*)  
 from tbl\_user  
 where user\_name=#{userName}  
</select>

查询用户的信息  
<select id="findPassword" parameterType="string" resultType="string">  
 SELECT password  
 from tbl\_user  
 where user\_name=#{userName}

用户的注册的

@RequestMapping("/registrerUser")  
public void registrerUser(User user, HttpServletResponse response)throws IOException  
{  
 String name=user.getUserName();  
 if(service.countUsername(name))  
 {  
 service.insertSelective(user);  
  
 response.getWriter().print("success");  
 }  
 else {  
  
 response.getWriter().print("error");  
 }  
  
}

用户的登录

@RequestMapping("/LoginServlet")  
public void Login(User user, HttpServletResponse response)throws IOException  
{  
  
  
 String passerod=service.findPassword(user.getUserName());  
  
 if(user.getPassword().equals(passerod))  
 {  
 response.getWriter().print("success");  
  
 }  
 else  
 response.getWriter().print("error");  
  
}

**2.注册的客户模块:**

public static HttpURLConnection getConnection3(String data) throws Exception {  
  
 //通过URL对象获取联网对象  
 URL url= new URL("http://192.168.43.37:8080/registrerUser");  
// URL url= new URL("www.baidu.com");  
 HttpURLConnection conn = (HttpURLConnection) url.openConnection();  
  
 conn.setRequestMethod("POST");  
 conn.setReadTimeout(5000);  
 conn.setConnectTimeout(5000);  
 //设置运行输入,输出:  
 conn.setDoOutput(true);  
 conn.setDoInput(true);  
 //Post方式不能缓存,需手动设置为false  
 conn.setUseCaches(false);  
 //我们请求的数据:  
  
 //这里可以写一些请求头的东东...  
 //获取输出流  
 OutputStream out = conn.getOutputStream();  
 out.write(data.getBytes());  
 out.flush();  
 return conn;  
 }

**3.登录模块**

public static HttpURLConnection getConnection(String data) throws Exception {  
  
 //通过URL对象获取联网对象  
 URL url= new URL("http://192.168.43.37:8080/LoginServlet");  
// URL url= new URL("www.baidu.com");  
 HttpURLConnection conn = (HttpURLConnection) url.openConnection();  
  
 conn.setRequestMethod("POST");  
 conn.setReadTimeout(5000);  
 conn.setConnectTimeout(5000);  
 //设置运行输入,输出:  
 conn.setDoOutput(true);  
 conn.setDoInput(true);  
 //Post方式不能缓存,需手动设置为false  
 conn.setUseCaches(false);  
 //我们请求的数据:  
  
 //这里可以写一些请求头的东东...  
 //获取输出流  
 OutputStream out = conn.getOutputStream();  
 out.write(data.getBytes());  
 out.flush();  
 return conn;  
 }

**4.主界面模块**

protected void jisuanqi(View view)  
{  
 startActivity(new Intent(this, Jisuanqi.class));  
}  
protected void jishiqi(View view)  
{  
 startActivity(new Intent(this, Jishiqi.class));  
}  
protected void tianqiyubao(View view)  
{  
 Intent intent=new Intent(this, WeaForecast.class);  
 startActivity(intent);  
}

**5.计算器模块**

@Override  
 public void onReceive(Context context, Intent intent) {  
  
 int a1=intent.getIntExtra("a1",0);  
 int a2=intent.getIntExtra("a2",0);  
 int a3=a1+a2;  
  
 message.getMsg(a3+"");  
 }  
  
public interface Message {  
 public void getMsg(String str);  
 }

**6.计时器模块**

protected void setTimedisplay()  
{  
 String display;  
 long hours;  
 long minutes;  
 long seconds;  
 long diff;  
 long timeNow;  
  
 if(timerRuning)  
 {  
 timeNow=System.currentTimeMillis();  
 }  
 else  
 {  
 timeNow=lastStopped;  
 }  
 diff=timeNow-startAt;  
 if(diff<0)  
 diff=0;  
  
 seconds=diff/1000;  
 minutes=seconds/60;  
 hours=minutes/60;  
 seconds=seconds%60;  
 minutes=minutes%60;  
  
 display=String.format("%d",hours)+" : "+  
 String.format("%02d",minutes)+" : "+  
 String.format("%02d",seconds);  
 textView.setText(display);  
}  
  
class UpdateTimer implements Runnable{  
  
 @Override  
 public void run()  
 {  
 if(handler!=null)  
 {  
 setTimedisplay();  
 handler.postDelayed(this,UPDATE\_EVERY);  
  
 }  
 }  
}

**7.天气预报模块**

public static HttpURLConnection getConnection2(String data) throws Exception {  
 String path="http://v.juhe.cn/weather/index"+data;  
 URL url = new URL(path);  
 HttpURLConnection conn = (HttpURLConnection) url.openConnection();  
 // 设置连接超时为5秒  
 conn.setConnectTimeout(5000);  
 // 设置请求类型为Get类型  
 conn.setRequestMethod("GET");  
 return conn;  
}

**8.相关流的处理**

public class StreamChangeStrUtils {  
  
 public static String toChange(InputStream inputStream) throws Exception {  
 ByteArrayOutputStream bos = new ByteArrayOutputStream();//数组流，在流的内部有个缓冲区，可以进行转换成字节  
 //下面是属于io流的知识，在此不再赘述  
 byte b[]=new byte[1024];  
 int len=-1;  
 while ((len=inputStream.read(b))!=-1){  
 bos.write(b,0,len);  
 }  
 inputStream.close();//关闭流,数组流会自动关闭，关闭是否都可以  
 String str = new String(bos.toByteArray());  
 //服务器默认返回的是gbk，如果要在android端解决乱码，可以在此设置为gbk,一般提倡的是服务器解决  
 // 让服务器给我们返回utf-8，因为在android本地默认的是utf-8  
 return str;  
 }  
  
}

**9.Jason的处理**

public static Weather handJson(String jsonstr) throws Exception  
{  
 JSONObject root = new JSONObject(jsonstr);  
 String resultcode=root.getString("resultcode");  
 String reason=root.getString("reason");  
 int error\_code=root.getInt("error\_code");  
  
 //城市瞎写  
 if(error\_code!=0)  
 return new Weather(resultcode,reason,error\_code,null);  
  
  
 JSONObject root2=root.getJSONObject("result");  
 JSONObject todayweather=root2.getJSONObject("today");  
 String temperature=todayweather.getString("temperature");  
 String weather=todayweather.getString("weather");  
 String wind=todayweather.getString("wind");  
 String dressing\_advice=todayweather.getString("dressing\_advice");  
 Weather weather1=new Weather(resultcode,reason,error\_code,new TodayWeather(temperature,weather,wind,dressing\_advice) );  
 return weather1;  
}

**10.辅助类的编写**

public class Weather {  
  
 private String resultcode;  
 private String reason;  
 private int error\_code;  
 private TodayWeather todayWeather;  
  
 public Weather() {  
 }  
  
 /\*\*  
 \*  
 \* @param resultcode  
 \* @param reason  
 \* @param error\_code  
 \* @param todayWeather  
 \*/  
 public Weather(String resultcode, String reason, int error\_code, TodayWeather todayWeather) {  
 this.resultcode = resultcode;  
 this.reason = reason;  
 this.error\_code = error\_code;  
 this.todayWeather = todayWeather;  
 }  
  
 public String getResultcode() {  
 return resultcode;  
 }  
  
 public void setResultcode(String resultcode) {  
 this.resultcode = resultcode;  
 }  
  
 public String getReason() {  
 return reason;  
 }  
  
 public void setReason(String reason) {  
 this.reason = reason;  
 }  
  
 public int getError\_code() {  
 return error\_code;  
 }  
  
 public void setError\_code(int error\_code) {  
 this.error\_code = error\_code;  
 }  
  
 public TodayWeather getTodayWeather() {  
 return todayWeather;  
 }  
  
 public void setTodayWeather(TodayWeather todayWeather) {  
 this.todayWeather = todayWeather;  
 }  
  
 @Override  
 public String toString() {  
 return "Weather{" +  
 "resultcode='" + resultcode + '\'' +  
 ", reason='" + reason + '\'' +  
 ", error\_code=" + error\_code +  
 ", todayWeather=" + todayWeather +  
 '}';  
 }

public class TodayWeather {  
  
 private String temperature;  
 private String weather;  
 private String wind;  
 private String dressing\_advice;  
  
 public TodayWeather() {  
 }  
  
 /\*\*\*  
 \*  
 \* @param temperature  
 \* @param weather  
 \* @param wind  
 \* @param dressing\_advice  
 \*/  
 public TodayWeather(String temperature, String weather, String wind, String dressing\_advice) {  
 this.temperature = temperature;  
 this.weather = weather;  
 this.wind = wind;  
 this.dressing\_advice = dressing\_advice;  
 }  
  
 public String getTemperature() {  
 return temperature;  
 }  
  
 public void setTemperature(String temperature) {  
 this.temperature = temperature;  
 }  
  
 public String getWeather() {  
 return weather;  
 }  
  
 public void setWeather(String weather) {  
 this.weather = weather;  
 }  
  
 public String getWind() {  
 return wind;  
 }  
  
 public void setWind(String wind) {  
 this.wind = wind;  
 }  
  
 public String getDressing\_advice() {  
 return dressing\_advice;  
 }  
  
 public void setDressing\_advice(String dressing\_advice) {  
 this.dressing\_advice = dressing\_advice;  
 }  
  
 @Override  
 public String toString() {  
 return "temperature='" + temperature + '\'' +"\n"+  
 "weather='" + weather + '\'' +"\n"+  
 "wind='" + wind + '\'' +"\n"+  
 "dressing\_advice='" + dressing\_advice + '\'';  
 }  
}

**五：运行结果截图**

1.注册：

刚开始我们登录需要app，需要我们填写账号密码，因为我们是第一次登app，所以需要先注册。注册有一个要求，就是用户名不能相同。如果相同的，我们会提示的。如果注册成功的话，我们会直接跳入登录界面。











2.登录模块：

输入账号密码，如果账号密码正确，就会跳转到主界面。如果账号密码不正确，它会提示的。



3.主界面主要提供功能的跳转



4.天气预报：

如果城市不存在的话，我们会提示该城市不存在。如果城市存在。我们将显示查询城市的温度。风向以及生活建议。







5.计时器：

按start的时候它就开始计时。按停止的时候，它就停止计时。



6.计算器：

主要是用广播的方法去实现。



**六、实验总结：**

（实验过程中的问题分析、产生的原因以及解决方法； 有待改进、优化思路）

1.问题一:Web端和安卓端不能正常通信。

**产生的原因**：web端:当时电脑连接的是hfuu。安卓端：当时用的说流量。连接hfuu的ip会变。所以就无法正常通信。

**解决方法**：手机开热点个电脑。这样电脑和手机就会处在同一个局域网，这样更改下URL的路径就可以正常通信了。

2.有待改进的地方：

设计的界面不是很美观。需要美化。

3.查询天气的功能：

只能查询当天的天气。应该让他查询未来七天的天气。

Web端:应该用webservice。