天津工业大学

毕业设计（论文）

基于安卓平台的 114 生活助手系统的设计与实现

姓　　名　钱晓闻

学　　院　计算机科学与软件

专　　业　软件工程

指导教师　杨晓光　张俊

职　　称　副教授　工程师

2015 年 06 月 01 日

天津工业大学毕业设计（论文）任务书

<table>
<tr><td>题目</td><td colspan="5">基于安卓平台的114生活助手系统的设计与实现</td></tr>
<tr><td>学生姓名</td><td>钱晓闻</td><td>学院名称</td><td>计算机科学与软件</td><td>专业班级</td><td>软件1103</td></tr>
<tr><td>课题类型</td><td colspan="5">实际课题</td></tr>
<tr><td>课题意义</td><td colspan="5">随着科技的发展，移动智能终端逐渐走进人们的视线，相关应用越来越广泛，并在人们的日常生活中扮演着越来越重要的角色。因此关键应用程序的开发，成为影响移动智能终端普及的重要因素，设计并开发实用、方便的应用程序，具有重要的意义和良好的市场前景。
114生活助手在语音服务的传统业务基础上，丰富了生活服务、商城购物、会员积分等服务内容，全力打造最方便快捷的一站式服务预订能力，为您提供挂号、订餐、机票、酒店代驾、鲜花、蛋糕、商城等便捷服务，从医、食、住、行、娱各个方面，全面服务您的日常生活。
利用现有的AndroidStudio开发环境和Andoird SDK完成114生活助手系统软件的各个功能模块的开发。学习掌握与现有第三方平台的对接。学习掌握Android在不同尺寸、不同分辨率、不同版本上设备的兼容。学习掌握Android 最新的Material Design的应用和设计。学习和掌握Android下的MVVM框架的开发方式。</td></tr>
<tr><td>任务与进度要求</td><td colspan="5">2015.3.9-2015.3.29 选题确认并完成开题报告、任务书的填写、提交、审核
2015.3.30-2015.4.12 深入了解课题内容、需求分析、确定系统框架、熟悉开发工具
2015.4.13-2015.5.10 完成课题概要设计和详细设计，代码编写、网页制作，完成系统的大部分功能，初稿完成
2015.5.11-2015.5.28 进行系统调试，并在调试中进一步完善系统的各项功能，二稿完成
2015.5.29-2015.6.6 毕业设计（论文）的审核、修改及定稿并装订
2015.6.7 答辩</td></tr>
</table>

主要参考文献	[1]卢秀芸．JAVA 异常处理的分析与研究．电脑知识与技术，2013，35 [2]戴歆．JAVATCP 网络通信编程研究．计算机科学，2014，15：17～20 [3]卢晓阳．MVC 设计模式的设计与实现．软件导刊杂志，2011，5：35～14 [4]刘树立．TCP 网络路由协议性能分析．电脑与信息技术杂志，2013，8：65 [5]刘雄鹰．MFC 数据控件绑定数据．电脑与信息技术杂志，2013，8：12
起止日期	2015 年 3 月 9 日至 2015 年 6 月 7 日
备注	

院长_____　　**教研室主任**_____　　**指导教师**_____

毕业设计（论文）开题报告表

2015年 3 月23日

姓名	钱晓闻	学院	计算机科学与软件	专业	软件工程	班级	软件1103
题目	基于安卓平台的114生活助手系统的设计与实现					指导教师	杨晓光 张俊

一、与本课题有关的国内外研究情况、课题研究的主要内容、目的和意义：

1. 与本课题有关的国内外研究情况

国内情况：

1) 作为目前移动端占有率最高的Android系统，能够在其上运行的生活工具类应用功能往往比较单一。有的只具有查询天气的功能。有的只具有订票的功能。并且这些工具提供的信息往往不够及时和准确。
2) 随着Android系统的不断更新升级。在最新的Google IO大会上，Google推出了最新的Andorid5.0系统和Metrail Design。然而目前市场上存在的生活助手类应用仍然采用了过时和老旧的设计。
3) 随着移动设备的大爆发，除了Android手机外。还有Android平板、Android电视、Android智能手表的出现。然而目前的生活工具类应用几乎只支持在Android手机上正常运行。
4) 随着移动设备的尺寸越来越大，屏幕分辨率越来越高。国内的大部分生活助手类工具在这些大屏幕，高分辨率的Android手机上运行效果都不尽如人意。

国外情况：

1) 国外的生活类工具应用虽然与国内相同存在功能单一，提供信息不够准确的缺陷。但是在机型适配上面要稍稍优于国内的同类产品。并且其中有一部分产品已经开始在尝试使用Android最新的Material Design了。
2) 由于相应的生态环境没有建立的原因。国外的生活助手类工具往往缺少P2P和O2O这样需要用户与用户之间对接，线上和线下对接的功能。因此这些工具往往缺乏活力和吸引力。

2. 课题研究的主要内容

1) 医疗版块：实现与现有医疗系统的对接。
2) 票务版块：将系统与铁路、公交、航班信息对接。
3) 餐饮版块：对合作餐饮公司、饭店的信息进行收集、修改和删除工作。
4) 家政版块：维护家政合作商家信息，提供保洁、月嫂、搬家在线电话服务。
5) P2P商城版块：用户可以注册为会员，会员可以自由刊登商品信息进行线下交易。

3. 课题研究的主要的目的和意义

114生活助手系统丰富了生活服务、商城购物、会员积分等服务内容，全力打造最方便快捷的一站式服务预订能力，为您提供挂号、订餐、机票、酒店代驾、鲜花、蛋糕、商城等便捷服务，从医、食、住、行、娱各个方面，全面服务您的日常生活。

4. 本次毕业设计应达到的目标

利用现有的AndroidStudio开发环境和Andoird SDK完成114生活助手系统软件的各个功能模块的开发。学习掌握与现有第三方平台的对接。学习掌握Android在不同尺寸、不同分辨率、不同版本上设备的兼容。学习掌握Android 最新的Material Design的应用和设计。

二、进度及预期结果：

起止日期	主要内容	预期结果
2015.3.9-2015.3.29	选题确认并完成开题报告、任务书的填写、提交、审核	完成
2015.3.30-2015.4.12	深入了解课题内容、需求分析、确定系统框架、熟悉开发工具	完成
2015.4.13-2015.5.10	完成课题概要设计和详细设计，代码编写、网页制作，完成系统的大部分功能，初稿完成	完成
2015.5.11-2015.5.28	进行系统调试，并在调试中进一步完善系统的各项功能，二稿完成	完成
2015.5.29-2015.6.6	毕业设计（论文）的审核、修改及定稿并装订	完成
2015.6.7	答辩	完成
完成课题的现有条件	**硬件：** meizu note1、google nexus7、macbook air MD720、HP CQ14-101 **软件：** AndroidStudio1.0、gradle2.3 **参考文献：** [1]卢秀芸．JAVA 异常处理的分析与研究．电脑知识与技术，2013，35 [2]戴歆．JAVATCP 网络通信编程研究．计算机科学，2014，15：17～20 [3]卢晓阳．MVC 设计模式的设计与实现．软件导刊杂志，2011，5：35～14 [4]刘树立．TCP 网络路由协议性能分析．电脑与信息技术杂志，2013，8：65 [5]刘雄鹰．MFC 数据控件绑定数据．电脑与信息技术杂志，2013，8：12	
审查意见	**指导教师：_____ _____年_____月_____日**	
学院意见	**主管领导：_____ _____年_____月_____日**	

天津工业大学本科毕业设计（论文）评阅表

（论文类）

<table>
<tr><td>题目</td><td colspan="5">基于安卓平台的 114 生活助手系统的设计与实现</td></tr>
<tr><td>学生姓名</td><td>钱晓闻</td><td>学生班级</td><td>软件 1103</td><td>指导教师姓名</td><td>杨晓光 张俊</td></tr>
<tr><td>评审项目</td><td colspan="3">指标</td><td>满分</td><td>评分</td></tr>
<tr><td rowspan="2">选题</td><td colspan="3">能体现本专业培养目标，使学生得到较全面训练。题目大小、难度适中，学生工作量饱满，经努力能完成。</td><td>10</td><td></td></tr>
<tr><td colspan="3">题目与生产、科研等实际问题结合紧密。</td><td>10</td><td></td></tr>
<tr><td>课题调研、文献检索</td><td colspan="3">能独立查阅文献以及从事其他形式的调研，能较好地理解课题任务并提出实施方案；有分析整理各类信息，从中获取新知识的能力。</td><td>15</td><td></td></tr>
<tr><td rowspan="2">论文撰写</td><td colspan="3">结构严谨，理论、观点、概念表达准确、清晰。</td><td>10</td><td></td></tr>
<tr><td colspan="3">文字通顺，用语正确，基本无错别字和病句，图表清楚，书写格式符合规范。</td><td>10</td><td></td></tr>
<tr><td>外文应用</td><td colspan="3">能正确引用外文文献，翻译准确，文字流畅。</td><td>5</td><td></td></tr>
<tr><td rowspan="3">论文水平</td><td colspan="3">论文论点正确，论点与论据协调一致，论据充分支持论点，论证过程有说服力。</td><td>15</td><td></td></tr>
<tr><td colspan="3">有必要的数据、资料支持，数据、资料翔实可靠，得出的结论有可验性。</td><td>15</td><td></td></tr>
<tr><td colspan="3">论文有独到见解或有一定实用价值。</td><td>10</td><td></td></tr>
<tr><td colspan="4">合计</td><td>100</td><td></td></tr>
<tr><td colspan="6">意见及建议：

评阅人签名： 2015 年 月 日</td></tr>
</table>

天津工业大学毕业设计（论文）成绩考核表

学生姓名	钱晓闻	学院名称	计算机科学与软件	专业班级	软件 1103
题目	基于安卓平台的 114 生活助手系统的设计与实现				

1．毕业论文指导教师评语及成绩：

成绩：

指导教师签字：_____ 2015 年 月 日

2．毕业论文答辩委员会评语及成绩：

成绩：

答辩主席（或组长）签字：_____ 2015 年 月 日

3．毕业论文总成绩：

a.指导教师 给定成绩	b.评阅教师 给定成绩	c.毕业答辩成绩	总成绩 (a×0.5+b×0.2+c×0.3)

摘　要

随着移动互联网的高速发展，Android操作系统在移动设备中地位已经被牢牢稳固。然而大量的Android设备高速普及过程中，与其配套的Android应用的开发速度和项目质量极为令人担忧。本课题的研究目的是通过114生活助手系统的开发，寻找Android应用快速迭代开发和高质量保证的开发方式。

在课题的研究过程中，通过对114生活助手系统的开发。实现了手机端的交通信息查询，建立了简易的在线交易市场，并实现了一套基于位置的周边信息查询系统。在开发的过程当中寻找和发现实现Android应用快速开发和高质量保证的技术和方法。在整个的课题研究过程中，尝试采用了众多最新的开源框架技术。这些技术包括EventBus、AndroidAnnotation、Robobinding、picasso、bolts、gson。并且采用了Android官方以及Android社区中普遍认同的最佳实践模式。这些模式包括大量使用Fragment实现解耦、使用Genymotion进行项目调试、等等。

在课题研究的最后阶段，发现采用这些最新的开源框架技术和这些新颖的实践模式，大大加快了过去原有的Android开发速度。同时保证了项目的质量。在各个模块之间的耦合也比按照原有开发模式有了巨大的提升。

关键词：生活助手；android；依赖注入

ABSTRACT

With the high-speed development of mobile Internet, the Android operating system status in the mobile devices has been firmly.A large number of Android devices appear. However, Android application development speed and quality is very bad.It make us very worry.For example,in the application market, the application like LifeAssistant always are bad.They are bad performance,ugly design,information wrong.this research purpose is to find the way to make the Android application development more efficient.

In the research progress , we find the way to Improve the quality and speed for the android application development by develop the 114 life assistant.In the project,i try to use some newest open source project.like EventBus、 AndroidAnnotation、 Robobinding、picasso、bolts、gson.And i have try to use the best practise which has been proposed in the Android official and Android community.these best practise include use fragment to decouping and use Genymotion to debug.

In the end of the research,it is be proved that use these open source project and best practise will greatly improve the develpoment of the android application.And make the application high quality.

Key words: LifeAssistant; Android ;DependencyInjection

目 录

第一章 绪论

随着移动互联网的井喷式爆发，自从 2008 年推出第一款 Android 手机之后，Android 操作系统以一种不可思议的速度不断的快速成长着。目前每天都有上百万部的 Andorid 手机被第一次激活，每月亿万次的 App 被下载。Android 操作系统已经成为当今发展最快的移动设备操作系统。其设备的覆盖已经不仅仅只是手机。随着最新的 Android5.0 的推出，Android 操作系统已经可以在手机、平板、电视、可穿戴设备、汽车上面运行。这些的成就都归功于 Google 公司和众多厂商的支持。然而在这个技术井喷的背景下面，Android 开发的技术要求也越来越高，项目也变的越来越复杂。为了能够更好的支持最新版本的操作系统，为了能够在众多不同尺寸，不同分辨率屏幕上面表现优异。开发者需要付出更多的精力和成本。

本课题的研究目的就是，通过针对 114 生活助手的系统开发，寻找研究 Android 的快速高效的开发方式。在本课题中将会采用最新的开源项目，这些项目都是专门针对 Android 系统设计，目的是简化众多 Android 开发中的繁琐的过程，提高开发效率，简化代码的耦合性。同时将会根据 Google 官方最新提供的开发指南以及 Android 开发社区中最新流行，并被证明有效的最佳实践方式进行项目的开发。这些框架和方式并没有被普遍的证明是行之有效的，或者是万能的。在课题的研究过程中，将会通过 114 生活助手这个项目本身检验这些方法，这些项目是否是真实有用。

该课题的研究过程当中，将会采用 Android 最新的开发技术。其包括使用 Gradle 进行项目的编译和打包，使用 AndroidStudio 替代 Eclipse 开发，使用 Genymotion 作为调试用的虚拟机。并且在开发中将会使用 git 作为版本管理工具，记录开发的所有历史记录。在涉及的开源项目当中，将会使用到 AndroidAnnotation，Gson，Picasso，Bolts，EventBus，ApacheCommons 这些开源项目。这些项目有的是专门针对 Andorid 的框架，有的则是针对 Java 的框架。使用它们的目的在于尽最大的可能简化项目的开发过程。

1.1 安卓应用开发的目前现状

如今，世界上的 190 多个国家当中运行着数以亿计的 Android 设备。他已经成为被安装最多并且成长速度最快的移动操作系统。每天都有百万的用户第一次开启他的 Android 设备并寻找应用和游戏。Android 为开发者提供了一个可以为全世界所有人提供应用和游戏的开发平台。这个基于 Linux 的开源操作系统，拥有超过 300 家的硬件，软件，运营合作伙伴[1]。Android 的开源政策也广受开发者和用户的喜爱。这也直接促进了 Android 应用的大量需求。Android 用户每月从 Google Play 中下载超过 1.5 亿的应用。在这些背景的驱使下，Android 持续不

断地推出最新的硬件和软件，为开发者和用户提供最新的功能。

在最近的 Google IO 大会上，Google 推出了他最新的 Android 操作系统－Android 5.0 Lollipop。这个版本的操作系统为开发者提供了数以千计的最新的 API。他适用于所有的 Android 设备，包括手机、平板、电视、可穿戴设备、汽车。在 Android5.0 当中，使用最新的 ART 虚拟机替代了原先的 Dalvik 虚拟机[2]。ART 虚拟机实现了 AOT，大大增强了 GC 的性能，提高了应用的调试能力。Android5.0 还带来了最新的 Material 设计以及配套的一系列开发工具。这使得开发 UI 更加的方便和高效。最新的 3D 技术使得你可以更容易的开发出实时阴影这样的效果。最新的 RenderThread 线程使得你可以更加平滑的运行应用的动画效果。

然而在这些新技术层出不穷的背后，是开发者技术的止步不前。最新最好的技术不断的出现，但是开发者却依旧停留在过去的开发模式当中。这些老的模式造成了开发效率的不断降低，开发质量的不断恶化，项目进度的不断推迟。同时由于 Android 的高速发展，其碎片化带来的问题使得 Android 的开发难度不断的增大。不同版本的操作系统，不通尺寸和分辨率的屏幕，不同类型的设备。这些由于高速发展带来的问题导致了 Android 开发的难度不断增大[3]。

以 Android 的操作系统为例，目前市场上普遍存在 10 个以上的不同版本的 Android 核心版本。其占有率如表 1-1 和图 1-1 所示。4.0 以上的版本已经占据安卓设备的 80%以上。而另外的份额依旧被 3.0 以下的份额所占据。而最新推出的 Android5.0 操作系统，所占据的份额不到 10%[4]。

表 1-1 Android 操作系统份额

Vesion	Codename	API	Distribution
2.2	Froyo	8	0.4%
2.3.3-2.3.7	Gingerbread	10	6.4%
4.0.3-4.0.4	Ice Cream	15	5.7%
4.1.x	Jelly Bean	16	16.5%
4.2.x		17	18.6%
4.3		18	5.6%
4.4	KitKat	19	41.4%
5.0	Lollipop	21	5.0%
5.1		22	0.4%

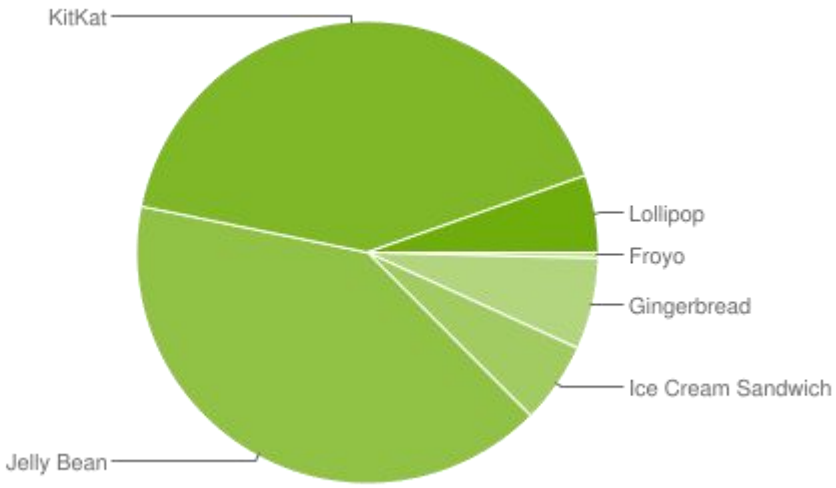

图 1-1 Android 操作系统份额

另外一方面，Android 设备的尺寸跨度非常巨大。如表 1-2，图 1-2 所示，适配一款 Android 应用需要考虑的屏幕尺寸大小需要分为 small、normal、large、Xlarge 四种。而分辨率则需要分为 ldpi、mdpi、hdpi、xhdpi、xxhdpi 六种。而大屏和高分辨率正在变为市场的主流趋势[5]。

表 1-2 Android 设备屏幕尺寸及分辨率

	Ldpi	Mdpi	Hdpi	Xhdpi	Xxhdpi	Total
Small	4.4%					4.4%
Normal		8.1%	39.3%	19.5%	15.9%	82.9%
Large	0.4%	4.8%	0.6%	0.6%		8.6%
Xlarge		3.2%	0.3%	0.6%		4.1%
Total	4.8%	16.1%	40.2%	20.7%	15.9%	

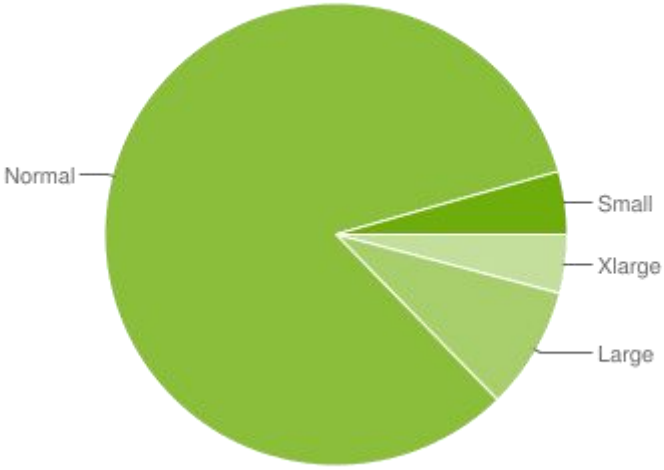

图 1-2 Android 设备屏幕尺寸

1.2 本课题的研究目的和意义

生活助手类应用是 Android 中普遍存在的应用。用户通过该类型的应用实时获取周边信息。从衣食住行各个方面获取最及时最准确的信息。应用应当满足简单易用，高效便捷，功能强大的需求。然而由于 Android 系统的高速迭代，Android 设备的高速发展，Android 应用的开发难度也不断大增加，项目的开发进度也不断的减缓。本课题的研究目的就是，通过实现 114 生活助手 Android 系统，寻找 Android 的快速开发模式。确保以这样的模式，可以迅速的开发出一款 Android 应用，并保证其应有的质量。

课题的研究成果将是一款简单易用，功能强大，高效便捷的生活助手应用。通过他可以方便的查询当前位置的周边信息，可以方便的获取交通信息。并且可以简单的在其平台上面实现 P2P 的在线交易。同时更加重要的是，通过这个项目，寻找到一个快速高效开发 Android 应用的模式。使得原先相对复杂的 Android 开发模式变得简单有效。

1.3 本课题的研究方式和手段

课题将会通过实现一个基于 Android 的 114 生活助手来实现课题的研究目的。在 114 生活助手这个项目的开发过程当中，将会尝试使用最新的开源项目和普遍认同的最佳实践方式进行项目的开发。这些使用到的开源项目有：AndroidAnnotation、Robobinding、Gson、Picasso、Bolts、EventBus。而使用的最佳实践方式是根据 Android 官方最新的开发指南，以及 Android 社区中最为推崇的开发模式总结而成。这些最佳实践模式包括：使用依赖注入实现代码解偶，使用 Fragment 替代 Activity 的部分功能、使用 Genymotion 作为调试工具。另外在开发过程当中，将会使用最新的开发工具开发。这些最新的开发工具包括：使

用 Gradle 进行编译和打包，使用 AndroidStudio 替换原来的 Eclipse，使用 git 作为版本管理工具[6]。

第二章 相关技术介绍

2.1 开发技术的选择

Android 移动操作系统是一个非常成熟的系统。在本项目当中，将会使用使用在 Google 2013 IO 大会上发布的最新的 Android 开发工具作为本项目的开发平台。本项目将使用 AndroidStudio 开发，在 Android Srudio 的第一个正式版本发布之后，其作为 Android 第一开发平台的地位已经被牢牢确定了。Gradle 是今年最为流行了的编译工具，其在具备 Maven 优秀的特性的同时，引入了更多令无数开发者梦寐以求的功能。其简单的 DSL 脚本语言和丰富的构建方式，使得项目的编译打包过程变得无比的自然和简单。

另外在项目当中将会使用 Genymotion 替换 Android 默认的虚拟机。Genymotion 是新一代的虚拟化开源项目，其采用 VirtualBox 作为技术支持，并采用 x86 核心而非 arm[7]。这些种种的改变，直接决定了在调试过程中，使用 Genymotion 将会带来更佳快速，平滑，稳定的体验。并且 Genymotion 提供更为简单的基于位置的调试解决方案，其可以非常简单的设置 GPS 的经纬度坐标，以方便用户进行项目调试。

在项目的开发当中，除了使用最新的 Android 开发框架之外，还大量使用了开源框架。在这些开源项目的帮助下，项目的整体架构变得更为简单，代码的书写变得更加的灵活方便，使得项目的开发效率被大大的提升，同时质量也被大大的增高。这所有的一切都源于开源社区的支持。在接下来的章节当中，将会简单介绍一下，这些被用到的优秀开源框架。

2.2 相关开源框架的使用

2.2.1 EventBus 的使用

EventBus 是由 Greenrobot 提供的一个用于模块间通信的工具，他可以被使用在 Activity，Fragment，Service，Thread 等任何组件之间，他使用方便，代码灵活。他的存在大大减少了原先用于项目模块之间的通信代码。EventBus 是 android 平台上最佳的基于推送和订阅的消息总线框架。

如图 2-1 所示，消息提交者将一个事件推送给 EventBus，之后 EventBus 将事件分发给该事件的订阅者，并回调他们的 onEvent 方法。EventBus 的这种特性使得模块间的通信变的异常的容易，并且执行效率非常的高，使用代码非常的简单，依赖包非常的小。EventBus 被超过一亿的应用使用，这些特性足以证明该框架优越性[8]。

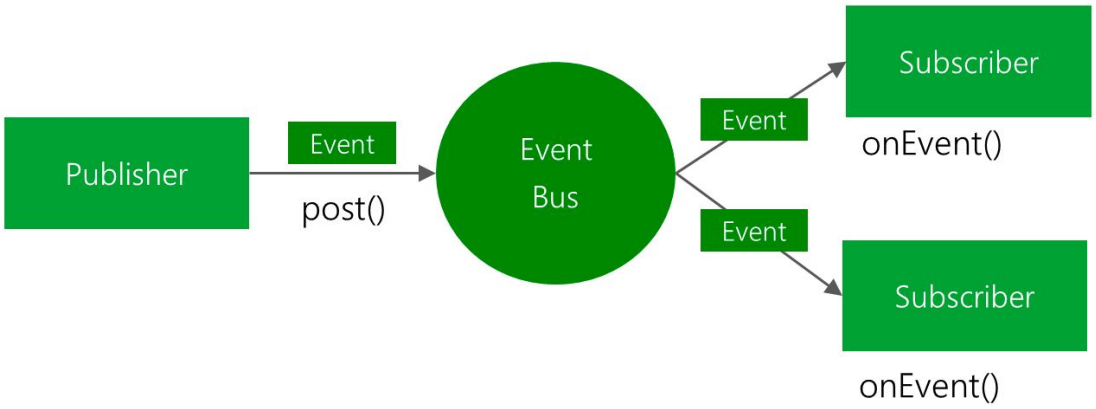

图 2-1 EventBus 的实现方式

同时集成使用 EventBus 的过程也非常的简单，因为 EventBus 已经在 Maven 核心仓库中出现了，因此可以非常容易在 Maven 和 Gradle 中使用 EventBus 框架。

2.2.2 AndroidAnnotation 的使用

AndroidAnnotation 是一个开源的框架，他使得 Android 开发变的更加的迅捷。他使你真正关注项目的核心内容，他使得你的代码变的简洁的同时，也使得项目变的更加的易于维护。他的目标是让开发者写出易于维护的代码，他相信简单的代码就是实现该目标的最佳方法。

在过去的 Android 开发过程当中，开发者会不禁疑问：为什么我们总是在书写重复的代码？为什么项目变的越来越难以维护？Context 和 Activity 这些神一样的对象，负责的线程机制，难以发现适合的 API，一大堆异步的监听类，成千上万的常量。这些问题难道就没有解决方案吗？

通过使用 Java 的注解特性，开发者可以表达出他的意图，并让 AndroidAnnotation 把这些意图在编译时转换成真实的代码。以下是他所有拥有令人神往的特性。

1. 依赖注入，可以注入 Views，Extras，System Service，几乎任何能想象到的东西。

2. 简单的线程模型，通过注解决定该方法是被运行在 UI 线程还是运行在后台线程。

3. 事件绑定，通过注解可以轻松的为 View 绑定事件，而不是过去那种丑陋的异步 Listener。

4. REST 客户端，只需要实现一个简单的 REST 接口，AndroidAnnotation 就可以帮助你生成正真的 REST 代码[9]。

```
@EActivity(R.layout.translate) // Sets content view to R.layout.translate
public class TranslateActivity extends Activity {

    @ViewById // Injects R.id.textInput
    EditText textInput;

    @ViewById(R.id.myTextView) // Injects R.id.myTextView
    TextView result;

    @AnimationRes // Injects android.R.anim.fade_in
    Animation fadeIn;

    @Click // When R.id.doTranslate button is clicked
    void doTranslate() {
        translateInBackground(textInput.getText().toString());
    }

    @Background // Executed in a background thread
    void translateInBackground(String textToTranslate) {
        String translatedText = callGoogleTranslate(textToTranslate);
        showResult(translatedText);
    }

    @UiThread // Executed in the ui thread
    void showResult(String translatedText) {
        result.setText(translatedText);
        result.startAnimation(fadeIn);
    }

    // [...]
}
```

图 2-2 AndroidAnnotation 的演示代码

图 2-2 显示的是 AndroidAnnotation 的演示代码。

2.2.3 Bolts 的使用

Bolts 是由 Parser 和 Facebook 维护的一款 Android 底层的开源框架。他的出现使得 Android 开发变的更加容易。他是 Parser 和 Facebook 真实使用的项目，当他们在内部大规模使用之后，他们决定将其开源。贡献给大家，使得任何人都可以使用到这样的技术。使用这个框架不需要任何的 Parser 服务，不需要 Parser 和 Facebook 的账号支持了。

Bolts 框架主要包括以下内容。

1. Tasks：一个帮助组织管理 java 复杂的异步代码的工具。一个 Task 就像是 JavaScript 的 Promise，但是却可以在 Andorid 中使用。

2. 一个用于 Application 之间沟通通信的工具，他用于帮助你实现应用之间的深链接[10]。

在 Android 项目的开发过程中，你将会需要执行众多不能运行在 UI 线程的复杂操作，以避免因此导致的 UI 线程的阻塞和等待。这意味这你需要在后台线程中运行这些程序。为了使得这个过程变得更加的简单，Bolts 创建了一个名为 Task 的类。一个 Task 代表着一个异步操作。一般来说，一个 Task 是被作为一个方法的执行结果返回的，此刻他已经开始执行他的工作了。一个 Task 不是传统意义上的线程模型，他对应的是一个工作的完成，而不是工作的执行。Task 相对于其他的异步编程技术有很多优势，像 Callback 和 AsyncTask，在 Task 面前变得不值一提。

以下是 Task 的特性：

1. 更少的系统资源消耗，因为他不会为了等待其他的任务而占据某个线程

2. 在一行代码当中使用多个 Task，不会像你使用 Callback 时那样造成代码量的增长

3. 他提供了丰富的语法，使得你可以处理分支，并排，和错误处理这些需求。而不需要像过去那样写出像意大利面一样层层叠叠的代码

4. 你还可以用 Task 的基础代码控制任务的执行顺序，相比过去那种需要你通过多个回调接口来控制业务逻辑的方式变的无比的简洁

在图 2-3，图 2-4，图 2-5 将会像你简单介绍 Task 的使用方式：

```
saveAsync(obj).onSuccess(new Continuation<ParseObject, Void>() {
  public Void then(Task<ParseObject> task) throws Exception {
    // the object was saved successfully.
    return null;
  }
});
```

图 2-3 Bolts 的演示代码

```
/**
 Gets a String asynchronously.
 */
public Task<String> getStringAsync() {
  // Let's suppose getIntAsync() returns a Task<Integer>.
  return getIntAsync().continueWith(
    // This Continuation is a function which takes an Integer as input,
    // and provides a String as output. It must take an Integer because
    // that's what was returned from the previous Task.
    new Continuation<Integer, String>() {
      // The Task getIntAsync() returned is passed to "then" for convenience.
      public String then(Task<Integer> task) throws Exception {
        Integer number = task.getResult();
        return String.format("%d", Locale.US, number);
      }
    }
  );
}
```

图 2-4 Bolts 的演示代码

```
saveAsync(obj).continueWith(new Continuation<ParseObject, Void>() {
  public Void then(Task<ParseObject> task) throws Exception {
    if (task.isCancelled()) {
      // the save was cancelled.
    } else if (task.isFaulted()) {
      // the save failed.
      Exception error = task.getError()
    } else {
      // the object was saved successfully.
      ParseObject object = task.getResult();
    }
    return null;
  }
});
```

图 2-5 Bolts 的演示代码

2.2.4 Android 开发最实践

然而即使如此，项目的开发过程当中依旧充斥着许多令人痛苦的过程，项目会在无形之间变的越来越难以维护，项目周期变的越来越长。正因如此，非常需要一个最佳的方式指导开发者完成这些工作。以下内容总结来自于 android 的开发社区，也正是本项目使用的方式[11]。

概括的可以总结为以下几点：

1. 使用 Gradle 和他推荐的项目架构。
2. 将密码和敏感的数据放置在 gradle.properties 中。

3. 不要使用你自己的 HTTP 客户端，而是使用 Volley 或者 OkHttp 开源框架。

4. 使用 Jackson 开源框架解析 JSON 数据。

5. 避免使用 Guava 和尽量少的使用第三方框架，因为有 65k 方法数的限制。

6. 使用 Fragment 来渲染 UI。

7. 使用 Activity 管理 Fragment。

8. 把 XML 布局文件当作代码看待，好好的管理他们。

9. 在 XML 布局文件当中避免使用过多的参数设置，而是使用 style 替代他们。

10. 使用多个 style 文件，而不是使用单一且庞大的一个 style。

11.保持你的 color.xml 短小简洁，仅仅只是定义调色板。

12. 不要直接使用深度嵌套的 ViewGroups。

13. 使用 Robolectric 用于单元测试，使用 Robotium 进行 UI 测试。

14. 使用 Genymotion 作为你的虚拟机。

15. 经常使用 ProGurad 活着 DexGuard。

关于项目架构，目前有两种选择。使用 Ant 和 Eclipse 的项目架构，使用 Gradle 和 AndroidStudio 的项目架构[12]。你应该选择最新的项目架构，如果你使用的是老的架构，那么强烈建议你抛弃他，并且尝试最新的架构[13]。图 2-6 老的项目架构，图 2-7 是新的项目架构

```
old-structure
├─ assets
├─ libs
├─ res
├─ src
│  └─ com/futurice/project
├─ AndroidManifest.xml
├─ build.gradle
├─ project.properties
└─ proguard-rules.pro
```

图 2-6 老的项目架构

```
new-structure
├ library-foobar
├ app
│  ├ libs
│  ├ src
│  │  ├ androidTest
│  │  │  └ java
│  │  │     └ com/futurice/project
│  │  └ main
│  │     ├ java
│  │     │  └ com/futurice/project
│  │     ├ res
│  │     └ AndroidManifest.xml
│  ├ build.gradle
│  └ proguard-rules.pro
├ build.gradle
└ settings.gradle
```

图 2-7 新的项目架构

第三章 需求分析与交互设计

基于安卓系统的 114 生活助手系统，在衣食住行各个方面为用户提供方便可靠的信息咨询。其主要提供的功能可以分为以下三类：

1．基于位置的周边信息。

2．覆盖航空，火车，长途大巴的交通信息查询系统。

3．在线交易市场。

其具体的功能又可以划分为以下几点。

3.1 城市选择功能

用户可以选择当前所在的城市，根据当前的城市信息，查询所需的生活信息。提供一个城市列表供用户选择，并且可以通过 GPS 定位到当前设备所在的城市，方便用户选择城市。城市选择用例图如 3-1 所示。

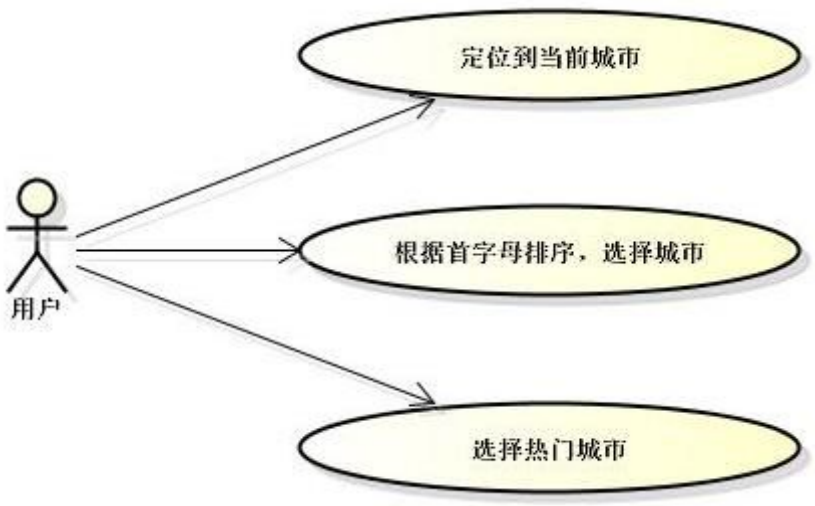

图 3-1 城市选择的用例图

3.2 查询医疗信息功能

用户可以查询到当前选择城市的相关医疗信息。这些信息主要包括周边的医院的地址，电话，介绍。通过这些信息为用户提供医疗服务。用户还可以添加相关的医疗信息，以丰富数据。其用例图如图 3-2 所示。

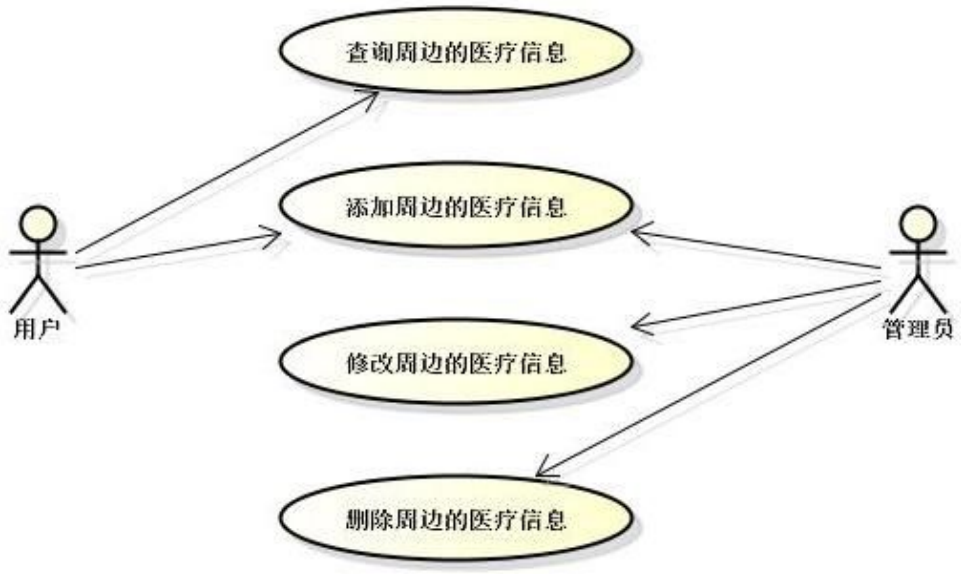

图 3-2 医疗信息的用例图

3.3 查询交通信息功能

用户可以查询当前的实时航班信息，查询到 12306 的实时火车票信息，查询到长途大巴票的信息。根据这些信息为用户提供出行指南。其用例图如图 3-3 所示。

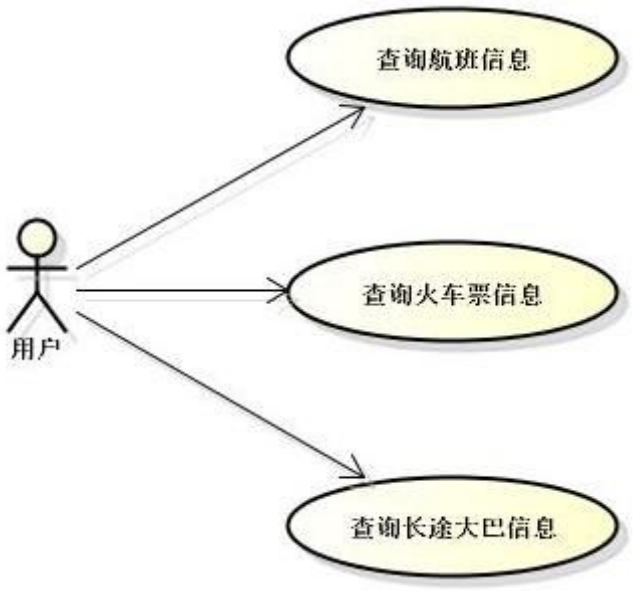

图 3-3 交通查询的用例图

3.4 查询酒店信息功能

用户可以根据当前所在的城市，查询周边的酒店信息。这些信息主要包括酒

店的地址，电话，介绍。通过这些信息用户可以随时找到合适的酒店安顿下来。当然，用户还可以添加相关的酒店信息，以丰富数据。其用例图如图 3-4 所示。

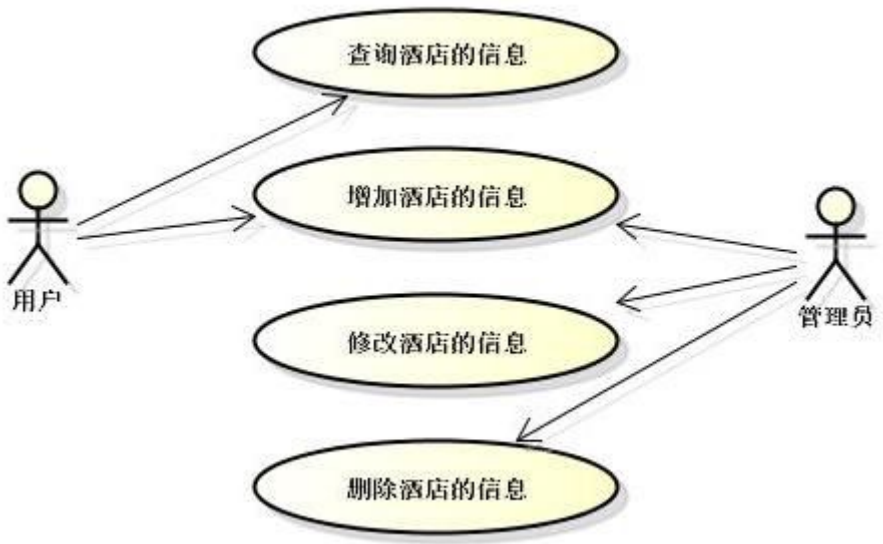

图 3-4 酒店信息的用例图

3.5 用户身份验证功能

用户可以登录到手机程序，如果用户没有该程序的账号，用户还可以注册一个新的账号。当用户成功进行身份验证之后，用户便可以添加新的信息，并可以根据用户的权限管理信息。其用例图如图 3-5 所示。

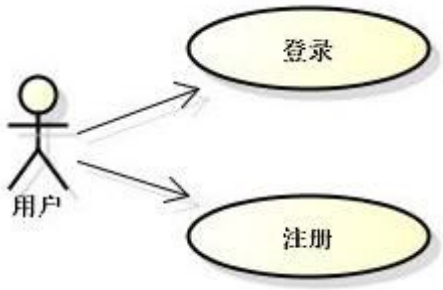

图 3-5 用户身份验证的用例图

3.6 查询餐饮信息功能

用户可以根据当前所在的城市，查询周边的餐馆和美食信息。这些信息主要包括餐馆的地址，电话，介绍。通过这些信息满足用户对于食物和美食的需求。当然，用户还可以添加相关的餐饮信息，以丰富数据。 其用例图如图 3-6 所示。

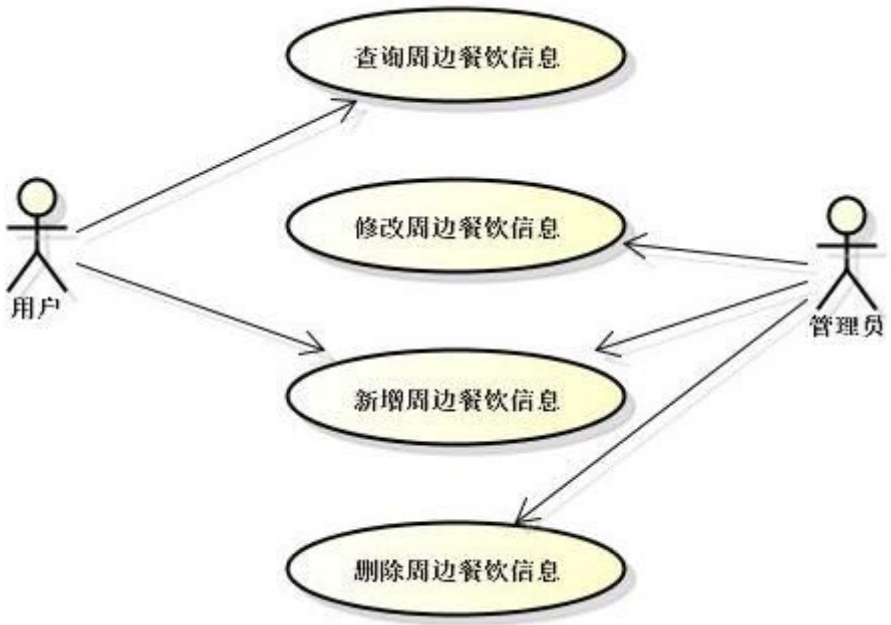

图 3-6 餐饮信息的用例图

3.7 在线市场交易功能

用户可以游览在线市场当中的商品信息。这些信息包括商品的价格，介绍，图等。如果用户需要其中的某一件商品则可以通过线下联系的方式进行交易。当然，用户也可以发布自己的商品信息。从而形成 P2P 的在线交易市场。当然，管理员身份的用户可以审核这些信息，并对其进行管理。其用例图如图 3-7 所示。

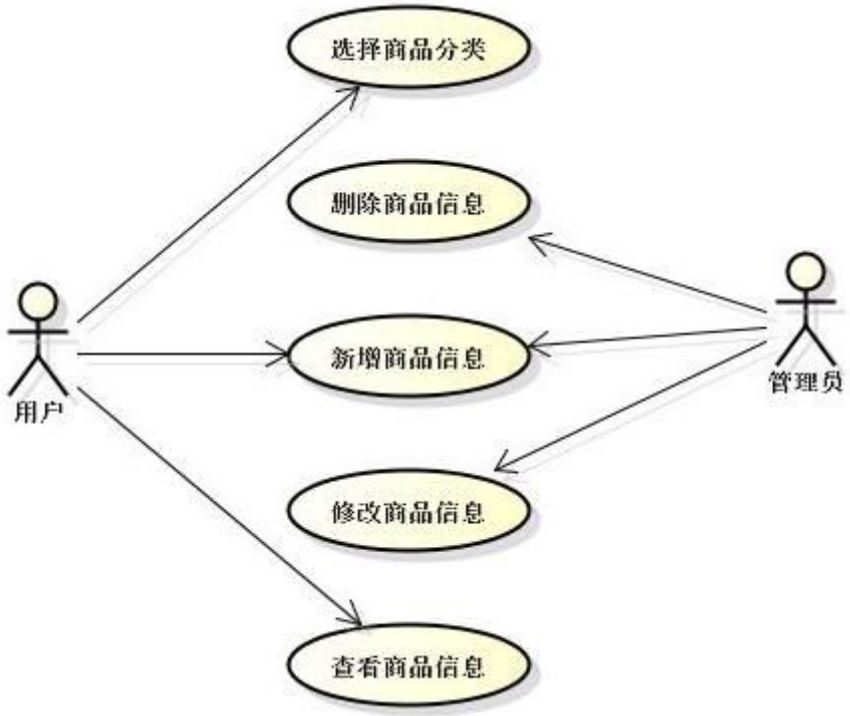

图 3-7 在线交易平台的用例图

3.8 查询家政服务功能

用户可以根据当前所在的城市，查询到关于清洁工，月嫂，搬家公司的信息。通过这些信息用户可以获取到自己所需要的服务。当然，如果用户是相关业务的从业者，也可以在程序当中发布自己的信息。其用例图如图 3-8 所示。

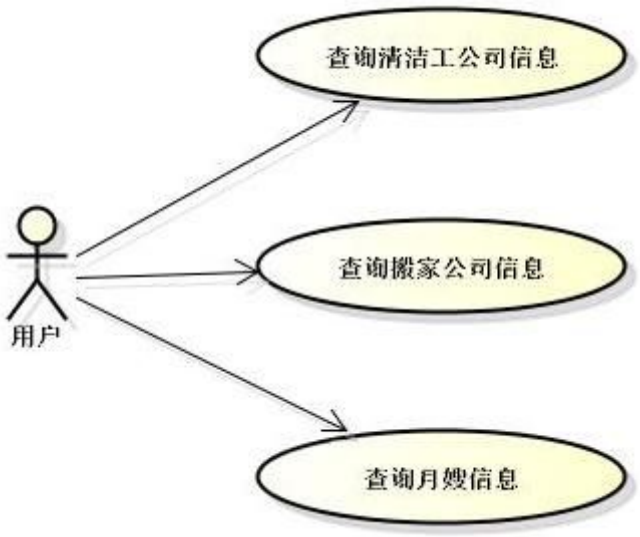

图 3-8 家政信息的用例图

第四章 架构设计与实现

4.1 整体架构设计

系统的整体架构分为三层，如图 4-1 所示，分别有 Activity，Fragment，Model 实现。这样的层次划分，降低了如图 4-2 所示的系统架构。这种过去将大量的业务逻辑全部封装在 Actvitiy 中的实现方式导致的项目臃肿，不易维护，大量重复代码的问题。其中 Activity 主要负责封装一整套的业务逻辑（例如：和交通信息相关的所有业务逻辑），同时负责管理和维护 Frament 的生命周期。而 Fragment 则负责具体的某个业务逻辑的实现。而 Model 则是真正负责实现该业务逻辑的具体代码。

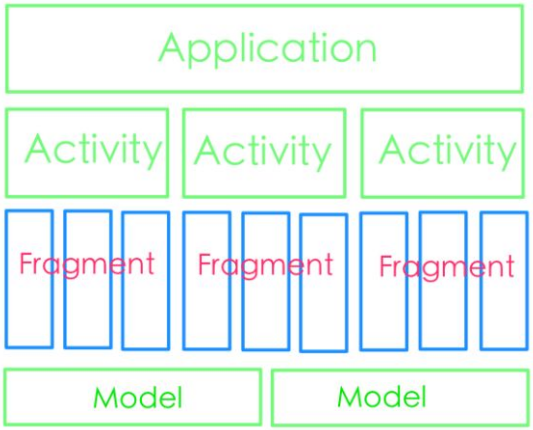

图 4-1 本项目采用的架构

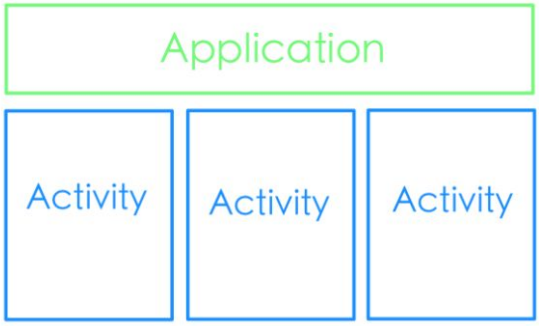

图 4-2 大量 Android 项目中采用的架构

正如上面所说一个模块是由一个 Activity 和若干 Fragment 构成。真正的业务逻辑由 Fragment 实现。而 Actvitiy 作为这些 Fragment 的容器和 Manger。如图 4-3 所示，若干 Activity 构成了整个应用的功能模块，而每个 Activity 由若干 Fragment 构成。

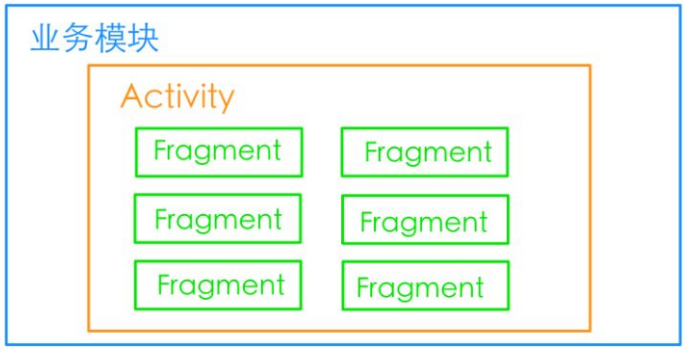

图 4-3 业务模块的实现

而在 Android3.0 之前，系统的功能实现只能由单个 Activity 实现，众多的 Activity 组合成一个模块，就如图 4-4 所示。这种架构的实现方式的缺点在于，Activity 之间的通信方式非常困难。同时也不利于程序代码的解耦工作。使用 Fragment 后的程序架构不仅变的更佳清晰，也有利于后期适配不同尺寸的工作。

图 4-4 业务模块的组成

而各个 Fragment 的具体业务逻辑又由相应的 BusinessModel，ServiceModel，PersistModel 实现。就如图 4-5 所示，若干 Model 支撑着 Fragment 的具体实现。在这些 Model 中，ServiceModel 主要负责与服务器对通信。PersistModel 主要负责数据持续化的工作。BusinessModel 主要负责专业的业务逻辑的工作。这些模块之间相互配合，相互调用，最终使得 Fagment 可以正常使用。

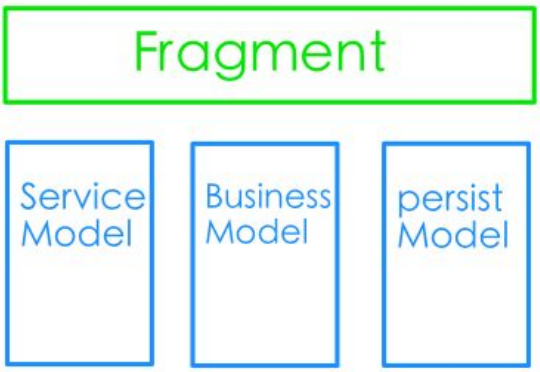

图 4-5 元功能模块的架构

4.2 模块之间的通信方式

4.2.1 Activity 之间的通信

而在这样的架构当中，其模块之间的通信又和传统的开发模式有所不同。在原有的完全基于 Activity 的架构当中，其模块之间通信如图 4-6 所示，Activity 之间通过 Intent 通信，通过 Bundler 传递数据。采用 StartActivityForResult 和 OnActivityResult 实现代码回调。这样的方式，在开发过程当中非常不便，代码不易阅读和维护。而在本项目当中使用的是如图 4-7 所示的通信方式。直接通过 BusinessModel 调用打开 Activity，而在调用 BusinessModel 的同时，直接将数据以方法参数的形式间接传递给 Activity。之后又通过 Bolts 框架的异步回调，实现 Activity 的异步回调。这样的通信架构设计解决了 Activity 与 Activity 之间原本糟糕的通信方式。

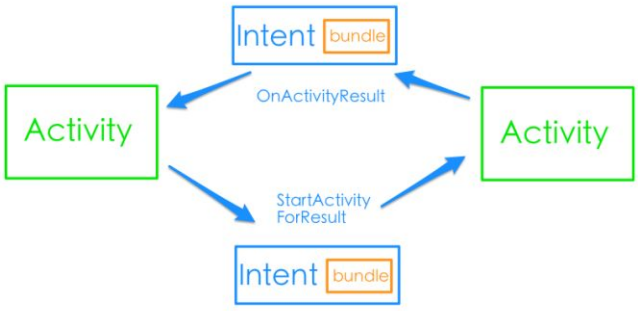

图 4-6 大量项目采用的通信方式

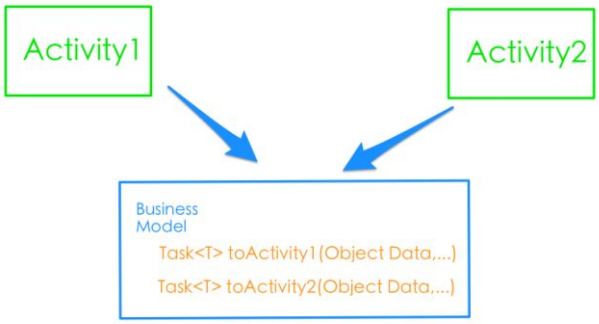

图 4-7 本项目采用的通信方式

4.2.2 Fragment 之间的通信

由于 Fragment 本身的设计相对于 Activity 来说要更加容易实现模块间通信。在本项目当中采用了构造者的设计模式，将需要传入的数据传递给 Builder，并把回调接口也传递给 Builder。之后再调用 create 方法，生成 Fragment。也就是说，Fragment 使用了直接传值和回调接口的方式实现了模块通信。当然最后依旧使用 BusinessModel 对这些过程进行分装。具体的流程如图 4-8 所示

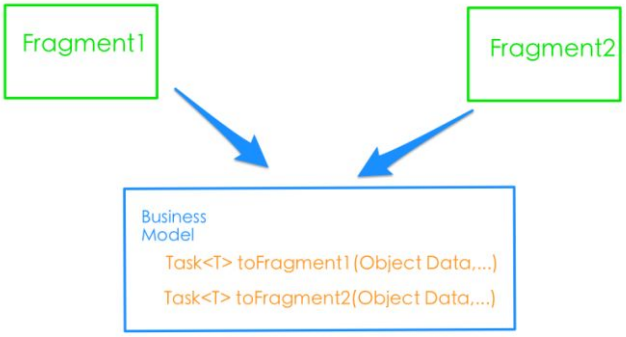

图 4-8 本项目采用的 Fragment 的通信方式

4.2.3 其他模块之间的通信

对于其他模块的通信主要分为同步通信和异步通信。而在这其中主要将会遇到的问题是异步通信。目前大多的应用采用的都是 AsyncTask 或者在方法中传递回调接口的方式实现异步通信。但两者的代码都会造成代码的臃肿和维护的苦难。在本项目当中采用的是 Bolts 的框架，通过该框架实现了优美的异步通信通信模式。

4.3 业务模块的划分

整个系统分为以下 5 个模块。如图 4-9 所示。

1．功能导航：实现应用的首页，为用户提供功能导航。

2．POI 模块：提供周边信息的检索，管理相应的周边信息。

3．交通信息模块：提供长途大巴，飞机票，火车票信息的检索。

4．P2P 模块：提供在线交易市场的信息检索与管理。

5．通用组件模块：提供供其他模块使用的的通用模块，包括：城市选择、日期选择、通知弹出、公告信息、用户身份验证。

图 4-9 功能模块划分示意图

4.4 服务端的实现方式

在本项目的实现当中，服务端的数据主要分为三大类。如图 4-10 所示，分为第三方 WebService、项目 Service、虚拟 Service。其中 WebService 使用的是聚合网的云服务和百度的 LBS 云服务，通过链接他们的 API，获取与周边信息，交通信息相关的全网数据。项目 Service，采用的是 LeanClound 和 qiniu 的云服务支持。其提供的是项目自身产生的数据，包括用户信息，P2P 信息等。虚拟 Service，则主要是用于项目调试使用的虚拟服务器。他是一个离线的伪服务器，通过他，在开发过程中可以抛弃服务端而直接进行开发，减少了与服务端的耦合。

图 4-10 本项目采用的服务端架构

在选择选择云服务的过程当中，主要考虑的内容有以下几点。

1．是否能够提供全面准确的数据。

2．API 接口是否足够简单易用。

3．访问速度是否足够理想。

4．是否收费。

聚合网云服务和百度的 LBS 云服务，很显然满足以上要求。聚合网拥有千量级别的海量数据，数据类型富含数百种。其中与生活服务相关等数据占据了数十种。另外除了提供封装完好的 SDK 之外，还提供获取 json、xml、jsonp 数据类型的 API。其数据的丰富程度和稳定性都另人满意。百度的 LBS 云服务，基于百度多年的数据积累。数据的丰富程度非常令人满意，唯独遗憾的事情是，百度的 API 接口不是非常完善，很多数据没有办法获取。但是，对于本项目而言已经足够。

LeanClound 的云服务，是一个 paas 的云服务系统。他过去的称呼是 AvosClound，后改名为 LeanClound。他为开发者提供了非常简单的 API 接口，使得开发者可以非常简单方便的为移动客户端提供数据的管理。而 qiniu 云服务则为本项目提供了一个云图床的服务。通过 qiniu 云服务，大大简化了项目在图片上传下载处理的技术瓶颈。本项目选择这些云服务的一个主要的考虑因素在于，Android 客户端的开发过程往往是需要与远程服务器配合的。但是大量的项目证明，完全建立自己的服务器是完全没有必要的。重复造轮子的过程将会大大增加项目的复杂程度，降低项目的质量，延长项目的周期。而这些云服务技术，为 Android 开发者提供了最为简单和高效的服务端的技术。

4.5 数据持续化的实现方式

在本项目当中，除了需要使用到网络数据之外，还需要使用到部分离线数据。这些数据都统一由持续化层来实现。如图 4-11 所示，本项目提供的持续化模块主要有以下这些。他们的工作就是通过代码的封装，将原本对于 File 和 SharedPreferences 的操作封装起来。为上层模块提供简单明了的 API 服务。在过去的项目当中，该层次的代码，往往被简易的放置在 Actviity 当中，这样的做法直接导致了离线数据的管理混乱和使用不便。时常会出现由于代码遗留的问题导致相同类型的数据被多次存放，或被错误存放。

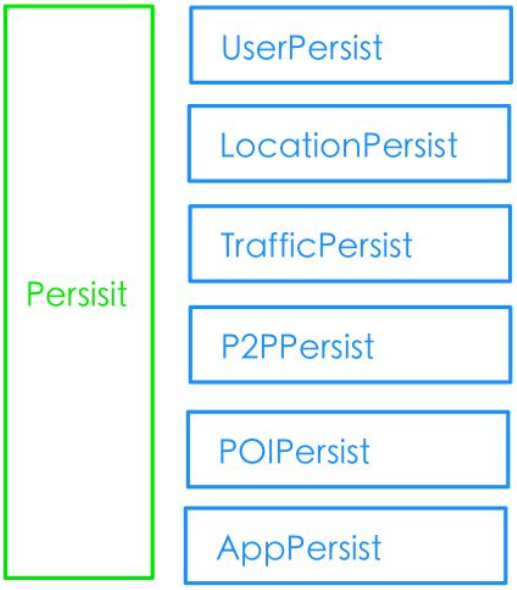

图 4-11 本项目采用的数据持续化架构

下面举例来说明数据持续化层的实现方式。在本项目当中 UserPersisit 主要负责存放和用户信息相关的数据。其背后的实现使用的是 SharedPreferences。通过存储 USER_NAME、USER_ID、USER_ICON 三个字段来持续化用户的信息。

LocationPersist 主要存放和位置相关的数据。其背后的实现使用的是 SharedPreferences 和 File 文件系统。通过 SharedPreferences 存储的是当前选择的城市，而 File 系统存储的是城市列表的 json 数据。

第五章 各个功能模块的具体实现

5.1 功能导航页面的实现

功能导航页作为程序的首页，负责引导用户前往其他功能模块。他由 HomeActivity 及其一系列模块组成。其中负责界面 UI 显示的有 fragment_home.xml,fragment_arround.xml,fragment_life.xml,fragment_personal。其中 fragment_home.xml 只包含一个 Tab 和一个 Pager。他们组成了一个 fragment 的容器，继而形成了当前的布局。而另外三个布局文件，则真正负责功能导航。在程序运行的过程中，ArroundFragment 会加载 fragment_arround.xml，LifeFragment 会加载 fragment_life.xml，PersonalFragment 会加载 fragment_personal.xml。而 HomeFragment 则会加载 home_fragment，并将 ArroundFragment，LifeFragment，PersonalFragment 加载到容器当中。其关系如图 5-1 所示。

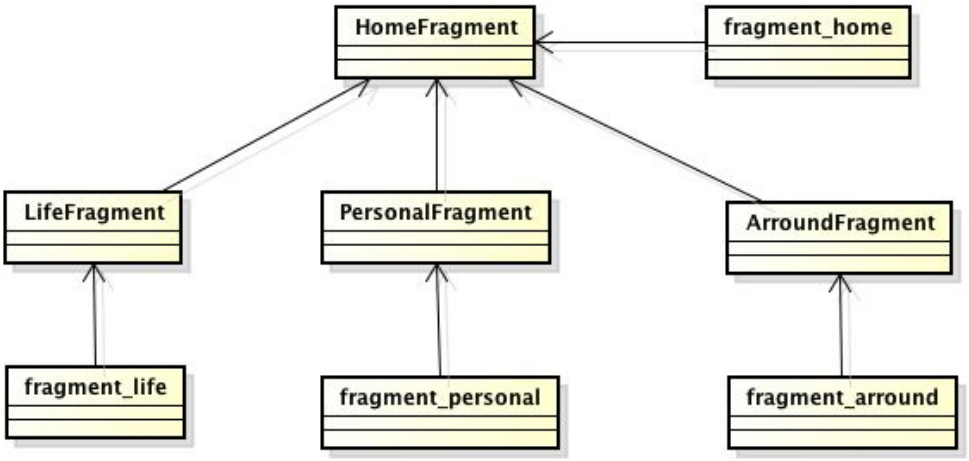

图 5-1 导航模块的界面组成

值得注意的是，HomeFragment 使用的 Tab 是一个第三方的开源框架。该框架简化了原 Tab 与 Pager 之间繁琐的设置，并且提供丰富的 API 设置，帮助开发者方便的设置 Tab 的细节。这些细节包括，Tab 的字体风格，Tab 区块的风格，Tab 滑块的风格。最最重要的是，该框架的使用，使得原先 50 多行的代码被缩减到了 10 行。其代码截图如图 5-2 所示

```
/**
 * 设置放置子页面的PagerView和Tab标签
 */
@AfterViews
public void setPagerAndTab() {
    PagerAndTabAdapter adapter = new PagerAndTabAdapter(this.getFragmentManager());
    pager.setAdapter(adapter);
    adapter.setData(homeHelper.getFragments());
    adapter.notifyDataSetChanged();
    tab.setShouldExpand(true);
    tab.setViewPager(pager);
    tab.setTextColorResource(R.color.app_text_color_light);
    tab.setIndicatorColorResource(R.color.tab_indicator);
    tab.setDividerColorResource(R.color.tab_divider);
}
```

图 5-2 设置 Actionbar

HomeFragmnet 除了负责作为 Fragment 的容器之外，其也要负责用户对当前城市的选择。这段代码被放置在了对 Toolbar 的设置当中。Toolbar 是 Android5.0 最新引入的特性，他的作用是替代原先 ActionBar 的作用。但是他比 ActionBar 更加灵活丰富。

在 ArroundFragment、LifeFragment、PersonalFragmnet 中真正负责导航的逻辑由各个 Model 模块负责，例如说，P2PBusiness 负责根据特定参数，跳转到 P2PActivity。其中的关系如图 5-3 所示。

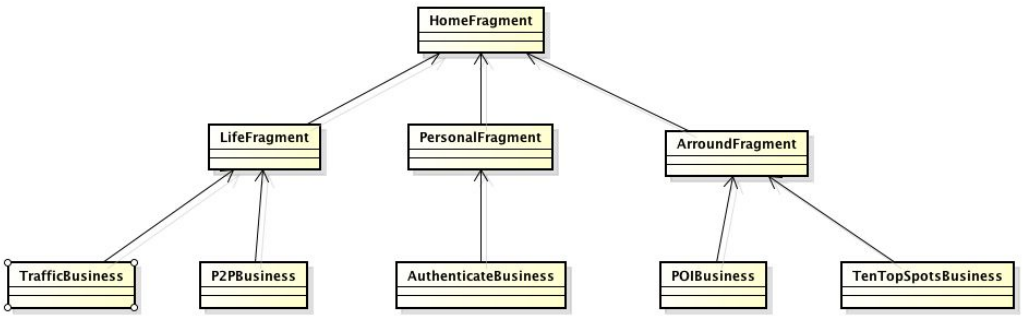

图 5-3 导航模块的实现

功能导航模块对应的 Activity 为 HomeActivity，其主要由 HomeFragment，ArroundFragment、LifeFragment、PersonalFragment 组成。HomeFragment 实现了一个带有 Tabs 和 Pager 的 Fragment，其通过包含另外三个 Fragment 实现功能导航。

而 ArroundFragment 主要负责的是周边信息的功能导航。他的业务逻辑主要由 POIBusiness，LocationBusiness，TopInfoBusiness 实现。它们分别负责 POI 的业务逻辑，定位业务逻辑，公告信息的业务逻辑。

LifeFragment 主要负责生活信息的功能导航。其主要的业务逻辑又由 LocationBusiness， TrafficBusiness， POIBusiness， P2PBusiness 实现。其中

TrafficBusiness 主要负责交通信息查询的业务逻辑，P2PBusiness 主要负责在线交易市场的业务逻辑。

PersonalFragment 主要负责个人信息的功能导航。但他的业务逻辑主要由 UserInfoBusiness ， AuthenticateBusiness ， P2PBusiness ， POIBusiness ， NoticeInfoBusiness 实现。其中 UserInfoBusiness，AuthenticateBusiness 主要负责用户信息和用户身份验证的业务逻辑。NoticeInfoBusiness 则主要负责公关信息的业务逻辑。

5.2 周边信息查询的实现

周边信息查询模块的组成如图 5-4 和 5-5 所示。Fragment 负责对 xml 中的界面渲染，同时调用 POIBusiness 和 POIService 中的方法，实现周边信息的获取和管理。在 POIService 中，他会负责根据当前的地理位置，和用户关注的周边信息关键字返回封装好的 POI 信息数据。这些数据被封装在 POIResultBean 当中，当 Fragment 获取到这些数据之后将会将这些数据渲染到界面当中。

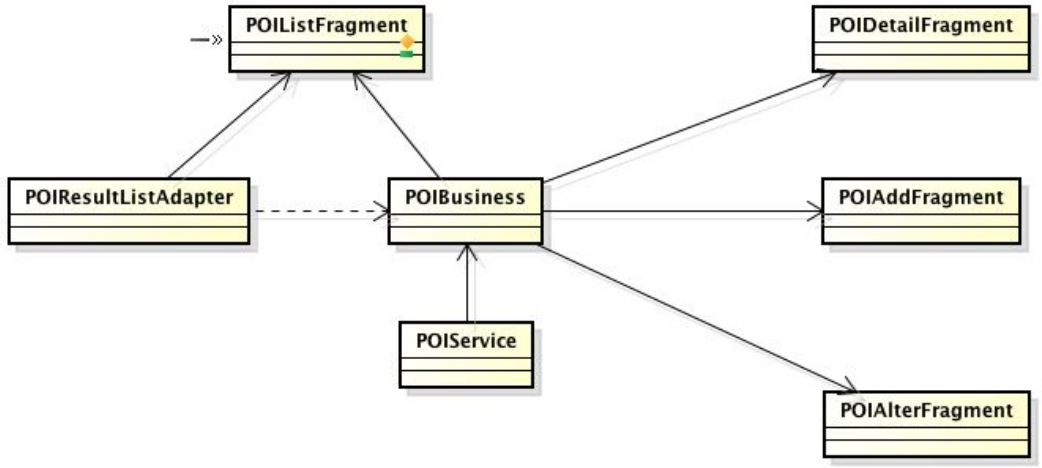

图 5-4 POI 模块的实现

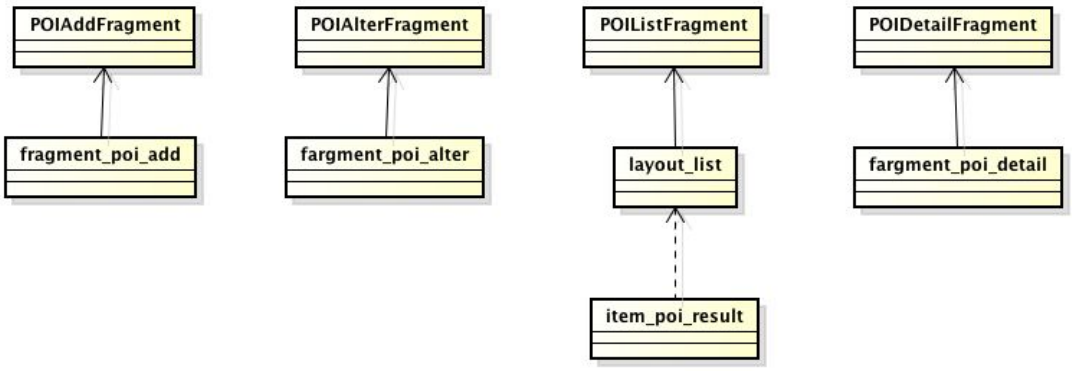

图 5-5 POI 模块的界面组成

值得注意的是 PersonalPOIListFragment 和 POIListFragment，PersonalPOIDetailFragment 和 POIDetailFragment 是继承关系。这两个特殊的 Fragment 除了继承了原本 Fragment 的特性之外，还具有对现有 POI 信息修改的能力。当然，普通用户只能修改自己产生的 POI 数据，而管理员用户可以修改所有自己产生的 POI 数据。正因为有了这两个 Fragment，程序拥有了数据管理和审核的功能。

而在 POIBusiness 的具体实现当中，由于周边信息的数据主要来自两方面。一个是百度提供的 LBS 云服务，另外一个是本应用自产生的数据。因此在整个数据获取的算法当中，其实现逻辑是首先获取本应用自产生的数据，之后获取百度 LBS 云服务的数据。通过这样的算法，强化了自产数据的价值。

POI 模块对应的 Activity 主要是 POIActivity。其主要是由 POIInfoListFragmnet，POIInfoDetailFragment，POIAddFragment，POIAlterFragment，PersonalPOIInfoListFragment，PersonalPOIInfoDetailFragment 实现。

其中 POIInfoListFragment，负责显示 POI 信息的列表。POIInfODetailFragment 负责显示详细的 POI 信息。而 PersonalPOIInfoListFragment 和 PersonalPOIInfoDetailFragment 分别继承于上述两个 Fragment。其显示的是当前登入用户提交的 POI 信息。

POIAddFragment 和 POIAlterFragment 分别负责 POI 信息的添加和管理的业务逻辑。在这些 Fragment 的内部实现当中，分别调用了 POIService，POIPersist，CameraBusiness，POIBusiness 这些模块。通过他们实现服务器的访问，数据的持续化，拍照，POI 的具体业务逻辑。其中图 5-6、图 5-7、图 5-8、图 5-9 分别是周边信息查询的完成效果图。

图 5-6 查询医疗信息的完成效果

图 5-7 查询酒店信息的完成效果

图 5-8 查询家政信息的完成效果

图 5- 9 查询餐饮信息的完成效果

5.3 交通信息查询的实现

交通查询模块的组成可以通过图 5-10 和图 5-11 得知。图中 TrainInfoRequestFragment 负责加载 fragment_train_info_request.xml，BusInfoResultFragment 负责加载 fragment_bus_info_request.xml，PlaneInfoRequestFragment 负责加载 fragment_bus_info_request.xml。而所有的

xxxInfoResultFragment 使用的都是 layout_list.xml。

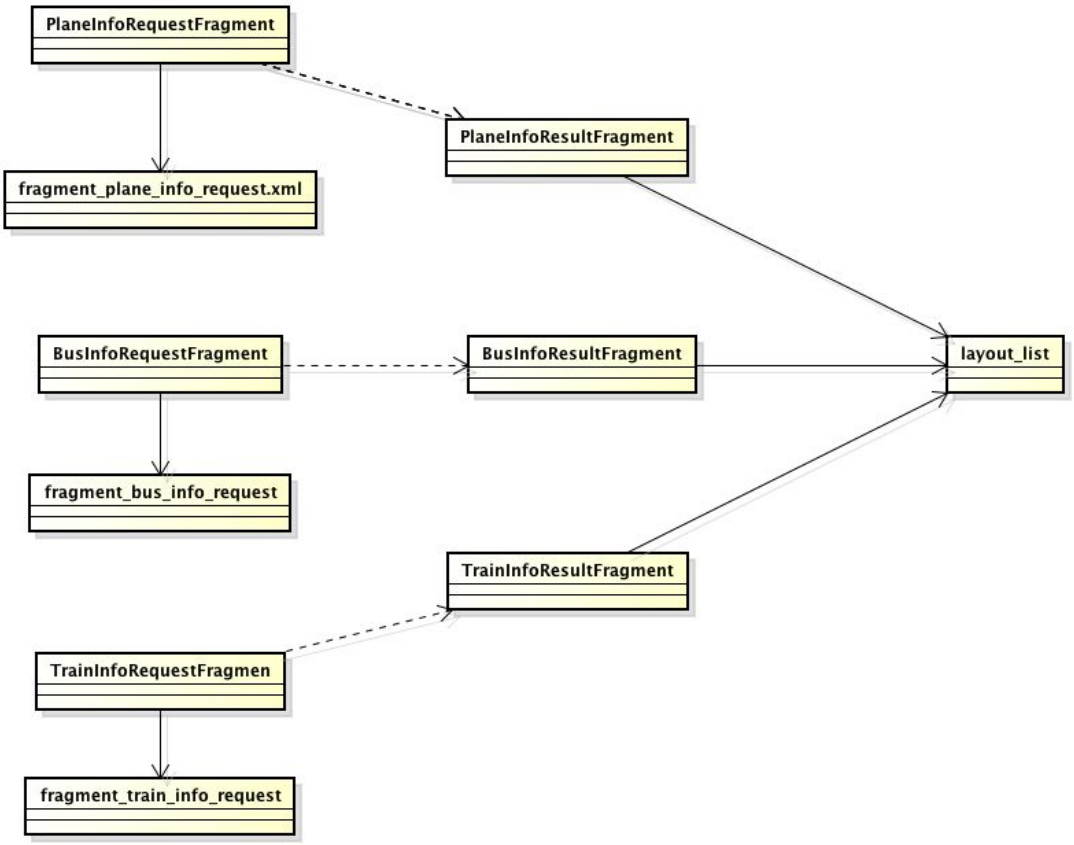

图 5-10 交通查询模块的界面组成

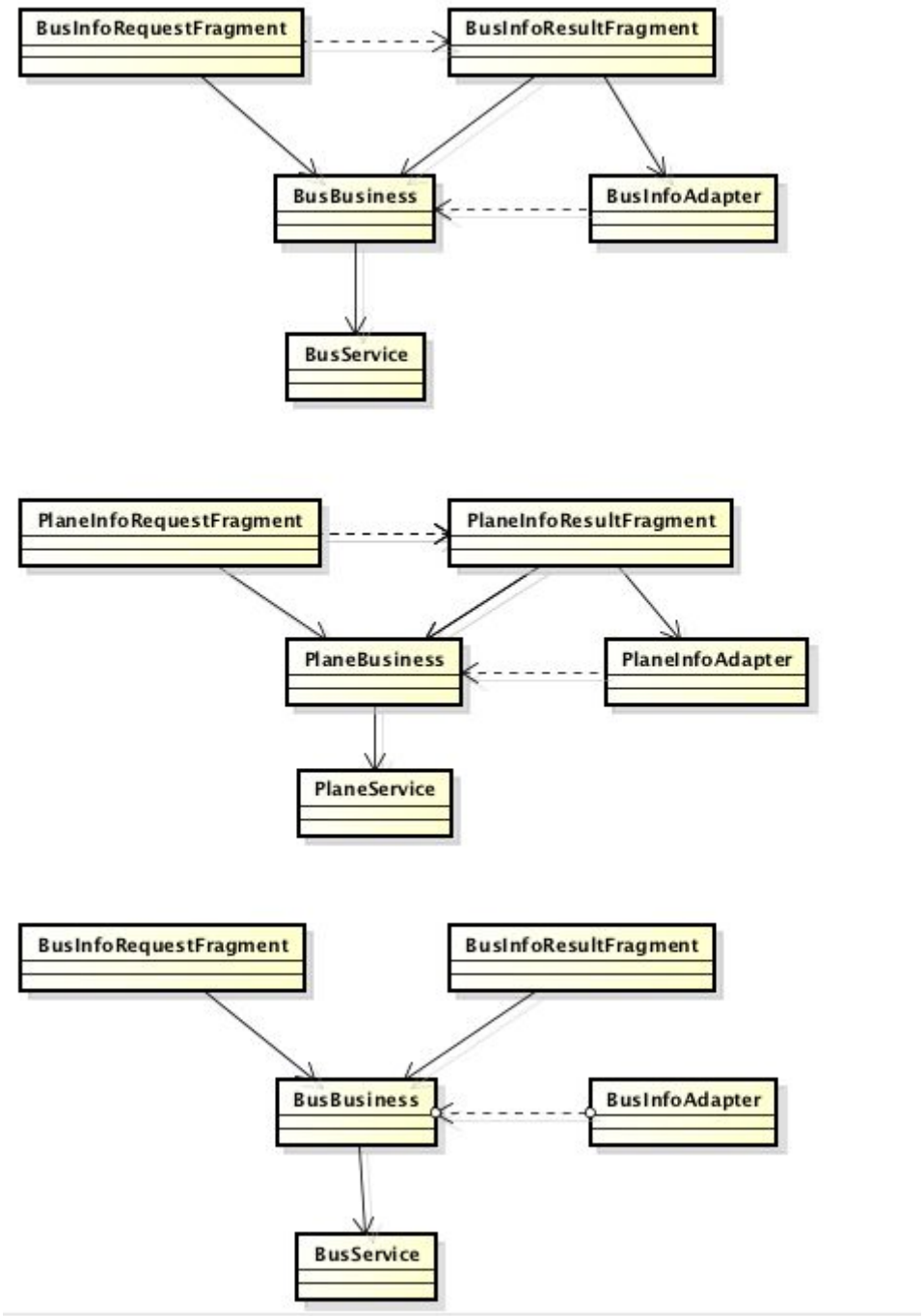

图 5-11 交通查询模块的实现

而在完成界面渲染之后，xxxInfoRequestFragment 通过 BusBusiness，TrainBusiness，PlaneBusiness 封装所要查询的交通数据的请求参数，并跳转到对应的 xxxInfoResultFragment 当中。而在 xxxInfoResultFragment 中则根据传递而来的请求参数数据，BusService，TrainService，PlaneService 请求获取相应的交通信息。其相关代码见图 5-12。

```
/**
 * 跳转到BusInfoResultFragment，查询长途大巴的信息
 *
 * @param startCity 出发城市
 * @param endCity   目的城市
 */
public void toBusInfoResultFragment(FragmentManager fragmentManager, CityBean startCity, CityBean endCity) {
    FragmentTransaction transaction = fragmentManager.beginTransaction();
    transaction.addToBackStack("");
    BusInfoResultFragment fragment = BusInfoResultFragment.generateFragment(startCity,
            endCity);
    transaction.replace(R.id.fragmrnt_container, fragment);
    transaction.commit();
}
```

图 5-12 跳转到交通查询结果的代码

交通查询模块对应的 Activity 主要是 TrafficActivity。他的业务逻辑主要由 PlaneInfoRequestFragment，PlaneInfoResultFragment，TrainInfoRequestFragment，TrainInfoResultFragment，BusInfoRequestFragment，BusInfoResultFragment 实现。其中 xxxRequestFragment 负责的是显示设置参数请求页，xxxResultFrgament 负责的是显示搜索结果页。以 Plane 为例，PlaneInfoRequestFragment 负责的是航班信息搜索参数的设置页面，PlaneInfoResultFragment 负责的是航班信息搜索结果页。

在这些 Fragment 的背后，其业务逻辑分别是由 PlaneService，BusService，TrainService，DateBusiness，LocationBuseinss 实现的。其中 PlaneService，BusService，TrainService 负责与相应的服务器联系，请求和获取数据。DateBusiness 负责选择日期。其中图 5-13 是交通信息查询的完成效果图

图 5-13 查询交通信息的完成效果

5.4 自由交易市场的实现

在线交易市场的组成如图 5-14 和 5-15 所示。其中 P2PListFragment 负责加载 fragment_p2p_list.xml，P2PDetailFragemt 负责加载 fragment_p2p_detail.xml，P2PAddFragment 负责加载 fragment_p2p_add.xml，P2PAlterFragment 负责加载 fragment_p2p_alter.xml。在各个 Fragment 加载好数据之后，将会调用 P2PService 和 P2PBusiness 获取在线交易市场的数据，并对这些数据进行管理。

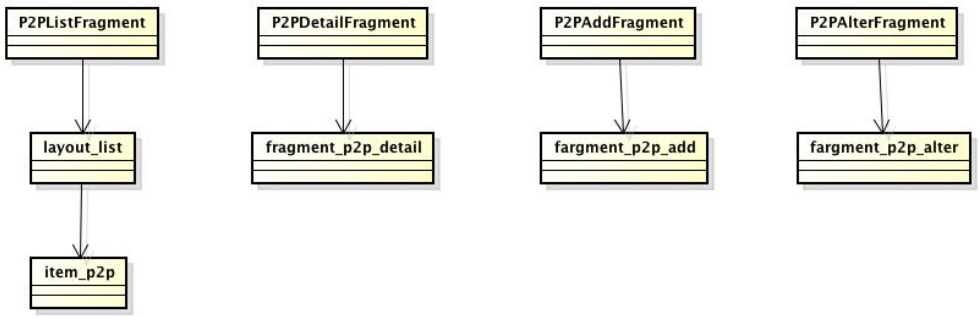

图 5-14 P2P 模块的界面组成

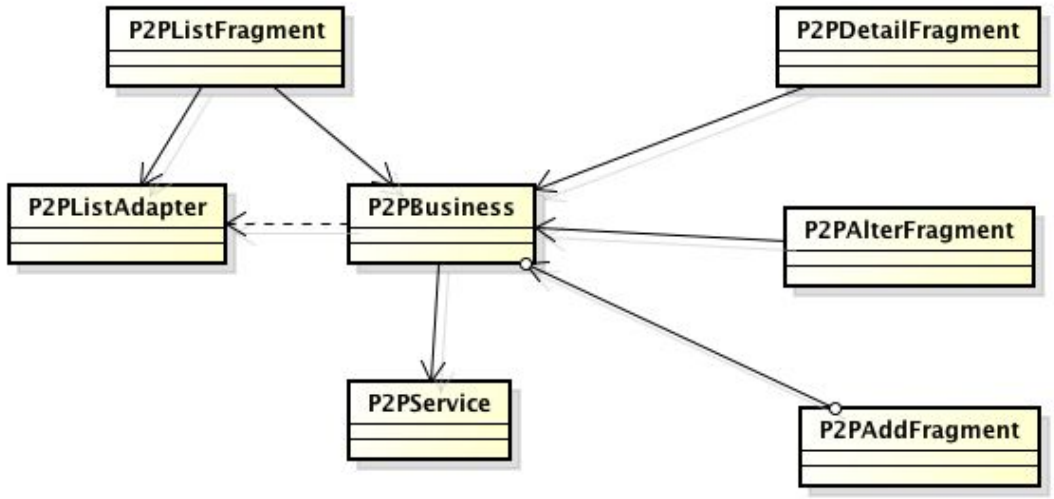

图 5-15 P2P 模块的功能实现

值得注意的是，PersonalP2PListFragment 和 P2PListFragment，PersonalP2PDetailFragment 和 P2PDetialFragment 之间是继承关系。他们在原有父类的基础上增加了对商品数据修改的功能，正是因为这些功能的存在，用户才可以对上述的数据进行修改和管理。当然如果当前的用户是管理员用户，他可以对所有的商品数据进行修改。

在 P2PService 中，将会负责访问服务器，获取相关的商品数据。服务器会根据相应的请求参数返回封装好的 json 数据。之后 P2PService 将会对这些 Json

数据进行解析，并将其封装成为 P2PItemBean 返回给响应的 Fragment。其中的解析数据的代码见图 5-16

```
/**
 * 解析P2P items的json数据
 */
private Task<List<P2PItemBean>> parseP2PItems(final String p2pItemsJson) {
    return Task.callInBackground(() -> {
            Gson gson = new Gson();
            List<P2PItemJsonBean> p2pItemJsonBeans = gson.fromJson(p2pItemsJson, new TypeToken<List<P2PItemJsonBean>>() {
            }.getType());
            List<P2PItemBean> p2pItemBeans = new ArrayList<>();
            for (P2PItemJsonBean p2pItemJsonBean : p2pItemJsonBeans) {
                p2pItemBeans.add(convertToP2PItemBean(p2pItemJsonBean));
            }
            return p2pItemBeans;
    });
}
```

图 5-16 解析商品数据

在 P2PBusiness 中，其主要负责在各个 P2P 页面中相互跳转。确保这样的页面跳转能够正常的运行。并且使得原本重复出现的代码被完好的封装，减少代码的冗余程度。其相关的代码可见图 5-17

```
/**
 * 前往P2P详情页
 */
public void toP2PDetailFragment(FragmentTransaction transaction, P2PItemBean p2pItemBean) {
    P2PDetailFragment fragment = new P2PDetailFragment.Builder()
            .setP2PItemBean(p2pItemBean)
            .create();
    transaction.add(android.R.id.content, fragment);
    transaction.commit();
}
```

图 5-17 跳转到商品详情页的代码

P2P 模块对应的 Activity 主要是 P2PActivity，其主要是由 P2PInfoListFragmnet，P2PInfoDetailFragment，P2PAddFragment，P2PAlterFragment，PersonalP2PInfoListFragment，PersonalP2PInfoDetailFragment 实现。

其中 P2PInfoListFragment，负责显示 P2P 信息的列表。P2PInfODetailFragment 负责显示详细的 P2P 信息。而 PersonalP2PInfoListFragment 和 PersonalP2PInfoDetailFragment 分别继承了上述两个 Fragment。其显示的是当前登入用户提交的 P2P 信息。

P2PAddFragment 和 P2PAlterFragment 分别负责 P2P 信息的添加和管理的业务逻辑。在这些 Fragment 的内部实现当中，分别调用了 P2PService，P2PPersist，CameraBusiness，P2PBusiness 这些模块。通过他们实现服务器的访问，数据的持续化，拍照，P2P 的具体业务逻辑。其中图 5-18、5-19、5-20 是自由交易市场模块完成的效果图。

图 5-18 查询商品信息的完成效果

图 5-19 发布新的商品的完成效果

图 5-20 信息审查管理的完成效果

5.5 选择城市和 GPS 定位的实现

当前选择城市模块的组成如图 5-21 和图 5-22 所示。Fragemnt 首先会加载 fragment_choose_city.xml，之后会通过 ChooseCityhelper 获取城市列表，并将其数据传递给 Adapter，Adapter 会将城市数据进行渲染。

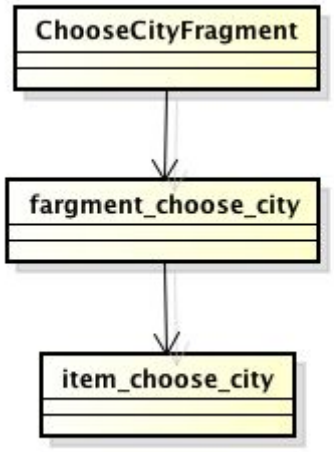

图 5-21 选择城市模块的界面组成

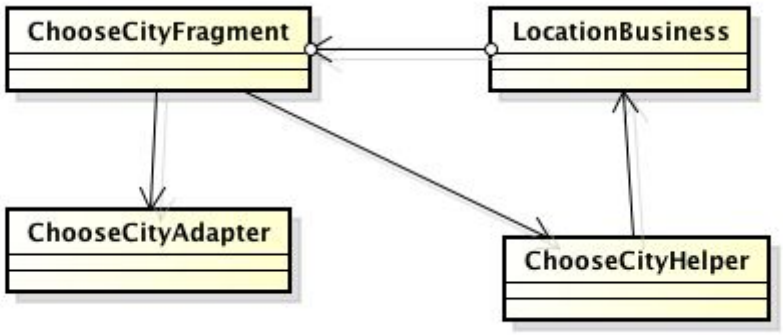

图 5-22 选择城市模块的实现

值得注意的是，ChooseCityHelper 所获取的数据是本地离线数据。由于城市列表数据是一个非常稳定数据。因此在程序设计之初将其写死在程序当中。通过 LocationPersist 获取封装城市信息的 json 数据。之后再使用 Gson 将数据解析成 CityBean 数据，其相关的代码见图 5-23。

```
/**
 * 获取以首字母分组排序的全部城市
 * <p/>
 * 该方法会为ChooseCity页面提供城市列表
 * 返回的map，key是城市的首字母，value是该首字母对应的城市的list集合
 */
public Map<String, List<CityBean>> getAllCity() {
    Map<String, List<CityBean>> cities = new ListOrderedMap<>();

    String citiesJson = locationPersist.getAllCitiesGroupByFirstChar();
    Gson gson = new Gson();
    List<CitiesGroupByFirstCharJsonBean> citiesGroupByFirstCharJsonBeans = gson.fromJson(citiesJson,
            new TypeToken<List<CitiesGroupByFirstCharJsonBean>>() {
            }.getType());
    for (CitiesGroupByFirstCharJsonBean citiesGroupByFirstCharJsonBean : citiesGroupByFirstCharJsonBeans) {
        List<CityBean> realCities = new ArrayList<>();
        for (CityJsonBean cityJsonBean : citiesGroupByFirstCharJsonBean.getCities()) {
            realCities.add(CityBean.generateCity(cityJsonBean.getName()));
        }
        if (realCities.size() > 0) {
            cities.put(citiesGroupByFirstCharJsonBean.getSection(), realCities);
        }
    }
    return cities;
}
```

图 5-23 获取城市数据的代码

而在程序的设计时，为了方便用户选择城市。城市选择界面采用了按照拼音首字母排列的方式。因此这给原本简单的城市选择列表的时间增加了小小的难度。为了实现该功能，我们对传递给 Adapter 的数据进行了处理。其中增加了 style 数据变量，程序将会根据该变量判断，该数据是城市数据还是字母标签数据。其代码如图 5-24 所示

```
case SECTION:
    itemChildViews.cityItem.setVisibility(View.GONE);
    itemChildViews.sectionItem.setVisibility(View.VISIBLE);
    itemChildViews.findLocationItem.setVisibility(View.GONE);
    sectionItemChildViews.titleTv.setText(((SectionItemBean) itemData).title);
    break;
case CITY:
    itemChildViews.cityItem.setVisibility(View.VISIBLE);
    itemChildViews.sectionItem.setVisibility(View.GONE);
    itemChildViews.findLocationItem.setVisibility(View.GONE);
    cityItemChildViews.cityNameTv.setText(((CityItemBean) itemData).cityName);
    break;
case FIND_LOCATION:
    itemChildViews.cityItem.setVisibility(View.GONE);
    itemChildViews.sectionItem.setVisibility(View.GONE);
    itemChildViews.findLocationItem.setVisibility(View.VISIBLE);
    findLoactionItemChildlViews.cityNameTv.setText(((FindLoactionItemBean) itemData).cityName);
    if (((FindLoactionItemBean) itemData).cityName.equals("正在定位……")) {
        findLoactionItemChildlViews.findLocationBn.setBackground(context.getResources().getDrawable(android.R.color.holo_orange_light));
    } else {
        findLoactionItemChildlViews.findLocationBn.setBackground(context.getResources().getDrawable(android.R.color.holo_red_light));
    }
    findLoactionItemChildlViews.findLocationBn.setVisibility(View.VISIBLE);
    findLoactionItemChildlViews.findLocationBn.setOnClickListener((v) → {
            callback.onFindCurrentLoactionCity();
    });
    break;
```

图 5-24 渲染城市数据的代码

另外在选择城市页面当中，当成功选择了当前城市之后，会触发设置的回调接口，通知相应的组件，已经成功的选择了城市。该回调接口的设置是在 Builder 中进行的，这样的设计确保了开启 ChooseCityFragment 时，必定设置了 Callback。继而避免了不应有的程序 Bug。

另外需要关注的是定位功能的实现。本项目的定位功能采用了 baidu 提供的 SDK，通过调用百度 SDK 的 API 获取当前设备所在的地理位置信息。使用该 SDK 的方法是，首先在 Applicatiion 中 onCreate 方法中初始化百度 SDK 的对象。之后使用 LocationClient 获取当前的位置信息，其相关的代码见 5-25。另外图 5-26 是城市选择模块的完成效果图。

```
/**
 * 通过定位获取当前设备所在位置。
 * <p/>
 * 在获取到当前设备所在位置后，会发送GetLocationEvent，以通知先关组件
 */
public Task<CityBean> findCurrentLocationCity() {
    final Task<CityBean>.TaskCompletionSource taskCompletionSource = Task.create();

    final LocationClient locationClient = new LocationClient(context);

    LocationClientOption locationClientOption = new LocationClientOption();
    locationClientOption.setIsNeedAddress(true);
    locationClientOption.setScanSpan(10001);
    locationClientOption.setLocationMode(LocationClientOption.LocationMode.Battery_Saving);
    locationClient.setLocOption(locationClientOption);

    BDLocationListener wrapperListener = (bdLocation) → {
            if (bdLocation != null && bdLocation.getCity() != null && bdLocation.getCityCode() != null
                    && !bdLocation.getCity().equals("") && !bdLocation.getCity().equals("null")
                    && !bdLocation.getCityCode().equals("") && !bdLocation.getCityCode().equals("null")) {

                Log.d(TAG, "get current location");
                locationClient.stop();
                CityBean currentLocationCity = CityBean.generateCity(bdLocation.getCity());
                taskCompletionSource.setResult(currentLocationCity);
                getEventBus().post(GetCurrentLocationCityEvent.generateEvent(currentLocationCity));
            }
    };
    locationClient.registerLocationListener(wrapperListener);

    locationClient.start();
    return taskCompletionSource.getTask();
}
```

图 5-25 使用 GPS 定位的代码

图 5-26 定位到当前城市的完成效果

5.6 日期选择的实现

选择日期模块的组成如图 5-27 所示，其组成部分相对简单，只有 DateBusiness 和 DatePcikerFragment。其中 DatePickerFragment 负责将日期渲染日期选择界面，并提供一个回调接口供调用者获取用户选择的日期。

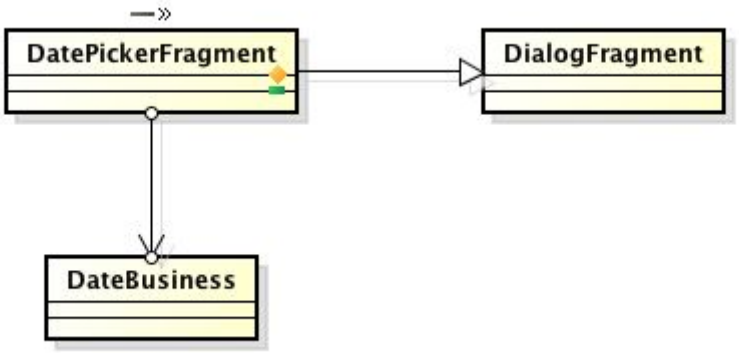

图 5-27 日期选择模块的实现

日期选择实际上采用的是系统默认的日期选择组件。但是区别于以往的是，在程序中没有直接使用 Dialog，而是使用 DialogFragment 封装了该模块。这样做的目的是为实现选择时间的回调接口，同时今后如果需要使用自定义的日期选择模块，则可以在不影响其他代码的情况下，无缝切换。其相关代码如图 5-28 所示。

```
@Override
public Dialog onCreateDialog(Bundle savedInstanceState) {
    Calendar baseCalendar = Calendar.getInstance();
    baseCalendar.setTime(baseDate);
    int year = baseCalendar.get(Calendar.YEAR);
    int month = baseCalendar.get(Calendar.MONTH);
    int day = baseCalendar.get(Calendar.DAY_OF_MONTH);
    return new DatePickerDialog(getActivity(), this, year, month, day);
}
```

```
@Override
public void onDateSet(DatePicker view, int year, int monthOfYear, int dayOfMonth) {
    Calendar calendar = Calendar.getInstance();
    calendar.set(year, monthOfYear, dayOfMonth);
    Date date = calendar.getTime();
    callback.onDateChoosed(date);
}
```

图 5-28 日期选择模块的核心代码

另外，同样为了实现日期选择的异步回调。日期选择模块和城市选择模块一样使用了回调接口的方式，实现日期选择结果的返回。并且回调接口被设计必须要通过 Builder 设置，通过这样的方法，避免了因为使用 DatePickerFragment 时，没有设置 Callback 而导致的 Bug。另外在 DateBusiness 则使用 Bolts 框架，对日期选择的操作进一步做了封装，使得调用者可以非常优雅的卸除日期选择程序的代码。

另外，在程序中提供了 DateUtil 程序，帮助使用者处理日期数据的格式转换的工作。其可以方便的根据日期格式的类型将数据转换成 Date 对象，或者将 Date 对象，转换成制定的日期格式。同时他还提供了获取制定日期格式的当前日期。

5.7 用户身份验证的实现

如图 5-29,5-30 示，用户身份验证模块由以下部分组成。AuthenticateFragment 负责加载 fragment_authenicate.xml，LoginFragment 负责加载 fragment_login.xml，SigninFragment 负责加载 fragment_signin.xml。其中 fragment_authenticate 的布局文件由一个 tab 和一个 pager 组成。其实现和 HomeFragment 一样。其目的是作为一个容器存在。

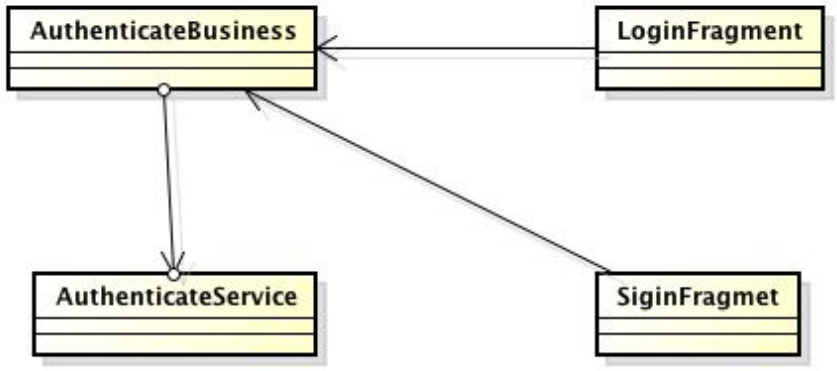

图 5-29 身份验证模块的实现

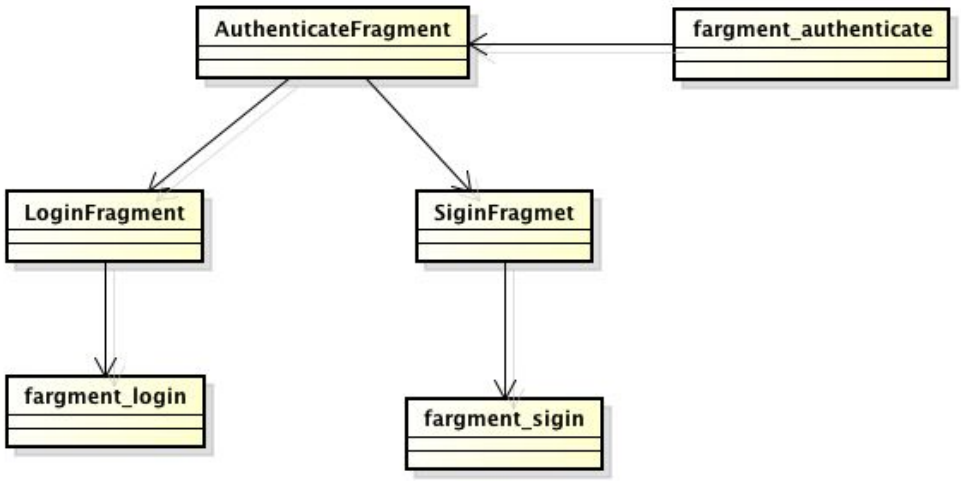

图 5-30 身份验证模块的界面组成

在 LoginFragment 的实现当中，其主要的业务逻辑便是加载 LoginFragment 和 SigninFragment。与此同时，设置 Pager 和 Tab。由于使用了 PagerSlidingTabStrip。因此相关的代码变的相对简单和简洁。

在 LoginFragment 中，其通过与 AuthenticateService 的通信，来验证当前输入的用户名和密码是否正确。如果正确，则会通过回调接口的方式，将用户数据 UserInfoBean 返回给相关代码。在这里，回调接口的设计依旧使用了 Builder 的方式，目的是为了减少在使用过程由于没有设置 Callback 导致的 Bug，其相关的代码如图 5-31。

```
@Click(R.id.login_bn)
void login() {
    String userName = userNameEt.getText().toString();
    String passowrd = passwordEt.getText().toString();

    dialogBusiness.showDialog(getFragmentManager(), new DialogBusiness.ProgressDialogBuilder().create(), "login");
    authenticationBusiness.login(userName, passowrd).onSuccess((task) -> {
            dialogBusiness.hideDialog("login");
            callback.onLoginSuccess(task.getResult());
            return null;
    });
}
```

图 5-31 登录的代码

在 SiginFragment 中，其通过与 AuthenticateService 的通信，根据输入的用户名和密码注册新的用户。如果注册成功，则会通过回调接口的方式，将用户数据 UserInfoBean 返回给相关代码。在这里，回调接口的设计依旧使用了 Builder 的方式，目的是为了减少在使用过程由于没有设置 Callback 导致的 Bug。

在 AuthenticateBusiness 中，程序提供了判断当前是否处于登入状态，如果除了查询登入状态还提供了获取当前登入用户的用户信息的接口。值得注意的是这些信息是通过 persisit 模块持续缓存在程序当中的。图 5-32 是用户验证模块完成的效果图。

图 5-32 用户身份验证的完成效果

5.8 公告信息的实现

图 5-33 是应用公告信息的实现。其实现方式非常的简单，由 NotifeInfoFragmet 加载 fragment_notife_info.xml 。然后 Frgament 再通过 NotifeInfoBusiness 获取相关的数据，并将数据渲染到界面当中去。其中 NotifeInfoBusiness 所获取的数据来自 Persist 的离线数据，该数据是被缓存在应

用当中的。

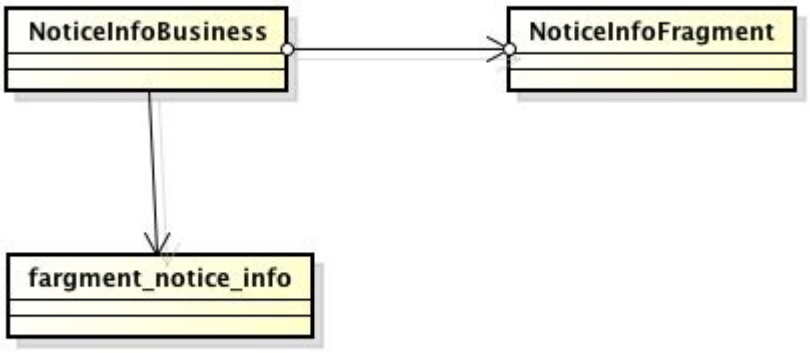

图 5-33 公告信息模块的实现

如图 5-34 所示，十大旅游景点由如下模块组成。其中 TenTopSpotsFragmet 负责加载 layout_list.xml，之后通过 TenTopSpotsBusiness 获取相关的数据。并将数据传递给 Adapter，由其渲染到界面当中。其中 TenSpotsBusines 获取的数据是由 Persist 离线数据获得的，中间经过了 json 解析的步骤。

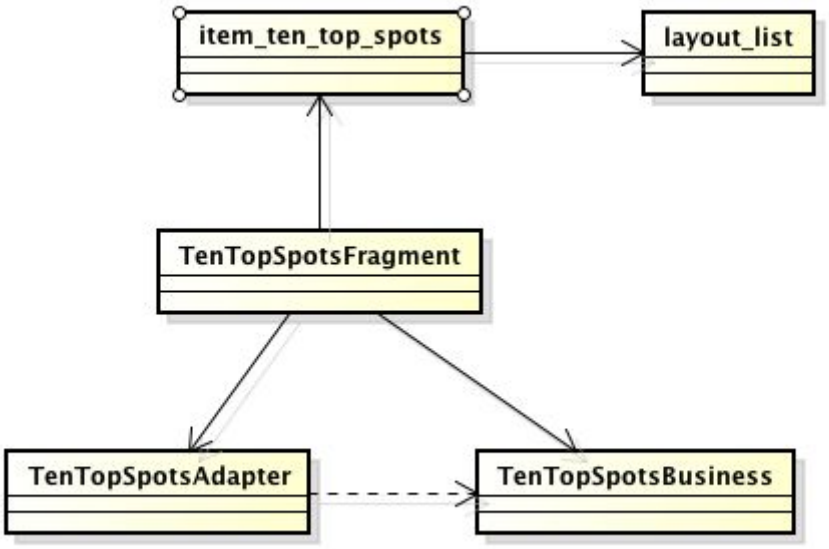

图 5-34 十大旅游城市模块的实现

第六章 测试

6.1 测试方法

本项目当中，将主要使用黑盒测试的方法对项目进行测试。黑盒测试是软件测试中普遍采用的一种方法。通过该方法对项目进行测试时，并不需要了解项目的内部结构和具体的实现方式。测试者所需要做的事情就是，根据测试用例的指导，对项目进行操作。并根据运行的结果，以及项目的文档判断该测试是否符合软件的要求。

6.2 测试用例

以下表格是各个功能模块的测试用例。其中表 6-1 负责对城市选择模块进行测试。表 6-2、表 6-4、表 6-5、表 6-10 分别是对周边信息查询模块的功能测试。表 6-3 是对交通信息查询模块的测试。表 6-6、表 6-7、表 6-9 是对现在交易市场模块的测试。表 6-8 是对用户身份验证模块的测试。

表 6-1　定位到当前城市测试用例

用例名称	定位到当前城市
用例标识	TEST_CASE_001
测试要点	通过 gps 定位和城市列表选择当前城市
操作步骤	1. 打开程序，进入首页 2.　点击 ActionBar 的城市按钮，跳转到选择城市页面 3. 选择目标城市，跳转回首页。
期望结果	成功选择了当前所在城市
实际结果	成功选择了当前所在城市

表 6-2　查询医疗信息测试用例

用例名称	查询医疗信息
用例标识	TEST_CASE_002
测试要点	更具当前选择的城市，查询周边的医疗信息
操作步骤	1. 打开程序，进入首页的周边信息页 2.点击周边查询的医疗按钮 3.跳转到医疗信息页面，显示周边的医疗信息
期望结果	能够查询到周边的医疗信息

实际结果	能够查询到周边的医疗信息

表 6-3　查询交通信息测试用例

用例名称	查询交通信息
用例标识	TEST_CASE_003
测试要点	查询到交通信息的实时信息
操作步骤	1. 打开程序，进入首页的生活信息页 2.　点击交通查询的飞机，火车，大巴按钮 3. 跳转到请求数据设置页，设置请求参数，点击查询 4. 跳转到交通信息查询结果页
期望结果	能够航班、火车、长途大巴的实时信息
实际结果	能够航班、火车、长途大巴的实时信息

表 6-4　查询酒店信息测试用例

用例名称	查询酒店信息
用例标识	TEST_CASE_004
测试要点	查询到周边的酒店信息
操作步骤	1. 打开程序，进入首页的周边信息页 2. 点击周边查询的酒店按钮 3. 跳转到酒店信息页，显示周边的酒店信息
期望结果	查看到当前选择城市的酒店信息
实际结果	查看到当前选择城市的酒店信息

表 6-5　查询家政信息测试用例

用例名称	查询家政信息
用例标识	TEST_CASE_005
测试要点	查询到月嫂、清洁、搬家公的家政信息
操作步骤	1. 打开程序，进入首页的生活信息页 2. 点击家政服务的搬家，月嫂，保洁按钮 3. 跳转到家政信息页，示周边的家政信息

期望结果	能够查询到当前选择城市的月嫂公司、清洁公司、搬家公司的信息
实际结果	能够查询到当前选择城市的月嫂公司、清洁公司、搬家公司的信息

表 6-6　查看在线交易市场测试用例

用例名称	查看在线交易市场
用例标识	TEST_CASE_006
测试要点	查询在线交易市场的商品信息
操作步骤	1. 打开程序，进入首页的生活信息页 2. 点击蔬菜家禽，电子商品，衣服鞋包按钮 3. 跳转到商品列表列表页面
期望结果	能够查询到交易市场的商品信息列表和商品详情
实际结果	能够查询到交易市场的商品信息列表和商品详情

表 6-7　发布新的商品测试用例

用例名称	发布新的商品
用例标识	TEST_CASE_007
测试要点	新增自己的商品信息
操作步骤	1. 打开程序，进入任意商品列表页 2. 点击 ActionBar 的增加按钮 3. 跳转到新增商品页面，填写商品信息 4. 点击发布，成功后可以在商品列表页面找到最新的商品
期望结果	新增自己的商品信息，并可以在商品列表中查询到自己发布的商品
实际结果	新增自己的商品信息，并可以在商品列表中查询到自己发布的商品

表 6-8　用户身份验证测试用例

用例名称	用户身份验证
用例标识	TEST_CASE_08

测试要点	可以成功的登录或者注册
操作步骤	1. 打开程序，进入首页的个人信息页 2. 点击头像图标进行登录 3. 跳转到身份验证页面，输入信息并提交 4. 成功登录后跳转回首页，显示登录的用户信息
期望结果	输入正确的账户和密码后完成用户验证
实际结果	输入正确的账户和密码后完成用户验证

表 6-9　信息审核和管理测试用例

用例名称	信息审核和管理
用例标识	TEST_CASE_09
测试要点	对发布的信息进行修改或者删除
操作步骤	1. 打开程序，以管理员身份登录程序 2. 跳转到个人信息页，点击我发布的商品，我发布的信息页面 3. 跳转到信息列表，长安需要管理的信息，跳转到信息修改页面。 4. 修改或者删除该信息
期望结果	成功的修改或删除发布的信息
实际结果	成功的修改或删除发布的信息

表 6-10　查询餐饮信息测试用例

用例名称	查询餐饮信息
用例标识	TEST_CASE_010
测试要点	对发布的信息进行修改或者删除
操作步骤	1. 打开程序，进入首页的周边信息页 2. 点击周边查询的美食按钮 3. 跳转到美食信息页，显示周边的餐饮信息
期望结果	根据当前选择的城市，查询到周边的餐饮信息
实际结果	根据当前选择的城市，查询到周边的餐饮信息

结论

在本次的毕业设计当中，完成了基于 Android 的 114 生活助手系统的设计与实现。并通过项目的开发，尝试寻找一种快速开发的开发方式。在最终阶段完成了课题任务书当中的所有目标和需求，完成并实现了生活助手系统所具备的周边信息查询，交通信息查询，在线交易平台等功能。根据完成的项目你可以查询到航班、火车、长途大巴的实时交通信息。另外你还可以根据自己选择的城市查询到相关的周边信息。这些周边信息包括：酒店信息、餐饮信息、医疗信息、家政信息。

在项目当中，我尝试使用了 Android 开发的最新前沿技术。这些技术有的来自 Andorid 官方，有的来自开源社区。经过这次实际项目的考验，这些原本只是文章中提及到的技术大多被证明是非常有效的。当然其中也有少许的技术并没有如项目开始时预期的那样完美，其中仍然有许多另人遗憾的地方存在。其中值得作为总结的内容如下：

1. 使用 AndroidAnnotation 这样的开发框架，将会大大降低重复的代码，让开发变的无比的顺畅。但是随着项目的无限增长，该框架的使用可能依旧会不可避免的造成代码的冗余。另外 AndroidAnnotation 项目作为 Android 开发的一个整体框架来说，其可能存在的最大的问题在于，其和其他开源框架的配合使用。在本项目的开发当中，已经不仅一次的遇到了 AndoridAnnotation 和其他开源项目的冲突问题。

2. 使用 Fragment 替换原来的 Activity 的部分功能，使得开发变的更加灵活高效。但是同时这也引入了新的问题，而最主要的问题是在于在 Andorid 的官方文档当中，对于 Fragment 的具体使用方式似乎还是相对限制的比较死板的，这造成了在很多的开发当中，开发者一层不变的按照官方文档的思路使用 Fragment。但是根据本次项目的实际经验来看，似乎这样的做法是不值得赞同的。Fragment 应该被更加灵活和丰富的使用。

3. 使用 Bolts 替代 Callback 和 AsyncTask，使得原先的代码变的更加清晰和容易阅读。该框架可以说是本项目中最值得注意的地方。Bolts 项目作为一个刚刚发布的项目，其知名程度和影响程度在开发者中还不是非常的普及，其带来的效率的提升和优雅的代码还没有被人所知。然而此次的项目为其价值做了最好的证明。

4. Picasso 框架的使用使得原先负责的图处理工作变的异常的简单。但是非常遗憾的事，Picasso 框架的稳定性依旧不是很好，在本项目的开发当中，发现 Picasso 项目对于部分的 Android 机型匹配支持的并不是非常的完美。虽然这种现象可以通过降低处理的图的大小来解决。但是其依旧作为问题而存在。

5. EventBus 项目，在本项目当中，被主要作为 Activity 间的通信工具使用。

的确正如 EventBus 开发者介绍的那样，EventBus 的使用的确解决了过去 Android 项目模块间沟通的问题。但是非常可惜的是，EventBus 并没有使用 java 的注解特性，这使得 EventBus 的语法在程序当中显得有点突兀。

6. 对项目架构的划分使得项目的架构变的非常的清晰。多层的架构划分，使得 Android 很好的实现了 MVC 的架构逻辑。将 UI 代码放置在 xml 布局文件当中。Model 层主要负责正真的业务逻辑。而 Activity 和 Fragment 作为 Control 层出现。这样的设计，要远远比单独使用 Activity 的要好很多。最最重要的是，这样的划分，也使得项目本身变的可以测试。当然，不能说这样的设计就是非常完美的。由于 Android 项目的特殊性，其层次的划分还不能被非常清晰的划分出来，在各个层次之间依旧存在很多交集的地方。

7. 由于课题的研究时间和经历的现实，本项目在单元测试上面所耗费的精力几乎可以为零。整个项目的开发都是朝着不断向前的步骤进行的，没有按照最佳实践的推荐对项目本身进行单元测试。同时在项目的过程当中，重构和代码回顾的过程也屈指可数，因此虽然在众多开源框架的帮助下，项目已经变的异常的简洁明了。但是其中依旧存在很多糟糕的设计需要改正。最后，由于设备的限制，项目最终的运行测试只有在魅蓝 note 和 google nexus7 上面进行。因此项目本项目的稳定性和适配性能另人担忧。

参考文献

[1] Hancock, Gregory,182 Most Asked Questions on Android: What You Need to Know,In Success Secrets.[S.l.] : Emereo Publishing,2014.

[2] Samuelson, Pamela1 Copyrightability of Java APIs Revisited, Communications of the ACM, 2015.

[3] Sverdlove, Harry, Feature: The Java vulnerability landscape, In Network Security:2014, 2014(4):9-14.

[4] Karakoidas, Vassilios.A type-safe embedding of SQL into Java using the extensible compiler framework J, In Computer Languages. Systems & Structures, 2015, 41:1-20.

[5] 李寅,范明钰,王光卫.基于反编译的 Android 平台恶意代码静态分析.计算机系统应用 (Computer Systems＆ Applications), 2012, 11: 187-189.

[6] 陈融.Android 安全研究综述.计算机应用与软件 (Computer Applications and Software), 2012, 29 (10): 205-210.

[7] 彭艳,杨欧.Android 平台的数据存储技术.计算机系统应用 (Computer Systems＆ Applications), 2012, 05: 192-194.

[8] 李维勇.Android 平台 Mash-Up 模式解析.中原工学院学报 (Journal of Zhongyuan Institute of Technology), 2012, 23 (01): 49-52.

[9] 余霖,任向林.极限编程及其 Android 开发应用.电脑编程技巧与维护 (Computer Programming Skills ＆ Maintenance), 2012, 18: 23-23.

[10] 徐尤华,熊传玉.Android 应用的反编译.电脑与信息技术 (Computer and Information Technology), 2012, 20 (01): 50-51.

[11] 杨帆,赵东东.基于 Android 平台的 WiFi 定位.电子测量技术 (Electronic Measurement Technology), 2012, 35 (09): 116-119.

[12] 邹燕飞,胡泽江.基于 Android 平台的 GDT 应用,西安文理学院学报：自然科学版 (Journal of Xi‘an University of Arts ＆ Science：Natural Science Edition), 2012, 15 (02): 75-78.

[13] DiMarzio.Android : A Programmer's Guide,In McGraw-Hill Professional.New York: McGraw-Hill, 2008.

附录文献翻译

1 英文原文

Android applications are written in the Java programming language. The compiled Java code — along with any data and resource files required by the application — is bundled by the aapt tool into an Android package, an archive file marked by an .apk suffix. This file is the vehicle for distributing the application and installing it on mobile devices; it's the file users download to their devices. All the code in a single .apk file is considered to be one application.

In many ways, each Android application lives in its own world:

1. By default, every application runs in its own Linux process. Android starts the process when any of the application's code needs to be executed, and shuts down the process when it's no longer needed and system resources are required by other applications.

2. Each process has its own virtual machine (VM), so application code runs in isolation from the code of all other applications.

3. By default, each application is assigned a unique Linux user ID. Permissions are set so that the application's files are visible only to that user and only to the application itself — although there are ways to export them to other applications as well.

It's possible to arrange for two applications to share the same user ID, in which case they will be able to see each other's files. To conserve system resources, applications with the same ID can also arrange to run in the same Linux process, sharing the same VM.

Application Components

A central feature of Android is that one application can make use of elements of other applications (provided those applications permit it). For example, if your application needs to display a scrolling list of images and another application has developed a suitable scroller and made it available to others, you can call upon that scroller to do the work, rather than develop your own. Your application doesn't incorporate the code of the other application or link to it. Rather, it simply starts up that piece of the other application when the need arises.

For this to work, the system must be able to start an application process when any part of it is needed, and instantiate the Java objects for that part. Therefore, unlike applications on most other systems, Android applications don't have a single entry

point for everything in the application (no main() function, for example). Rather, they have essential *components* that the system can instantiate and run as needed. There are four types of components:

Activities

An *activity* presents a visual user interface for one focused endeavor the user can undertake. For example, an activity might present a list of menu items users can choose from or it might display photographs along with their captions. A text messaging application might have one activity that shows a list of contacts to send messages to, a second activity to write the message to the chosen contact, and other activities to review old messages or change settings. Though they work together to form a cohesive user interface, each activity is independent of the others. Each one is implemented as a subclass of the Activity base class.

An application might consist of just one activity or, like the text messaging application just mentioned, it may contain several. What the activities are, and how many there are depends, of course, on the application and its design. Typically, one of the activities is marked as the first one that should be presented to the user when the application is launched. Moving from one activity to another is accomplished by having the current activity start the next one.

Each activity is given a default window to draw in. Typically, the window fills the screen, but it might be smaller than the screen and float on top of other windows. An activity can also make use of additional windows — for example, a pop-up dialog that calls for a user response in the midst of the activity, or a window that presents users with vital information when they select a particular item on-screen.

The visual content of the window is provided by a hierarchy of views — objects derived from the base View class. Each view controls a particular rectangular space within the window. Parent views contain and organize the layout of their children. Leaf views (those at the bottom of the hierarchy) draw in the rectangles they control and respond to user actions directed at that space. Thus, views are where the activity's interaction with the user takes place.

For example, a view might display a small image and initiate an action when the user taps that image. Android has a number of ready-made views that you can use — including buttons, text fields, scroll bars, menu items, check boxes, and more.

A view hierarchy is placed within an activity's window by the Activity.setContentView() method. The *content view* is the View object at the root of the hierarchy. (See the separate User Interface document for more information on

views and the hierarchy.)

Services

A *service* doesn't have a visual user interface, but rather runs in the background for an indefinite period of time. For example, a service might play background music as the user attends to other matters, or it might fetch data over the network or calculate something and provide the result to activities that need it. Each service extends the Service base class.

A prime example is a media player playing songs from a play list. The player application would probably have one or more activities that allow the user to choose songs and start playing them. However, the music playback itself would not be handled by an activity because users will expect the music to keep playing even after they leave the player and begin something different. To keep the music going, the media player activity could start a service to run in the background. The system would then keep the music playback service running even after the activity that started it leaves the screen.

It's possible to connect to (bind to) an ongoing service (and start the service if it's not already running). While connected, you can communicate with the service through an interface that the service exposes. For the music service, this interface might allow users to pause, rewind, stop, and restart the playback.

Like activities and the other components, services run in the main thread of the application process. So that they won't block other components or the user interface, they often spawn another thread for time-consuming tasks (like music playback). See Processes and Threads, later.

Broadcast receivers

A *broadcast receiver* is a component that does nothing but receive and react to broadcast announcements. Many broadcasts originate in system code — for example, announcements that the timezone has changed, that the battery is low, that a picture has been taken, or that the user changed a language preference. Applications can also initiate broadcasts — for example, to let other applications know that some data has been downloaded to the device and is available for them to use.

An application can have any number of broadcast receivers to respond to any announcements it considers important. All receivers extend the BroadcastReceiver base class.

Broadcast receivers do not display a user interface. However, they may start an activity in response to the information they receive, or they may use the

NotificationManager to alert the user. Notifications can get the user's attention in various ways — flashing the backlight, vibrating the device, playing a sound, and so on. They typically place a persistent icon in the status bar, which users can open to get the message.

Content providers

A *content provider* makes a specific set of the application's data available to other applications. The data can be stored in the file system, in an SQLite database, or in any other manner that makes sense. The content provider extends the ContentProvider base class to implement a standard set of methods that enable other applications to retrieve and store data of the type it controls. However, applications do not call these methods directly. Rather they use a ContentResolver object and call its methods instead. A ContentResolver can talk to any content provider; it cooperates with the provider to manage any interprocess communication that's involved.

See the separate Content Providers document for more information on using content providers.

Whenever there's a request that should be handled by a particular component, Android makes sure that the application process of the component is running, starting it if necessary, and that an appropriate instance of the component is available, creating the instance if necessary.

II 中文译文

Android 应用程序使用 Java 编程语言开发。aapt 工具把编译后的 Java 代码连同应用程序所需的其他数据和资源文件一起打包到一个 Android 包文件中，这个文件使用.apk 作为扩展名。此文件是分发并安装应用程序到移动设备的载体；是用户下载到他们的设备的文件。单一.apk 文件中的所有代码被认为是一个应用程序。从多个角度来看，每个 Android 应用程序都存在于它自己的世界之中：

1 默认情况下，每个应用程序均运行于它自己的 Linux 进程中。当应用程序中的任何代码需要被执行时，Android 启动此进程，而当不再需要此进程并且其它应用程序又请求系统资源时，则关闭这个进程。

2 每个进程都有其独有的虚拟机（VM），所以应用程序代码与所有其它应用程序代码是隔离运行的。

3 默认情况下，每个应用程序均被赋予一个唯一的 Linux 用户 ID，并加以权限设置，使得应用程序的文件仅对此用户及此应用程序可见——尽管也有其它的方法使得这些文件同样能为其他应用程序所访问。

1 应用程序组件

Android 的一个核心特性就是一个应用程序可以使用其它应用程序的元素（如果那个应用程序允许的话）。例如，如果你的应用程序需要显示一个图片卷动列表，而另一个应用程序已经开发了一个合用的而又允许别的应用程序使用的话，你可以直接调用那个卷动列表来完成工作，而不用自己再开发一个。你的应用程序并没有吸纳或链接其它应用程序的代码。它只是在有需求的时候启动了其它应用程序的那个功能部分。

为达到这个目的，系统必须能够在一个应用程序的任何一部分被需要时启动一个此应用程序的进程，并将那个部分的 Java 对象实例化。因此，不像其它大多数系统上的应用程序，Android 应用程序并没有为应用程序提供一个单独的入口点（比如说，没有 main()函数），而是为系统提供了可以实例化和运行所需的必备组件。一共有四种组件类型：

1 Activity

activity 是为用户操作而展示的可视化用户界面。例如，一个 activity 可以展示一个菜单项列表供用户选择，戒者显示一些包含说明文字的照片。一个短消息应用程序可以包括一个用于显示要发送消息到的联系人列表的 activity，一个给选定的联系人写短信的 activity 以及翻阅以前的短信或改变设置的其他 activity。尽管它们一起组成了一个内聚的用户界面，但其中每个 activity 都不其它的保持独立。每一个都实现为以 Activity 类为基类的子类。

一个应用程序可以只有一个 activity，戒者，如刚才提到的短信应用程序那样，包含很多个。每个 activity 的作用，以及有多少个 activity，当然是取决于应用程

序及其设计的。一般情况下，总有一个应用程序被标记为用户在应用程序启动的时候第一个看到的。从一个 activity 转向另一个靠的是用当前的 activity 启动下一个。

每个 activity 都被给予一个默认的窗口以进行绘制。一般情况下，这个窗口是满屏的，但它也可以是一个小的位于其它窗口之上的浮动窗口。一个 activity 也可以使用附加窗口——例如，一个在 activity 运行过程中弹出的供用户响应的对话框，戒是一个当用户选择了屏幕上特定项目后显示的必要信息的窗口。

窗口显示的可视内容是由一系列层次化 view 构成的，这些 view 均继承自 View 基类。每个 view 均控制着窗口中一块特定的矩形区域。父级 view 包含并组织其子 view 的布局。叶节点 view（位于层次结构最底端）在它们控制的矩形区域中进行绘制，并对用户直达其区域的操作做出响应。因此，view 是 activity 与用户进行交互的界面。例如，view 可以显示一个小图片，并在用户指点它的时候产生动作。Android 有一些预置的 view 供开发者使用——包括按钮、文本域、滚动条、菜单项、复选框等等。

view 层次结构是由 Activity.setContentView() 方法放入 activity 的窗口之中的。content view 是位于层次结构根位置的 View 对象。（参见独立的用户界面文档以获取关于 view 及层次结构的更多信息。）

2 Service

service 没有可视化的用户界面，而是在一段时间内在后台运行。例如，一个 service 可以在用户做其它事情的时候在后台播放背景音乐、从网络上获取数据或者计算一些东西并提供给需要这个运算结果的 activity 使用。每个 service 都继承自 Service 基类。

一个媒体播放器播放播放列表中的曲目是一个不错的例子。播放器应用程序可能有一个或多个 activity 来给用户选择歌曲并进行播放。然而，音乐播放这个任务本身丌应该由任何 activity 来处理，因为用户期望即使在他们离开播放器应用程序而开始做别的事情时，音乐仍在继续播放。为达到这个目的，媒体播放器 activity 可以启动一个运行于后台的 service。系统将在这个 activity 不再显示于屏幕乀后，仍维持音乐播放 service 的运行。

连接至（绑定到）一个正在运行的 service（如果 service 没有运行，则启动之）是可能的。连接之后，你可以通过那个 service 暴露出来的接口不 service 进行通讯。对于音乐 service 来说，这个接口可以允许用户暂停、回退、停止以及重新开始播放。

如同 activity 和其它组件一样，service 运行于应用程序进程的主线程内。所以它不会对其它组件或用户界面有任何妨碍，它们一般会派生一个新线程来执行一些时间消耗型任务（比如音乐回放）。参见稍后的进程和线程。

3 Broadcast receiver

broadcast receiver 是一个与注于接收广播通知信息，并做出相应处理的组件。许多广播是由系统代码产生的——例如，通知时区改变、电池电量低、拍摄了一张照片或者用户改变了语言选项。应用程序也可以发起广播——例如，通知其它应用程序一些数据已经下载到设备上并处于可用状态。

一个应用程序可以拥有任意数量的 broadcast receiver，以对所有它认为重要的通知信息予以响应。所有的 receiver 均继承自 BroadcastReceiver 基类。

broadcast receiver 没有用户界面。然而，它们可以启动一个 activity 来响应它们收到的信息，或者也可以使用 NotificationManager 来通知用户。通知可以用多种方式来吸引用户的注意力──闪动背光灯、震动设备、播放声音等等。通知一般是在状态栏上放一个持丽的图标，用户可以打开它并获取消息。

4 Content provider

content provider 将一些特定的应用程序数据供给其它应用程序使用。数据可以存储于文件系统、SQLite 数据库或其它有意丿的方式。content provider 继承于 ContentProvider 基类，实现了一套使得其他应用程序能够检索和存储它所管理的类型数据的标准方法。然而，应用程序并不直接调用返些方法，而是使用一个 ContentResolver 对象，调用它的方法作为替代。ContentResolver 可以与任何 content provider 进行会话；与其合作对任何相关的进程间通讯进行管理。参阅独立的 Content Providers 文档以获得更多关于使用 content provider 的信息。

每当出现一个需要被特定组件处理的请求时，Android 会确保那个组件的应用程序进程处于运行状态，必要时会启动它，并确保那个组件的一个合适的实例可用，必要时会创建那个实例。

谢辞

本次的毕业设计进行的非常顺利，这背后除了作者本人之外还需要感谢很多在背后给予帮助和支持的同学和老师，以及那些不知名的朋友们。正是由于你们的帮助和付出，才使得这次的毕设按照计划的进行下去。正是由于你们的鼓励，才能使得我可以在困难面前继续坚持下去。非常感谢你们。

感谢杨晓光老师指导和支持，正是由你的把关，我才能按照要求完成论文的拟写工作。

感谢张俊老师的悉心指导，在你的帮助下面，项目才得以一步步的发展下去。

感谢所有东软的老师，每次的毕业指导课程都非常有价值的意义。虽然在谈论当中不免会出现冲突，但是正是由于这些问题使得这次毕设变的更好。

感谢陈立辉同学，陈蕾同学，娄巍屹同学，李宝昌同学，刘德强同学。是你们的鼓励和支持帮助我完成了本次毕设。

感谢 GreenRobot 组织，你们的 EventBus 项目解决 Android 项目通信的问题。

感谢 Excilys 组织，你们的 AndroidAnnotation 框架，大大减少了过去 Android 开发当中重复的代码，使得我可以将开发的主要精力放在项目的核心代码上面

感谢 Facebook 和 Parser 的程序员们，正是你们大度的将 Bolts 这样如此优秀的项目开源出来，才使得本项目当中涉及到的异步回调代码变的如此优雅和简单。

感谢 Square 的程序员们，你们对于 Android 开源项目的鼎力支持，促进了 Android 项目的开发效率和质量。同时也帮助我解决了和图相关代码的许多问题。

感谢 Google 公司，正是由于你们的努力，为这个世界带来了如此优秀的移动操作系统。感谢你们的付出。

感谢所有帮助过我，但是没有在这里提及到的人们，谢谢你们对我的帮助。