**分类号： U D C：**

**密 级：公开 编 号：2014121096**

**成都信息工程大学**

**学位论文**

**天气应用App的设计与实现**

|  |  |
| --- | --- |
| **论文作者姓名:** | **谭鹏程** |
| **申请学位专业:** | **网络工程** |
| **申请学位类别:** | **工学学士** |
| **指导教师姓名（职称）:** |  |
| **论文提交日期:** |  |

**天气应用App的设计与实现**

**摘 要**

目前市场上天气相关的Android App很多，但是大多数都充斥着许多广告推送和一些不必要的后台活动，严重造成Android系统卡顿和浪费用户的流量费用，更有流氓App靠此收集用户信息。

基于目前国内Android天气类App存在的问题和用户对于实时天气信息的需求。本此课题设计一款轻后台的Android天气应用，用户可以根据自己对于天气的不同需求选择不同的后台更新频率或者选择关闭后台。同时用户可以在设置中开启通知栏和在桌面添加小部件，在不点击进入天气App的情况下获取最新的天气信息。还可以在常用城市中选择加载当前位置的天气，通过GPS获取位置信息后请求天气信息。更加适合在多个城市奔波的用户，提高用户的使用体验。本系统使用了当前Android开发流行的开源库和设计模式，包括依赖注入库Dagger2、网络请求库Volley、数据库ORM库ORMLite、图片加载库Glide、MVP设计模式。系统的可扩展性强，同时这些开源库有大量的开发者使用，稳定性可以得到保障，后期的维护成本低。

**关键词：**Android；天气App；Dagger2；Volley；数据库；ORMLite ；Glide；MVP设计模式

**Design and Implementation of Weather App**

**Abstract**

The weather apps are very common in app store and most of them are bound by adware and unnecessary background service. It coasts more extra hardware and system resource, and causes the problem with phone stuttering. The users need assume more flow coast. Some of them collect users’ infomation by this way.

Because of common weather apps’ problem and the user’s needs to weather infomation, This app is designed with a lightweight background and give the switch of service to users. The user can set the rate of service update or close service by users’ needs. If users need, can set widgets on Android’s desktop or notifications. The users can get weather information without opening apps. This weather app also get location info by GPS, and can dynamiclly load the latest weather information with changing location. With many Android open source libraries and design modes which are popular with Android developers in this system, For example, Dagger2,Volley,ORMLite,Glide,MVP design mode and etc. The whole system proves to be easy extensibility and low maintenance cost.

**Key words:** Android; weather app; Dagger2; Volley; ORMLite; Glide; MVP design mode

**目 录**

论文总页数：××页

[**1引言** 1](#_Toc514358611)

[**1.1 背景** 1](#_Toc514358612)

[**1.2 国内外研究现状** 1](#_Toc514358613)

[**1.3 本课题研究意义** 1](#_Toc514358614)

[**1.4 本课题研究目的** 1](#_Toc514358615)

[**2技术介绍** 1](#_Toc514358616)

[**2.1 开发环境** 1](#_Toc514358617)

[**2.1.1 Android Studio** 1](#_Toc514358618)

[**2.1.2 GitHub** 2](#_Toc514358619)

[**2.1.3 IDEA** 2](#_Toc514358620)

[**2.1.4 MySQL** 2](#_Toc514358621)

[**2.2 Android MVP 设计模式** 2](#_Toc514358622)

[**2.3 依赖注入库--Dagger2** 3](#_Toc514358623)

[**2.4 Android ORM框架—ORMLite** 3](#_Toc514358624)

[**2.5 Material Design** 3](#_Toc514358625)

[**3需求分析** 3](#_Toc514358626)

[**3.1 用户需求** 3](#_Toc514358627)

[**3.2 功能需求** 4](#_Toc514358628)

[**3.3可行性分析** 4](#_Toc514358629)

[**3.3.1技术可行性分析** 4](#_Toc514358630)

[**3.3.2社会可行性分析** 4](#_Toc514358631)

[**3.3.3经济可行性分析** 4](#_Toc514358632)

[**3.3.4法律可行性分析** 4](#_Toc514358633)

[**4概要设计** 4](#_Toc514358634)

[**4.1 主界面初始化流程** 4](#_Toc514358635)

[**4.2 天气消息通知流程** 5](#_Toc514358636)

[**4.3 登陆流程** 7](#_Toc514358637)

[**4.4 数据结构设计** 8](#_Toc514358638)

[**4.4.1 客户端城市信息数据结构** 8](#_Toc514358639)

[**4.4.2 客户端SharedPreferences数据结构** 8](#_Toc514358640)

[**4.4.3 服务端数据结构** 9](#_Toc514358641)

[**4.5 系统功能模块** 10](#_Toc514358642)

[**5详细设计与实现** 10](#_Toc514358643)

[**5.1 系统结构** 10](#_Toc514358644)

[**5.2 进入主界面数据初始化** 12](#_Toc514358645)

[**5.3 后台天气信息自动更新** 14](#_Toc514358646)

[**5.4 桌面小部件和通知栏** 15](#_Toc514358647)

[**5.5 全国城市信息获取和添加常用城市** 16](#_Toc514358648)

[**5.6 设置信息的储存** 16](#_Toc514358649)

[**5.7 用户信息保存到服务端** 16](#_Toc514358650)

[**6测试** 16](#_Toc514358651)

[**6.1 初次数据初始化测试** 16](#_Toc514358652)

[**6.2 添加常用城市天气信息测试** 16](#_Toc514358653)

[**6.3 获取当前位置天气信息测试** 16](#_Toc514358654)

[**6.4 小部件和通知栏信息同步测试** 16](#_Toc514358655)

[**6.5 和服务端数据同步测试** 16](#_Toc514358656)

[**结 语** 17](#_Toc514358657)

[**参考文献** 18](#_Toc514358658)

[**致 谢** 19](#_Toc514358659)

[**声 明** 20](#_Toc514358660)

**1引言**

**1.1 背景**

2008年Google在I/O大会上发布了Android 1.0，由于其开源的特性，受到了很多开发者的青睐。随后几年越发壮大，干倒了当时的霸主——Symbian，成为全球占有率最高的移动操作系统。Android的市场占有率提高后，Android的App市场也是百花齐放，出现了很多实用的App。

**1.2 国内外研究现状**

2010年Google决定退出中国市场，随后中国的Android应用市场也失去了一个权威的应用市场管理机构。中国用户只有选择在类似应用宝一样平台下载应用，相对于Google管理的应用市场，国内的应用市场监管不够严格。许多的流氓App也充斥在国内的应用市场中。大多数的天气类App功能已经能够满足大多数用户的日常使用，但是嵌套了许多用户不适用的功能和广告插件，浪费系统资源浪费和加剧系统碎片化，造成系统卡顿。

**1.3 本课题研究意义**

天气的变化容易引起人感冒，耽误工作和学习。及时的天气信息可以提醒用户增添或者减少衣物，降低天气变化对人身体的影响。同时天气变化也会影响人们日常生活的出行，提前预知突发天气，做好准备。

当前天气的信息还是非常的准确，但是准确的天气信息不总是能够第一时间传送到用户。近些年移动互联网蓬勃发展，智能手机普及率也是很高。同时手机的消息通知很及时。所以本次系统选择了使用全球最大的移动操作系统Android作为开发环境。

**1.4 本课题研究目的**

结合目前国内Android天气类App的现状和用户对于及时天气信息的需求，本次课题主要研究一种轻量级的Android天气App，同时满足各种各类人群对于及时天气信息的需求。本系统把天气Service的开关交给用户，用户可以根据自己的需求开启或者关闭自动更新后台，设置后台更新的频率，同时桌面还可以添加通知栏和桌面小部件，同步显示天气信息。本系统还可以通过GPS加载当前位置信息，从而通过变化的位置动态加载天气信息。不用用户手动选择位置，提高用户的体验。

**2技术介绍**

**2.1 开发环境**

**2.1.1 Android Studio**

Android Studio作为ADT的替代品，在各种方面都优于ADT。例如：智能的代码提示，强大的代码查找功能，方便的依赖管理等。IDEA为Android 提供技术支持。在IDEA的基础上Android还提供以下主要功能：1.用Gradle作为编译工具和依赖添加工具。2.基于Android开发特有的代码提示与修复。 3.强大的UI编辑和预览。

**2.1.2 GitHub**

GitHub是一个免费提供git代码仓库托管和提供web页面管理的版本控制平台。其中也包括很多的知名开源项目，例如：jQuery、python、OkHttp等。每天全球有超过900万用户通过GitHub学习和交流。

**2.1.3 IDEA**

IDEA是JetBrains公司开发的一款java语言开发的集成环境，IDEA以智能著称，尤其在智能的代码提示、重构、j2ee支持、Ant、Junit、代码审查、友好UI等方面超越许多市面上的同类的产品。同时IDEA还支持很多的插件，适合各类的java程序开发者。

**2.1.4 MySQL**

MySQL是一种免费开源的关系型数据库的管理系统。相对于Sql Server的大型数据库系统，MySql还是有很多不足的地方。由于其体积小、速度快、免费等特点，MySQL在中小型企业中很受欢迎。

**2.2 Android MVP 设计模式**

MVP模式简单结构图如图2-1。



图2-1 MVP模式简单结构图

在传统的Android开发设计中，运用的是MVC设计模式，在Activity中要扮演Controller和View两种角色，Activity需要直接依赖Model和刷新View。当项目的业务复杂后，Activity代码会成倍的增多，降低代码的可读性和后期的维护成本。

MVP模式的出现从一定程度上解决了这种问题。在MVP模式中，View为Activity或者Fragment扮演，只是刷新界面不做业务逻辑，不再依赖于Model层，而是依赖于Presenter；Presenter直接依赖于Model，参与主要业务逻辑判断，控制View刷新界面；Model主要是通过Database或者网络请求获取需要的数据。

**2.3 依赖注入库--Dagger2**

当一个类中需要某一个依赖的时候，不是通过自己创建，而是通过第三方创建，注入到该类中，这就是依赖注入。最简单的注入方式就是通过setter方法替换掉原来的成员变量。通过依赖注入的方式添加依赖有以下几点好处：1.复杂的依赖关系变得简单。2.易于单元测试。3.根据需求控制依赖创建和回收。著名的依赖注入库，例如JavaEE的Spring框架。

Dagger2是Google官方推荐的依赖注入库，Dagger2主要有Module、Component通过这两大模块完成依赖注入。Module：是依赖创建的地方，同时控制依赖的回收和是否重新创建。Component：将Module创建的实例注入到需要依赖的地方，Component就像是注入器，Module 就像是注入器中的药剂。

**2.4 Android ORM框架—ORMLite**

Android中内置的数据库SQLite和市面上大多数的数据库一样是关系型数据库。但是现在大多数的软件都是基于面向对象编程。面向对象开发方法，有以下特点：1.易维护，2.质量高，3.效率高，4.易扩展。ORMLite将关系型的数据库表转换成一个Java类，通过对Java类的操作映射到数据库，完成对数据库的增、删、查、改。

**2.5 Material Design**

Google在2014年I/O大会上着重展示了Material Design，为用户创造了一种全新的设计语言。同时，遵循经典设计定则，还汲取了最新的科技，秉承创新的设计理念。Material Design提倡扁平的设计风格，更加简洁，突出重要信息，同时有很多眼前一亮的动画和阴影。现在在Android应用和web应用中使用很广泛。

**3需求分析**

**3.1 用户需求**

经过调研，用户对于及时的实时天气信息的信息需求很大，同时没有广告插件，可以任意的关闭和开启后台，能够在不进入App的情况下也能获取部分重要天气信息，能够获当前时刻的天气和未来天气的预报，可以很方便的添加城市信息和获取当前位置信息，App的UI友好，最好还有一些人性化的扩展功能。

**3.2 功能需求**

根据用户需求将本系统分为以下5大模块：

1.展示UI：主要界面展示天气，小部件展示天气，通知栏展示天气，获取桌面美图。

2.天气信息服务：获取当前时刻天气信息，获取未来十天的天气预报。

3位置服务：加载全国天气信息，获取当前位置信息。

4服务端同步：用户注册，用户登录，获取用户储存到服务端的信息，上传用户信息到服务端。

5.设置：后台服务开关，美图加载开关，天气信息更新频率，通知栏开关，背景亮度。

**3.3可行性分析**

**3.3.1技术可行性分析**

Android 1.0版本从发布至今已经有10年的时间了，10年的时间里Android不断壮大，出现了很多优秀的开源框架。Android 自己的优化更新也没有停止，目前最新的版本已经到了9.0。本系统在客户端网络请求库使用Volley，依赖注入库使用Dagger2，ORM框架使用ORMLite。服务端使用Spring MVC + mybatis框架，服务器使用tomcat web应用服务器。客户端和服务端都是使用面向对象的开发语言JAVA，JAVA具有良好的可读性，较好的安全性和平台无关性。

**3.3.2社会可行性分析**

在互联网+和移动开发技术成熟的环境下，更多的用户愿意使用手机获取日常重要的信息。特别是类似于天气信息一样，需要及时性和准确性。相对于电视和电脑这样的信息获取设备，手机具有便于携带性和用户使用频率高的特点。

**3.3.3经济可行性分析**

本系统所使用的技术都是属于开源免费的，在经济投入方面几乎为零。人力投入方面，主要是代码实现的人力投入。本系统的主要目的在于方便大众对于天气信息的获取，未考虑收益方面。

**3.3.4法律可行性分析**

本系统所使用的技术都是开源免费的，在网络上供大家参考和学习。系统也是在指导老师的指导下完成，不会存在有任何的侵权行为。

**4概要设计**

**4.1 主界面初始化流程**

主界面初始化流程图如图4-1。



图4-1 主界面初始化流程图

初始化简单描述：

进入系统查询是否添加常用城市，添加过常用城市直接获取天气，如果没有添加常用城市（初次进入没有常用城市），先通过GPS获取到位置的精度和纬度，然后请求百度的API获取当前城市，然后储存到常用位置中。有常用位置后，请求雅虎天气的API获取天气信息，一共有4个天气相关的接口，大气相关，当前温度，未来10天预报，风相关。

**4.2 天气消息通知流程**

天气消息通知流程图如图4-2。



图4-2 消息通知流程图

通知流程简单描述：

该功能的前提需要在设置中开启后台。检测后台是否开启，如果没有开启先开启后台服务，进入后台服务，读取设置中的更新频率，根据更新频率每隔一段时间执行更新天气信息任务。天气信息请求成功或者失败，都会重新获取更新频率进入重复天气请求任务，请求成功后，就会刷新主界面，通知栏，小部件等三个主要显示UI界面。

**4.3 登陆流程**

登录流程图如图4-3。



图4-3 登录流程图

登录流程简单描述：

用户输入登录信息，检测用户输入是否合法，不合法则重新输入，合法则将登录信息封装上传到服务器，根据服务端返回的信息判断是否登录成功，如果失败则提示用户重新登录。成功后读取登录用户设置信息，重新写入到本地。

**4.4 数据结构设计**

**4.4.1 客户端城市信息数据结构**

全国城市信息表，用于储存从中国天气网上获取到的全国城市信息，实体图如图4-4，表结构如表4-1。



图4-4 城市信息结构图

表4-1 city

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 描述 |
| code | char | 11 | 主键，确定城市的唯一代码。 |
| parent\_code | char | 11 | 对应父级城市代码 |
| level | integer | 11 | 城市对应等级 |
| city\_name | varchar | 11 | 城市名 |

**4.4.2 客户端SharedPreferences数据结构**

SharedPreferences主要用户储存用户的常用地址、当前使用城市和一些设置信息，数据储存结构如表4-2。

表4-2 SharedPreferences数据结构

|  |  |  |
| --- | --- | --- |
| 键名 | 数据类型 | 描述 |
| current\_city | String | 当前使用城市 |
| common\_use\_cities | String | 常用城市 |
| wallpaper\_alpha | int | 背景图片亮度（0-100） |
| update\_rate | int | 更新频率（单位：小时） |
| load\_wallpaper | boolean | 是否加载美图 |
| open\_service | boolean | 是否开启后台 |
| open\_notification | boolean | 是否开启通知栏 |

**4.4.3 服务端数据结构**

服务端实体关系图如图4-5，用户表如表4-3，设置表如表4-4，城市表如表4-5。



图4-5 服务端实体关系图

表4-3 user1

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 描述 |
| id | int | 11 | 主键，用户id |
| name1 | text |  | 用户名 |
| password1 | text |  | 密码 |
| sex | char | 5 | 性别 |

表4-4 city

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 描述 |
| id | int | 11 | 主键 |
| user\_id | int | 11 | 外键，用户ID |
| current\_city | text |  | 当前使用城市 |
| common\_cities | text |  | 常用城市 |

表4-5 setting

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 描述 |
| id | int | 11 | 主键 |
| user\_id | int | 11 | 外键，用户ID |
| load\_service | tinyint | 1 | 是否加载后台 |
| load\_notification | tinyint | 1 | 是否加载通知栏 |
| load\_image | tinyint | 1 | 是否加载美图 |
| update\_rate | int | 11 | 更新频率 |
| image\_alpha | int | 11 | 桌面背景亮度 |

**4.5 系统功能模块**

系统功能模块图如图4-6。



图4-6 系统模块图

**5详细设计与实现**

**5.1 系统结构**

系统结构图如图5-1。



图5-1 系统结构图

系统结构描述：

Activity实现了IView接口，Presenter都实现了IPresenter接口，Activity和Presenter是关联关系，Presenter依赖于Model，Model中主要包括WeatherInfoManager、NetRequestUtils、SpManager、ChinaCitiesManager、SettingsManager、LoginManager等几个部分。Model依赖Android数据库和网络服务。Background Service包括Widget、Notification和UpdateService，Background Service直接依赖于Model。整个系统都依赖于像 Volley和Dagger2一样的基础组件。

MainActivity：主要显示当前天气信息和预报信息，在侧滑栏中有用户的城市信息和设置一些选项。业务逻辑都是Presenter完成，后面提到的Activity也是一样，就不再赘述。

AddCityActivity：从数据库中获取全国天气信息，根据用户的选择添加常用城市到本地。

SettingsActivity：主要提供修改设置信息入口，修改保存后先保存到本地，如果用户登录过，还要上传设置信息到服务器后台。

Background Service：定时刷新天气信息，同时通知系统三个主要天气UI展示刷新。

NetRequestUtils：负责Http请求相关功能。

WeatherInfoManager：通过NetRequestUtils请求最新的天气，然后储存到内存中，成功后，通知设置了天气监听的模块。

ChinaCitiesManager：初始化的时候通过网络请求全国天气信息，然后存储到本地数据库，获取当前位置信息，读取和储存用户常用位置。

SpManager：通过SharedPreferences储存和读取数据。

SettingsManager：管理设置信息，下发设置改变的通知。

LoginManager：管理用户登录和注册相关，将本地数据上传到服务端，同步服务端数据到本地。

在WeatherInfoManager、ChinaCitesManager、SpManager、SettingsManager中都用了观察者模式，依赖的地方可以添加监听，当其中的数据发生改变时这些Manager就会回调这些监听中的方法，从而在依赖的地方做一些处理。

**5.2 进入主界面数据初始化**

用户进入应用后，通过SettingsManager检测设置中是否加载后台和加载背景美图，检测是否有当前位置信息，如果没有通过ChinaCitiesManager获取到精度和纬度，请求百度API获取城市，成功后储存到本地。然后通过WeatherInfoManager请求网络获取天气信息。加载中和加载失败如图5-2，加载成功如图5-3，侧滑菜单如图5-4。

天气获取成功后刷新界面代码实现：

mView.setWeatherViewEnable(true);

mView.refreshWeatherInfo(createWeatherVo());

if(mView.isRefreshing()) {

mView.closeRefreshing();

}

天气获取失败后界面刷新代码：

mView.setWeatherViewEnable(false);

ToastUtils.getInstance().showShortText(e.getMessage());

if(mView.isRefreshing()) {

mView.closeRefreshing();

}

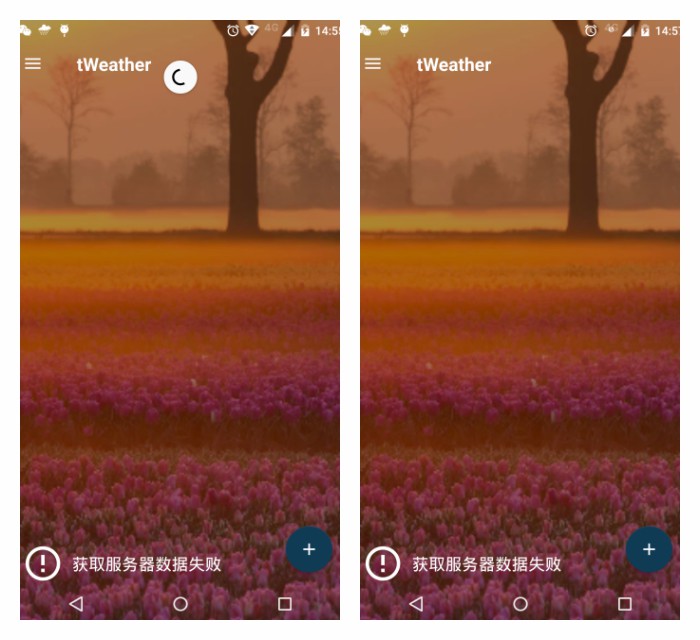


图5-2 加载中和加载失败



图5-3 成功加载

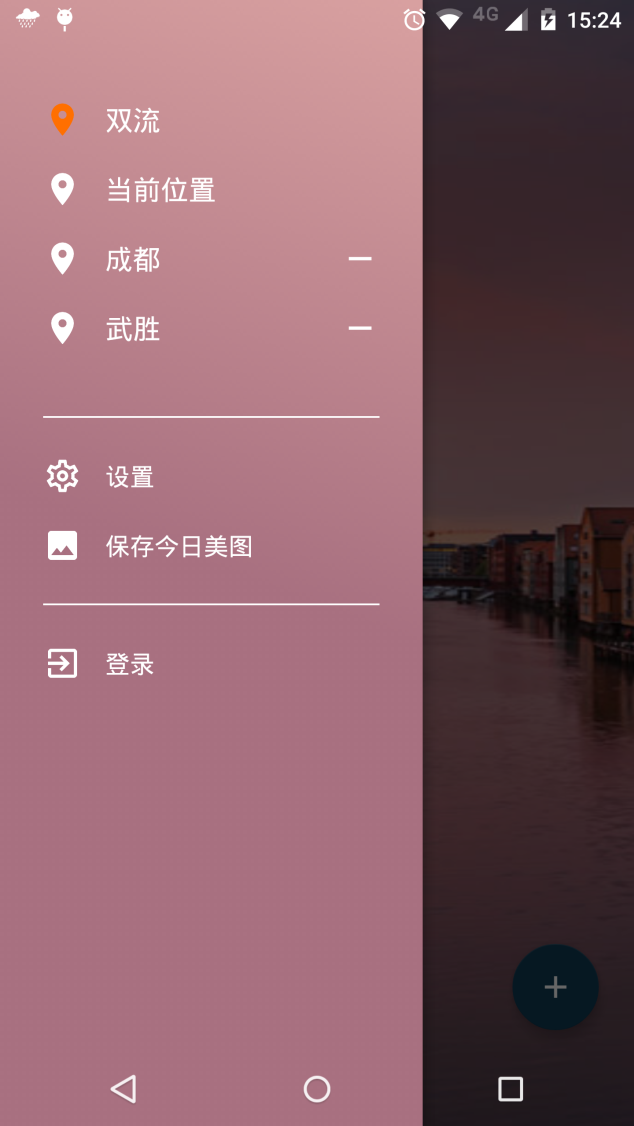


图5-4 侧滑菜单

**5.3 后台天气信息自动更新**

后台的自动更新的前提是要在设置界面中开启后台，并设置更新频率，在MainActivity初始化中就会检测后台，并开启。代码如下：

if(settingsManager.isOpenService()) {

startService();

}

在Service中会注册一个定时动态广播，发送的时间间隔通过用户在设置中的设置。但广播收到消息后，会请求WeatherInfoManager更新天气信息，同时再开启循环后台。

循环任务核心代码如下：

long triggerTime = SystemClock.elapsedRealtime()

+ AN\_HOUR\*settingsManager.getRate();

Intent intent1 = new Intent();

intent1.setAction(UPDATE\_WEATHER);

PendingIntent pendingIntent = PendingIntent.getBroadcast(this, 0,

intent1, PendingIntent.FLAG\_UPDATE\_CURRENT);

alarmManager.set(AlarmManager.ELAPSED\_REALTIME\_WAKEUP,

triggerTime, pendingIntent);

**5.4 桌面小部件和通知栏**

桌面小部件和通知栏效果如图5-5。

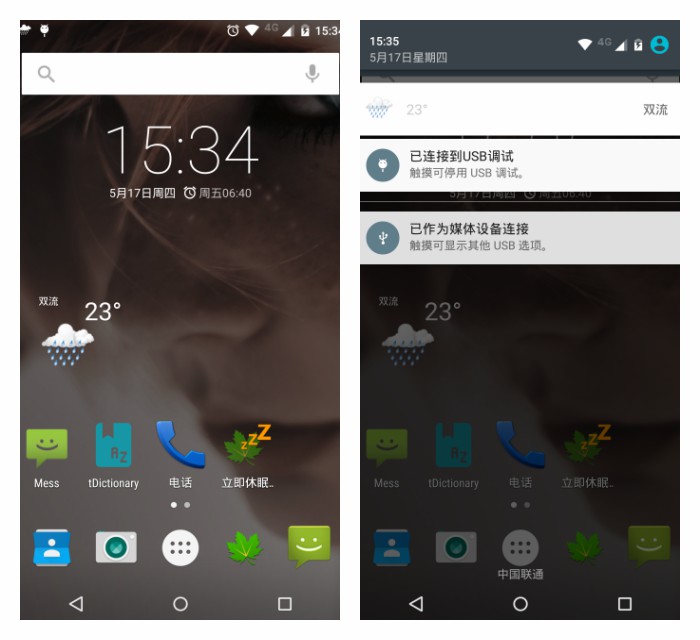


图5-5 桌面小部件和通知栏

小部件和通知栏都是同时依赖于WeatherInfoManager中信息下发，但是消息获取的原理不同。

小部件在Android中的实现原理是广播。本系统的小部件刷新原理也是基于自定义广播。WeatherInfoManager中天气信息刷新成功后会发送这个自定义广播，在添加接收这个广播。

在WeatherInfoManager发送广播代码如下：

Intent intent = new Intent();

intent.setAction(WeatherInfoWidget.UPDATE\_WEATHER);

mContext.sendBroadcast(intent);

小部件接收广播代码如下：

if (intent.getAction().equals(UPDATE\_WEATHER))

updateAllWidget(context);

通知栏和后台更新的Service绑定在一起的，优先级较高，不容易被杀死。如果需要关闭通知栏，需要在设置界面设置。通知栏在SettingsManager和WeatherInfoManager中设置了监听。当设置中的信息和天气信息改变后会通知到通知栏，然后做出相应UI刷新。

天气监听主要代码如下：

if(settingsManager.isOpenNotification()) {

refreshNotification();

}

设置监听主要代码如下：

if(settingsManager.isOpenNotification()) {

showNotification();

} else {

cancelNotification();

}

**5.5 全国城市信息获取和添加常用城市**

在主界面初始化的时候ChinaCitesManager会检测数据库中是否有全国城市信息，如果没有就会从中国天气网API中获取城市信息，然后添加到数据中。该方法是通过递归的方式完成中国所有城市信息的获取。

该方法实现主要代码如下：

String s = result.toString();

String[] cities = splitCityString(s);

for(int i=0;i< cities.length;i++) {

String[] city = splitCityNameAndCode(cities[i]);

saveCity(city,parentCityCode,level);

if(level < END\_LEVEL) {

initCitiesInfo(city[0],level+1);

}

}

在设置添加界面城市一共分成3个等级，等级1是省级，等级2是市级，等级3是县级。一个省级包括若干个市级，一个市级包括若干个县级。例如：四川->成都->双流。在城市添加界面点击1级和2级城市时会刷新页面显示下一级城市，如果是3级城市同时没有添加到本地常用城市就会添加该城市，并关闭该添加界面，添加成功后可以在主界面的侧滑栏中查看。添加界面如图5-6。

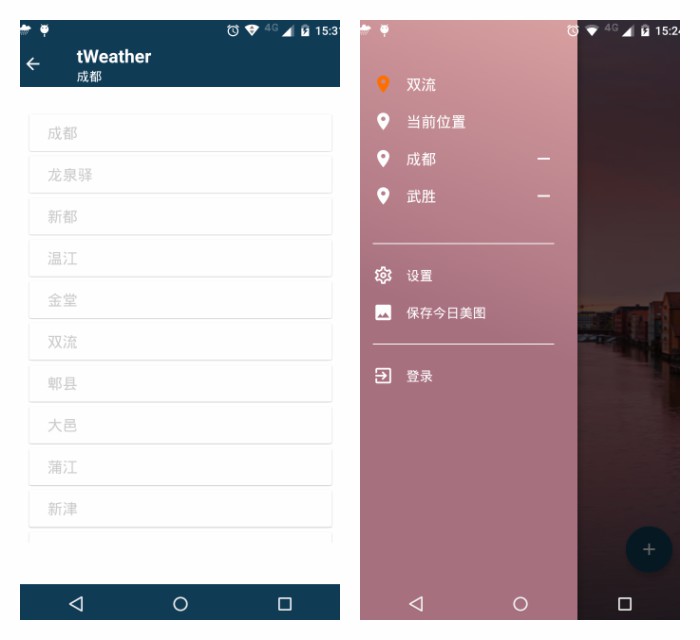


图5-6 常用城市添加界面

**5.6 设置信息**

在主界面的侧滑栏中点击设置就可以进入设置界面，设置界面如图5-7。在设置界面一共有加载Bing美图、打开通知、后台Service、Service更新频率、背景亮度等几个设置项。点击重置按钮就会重置用户的修改项，点击保存就会把设置保存到本地，同时根据设置信息调整相应的UI界面，如果用户登录了，还会同步数据到服务端。

保存设置信息到本地的代码如下：

settingsManager.setOpenService(openService);

settingsManager.setLoadImage(loadImage);

settingsManager.setRate(rate);

settingsManager.setAlpha(alpha);

settingsManager.setOpenNotification(openNotification);

settingsManager.save();

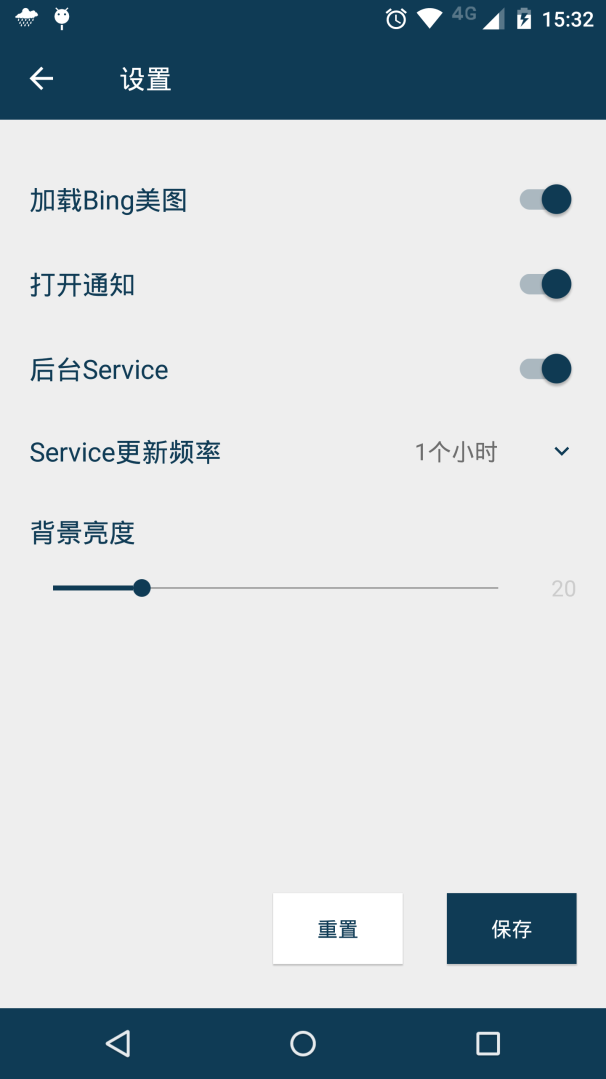


图5-7 设置界面

**5.7 用户信息保存到服务端**

**6测试**

**6.1 初次数据初始化测试**

**6.2 添加常用城市天气信息测试**

**6.3 获取当前位置天气信息测试**

**6.4 小部件和通知栏信息同步测试**

**6.5 和服务端数据同步测试**

**结 语**

[**参考文献**](file:///D:\android\tWeather\important\论文格式摸板(2007).doc)

**致 谢**

**声 明**