

**应用技术学院**

**移动平台开发技术与应用实训**

**题 目** SunnyWeather天气预报

**姓 名** 韩开元、成凯、时佳乐

**学 号** 201833100035

201833050025

201933050014

**院 系** 应用技术学院

**年 级** 2019级

**专业班级** 计科1班

**指导教师** 方忠进

**二Ｏ二二 年 12 月 10日**

SunnyWeather天气预报

韩开元、成凯、时佳乐

南京信息工程大学应用技术学院，江苏 南京

# 1实习目的

通过《SunnyWeather天气预报》课程设计实习过程，进一步巩固《移动平台开发》课程所学的理论知识，在有限的时间里，运用自己学到的知识，完成了App功能的设计与实现，积累了宝贵的经验，不仅加深了对所学知识的印象，同时在各科知识的融合上都更进一步。本次课程设计融MySql、Android、Kotlin等于一体，运用Android studio、eclipse、Navicat Premium、Gitee等软件，将各科知识的嵌套与融合，让App的功能更加强大。在实验的前期都是前端页面的设计，布局等，比较简单，后期的后端就向对比较麻烦，这也是本次课程设计的进度慢的原因之一。在本次课程设计中，认识到在遇到不会或者难以解决的问题时，通过团队之间讨论或者上论坛里查找资料是个很好的解决办法。

# 2实验内容：

2.1设计背景

实时的天气预报极大地方便人们的生活和工作，而智能电子产品的普及为实时天气预报系统提供了更加方便有效的平台。目前，Android 已经成为重要的手机软件系统, 基于Android 平台上开发的应用软件也越来越多。本款应用程序不仅仅是一款单纯的天气预 报软件，更是将短信与天气预报相结合，拉近了人与人之间的 亲情，使得人与人之间的感情更加亲密并且帮助那些使用非 智能机却想知道每天天气情况的人。具有操作简单，功能丰 富，界面友好等特点，为人们的出行和工作提供了极大的便利。

## 2.2功能需求分析

需求分析是软件开发前期进行的关键一步，调查用户的需求进而得出结论，确定要完成的内容，以及预期达到的目标，才能真正开发出让用户满意的产品。通过对用户需求的调查 研究，基于 Android 平台的天气预报系统，至少需要具备四个功能项：

可以搜索全球大多数国家的各个城市数据；

可以查看全球绝大多数城市的天气信息；

可以自由地切换城市，查看其他城市的天气；

可以手动刷新实时的天气。

天气界面需求分析。用户在天气界面中可以直接、快捷的查看当前所在城市的天气情况，定位当前用户所在的地理位置，并加入到城市数据库中；根据用户所在的城市，动态更新天气数据；用户可以添加其他的城市，查看其天气情况，也可以删除城市；用户可重新定位并可立即刷新当前城市天气。

生活界面需求分析。查看感冒指数并提醒注意事项；查看穿衣指数并且查看穿衣建议； 查看紫外线指数并且查看紫外线强度详情；查看洗车指数并且提供洗车建议；

短信发送界面需求分析。手动输入电话号码或者读取联 系人列表；当从联系人列表选择联系人后应自动确定此联系人的城市，并自动获取天气信息；可以添加日历，在某个确定的时间提醒；可以调用短信模板，对模板的增删改查。

想要全部实现这些功能，需要用到UI、网络、数据 存储、异步处理等技术，非常考验综合应用能力。

## 2.3行技术可行性分析

天气软件最重要的问题就是如何才能得到全球大多数国家的城市数据，以及如何才能获取每个城市的天气信息。遗憾的是，现在网上免费的天气预报接口已经越来越少，很多接口也慢慢关闭了。为了能够提供功能强大且长期稳定的服务器接口，我们最终选择了彩云天气。 彩云天气是一款非常出色的天气预报App，我们就是以彩云天气为范本开发的这款天气软件的。另外，彩云天气的开放API还提供了全球100 多个国家的城市数据，以及每个城市的实时天气预报信息，并且这些API接口是长期稳定且可用的，从而把前进的道路都铺平了。而且彩云天气的API每天最多提供1万次的免费请求，这是非常友好的。

# 3系统设计方案

## 3.1系统总体设计

此款基于 Android 平台的天气预报系统主要包括天气信息、生活指数信息、地图及地理位置信息。其中，天气信息又包括添加和删除城市、定位功能、随时刷新天气状态等；生活指数信息中又包括当前城市的四种生活指数及建议、短信功能、短信模板管理、定时任务、读取联系人信息等。功能模块如图 1 所示。

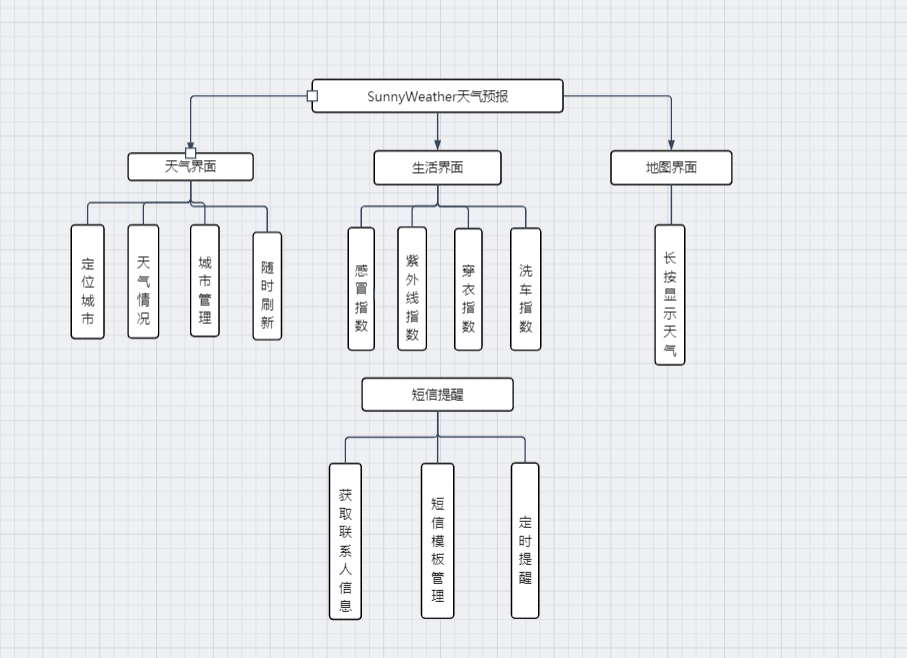


图1 功能模块

MVVM（Model- View-ViewModel ）是一种高级项目架构模式，目前已被广泛应用在Android 程序设计领域，类似的架构模式还有MVP、MVC等。简单来讲，MVVM架构可以将程序结构主要分成3部分：Model 是数据模型部分；View 是界面展示部分；而ViewModel比较特殊，可以将它理解成一个连接数据模型和界面展示的桥梁，从而实现让业务逻辑和界面展示分离的程序结构设计。当然，一个优秀的项目架构除了会包含以上3部分内容之外，还应该包含仓库、数据源等，此项目便遵循了这样的体系。

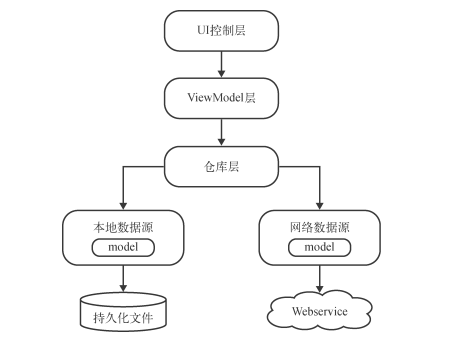


图2 MVVM架构

## 3.2系统功能设计

### 3.2.1 天气模块功能设计

定位模块

软件在第一次启动后，将自动定位当前用户所在的城市信息，可定位到县级及县级以上城市并自动加入城市数据库。并且，当用户添加多个城市时，使用定位可以跳转到用户所在城市。使用定位模块可以减少用户的一部分时间。天气信息处理模块。由定位模块获取到用户所在城市之后，通过软件所自带的城市与代码对照表获取该城市所对应的 城市代码，通过异步网络请求，调用中国气象局的API接口获取 天气数据信息并显示在手机屏幕上。该模块增加缓存机制，用户在未连接网络时，显示之前已缓存的天气并显示之前的时间。用户可以查看未来三天的天气情况。若有多个城市，用户可以 左右滑动来切换城市并查看信息。用户可以随时更新天气信息。

城市管理模块

在此模块中，用户可以管理城市，包括添加和删除城市，可以添加任意多个城市，点击城市后，展示该城市的天气信息。

### 3.2.2 生活模块功能设计

生活指数模块。生活指数提供人们日常所需的一些建议。 晨练指数提供晨练的适宜情况及一些晨练的建议；舒适度指数提供舒适度的适宜情况以及详细的信心；穿衣指数提供穿衣的适宜情况以及各个年龄段的穿衣建议。短信提醒模块。此模块中，用户可以手动输入联系人的 电话号码或者从手机联系人中选择以个联系人，当用户是选择联系人操作时，可以自动查询此联系人所在的城市并查询 天气信息；用户可以选择短信的模板。对于短信模板管理，用 户可以增减模板、删除模板，也可以自定义模板内容；用户可 以设置发送短信提醒的时间，设置日历提醒的标题、内容、是 否为全天任务、以及提醒次数等。

### 3.2.3 数据库设计

数据库设计是指对于一个给定的应用环境，构造最优的 数据库模式，建立数据库及其应用系统，使之能够有效地存储 数据，满足各种用户的应用需求。根据软件的使用方法和数 据，分析数据库的设计[4]。本软件使用 SQLite 数据库作为软件 的数据库，所涉及的表及文件如下： 天气城市表：用于记录用户所添加城市的基本信息，包括 城市 ID，城市的名称，是否为默认城市。城市ID为该表的主键，天气城市表（weather）如表 1 所示。



表1 天气城市表

短信模板表：用于记录短信发送操作的短信内容所应用 于的对象。包括模板 ID，模板的标题，模板的内容以及是否为 当前所使用的模板，短信模板表（muban）如表 2 所示。

****

**表2 短信模板信息表**

城市代码表：此表用于将定位或者手动添加的城市转化为对应的城市代码，以便请求天气数据。此表是一个本地表， 包括ID、城市代码、城市天气代码、省份、市级城市、县级城市， 城市代码表（city\_info）如表 3 所示。



表3 城市代码表

# 4系统实现

## 4.1搜索全球城市数据

根据之前的技术可行性分析，要想实现查看天气信息的功能，首先要能搜索到具体的城市数 据，并获取该地区的经纬度坐标。因此，我们第一阶段的开发任务就是先来实现搜索全球城市的数据信息。根据合理的开发方式，实现过程应该主要分为ui层实现和逻辑层实现两部分。

UI层代码：

<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="?android:windowBackground">  
  
 <ImageView  
 android:id="@+id/bgImageView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentBottom="true"  
 android:src="@drawable/bg\_place"/>  
  
 <FrameLayout  
 android:id="@+id/actionBarLayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="60dp"  
 android:background="@color/colorPrimary">  
  
 <EditText  
 android:id="@+id/searchPlaceEdit"  
 android:layout\_width="match\_parent"  
 android:layout\_height="40dp"  
 android:layout\_gravity="center\_vertical"  
 android:layout\_marginStart="10dp"  
 android:layout\_marginEnd="10dp"  
 android:paddingStart="10dp"  
 android:paddingEnd="10dp"  
 android:hint="输入地址"  
 android:background="@drawable/search\_bg"/>  
 </FrameLayout>  
  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/recyclerView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_below="@id/actionBarLayout"  
 android:visibility="gone"/>  
</RelativeLayout>

逻辑层代码:

class PlaceFragment : Fragment() {  
  
 val viewModel by lazy { ViewModelProviders.of(this).get(PlaceViewModel::class.java) }  
  
 private lateinit var adapter: PlaceAdapter  
  
 override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View? {  
 return inflater.inflate(R.layout.fragment\_place, container, false)  
 }  
  
 override fun onActivityCreated(savedInstanceState: Bundle?) {  
 super.onActivityCreated(savedInstanceState)  
 if (activity is MainActivity && viewModel.isPlaceSaved()) {  
 val place = viewModel.getSavedPlace()  
 val intent = Intent(context, WeatherActivity::class.java).apply {  
 putExtra("location\_lng", place.location.lng)  
 putExtra("location\_lat", place.location.lat)  
 putExtra("place\_name", place.name)  
 }  
 startActivity(intent)  
 activity?.finish()  
 return  
 }  
 val layoutManager = LinearLayoutManager(activity)  
 recyclerView.layoutManager = layoutManager  
 adapter = PlaceAdapter(this, viewModel.placeList)  
 recyclerView.adapter = adapter  
 searchPlaceEdit.addTextChangedListener { editable ->  
 val content = editable.toString()  
 if (content.isNotEmpty()) {  
 viewModel.searchPlaces(content)  
 } else {  
 recyclerView.visibility = View.GONE  
 bgImageView.visibility = View.VISIBLE  
 viewModel.placeList.clear()  
 adapter.notifyDataSetChanged()  
 }  
 }  
 viewModel.placeLiveData.observe(this, Observer{ result ->  
 val places = result.getOrNull()  
 if (places != null) {  
 recyclerView.visibility = View.VISIBLE  
 bgImageView.visibility = View.GONE  
 viewModel.placeList.clear()  
 viewModel.placeList.addAll(places)  
 adapter.notifyDataSetChanged()  
 } else {  
 Toast.makeText(activity, "未能查询到任何地点", Toast.LENGTH\_SHORT).show()  
 result.exceptionOrNull()?.printStackTrace()  
 }  
 })  
 }  
}

初始界面：



## 4.2 显示天气信息

在第二阶段中，我们就要开始去查询天气，并且把天气信息显示出来了。实现的过程也是类似的，同样主要分为逻辑层实现和UI层实现两部分。

UI层代码：

<androidx.drawerlayout.widget.DrawerLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:id="@+id/drawerLayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <androidx.swiperefreshlayout.widget.SwipeRefreshLayout  
 android:id="@+id/swipeRefresh"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <ScrollView  
 android:id="@+id/weatherLayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:overScrollMode="never"  
 android:scrollbars="none"  
 android:visibility="invisible">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical">  
  
 <include layout="@layout/now" />  
  
 <include layout="@layout/forecast" />  
  
 <include layout="@layout/life\_index" />  
  
 </LinearLayout>  
  
 </ScrollView>  
  
 </androidx.swiperefreshlayout.widget.SwipeRefreshLayout>  
  
 <FrameLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_gravity="start"  
 android:clickable="true"  
 android:focusable="true"  
 android:background="@color/colorPrimary">  
  
 <fragment  
 android:id="@+id/placeFragment"  
 android:name="com.sunnyweather.android.ui.place.PlaceFragment"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_marginTop="25dp"/>  
  
 </FrameLayout>  
  
</androidx.drawerlayout.widget.DrawerLayout>

逻辑层代码：

class WeatherActivity : AppCompatActivity() {  
  
 val viewModel by lazy { ViewModelProviders.of(this).get(WeatherViewModel::class.java) }  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 if (Build.VERSION.SDK\_INT >= 21) {  
 val decorView = window.decorView  
 decorView.systemUiVisibility = View.SYSTEM\_UI\_FLAG\_LAYOUT\_STABLE or View.SYSTEM\_UI\_FLAG\_LAYOUT\_FULLSCREEN  
 window.statusBarColor = Color.TRANSPARENT  
 }  
 setContentView(R.layout.activity\_weather)  
 if (viewModel.locationLng.isEmpty()) {  
 viewModel.locationLng = intent.getStringExtra("location\_lng") ?: ""  
 }  
 if (viewModel.locationLat.isEmpty()) {  
 viewModel.locationLat = intent.getStringExtra("location\_lat") ?: ""  
 }  
 if (viewModel.placeName.isEmpty()) {  
 viewModel.placeName = intent.getStringExtra("place\_name") ?: ""  
 }  
 viewModel.weatherLiveData.observe(this, Observer { result ->  
 val weather = result.getOrNull()  
 if (weather != null) {  
 showWeatherInfo(weather)  
 } else {  
 Toast.makeText(this, "无法成功获取天气信息", Toast.LENGTH\_SHORT).show()  
 result.exceptionOrNull()?.printStackTrace()  
 }  
 swipeRefresh.isRefreshing = false  
 })  
 swipeRefresh.setColorSchemeResources(R.color.colorPrimary)  
 refreshWeather()  
 swipeRefresh.setOnRefreshListener {  
 refreshWeather()  
 }  
 navBtn.setOnClickListener {  
 drawerLayout.openDrawer(GravityCompat.START)  
 }  
 drawerLayout.addDrawerListener(object : DrawerLayout.DrawerListener {  
 override fun onDrawerStateChanged(newState: Int) {}  
  
 override fun onDrawerSlide(drawerView: View, slideOffset: Float) {}  
  
 override fun onDrawerOpened(drawerView: View) {}  
  
 override fun onDrawerClosed(drawerView: View) {  
 val manager = getSystemService(Context.INPUT\_METHOD\_SERVICE) as InputMethodManager  
 manager.hideSoftInputFromWindow(drawerView.windowToken, InputMethodManager.HIDE\_NOT\_ALWAYS)  
 }  
 })  
 }  
  
 fun refreshWeather() {  
 viewModel.refreshWeather(viewModel.locationLng, viewModel.locationLat)  
 swipeRefresh.isRefreshing = true  
 }  
  
 private fun showWeatherInfo(weather: Weather) {  
 placeName.text = viewModel.placeName  
 val realtime = weather.realtime  
 val daily = weather.daily  
 // 填充now.xml布局中数据  
 val currentTempText = "${realtime.temperature.toInt()} ℃"  
 currentTemp.text = currentTempText  
 currentSky.text = getSky(realtime.skycon).info  
 val currentPM25Text = "空气指数 ${realtime.airQuality.aqi.chn.toInt()}"  
 currentAQI.text = currentPM25Text  
 nowLayout.setBackgroundResource(getSky(realtime.skycon).bg)  
 // 填充forecast.xml布局中的数据  
 forecastLayout.removeAllViews()  
 val days = daily.skycon.size  
 for (i in 0 until days) {  
 val skycon = daily.skycon[i]  
 val temperature = daily.temperature[i]  
 val view = LayoutInflater.from(this).inflate(R.layout.forecast\_item, forecastLayout, false)  
 val dateInfo = view.findViewById(R.id.dateInfo) as TextView  
 val skyIcon = view.findViewById(R.id.skyIcon) as ImageView  
 val skyInfo = view.findViewById(R.id.skyInfo) as TextView  
 val temperatureInfo = view.findViewById(R.id.temperatureInfo) as TextView  
 val simpleDateFormat = SimpleDateFormat("yyyy-MM-dd", Locale.getDefault())  
 dateInfo.text = simpleDateFormat.format(skycon.date)  
 val sky = getSky(skycon.value)  
 skyIcon.setImageResource(sky.icon)  
 skyInfo.text = sky.info  
 val tempText = "${temperature.min.toInt()} ~ ${temperature.max.toInt()} ℃"  
 temperatureInfo.text = tempText  
 forecastLayout.addView(view)  
 }  
 // 填充life\_index.xml布局中的数据  
 val lifeIndex = daily.lifeIndex  
 coldRiskText.text = lifeIndex.coldRisk[0].desc  
 dressingText.text = lifeIndex.dressing[0].desc  
 ultravioletText.text = lifeIndex.ultraviolet[0].desc  
 carWashingText.text = lifeIndex.carWashing[0].desc  
 weatherLayout.visibility = View.VISIBLE  
 }  
}

初始化界面：



然后我们还可以向下滑动查看更多天气信息



由于界面上显示的天气信息有可能会过期，因此用户需要一种方式来手动刷新天气。那么具体应该如何触发刷新事件呢？这里我准备采用下拉刷新的方式，使用下拉刷新控件的用法，实现起来会比较简单。



完成了手动刷新天气的功能，接下来继续实现切换城市功能。 既然是要切换城市，那么就肯定需要搜索全球城市的数据，而这个功能已经完成了，并且为了方便后面的复用，当时特意选择了在Fragment 中实现。因此，我们其实只需 要在天气界面的布局中引入这个Fragment ，就可以快速集成切换城市功能了。 虽说实现原理很简单，但是显然我们也不可能让引入的Fragment 把天气界面遮挡住，将Fragment 放入滑动菜单中就可以实现这个需求，正常情况下它不占据主界面的任何空间，想要切换城市的时候，只需要通过滑动的方式将菜单显示出来就可以了。



## 4.3 短信管理

此部分功能为获取用户指定联系人的城市天气信息，并 可以为不同的联系人指定不同的天气信息，然后以短信的方 式发送给指定联系人。此部分的设计为一个选择联系人按钮、 一个搜索按钮、模板选择以及设置提醒方式。当点击选择联 系人按钮后，软件将跳转到手机的联系人界面，当用户选择了 一个联系人后，软件将回到此界面并将联系人的手机号码显 示在选择联系人按钮左边的编辑框中，并且，软件会自动根据 联系人判断归属地。之后，软件根据获取的归属地自动查询 天气信息。若此联系人绑定了短信模板的内容，软件将自动 更换天气模板信息。当点击在指定时间联系我时，软件将跳 转到日历添加事件界面，设置好后软件将在指定时间提醒用 户发送短信。 短信模板管理的主要功能为设置常用的联系人短信内容， 比如家人模板、朋友模板、同事模板等等。使用模板功能可快 速定义短信内容。此部分的设计主要是使用 ListView 显示数 据并使用 BaseAdapter 为 ListView 绑定数据，BaseAdapter中的数据是由 muban数据库提供的。在此界面中，用户可以通 过界面上的编辑和删除按钮来对模板进行操作。本质上是对muban数据库的修改增加和删除。

## 4.4 地图模块

此模块实现了地图任意地点查询天气的功能。如需要查看其他地点的城市，但是却不想加入到常看的城市中，那 么此功能将显示它的价值。当用户长按地图上的某个点时， 此界面将显示此地点的天气信息。此模块实现的原理是：首 先，嵌入百度地图和定位数据并显示到用户界面；然后添加地 图长按事件，在此事件中，软件将用户按下位置的经纬度经过 地理位置反编译，得到该点的城市区域信息；最后根据该点的 城市，通过申请中国气象局的 API 接口，返回后通过一个显示 框显示给用户。地图模块如图所示。



# 5总结

这学期的安卓开发使用了新的语言Kotlin，相对Java，kotlin的资料是非常少的，案例也是很少，在这次课程设计中我也是花了很多时间找相关资料，《第一行代码》这本书前前后后也翻了好几遍，也渐渐明白了Kotlin语法糖的一些特性，编写代码的习惯也有了较好的规范。在后端方面，因为对SpringBoot有一定的实战经验，所以此次我包括了后端开发，后端功能我写的并不多，但现有的功能的逻辑性细节还是挺强挺多的，比如用户这一方面，除了个人信息之外，还添加了密码加盐加密，携带令牌Token等，后端增加了短信验证码功能。我比较喜欢探究新的东西，这次界面设计上，还运用了完全的卡片式布局，消息推送，启动页广告，夜间模式等功能，这些功能实现起来并不容易，但是也挺有趣的。

这次课程设计总体来说也是摸爬滚打中学习，挺不容易的，同时和队友的沟通帮助也增加了我的团队协作能力，也在收尾的时候，感谢坚持的自己，感谢队友的积极合作，同时也感谢老师平时实践性的课程，这样知识点了解得更加清晰明白。

# 6参考文献

[1] 李泽椿,毕宝贵,金荣花,等.近 10 年中国现代天气预报的 发展与应用[J].气象学报, 2014, 72(6):1069-1078.

[2] 罗布·佩格拉诺, 常玉田.安卓系统手机六大变化[J].中国 翻译, 2017(2):115-116.

[3] (美)]Karl Wiegers,Joy Beatty 著, 李忠利,李淳,霍金健,孔晨 辉译.软件需求（第三版）[M].北京:清华大学出版社, 2016.

[4] (英)戴特著,卢涛译. 数据库设计与关系理论.[M].北京: 机 械工业出版社,2013.

[5] 范美英.Android 用户体验与 UI 设计 [M].北京:知识产权 出版社,2015