# 目录

# 1.系统需求分析

## 1.1 系统功能及框图

### 1.1.1 功能需求

1. 显示当前天气状况。可以在主界面以图片和文字的形式看到当前温度、当天最高及最低气温，预报未来 24 小时天气情况，包括温度、风向及雨雪情况等，提供适当生活建议，如穿衣指数、旅游指数、洗车指数等。

2. 预报未来 15 天天气状况，包括最高最低气温及其折线图。

3. 提供城市选择及添加功能。添加多个城市后，可以随时切换不同城市显示当地天气及预报。

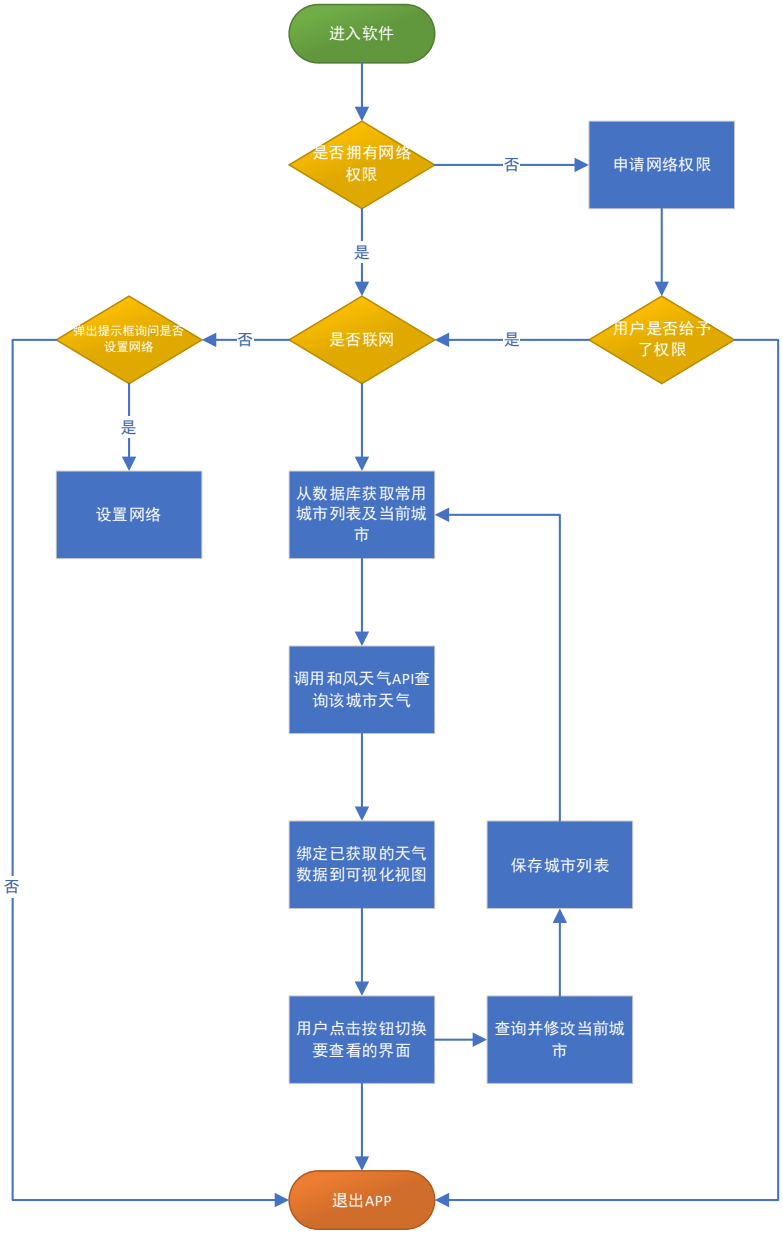


图 1-1 流程图

## 1.2 系统需求

### 1.2.1 业务需求

该应用帮助使用者了解本地以及用户所设置城市的天气，打开应用可以通过定位看到本地的最近几天的天气情况并可以通过输入城市名查询其他城市天气。在详情中可以查询当日二十四小时分段温度、生活指数等信息，在城市中可以选择用户想要了解天气情况的城市，即可以获取相应城市天气信息。

### 1.2.2 面向的用户类型

可以基本使用智能机的人

### 1.2.3 用户需求

可以对城市进行选择，以了解这个城市最近几天的天气情况
可以查看当前城市实时天气，为今日出行提供规划依据
可以生活指数了解城市景点、穿衣指数等信息
通过注册、登录记录保留用户使用数据

### 1.2.4 软件需求

需要通过相应接口，可以实现对天气的查询；
需要给予用户友好的交互提示；

## 1.3 该项目涉及到的技术点

### 1.3.1 GUI 界面设计

布局以 LinearLayout 为主，嵌套 RelativeLayout；利用 ScrollView 及 HorizontalScrollView 实现滚动条查看当日 24 小时天气预报等数据信息；RadioGroup 实现底部导航栏；AutoCompleteTextView 在添加城市时实现自动补全文本；其他常见组件，如 Button、TextView 等。

### 1.3.2 数据适配器 BaseAdapter

BaseAdapter 是一个抽象类，常用于和 Android 中的一些控件如 ListView/GridView/ExpandableListview/Spinner 等控件结合来显示数据的一种控件，在本项目中用于城市设置。

### 1.3.3 数据库存储 SQLite

本软件中有两部分数据需要存储，一个是显示页面的数据，另一个是详细页面的数据。但是数据量都不是很大，因此可以选择 SQLite 数据库作为存储数据的方法，建立数据库。

### 1.3.4 数据格式转化 Gson

Gson 是 Google 提供的用来在 Java 对象和 JSON 数据之间进行映射的 Java 类库。可以将一个 JSON 字符串转成一个 Java 对象，或者反过来。
利用 AndroidStudio 中的工具，能够快速将 JSON 字符串转换成一个 Java Bean，免去我们根据 JSON 字符串手写对应 java Bean 的过程。

### 1.3.5 图表库 Hellocharts

为了实现天气数据的可视化，我们引用了适用于 Android 的开源图标库 HelloCharts【https://github.com/lecho/hellocharts-android】，它可以实现折线图（三次线、实线、散点）、柱状图（分组、堆积、负值）、饼形图等多种图的快速绘制。

## 1.4 项目分工及组内互评得分

| 序号 | 姓名 | 主要完成任务 | 组内评分（满分 100 分） |
|---|---|---|---|
| 1 | 张培风 | 查找资料及代码编写 | |
| 2 | 潘肖男 | 查找资料及完成作业报告、PPT | |
| 3 | 王乐云 | 查找资料及完成作业报告 | |

# 2.数据存储设计

本软件中仅有常用城市需要进行存储，数据量不是很大，因此可以选择 SQLite 数据库作为存储数据的方法，建立数据库 db 存储页面的数据。

## 2.1 SQLite 存储介绍

1.建表：

```
public static final String CREATE_USERDATA = "create table if not exists
CitySQ(name varchar(80) primary key,  isSelect varchar(8))";

public void onCreate(SQLiteDatabase db) {
        db.execSQL(CREATE_USERDATA);
    }
```

2.调用数据库对象：

```
dbHelper = new MySQHelper(context, "CitySQ", null, 1);
db = dbHelper.getWritableDatabase();
```

3.调用 cursor 进行查询：

```
Cursor c = db.rawQuery("select * from CitySQ", null);
```

4.添加：

```
db.insert("CitySQ",null, cValue);
```

5.修改：

```
db.update("CitySQ", cValue, whereClause, whereArgs);
```

6.删除：

```
db.execSQL("delete from CitySQ where name=?", new String[] { name });
```

## 2.2 数据表结构

| | | | |
|---|---|---|---|
| primary key | name | varchar（80） | 记录城市名字 |
| | isSelect | varchar（8） | 记录是否被选中为当前位置 |

表 2-1 数据表 CitySQ

# 3.具体编码及截图

## 3.1 整体实现

### 3.1.1 Fragment 实现分页功能

1.技术介绍

Fragment 是一种可以嵌入在活动中的 UI 片段，能够让程序更加合理和充分地利用大屏幕的空间，出现的初衷是为了适应大屏幕的平板电脑，可以将其看成一个小型 Activity，又称作 Activity 片段。使用 Fragment 可以把屏幕划分成几块，然后进行分组，进行一个模块化管理。不能够单独使用 Fragment，需要嵌套在 Activity 中使用，其生命周期也受到宿主 Activity 的生命周期的影响。

在天气预报 APP 中，我们使用 Fragment 功能对页面进行分块，使每个界面由多个模块组成，Fragment 有自己的生命周期和接受、处理用户的事件，这样我们就不需要在 Activity 写一堆控件的事件处理的代码了，同时，我们可以动态的添加、替换和移除某个 Fragment。

2.具体实现

本 APP 主要设计了三个主界面：天气预报、天气详情、常用城市，所以我们建立了三个继承于 Fragment 的对象，每个对象单独调用数据，对应不同的布局文件进行相应展示。

### 3.1.2 ViewPager 实现滑动翻页

1.技术介绍

Viewpager，视图翻页工具，提供了多页面切换的效果。Android 3.0 后引入的一个 UI 控件，位于 v4 包中。低版本使用需要导入 v4 包，但是现在我们开发的 APP 一般不再兼容 3.0 及以下的系统版本，另外现在大多数使用 Android studio 进行开发，默认导入 v7 包，v7 包含了 v4，所以不用导包，越来越方便了。

Viewpager 使用起来就是我们通过创建 adapter 给它填充多个 view，左右滑动时，切换不同的 view。Google 官方是建议我们使用 Fragment 来填充 ViewPager 的，这样可以更加方便的生成每个 Page，以及管理每个 Page 的生命周期。

而FragmentPagerAdapter则为Fragment页面提供了相对应的页面适配器创建方法，我们可以直接继承使用。

2.具体实现

1）在 xml 中进行引用

```
<android.support.v4.view.ViewPager
    android:id="@+id/vpager"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_above="@id/div_tab_bar" />
```

2）设置各个 page 界面

3）创建适配器

```
public MyFragmentPagerAdapter(FragmentManager fm, List<Fragment> fragments)
    super(fm);

    myFragment1 = new Fragment1();
    myFragment2 = new Fragment2();
    myFragment3 = new Fragment3();
}
```

图 3-1 MyFragmentPagerAdapter 适配器定义

4）设置适配器

```
mAdapter = new MyFragmentPagerAdapter(getSupportFragmentManager(),fragments);
vpager = findViewById(R.id.vpager);
vpager.setAdapter(mAdapter);
```

图 3-2 适配器设置

5）利用 onPageScrollStateChanged 方法来监听滑动状态，并根据当前所在按钮状态相应显示不同的分页

```
@Override
public void onPageScrollStateChanged(int state) {
    //state的状态有三个，0表示什么都没做，1正在滑动，2滑动完毕
    if (state == 2) {
        switch (vpager.getCurrentItem()) {
            case PAGE_ONE:
                rb0.setChecked(true);
                break;
            case PAGE_TWO:
                rb1.setChecked(true);
                break;
            case PAGE_THREE:
                rb2.setChecked(true);
                break;
        }
    }
}
```

图 3-3 滑动状态监听

### 3.1.3 RadioButton 实现底部栏

1. 技术介绍

RadioButton 是单选按钮，允许用户在一个组中选择一个选项。同一组中的单选按钮有互斥效果。但是除了单选框以外，由于它的唯一可选性，它还经常被用来作为底部导航栏。

2.具体实现

1）xml 布局

```
<RadioGroup
    android:id="@+id/rg_tab_bar"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:layout_alignParentBottom="true"
    android:background="@color/bg_white"
    android:orientation="horizontal">

    <RadioButton
        android:id="@+id/rb0"
        style="@style/tab_menu_item"
        android:drawableTop="@drawable/forecast"
        android:text="@string/ForecastBtn"
        android:textSize="12sp"
        android:background="@color/color_radiobutton"/>

    <RadioButton
        android:id="@+id/rb1"
        style="@style/tab_menu_item"
        android:drawableTop="@drawable/detail"
        android:text="@string/DetailBtn"
        android:textSize="12sp"
        android:background="@color/color_radiobutton"/>

    <RadioButton
        android:id="@+id/rb2"
        style="@style/tab_menu_item"
        android:drawableTop="@drawable/city"
        android:text="@string/CityBtn"
        android:textSize="12sp"
        android:background="@color/color_radiobutton"/>
```

图 3-4 RadioButton 样式代码

2）设置按钮选中变色样式

通过 color_radiobutton 实现点击底部按钮变色，xml 代码如下所示：

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">


    <item android:state_checked="true" android:color="#679ED5"/>
    <!-- not selected -->
    <item android:color="#104E8B"/>


</selector>
```

图 3-5 底部栏颜色样式

我们通过设置按钮颜色实现按钮点击变色功能，一般情况下，按钮颜色被设置为“#104E8B”，当按钮被点击时，颜色变为“#679ED5”。

### 3.1.4 ScollView 实现界面滚动

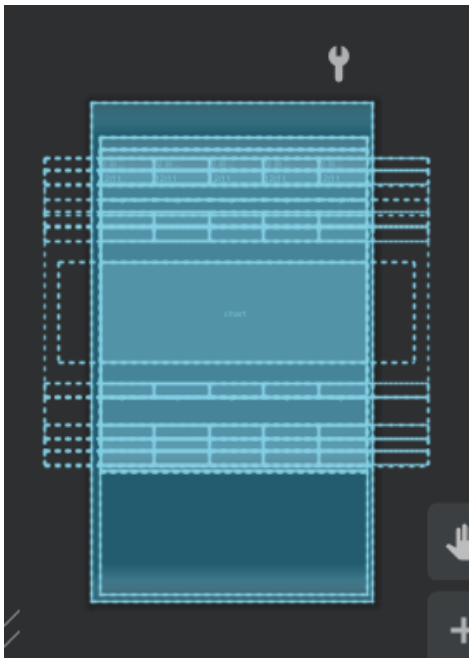

图 3-6 界面 1 布局图

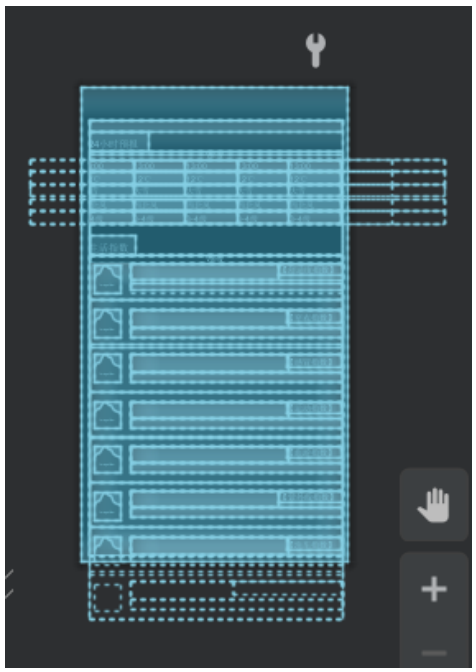

图 3-7 界面 2 布局图

如上图所示，由于屏幕大小限制，我们的所有天气相关控件无法在单屏上完整显示，为了解决该问题，我们使用了滚动视图 Scollview，它的作用是当在一个屏幕的像素显示不下绘制的 UI 控件时，可以采用滑动的方式，使控件显示。

Xml 布局文件代码如下图：

```
<ScrollView
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">
        <HorizontalScrollView
            android:scrollbars="none"
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="10dp"
                android:layout_marginBottom="10dp"
                android:orientation="vertical">

                <LinearLayout
                    android:layout_width="match_parent"
                    android:layout_height="20dp"
                    android:orientation="horizontal">
```

图 3-8 上下滚动样式

在使用 Scollview 后，我们可以通过上下滑动的方式查看完整的气温折线图及具体天气情况等，同时考虑到用户手机屏幕宽度不同，APP 可能无法兼容的问题，我们又利用了相似的水平滚动试图 HorozontalScollview 解决控件超出左右屏幕的情况

```
<HorizontalScrollView
    android:scrollbars="none"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:layout_marginBottom="10dp"
        android:orientation="vertical">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="20dp"
            android:orientation="horizontal">



            <TextView
                android:id="@+id/day2"
                style="@style/compentToday"
                android:gravity="center"
```

图 3-9 左右滚动样式

## 3.2 主界面

### 3.2.1 一周天气预报界面

1. 运行界面截图


图 3-10 当前城市天气预报界面

2. 实现功能：
1）利用主体组件展示当前所选城市及其实时温度
2）通过点击左上角按钮进入城市添加功能
3）用文字详细展示今明两天的天气状况预报
4）用折线图可视化的方式展示未来多天的最高最低气温以及其具体日期等
5）未来天气情况折线图可在当前页面上下以及左右滑动查看

### 3.2.2 今日天气详情界面

1. 运行界面截图


图 3-11 今日天气详情及生活指数界面

2. 实现功能

1）显示所选城市 24 小时内的天气预报

2）针对舒适度、紫外线等数据为用户提供舒适度指数

3）针对较长的生活指数提示可实现上下滚动查看详情

### 3.2.3 城市列表界面

1. 运行界面截图


图 3-12 常用城市列表界面

2. 实现功能：

1）展示用户当前所选城市

2）设置其他常用城市为当前城市，同时刷新天气数据

3）删除常用城市

### 3.2.4 城市添加界面

1. 运行界面截图

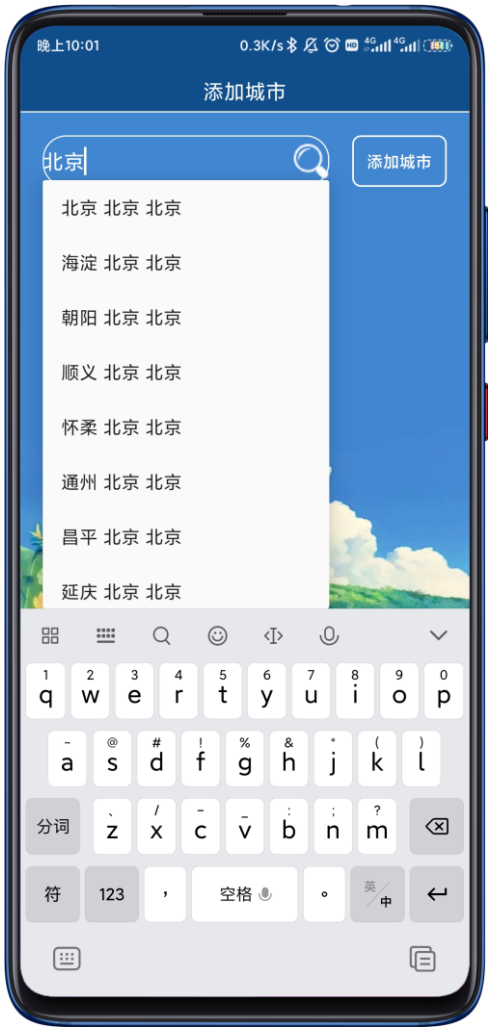


图 3-13 城市添加界面

2. 功能实现

1）搜索中国省份/城市/区

2）自动联想并补齐城市名称

3）将选定城市设为当前城市，并刷新天气数据

### 3.2.5 离线状态界面

1. 运行界面截图


图 3-14 网络设置界面图

2. 实现功能

1）当用户进入软件时，检测其联网状态

2）若无法联网，则跳转到系统网络设置界面

3）取消该提示窗，退出软件

## 3.3 各功能模块

### 3.3.1 实时温度模块


图 3-15 实时温度模块

1.调用 API

我们在 APP 中调用了和风天气的开发者 API，用于获取天气数据。

1）首先从和风天气官网注册获取到开发者 API

2）然后通过访问 API 可以直接获取到所查询内容的 Json 格式数据，如下图所示：


```
"data": [
  "HeWeather6": [
    {
      "basic": {
        "cid": "CN101251001",
        "location": "岳阳",
        "parent_city": "岳阳",
        "admin_area": "湖南",
        "cnty": "中国",
        "lat": "29.37029076",
        "lon": "113.13285828",
        "tz": "+8.00"
      },
      "update": {
        "loc": "2020-02-05 21:09",
        "utc": "2020-02-05 13:09"
      },
      "status": "ok",
      "now": {
        "cloud": "96",
        "cond_code": "104",
        "cond_txt": "阴",
        "fl": "11",
        "hum": "63",
        "pcpn": "0.0",
        "pres": "1014",
        "tmp": "13",
        "vis": "16",
        "wind_deg": "26",
        "wind_dir": "东北风",
        "wind_sc": "2",
        "wind_spd": "8"
      },
      "daily_forecast": [
```


图 3-16 部分 Json 返回数据

3）通过查询可得和风 API 可提供的数据如下图所示：

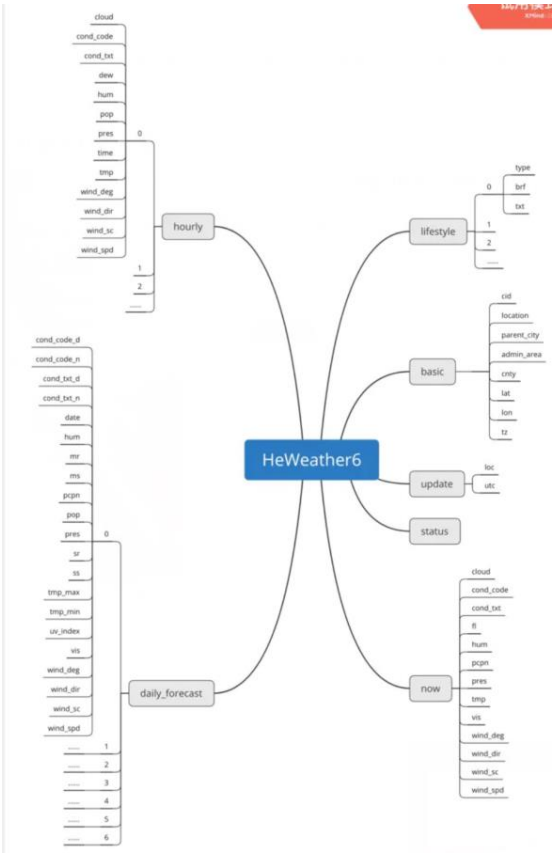


图 3-17 API 提供数据

4）再通过 Gson 将需要获取的 Json 数据进行格式转化，转为可利用的 Java 对象：

```
void getData(){
    (Thread) run() → {
            try {
                CityOperator cityOperator = new CityOperator(getContext());
                string_city = cityOperator.getIsSelectCity().toString2();
                string_hourly_weather_forcast = GetData.getJson( path: "https://free-api.heweather.com/s6/weather/hourly?location=" + string_city + "&
                string_Lifestyle_forcast = GetData.getJson( path: "https://free-api.heweather.com/s6/weather/lifestyle?location=" + string_city + "&ke
            } catch (Exception e) {
                e.printStackTrace();
            }
            handler.sendEmptyMessage( what: 0x001);
        }.start();
}

void parseData(){
    Gson gson = new Gson();
    hourlyWeatherForecast = gson.fromJson(string_hourly_weather_forcast,HourlyWeatherForecast.class);
    lifestyleForecast = gson.fromJson(string_Lifestyle_forcast,LifestyleForecast.class);
    draw();
}
```

图 3-18 调用 API 获取数据

5）接着利用 GsonFormat 工具将获取的 json 数据解析为 com.windy.forecast.request 包下的多个实体类

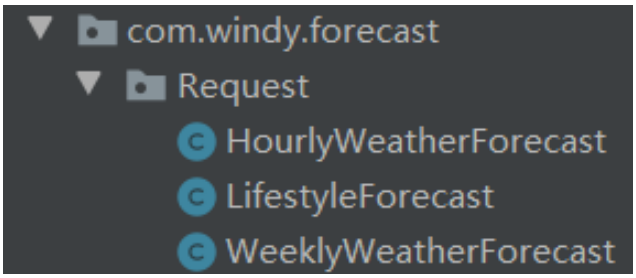


图 3-19 生成的实体类

6）获取具体数据并绑定到相应的 id 上：

*hourlyWeatherForecast.getHeWeather6().get(0).getHourly().get(i).getTmp()*

*weeklyWeatherForecast.getHeWeather6().get(0).getDaily_forecast().get(2).getDate()*

7）最后我们在 xml 中设置相应组件及其样式

## 3.3.2 一周天气预报模块

1.温度折线图

1）xml 布局

```
<lecho.lib.hellocharts.view.LineChartView
    android:id="@+id/chart"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

图 3-20 hellocharts 样式

2）获取温度数据并调用实例化的 LineChartView 对象，设置其 x 轴及 y 轴的值

```
List<Line> lines = new ArrayList<>();
LineChartView chart = view.findViewById(R.id.chart);
Line line1 = new Line(value1).setColor(Color.YELLOW);
line1.setHasLines(true);
line1.setHasPoints(true);
line1.setHasLabels(false);
Line line2 = new Line(value2).setColor(Color.BLUE);
line2.setHasLines(true);
line2.setHasPoints(true);
line2.setHasLabels(false);
line1.setPointColor(Color.WHITE);
line2.setPointColor(Color.WHITE);
line1.setPointRadius(4);
line2.setPointRadius(4);
lines.add(line1);
lines.add(line2);
line1.setStrokeWidth(2);
line2.setStrokeWidth(2);
line1.setCubic(false);
line2.setCubic(false);
LineChartData data = new LineChartData(lines);
data.setValueLabelBackgroundEnabled(false);
data.setValueLabelsTextColor(R.color.text_yellow);
data.setBaseValue(Float.NEGATIVE_INFINITY);
data.setValueLabelsTextColor(Color.WHITE);
data.setValueLabelTextSize(15);
chart.setLineChartData(data);
chart.setInteractive(false);
```

图 3-21 折线图绘制代码

2. API 获取天气数据：原理如上 3.3.1 所示

### 3.3.3 生活指数模块

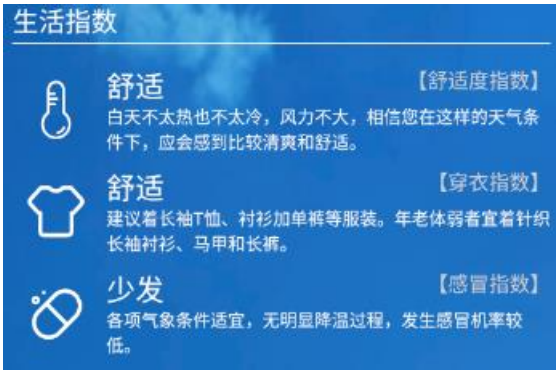


图 3-22 生活指数模块

1. 相应图标

为了保证 APP 界面的美观度，能够让用户尽量在第一眼就能获取到需要的信息，我们给每个不同的指数模块设置了相应的图标，如衣服图标代表穿衣指数

2. API 获取天气数据：原理如上 3.3.1 所示

### 3.3.4 城市列表

1. 圆角显示：
使用圆角样式主要是为了让布局中的控件看起来更美观一点。所以，我们在 drawable 定义 xml 文件，如下所示：

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle"
    >

    <stroke android:width="1dip"
     android:color="#ffffff"
     />

    <!-- 圆角 -->
    <corners
        android:bottomRightRadius="20dp"
        android:bottomLeftRadius="20dp"
        android:topLeftRadius="20dp"
        android:topRightRadius="20dp"
        />
</shape>
```

图 3-23 圆角样式

在需要此显示此样式时，将圆角的布局文件中引入本配置作为背景即可。

2. 管理城市
1）设立城市适配器

```java
public class CityAdapter extends BaseAdapter{
    private Context context;
    private List<City> listCity;
    private CityOperator cityOperator;

    public CityAdapter(List<City> listCity, Context context){
       this.listCity = listCity;
       this.cityOperator = new CityOperator(context);
    }
```

图 3-24 城市适配器

2）绑定到视图
3）监听视图删除/设置当前城市按钮
4）调用方法对数据库进行操作

```java
else  if(listCity.get(position).getIsSelect().equals("否")){
    viewHolder.citySet.setText("设为当前位置");
    viewHolder.citySet.setOnClickListener((v) → {
            City city1 = cityOperator.getIsSelectCity();
            cityOperator.update(city1);
            City city2 = new City(listCity.get(position).getName(),  isSelect: "否");
            cityOperator.update(city2);
            listCity = cityOperator.getItemCity();
            notifyDataSetChanged();
    });
    viewHolder.cityDelete.setText("删除");
    viewHolder.cityDelete.setOnClickListener((v) → {
            cityOperator = new CityOperator(context);
            cityOperator.delete(listCity.get(position).getName());
            listCity = cityOperator.getItemCity();
            notifyDataSetChanged();
```

图 3-25 对城市进行操作

## 3.3.5 添加城市

1. 自动补齐

1）设置 xml 组件

```
<AutoCompleteTextView
    android:layout_width="250dp"
    android:layout_height="45dp"
    android:id="@+id/autoCompleteTextView"
    android:textColor="#ffffff"
    android:completionThreshold="1"
    android:drawableRight="@drawable/search"
    android:singleLine="true"
    android:background="@drawable/app_list_corner_circle"
    android:hint="  请点击按钮添加城市"
    android:textColorHint="#ffffff"/>
```

图 3-26 AutocompleteTextView 组件

2）获取城市列表，自动提示补齐

```
private void bindView(){
    this.autoCompleteTextView = this.findViewById(R.id.autoCompleteTextView);
    Resources resources = this.getResources();
    String[] country = resources.getStringArray(R.array.city_array);
    ArrayAdapter<String> adapter = new ArrayAdapter<~>( context: this,
            android.R.layout.simple_list_item_activated_1,
            country
    );
    this.autoCompleteTextView.setAdapter(adapter);
    this.autoCompleteTextView.setOnClickListener(this);
    autoCompleteTextView = findViewById(R.id.autoCompleteTextView);
    btn_add = findViewById(R.id.btn_add);
    btn_add.setOnClickListener(this);
}
```

图 3-27 城市数组绑定视图

## 3.3.6 网络状态检测及设置

1. 网络权限检测

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

图 3-28 权限申请

2. 若无网络则，弹窗提示需要联网
3. 隐式启动系统网络设置

```
btn_setting.setOnClickListener((v) → {
        Intent intent = new Intent();
        intent.setAction(Settings.ACTION_DATA_ROAMING_SETTINGS);
        startActivityForResult(intent, requestCode: 0);
        finish();
});
```

图 3-29 启动网络设置


图 3-30 运行结果

# 4.总结

## 4.1 课程收获

从移动终端开发的课程中，我们学到了安卓开发的基础知识，如四大组件及页面的生命周期等，了解了一个 APP 从用户界面到数据库以及前后端交互的基本开发流程。

## 4.2 大作业完成过程

在大作业中，我们选择了以模拟天气预报 APP 为主题，主要完成了一款基于 android 平台的天气预报软件的设计与实现，通过调用天气相关 API 来获取天气数据并实现天气数据的可视化展示，提出了 android 用户界面设计、获取并解析城市列表数据的办法，给出了在用户界面上的原理与实现过程，最后通过模拟器和实体机分别进行了应用程序的调试。

在本次实验中，通过小组合作完成选题、需求分析、设计、调试、运行、文档书写、答辩等软件开发的完整过程，使我们对 Android 开发有了更加总览性的了解，锻炼了我们的动手能力，培养了我们的创新精神，在动手实践的过程中，我们在上课学习的现有知识的基础上，根据项目开发的具体要求进行知识拓展，通过网络学习新知识用于项目开发。

## 4.3 开发遇到的问题及深入思考

在本次实践中，我们也遇到了一些困难。比如说在针对 Listview 进行小时天气数据绑定时，因每小时天气数据 textview 的 id 各不相同，在 java 文件中为其赋值时需要逐一定义再绑定，使得代码十分繁琐。所以，经过查询资料，我们发现安卓中有一个方法 xxx 能循环取出一个 layout 中的所有子组件 id，于是我们的问题也就迎刃而解，可依靠此方法实现循环赋值。在此次问题中，其实功能早已经基本实现，但是我们为了保证代码的可读性与后期的可维护性，对代码中书写不合理的地方实现了进一步的改善，使得软件的运行效率更高，项目的完成度也更高。

虽然上面举出的实践过程中的问题看似对功能的实现影响不大，但实则这也是我们对工匠精神的一种坚守。十九大报告中提出“建设知识型、技能型、创新型劳动者大军，弘扬劳模精神和工匠精神，营造劳动光荣的社会风尚和精益求精的敬业风气”。报告中的“工匠精神”，它是一种职业精神，同时又是职业道德、职业能力、职业品质的体现，是从业者的一种职业价值取向和行为表现，主要包括爱岗敬业的职业精神、精益求精的品质精神、协作共进的团队精神、追求卓越的创新精神这四个方面的内容。其中，爱岗敬业的职业精神是根本，精益求精的品质精神是核心，协作共进的团队精神是要义，追求卓越的创新精神是灵魂。作为当代大学生，我们更应该严格要求自己，学习一门语言、一种开发技巧，不应该仅仅是“完成功能”就行，在更深层次上，我们还需要思考软件的运行效率或者实现方式上有没有可能再进一步优化。不断提高自身专业素养和道德素养，以更高的标准约束自己，激励自己，这才是我们对待所有知识的态度。

## 4.4 未来需完善的地方

本程序能实现当日 24 小时及未来几天的天气预报，有良好的稳定性及扩展性，但是有待完善的地方依然很多，未来改进时可以考虑一下几点：根据不同天气状态动态变化背景图片；可以直接获取用户当前位置；可以添加用户及其常用城市；将软件做成桌面小窗口方便查询。这些功能都需要在日后学习中不断探索研究，以建立实用的天气预报系统，给用户带来更好的体验与便捷的生活服务。