

华东师范大学软件学院
2016 年软件工程学士学位论文

基于 Android 的移动点餐系统服务器端设计

Design for the Server-side of Mobile Ordering System based on Android

姓 名： 周则宙
学 号： 10122510259
班 级： 2012 级 2 班
指导教师姓名： 全红艳
指导教师职称： 副教授

2016 年 5 月

目 录

华东师范大学软件学院.....	I
摘 要.....	I
ABSTRACT.....	II
一、 绪论	1
(一) 课题研究的背景及意义	1
(二) 国内外研究动态	3
1. 国外研究动态	3
2. 国内研究动态	3
(三) 核心技术介绍	4
(四) 论文主要内容及章节结构	6
二、 系统分析与设计.....	8
(一) 需求分析	8
(二) 系统设计	19
三、 系统实现.....	33
(一) 系统开发环境及方法工具	33
(二) MYSQL 数据库连接.....	33
(三) ANDROID 后台管理系统 SQLite 数据库建立.....	33
(四) CONTENTPROVIDER 为其它应用提供数据访问接口.....	34
(五) VOLLEY 框架的使用	34
(六) 用户注册功能的实现	34
(七) 用户登录功能的实现	36
(八) 查看菜单功能的实现	37
(九) 点餐下单功能的实现	37
(十) 同步数据功能的实现	38
(十一) 管理员登录及餐厅数据统计的实现.....	40
(十二) 菜品修改功能的实现.....	41
(十三) 菜品添加功能的实现.....	43
(十四) 菜品删除功能的实现.....	45
四、 系统测试.....	46
(一) 功能测试	46
(二) 测试总结	48
五、 总结和展望.....	49
参考文献.....	50
附录.....	52
致谢.....	77

摘 要

目前餐饮企业在日常运营中碰到了诸多挑战,比如外卖软件的市场抢夺,人力成本的日益昂贵等问题。在互联网迅猛发展的今天,各式移动 App 层出不穷,通过对 Android 市场份额的调查,一款基于 Android 的移动点餐系统必将受大众欢迎。为了解决餐饮企业面临的问题并满足餐饮业的变化及管理需求,服务器端的开发也变得尤为重要,为了提高餐饮企业的管理效率并使点餐变得更加高效且新颖,本文研究开发的移动点餐系统服务器端应运而生,让传统餐饮行业体会到技术革新带来的魅力。

本文结合餐饮业的市场背景以及 Android 的发展趋势,以数据为支撑进行系统的立项开发,并对系统运用到的核心知识,如 C/S 模式,Servlet 技术,Android SQLite 数据库进行了说明。在研究当中,本系统严格遵循软件工程思想,首先利用 UML 统一建模语言,E-R 模型等工具详细分析需求,总结用例并对具体模块进行完备的设计,为之后的开发打好坚实的基础;之后使用 Java 语言,以 MyEclipse10,Android Studio 为工具开发了服务器端 web 项目以及 Android 后台管理客户端,最后成功将 web 项目部署于阿里云服务器上供客户端访问从而实现了用户登录,用户注册,查看菜单,点餐下单的后台接口以及管理员登录,数据同步,菜品添加,修改,删除等功能。最后通过功能测试保证系统的完备可用。达到了预期的目标并能满足日常使用的功能。最终为当今的餐饮企业的移动点餐系统提供服务器端的接口支持以及更高效的后台管理方案。

关键词: 移动点餐, 服务器端, 后台菜品管理, Android

Abstract

Currently catering enterprises are facing a serious challenge in their daily operations, such as the market snatch of the take out applications, increasingly expensive labor costs. Nowadays, the Internet develops rapidly. And there emerges a lot of excellent mobile applications. An Android-based mobile ordering system will welcomed by the public by the survey of Android market share. In order to solve the challenges that catering enterprises faced and meet the changing and management needs of the catering enterprises, the development of server-side becomes more and more important. So the server-side of Android-based mobile ordering system in this paper can provide a more efficient and novel stylish way to order, provide a more convenient and effective way for restaurant management at the same time. Finally, bring charm of technological innovation to traditional catering enterprises.

In this paper, I combine the background of catering industry and the development trend of Android, setting up the project and developing on it by support of data. And introduce some core knowledge such as C/S Mode, Servlet technology, Android SQLite Database. During the study, this system in strict accordance with software engineering development. First of all, carry on the demanding analysis for this system, summarize the use cases relates to this system and carry on complete design by UML, E-R Model and so on.. Then develop a web project for server and a backstage management client based on Android by Java, MyEclipse 10, Android Studio and so on. After that deploy the web project in Alibaba Cloud for client access to achieve the background interface of user login, user registered, view the menu and ordering as well as serveral functions such as administrator login, data synchronization, meal adjunction, modification and deletion. Finally, through a functional test ensure the complete and available of the system. Ultimately providing server interfaces and a more efficient backstage management solution for today's mobile ordering system of restaurant.

Keywords: Mobile ordering, Server-side, Background Meal Management, Android

一、 绪论

(一)课题研究的背景及意义

俗话说“民以食为天”，餐饮业一直以来都备受关注，并且作为服务业的一个重要组成部分，为我国经济发展起到了无法替代的作用。2011 年我国的餐饮业总额占零售总额的 11.22%，达到 20635 亿元以上，相比 2010 年增长 16.9%，对我国经济做出了巨大的贡献。2014 年餐饮销售总额增长 9.7%，超过 27000 亿元。然而，目前盈利能力下降，各式成本升高等问题逐渐成为阻碍餐饮业发展的障碍而其中最突出的问题便是人工成本的日益提高，餐饮业亟需转型，优化结构。为了有效地降低人工成本，并且为人们提供更加方便快捷的服务，如何改善经营和管理方法，成为了目前我国餐饮行业面对的主要难题之一，其中技术的革新成为了一个新的方向^[2]。

随着现代科技的迅猛进步，基本上人人都有一部智能手机或平板，普及程度非常高，据统计，我国的移动智能终端早在 2014 年就已经达到 10.6 亿。因为智能终端手持设备的价格成本的降低，许多餐饮企业已经使用 PDA 设备电子点餐取代原始的手工人力点餐的旧方式。然而，传统的 PDA 点餐设备仍然需要餐厅服务人员进行使用，不仅提高了餐厅服务人员的学习成本而且关键并没有达到降低人力成本的目的，现状亟需改进。使用一款普及率高，大众普遍拥有的智能终端来进行点餐可以有效的解决这个问题。

根据 2012 年的数据统计，11 月份全球智能终端份额 Android 占到其中的 76%，到了 2015 年 11 月，Android 操作系统所占份额高达 84.7%，截止今年 2 月，如图 1-1 所示，Android 在中国的市场份额占据 76.4%，蚕食着其它系统的市场，乔布斯去世以后，iPhone 渐渐有所没落，在各大安卓企业的努力下，安卓将逐渐成为智能终端的主流操作系统^[3]。基于 Android 的移动点餐系统在这样的情景下，才能更好的发挥出作用，让餐饮企业以及人民大众体会到科技进步带来的便利。

USA	3 m/e Feb 15	3 m/e Feb 16	% pt. Change
Android	55.6	58.9	3.3
iOS	38.8	38.3	-0.5
Windows	4.8	2.6	-2.2
Other	0.8	0.2	-0.6
China	3 m/e Feb 15	3 m/e Feb 16	% pt. Change
Android	73.0	76.4	3.4
iOS	25.4	22.2	-3.2
Windows	0.9	0.9	0.0
Other	0.7	0.5	-0.2
Australia	3 m/e Feb 15	3 m/e Feb 16	% pt. Change
Android	51.3	55.1	3.8
iOS	37.9	38.2	0.3
Windows	9.1	5.8	-3.3
Other	1.7	0.9	-0.8
Japan	3 m/e Feb 15	3 m/e Feb 16	% pt. Change
Android	47.9	48.2	0.3
iOS	49.8	50.2	0.4
Windows	0.3	0.5	0.2
Other	2.0	1.1	-0.9
EU5	3 m/e Feb 15	3 m/e Feb 16	% pt. Change
Android	67.7	74.3	6.7
iOS	20.9	19.1	-1.8
Windows	10.1	5.9	-4.2
Other	1.4	0.6	-0.7

图 1-1 Android 市场份额变化图

Figure1-1 Change of Android Market Share

对于餐饮企业来说使用基于 Android 的移动点餐系统相对于原始的人工手动点餐，极大地降低了人力成本；与传统的 PDA 电子点餐相比，减少了设备的购买维护成本。此外，移动点餐相对于传统人工下单，查询，上菜等复杂繁琐的步骤极大地提高了效率，减少了人工干预。

而对于顾客来说，使用本系统可以在自己的 Android 手机上进行点餐，下单更加省时省力。Android 因其组件的丰富性，能让用户使用操作的更加流畅，界面相比于 PDA 设备也更加美观友好。

综上所述，本系统具有极大的研究意义，成功的可能性也异常之大。

(二)国内外研究动态

1. 国外研究动态

在二十世纪中旬,原始的手工服务模式仍是餐饮业的主要方式。二十世纪末,计算机行业迅猛发展,国外的一些发达国家,率先察觉到信息技术的前景以及优点,在餐饮行业实现了一些技术革新以及尝试。近些年来,移动智能点餐方式在越来越多的国家地区得以实施。

根据美国 NRA 协会的统计,美国餐饮业的极速增长已经持续了 12 年,2012 年销售额已经超过 6320 亿美元,较去年增长 3.5%。然而他们并不满足于此,每年他们仍会拿出部分资金进行投资研究,为了更加创新,为了技术的更新。

2013 年,美国丹佛大学联合内华达大学研究了客户互动技术,对餐饮业的未来方向进行了总结,即移动网站和 APP 将成酒店餐饮必备装备。在餐饮业中,移动技术已经越来越重要,直接影响了经营以及客户互动,对业绩举足轻重。网站以及移动应用受到美国餐饮业的重视,被越来越多的企业投资开发。在 2015 年,拥有移动应用的餐厅在全美已经达到 75%,而网站则达到惊人的 91%。

可以看出在国外,尤其是欧美地区,对于餐饮行业的技术革新以及创新性的研究从未停止过,每年都会花费大量资金进行相关技术的探索,其中移动 APP 应用更是他们研究的重点,移动点餐将会是将来餐饮行业的发展趋势,而一个完整的移动点餐系统更是将来餐饮业所不可或缺的一部分。

2. 国内研究动态

国内同样在餐饮行业的软件方面进行了大量的研究探索,无线点菜渐成主流,目前国内主流的几种新型点菜方式为:无线点餐系统,触摸屏点菜系统,手持式点菜系统^[3]。其中移动点餐软件层出不穷,下面介绍部分优秀的移动点餐软件。

(1) 天子星:国内第一个获得 IBM Ready for POS 认证的餐饮软件

天子星是一款非常优秀的餐饮软件,在亚太地区获得了 IBMReadyforRIF 认证,该品牌针对不同的餐饮业态,划分了多项子产品,包括大型的餐品连锁信息系统,中小型餐饮连锁信息系统,正餐、快餐等七个版本^[18]。

(2) 餐行健:天行健,君子以自强不息

“餐行健”系列产品众多,其中包括电子菜谱、智能点餐宝、智能餐饮管理软件等,为大小型餐饮企业提供了各种丰富的软件,基本满足餐馆需求。同时该系列产品获得了多项国家专利,以及软件著作权。

其它还有众多优秀的点餐软件如饮食通等。综上，国内在点餐软件模块已经有了非常多的探索以及成果，起到了很好的榜样作用。

(三)核心技术介绍

1. C/S 架构

本移动点餐系统采用 C/S 结构（客户端与服务器端结构）。本系统根据 C/S 模式分为两大部分：前端（本系统中的 Android 客户端）和后端（我负责的服务器端）^[5]。服务器端为 Android 端提供数据接口，接收相关数据并返回 Android 端所需数据。在本系统中，Android 移动点餐客户端为用户提供一个友好的 GUI 供用户输入，Android 客户端对用户的输入进行合法性的验证，然后提交输入进行查询，并显示查询结果，而服务器端则对客户端提供的输入数据进行逻辑处理，从数据库获取所需数据并返回给客户端。本移动点餐系统的总体架构如图 1-2 所示。本系统采用了三层 C/S 模式，将整体分为表示层，功能层和数据层。Android 客户端为表示层，负责输入数据并向服务器发出请求并显示结果，而服务器端则分为功能层和数据层，在数据层对数据库进行操作，而功能层则对数据进行逻辑处理最终返回给表示层即客户端。整体业务流程如图 1-3。C/S 模式交互性强，适合处理大量数据并且适合使用范围小地点固定的场合，对于移动点餐系统非常适合^[8]。

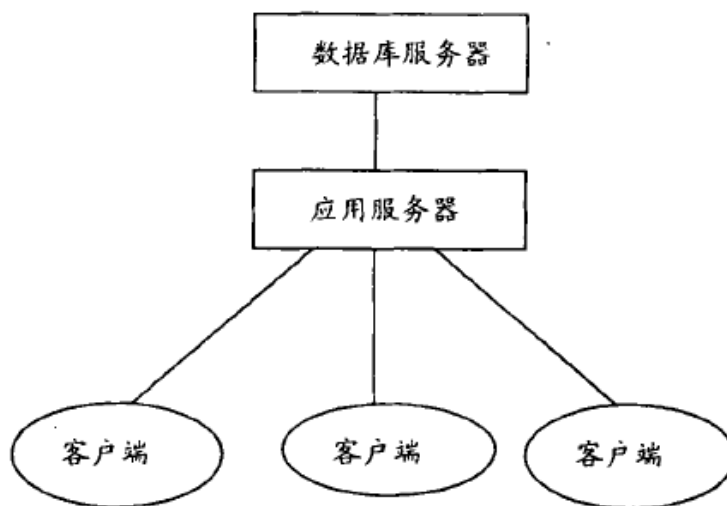


图 1-2 两层 C/S 模式

Figure1-2 Two Layers of C/S Mode

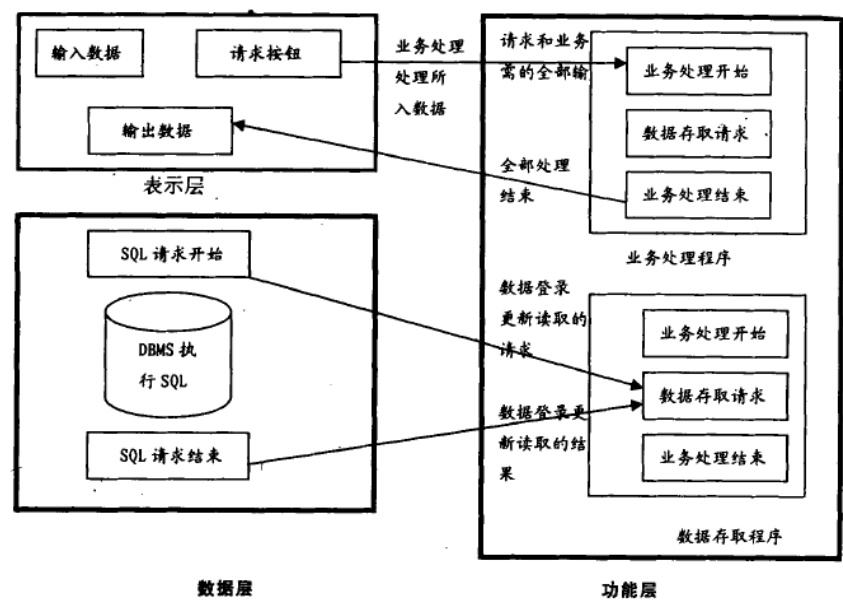


图 1-3 业务处理流程

Figure1-3 Business Processes

2. 服务器端 Servlet 技术

在服务器端的选择上，我采用了目前比较成熟且本人较为熟悉的 JavaWeb 项目，而它的关键技术则是 Servlet。

JAVA Servlet 提供了一种简便可靠的机制来扩展服务器的功能^[8]。它没有运行界面，Servlet 是服务器端的，与平台无关，因此可以适用于本系统。Servlet 可以使用 Java 所有的 api，同时也具有 java 的所有优点:轻便，可复用等特点。Servlet 即运行在后台的 Java 类，并且可以接受处理 HTTP 请求^[9]。Java Servlet 比一般的 CGI 技术更加高效，易用，彪悍^[10]，非常适合 C/S 模式。为了使用 Servlet，需要一个支持 Servlet 的服务器，目前支持 Servlet 的服务器很多，本系统采用了现在较为主流的 Tomcat 服务器。

3. 后台管理系统 Android，SQLite 数据库，ContentProvider

本系统为管理员提供了后台管理的 Android 客户端，其中该客户端创建了本地 SQLite 数据库与服务器端数据进行同步，并通过 ContentProvider 为顾客点餐客户端提供数据，下面介绍一下 Android 体系结构,SQLite 数据库以及 ContentProvider。

Android 系统框架体系结构如图 1-4，SQLite 处于其中的 Libraries 层，而在我们的开发中接触最多的是应用层（Applications）即基本使用 Java 开发。

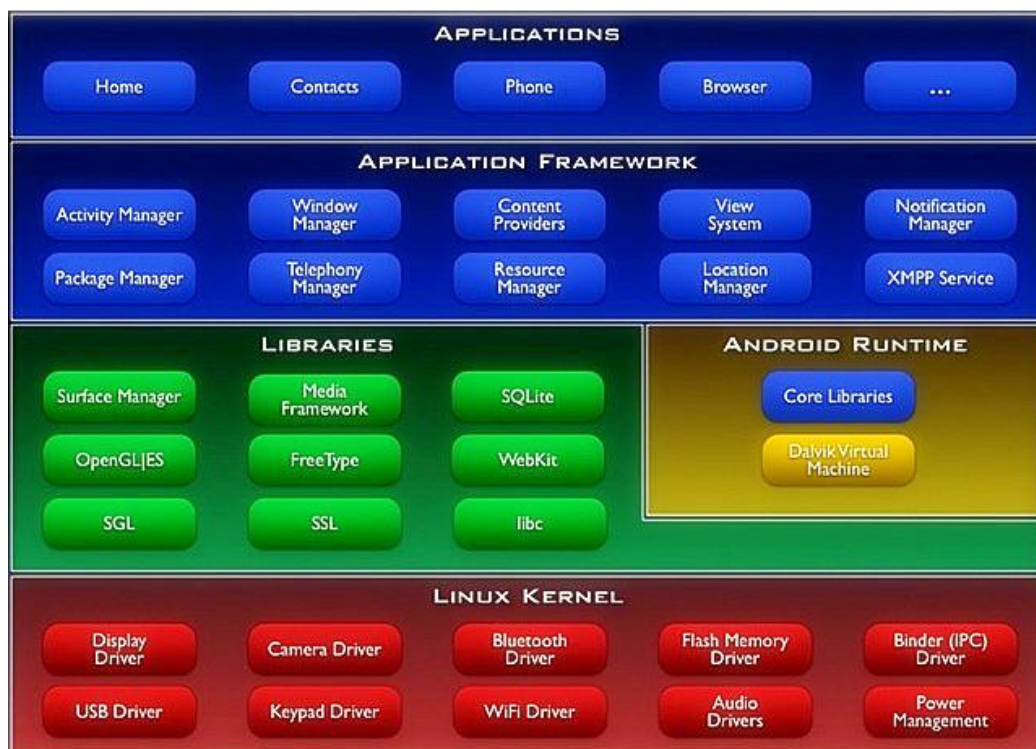


图 1-4 安卓体系结构框架

Figure1-4 Android Architecture Framework Map

SQLite 数据库是 Android 平台上的一个轻量级的数据库，虽然号称轻量级，却仍然实现了一个可以适应各种情况的数据库，具有数据库处理事务的原子性，一致性，隔离性和持久性四个基本特征。SQLite 只支持部分 SQL 语句^[14]。SQLite 有许多优点，它操作简单、体积小、效率惊人同时比较稳定^[16]。Android 为开发者提供了非常多便利的 api 来对 SQLite 进行操作，避免使用 JDBC 方式，适用于手机平板这类内存受限的设备。

在 Android 系统中 ContentProvider 可以将应用中的数据共享给其他应用，其他应用可以通过 ContentProvider 对应用中的数据进行增删改查如服务器端的作用一样，此时 SQLite 本地数据库即可作为数据层使用。ContentProvider 使外界应用可以通过规定的接口以更加方便的方式访问数据，更加严谨从而加强了规范性。

(四)论文主要内容及章节结构

本文根据移动点餐系统服务器端需要为客户端提供的服务进行需求分析，围绕现有需求使用 Servlet+Tomcat 进行服务器端的设计实现，并且通过 Android 实现后台菜品管理功能，研究使用 SQLite 创建本地数据库与服务器端同步并通过 ContentProvider 为移动点餐客户端提供另外一种数据获取方式。

本文内容安排如下：

第一章， 绪论

介绍目前国内外餐饮行业发展现状以及 Android 系统的发展现状，并通过对各种新型点餐系统在国内外的研究现状进行分析，表现出本系统研究的意义。介绍系统开发所用到的核心技术，包括服务器端以及后台管理客户端所用到的主要技术，最后说明主要研究内容及章节结构。

第二章， 系统分析与设计

针对移动点餐的场景进行需求分析，通过各种 UML 工具进行较为详尽的架构设计，功能设计，数据库设计等，为后续实现提供前期准备。

第三章， 系统实现

进行系统的详细编码实现，使用 Servlet+Tomcat 在阿里云服务器上搭建服务器端，并通过 Android 进行后台管理客户端开发，使用 Volley 框架进行网络请求。从而为移动点餐客户端提供用户注册登录，菜单获取，点餐下单的后台接口，并实现后台管理客户端的菜品管理，数据同步，餐厅数据统计，为移动点餐客户端提供数据等功能。

第四章， 系统测试

设计详尽的测试用例，对服务器端接口进行测试，并为 Android 客户端进行功能测试，根据结果进行维护与二次开发，保证系统的稳定性与质量。

第五章， 总结与展望

对本文所做工作进行总结，列出完成的主要工作内容，并对存在的不足进行总结反思，阐述本文后续研究的价值，并对未来工作进行展望。

二、 系统分析与设计

(一)需求分析

1. 概述

移动点餐正逐渐成为餐饮企业点餐方式的新潮流，传统人工服务点餐以及手持 PDA 设备点餐的方式因为其人力成本及设备成本的昂贵渐渐要被行业摒弃，因此需要开发移动点餐系统。而为了保证数据的统一，便于对数据进行管理，于是需要设计实现移动点餐系统服务器端。从而满足移动点餐客户端数据获取，后台菜品管理等需求。

2. 系统面向用户群体

客户方：某餐饮企业

餐饮企业客户方希望产品能够提供数据的稳定储存以及访问，并希望能通过后台管理系统进行数据以及菜品管理，使餐厅顾客能够愉快地进行点餐，提升餐厅点餐效率，并节约成本。

用户方 1：移动点餐客户端

移动点餐客户端需要向后台发送 HTTP 请求，希望本产品能够及时处理输入请求进行处理并且返回需要的数据。保证数据获取的便捷稳定。

用户方 2：餐厅管理员

管理员希望通过本产品获取餐厅的数据统计信息，并通过本产品进行菜品的管理，数据的同步，从而有效进行餐厅的管理。

3. 客户信息

本系统针对的客户方是那些仍在使用原始人工点菜或者使用传统手持 PDA 点餐的餐厅，这些餐厅由于点餐方式的落后，花费了大量的人力成本以及设备成本同时点餐效率也不高，正需要我们的系统。

4. 系统目标与范围

基于 Android 的移动点餐系统服务器端的目标是为客户端提供两种数据获取的方式，使点餐客户端能够获取最新的菜品信息并且快捷下单。本系统的主要功能包括客户端用户登录验证，菜品信息数据提供，订单管理，后台菜品管理，本地与服务器端数据库同步等功能，范围主要覆盖在后台为客户端提供接口，并为餐厅管理人员提供后台服务。

5. 业务分析描述

(1) 业务描述

本系统业务包括：接收 Android 移动点餐客户端向本系统发送的用户登录，注册，菜单获取，下订单等请求，并进行逻辑处理，返回结果给客户端；管理员通过 Android 后台管理客户端同步服务器端数据，查看餐厅统计的数据，并对菜品进行增删改操作。

(2) 业务数据流图

本系统业务流程如图 2-1 所示，Android 移动点餐客户端能够向服务器端及本地数据库发送众多请求，从而获取相应数据，Android 后台管理客户端则与服务器端数据保持同步并进行菜品管理，服务器端处理各种 HTTP 请求并返回结果。

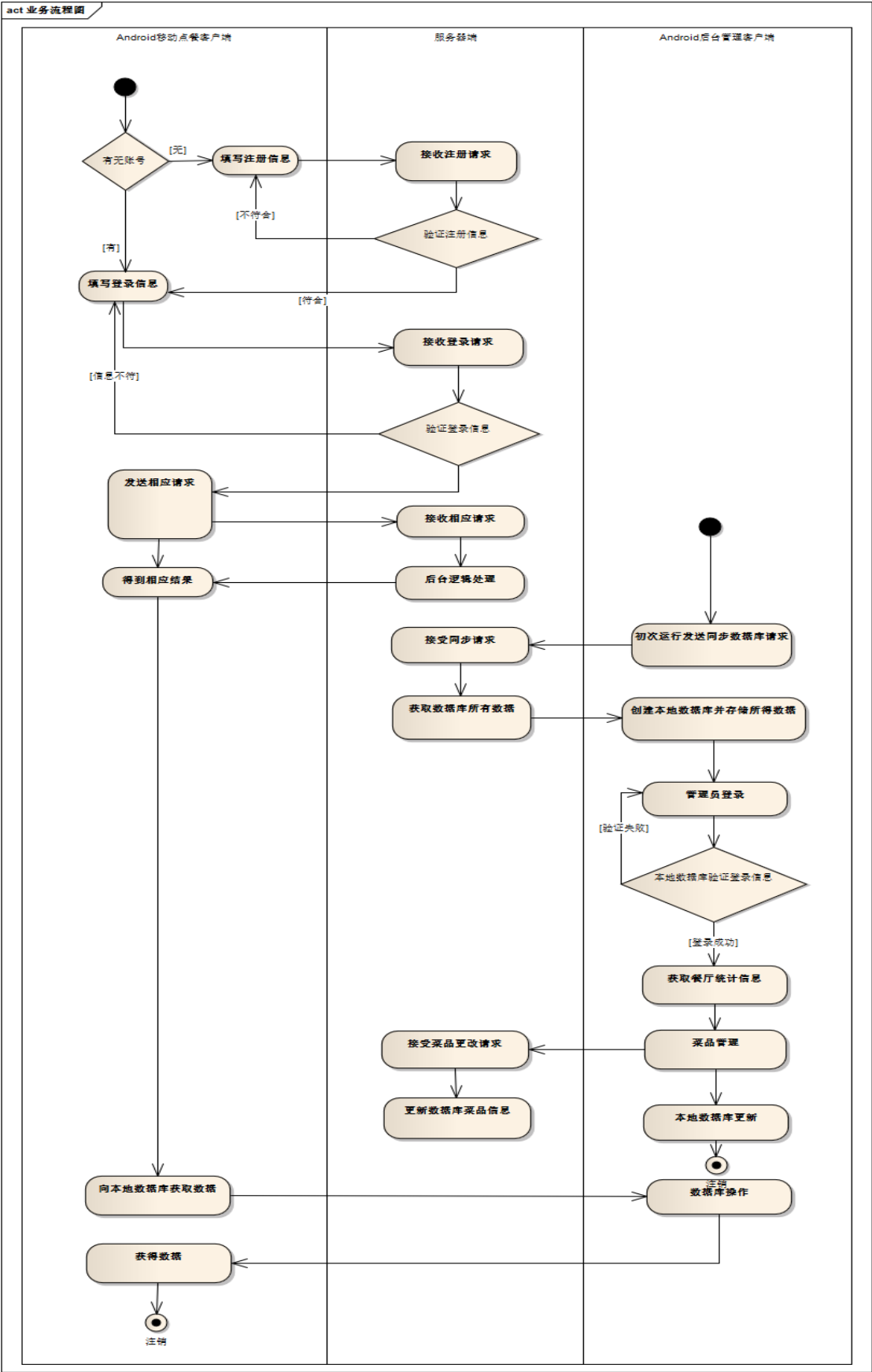


图 2-1 业务数据流图

Figure2-1 Business Data Flow Diagram

(3) 业务说明

用户注册：用户名，密码，手机号等注册信息由客户端通过 HTTP 请求发送给服务器端接口，用户名被数据库验证为不存在时便予以注册，否则注册失败。

用户登录：用户名，密码由客户端通过 HTTP 请求发送给服务器端接口，登录信息在后台进行逻辑验证是否匹配，从而返回登录是否成功的信息。

获取菜单：服务器端以及后台管理客户端接受菜单获取请求，根据所需菜品类型，查询数据库获取相应菜品数据返回移动点餐客户端。

用户下单：服务器端接受用户下单请求，获取下单信息，并存储在数据库中。

同步数据：管理员第一次打开或者手动选择同步数据功能后，数据同步请求通过客户端以 HTTP 请求形式发送给服务器接口接收，将所有数据从数据库中取出并返回给 Android 后台管理系统^[12]。

管理员登录：管理员使用 Android 后台管理客户端，输入用户名密码，在本地 SQLite 中进行查询验证。

餐厅数据统计：Android 后台管理系统统计整理 SQLite 数据库中的关键可用数据，并展示给管理员。

菜品管理：管理员通过 Android 后台管理客户端对菜品进行增删改操作，对应操作都要及时向服务器端发送，更新服务器端数据以及本地数据库。

6. 系统中的角色

系统中的角色如表 2-1 中所示

表 2-1 系统角色表

Table 2-1 System Actors

角色名称	职责描述
Android 移动点餐客户端	注册账号，账号登录，查看菜单，点餐下单
餐厅管理员	登录客户端，同步数据，统计餐厅数据，菜品管理

7. 系统的功能性需求

(1) 功能性需求分类

通过 UML 分析需求得到相应的用例，如图 2-2 所示

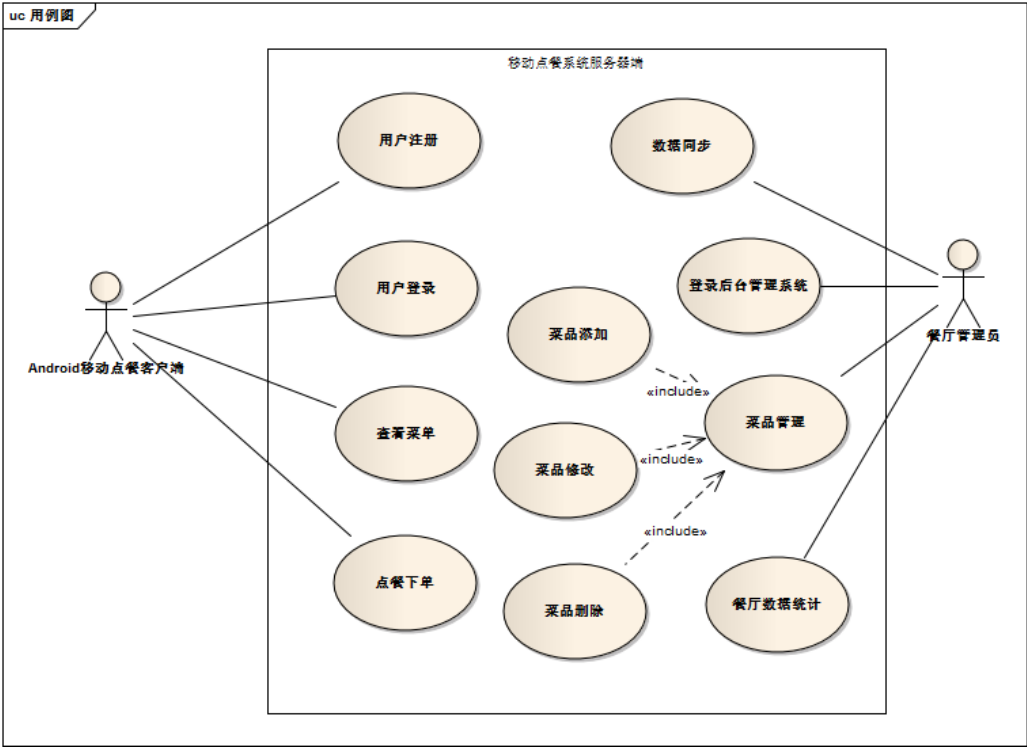


图 2-2 系统用例图

Figure 2-2 System Use Case

(2) 用例

- 用例名称： 用户注册
- 参与者： Android 移动点餐客户端
- 前置条件： 客户端网络连接通畅
- 后置条件： 客户端获取注册结果
数据库中新增用户
- 主干过程： 1. 移动点餐客户端发送注册请求
2. 服务器端接受请求
3. 获取验证注册信息
4. 数据库新增用户
5. 返回客户端注册成功
- 异常： 1a. 网络原因请求发送失败
1b. 客户端进行处理并重新发送请求
3a. 注册用户名已存在
3b. 返回客户端注册失败信息

用例名称： 用户登录

参与者： Android 移动点餐客户端

前置条件： 客户端网络连接通畅

后置条件： 客户端获取登录结果

主干过程： 1. 移动点餐客户端发送登录请求

2. 服务器端接受请求

3. 获取登录信息

4. 数据库中匹配用户名密码

5. 返回客户端登录成功

异常： 1a. 网络原因请求发送失败

1b. 客户端进行处理并重新发送请求

4a. 用户名与密码不匹配

4b. 返回客户端登录失败信息

用例名称： 查看菜单

参与者： Android 移动点餐客户端

前置条件： 客户端网络连接通畅

后置条件： 客户端获取菜单列表

主干过程： 1. 移动点餐客户端发送查看菜单请求

2. 服务器端接受请求

3. 获取所需菜单类型信息

4. 根据菜单类型查询数据库获取相应菜单数据

5. 返回客户端菜单数据

异常： 1a. 网络原因请求发送失败

1b. 客户端进行处理并重新发送请求

用例名称： 点餐下单

参与者： Android 移动点餐客户端

前置条件： 客户端网络连接通畅

用户已登录客户端

后置条件: 客户端获取下单结果

数据库中新增订单

主干过程: 1. 移动点餐客户端发送下单请求

2. 服务器端接受请求

3. 获取订单信息

4. 数据库中新增订单

5. 返回客户端下单成功

异常: 1a. 网络原因请求发送失败

1b. 客户端进行处理并重新发送请求

用例名称: 同步数据

参与者: 餐厅管理员

前置条件: 打开 Android 后台管理系统

后置条件: 本地 SQLite 与服务器 MySQL 数据同步

主干过程: 1. 客户端向服务器端发送数据同步请求

2. 服务器端接收请求

3. 查询数据库获得所有数据

4. 返回数据给客户端

5. 客户端接收数据创建本地数据库

6. 将数据存储到本地数据库中

异常: 1a. 网络原因请求发送失败

1b. 客户端进行处理并重新发送请求

用例名称: 登录后台管理系统

参与者: 餐厅管理员

前置条件: Android 后台管理系统已与服务器端同步数据

后置条件: 管理员登录系统

主干过程: 1. 餐厅管理员输入用户名及密码

2. 在本地数据库中验证登录信息

3. 登录成功

- 异常:
- 1a. 用户名或密码为空
 - 1b. 提示管理员输入完整登录信息
 - 1c. 管理员重新填入完整登录信息
 - 2a. 用户名与密码不匹配
 - 2b. 返回客户端登录失败信息
 - 2c. 重新输入用户名及密码

用例名称: 餐厅数据统计

参与者: 餐厅管理员

前置条件: Android 后台管理系统已与服务器端同步数据
餐厅管理员已登录系统

后置条件: 管理员查看餐厅统计数据

主干过程:

- 1. 管理员进入系统查看页面
- 2. 本地 SQLite 查询并统计相关信息
- 3. 统计结果显示在 Android 页面中

用例名称: 菜品添加

参与者: 餐厅管理员

前置条件: 管理员登录系统
进入菜品管理页面

后置条件: 服务器端数据库新增菜品
本地数据库新增菜品

主干过程:

- 1. 管理员选择添加菜品
- 2. 在本地选择菜品图片, 并填写菜品相关信息
- 3. 确定添加, 向服务器端发送菜品添加请求
- 4. 服务器接受请求获取菜品信息
- 5. 服务器存储菜品图片, 数据库中存储菜品信息
- 6. 返回客户端存储成功
- 7. 客户端将菜品信息存储到本地数据库

异常:

- 3a. 菜品信息不完整

3b. 提示管理员输入完整菜品信息

3c. 管理员重新填入完整菜品信息

4a. 网络原因请求发送失败

4b. 客户端重新发送请求

用例名称: 菜品修改

参与者: 餐厅管理员

前置条件: 管理员登录系统

进入菜品管理页面

后置条件: 后台 MySQL 菜品信息更新

本地数据库菜品信息更新

主干过程: 1. 管理员选择菜品

2. 在编辑页面修改对应菜品信息

3. 确定添加, 向服务器端发送菜品修改请求

4. 服务器端接受请求获取菜品修改信息

5. 服务器端 MySQL 更新菜品信息

6. 返回客户端更新成功

7. 客户端将菜品信息存储到本地 SQLite

异常: 3a. 菜品信息不完整

3b. 提示管理员输入完整菜品信息

3c. 管理员重新填入完整菜品信息

4a. 网络原因请求发送失败

4b. 客户端重新发送请求

用例名称: 菜品删除

参与者: 餐厅管理员

前置条件: 管理员登录系统

进入菜品管理页面

后置条件: 服务器端 MySQL 删除对应菜品条目

本地 SQLite 删除对应菜品条目

- 主干过程：
1. 管理员选择菜品

2. 点击删除按钮

3. 对话框中选择确定删除，向服务器提交删除请求

4. 服务器端接受请求获取所要删除菜品信息

5. 服务器端 MySQL 删除对应菜品条目

6. 返回客户端删除成功

7. 客户端本地 SQLite 删除对应菜品条目
- 异常：
- 4a. 网络原因请求发送失败

4b. 客户端重新发送请求

8. 产品的非功能性需求

(1) 用户界面需求

为使用户对界面感觉更友好，用户界面需求如表 2-2 所示

表 2-2 用户界面需求表

Table 2-2 User Interface Requirements

需求名称	详细要求
规范性	Android 客户端界面遵守 Material Design 设计规范
易用性	用户能非常清晰使用产品，具有很强的引导能力，简单易用容易上手
安全性	界面中不透露任何用户私密信息，如密码
简洁性	界面无冗余信息，没有额外广告，简洁清爽

(2) 软硬件环境需求

为满足系统需要，软硬件环境需求如表 2-3 所示

表 2-3 软硬件环境需求表

Table 2-3 Hardware and Software Environment Requirements

需求名称	详细要求
软件环境	服务器端使用 Tomcat 作为 web 服务器，数据库使用 MySql，后台管理系统使用 Android Studio 开发工具。
硬件环境	使用阿里云服务器 24 小时部署服务器，客户端要求 Android 智能终端，系统在 5.0 以上最佳。

(3) 产品质量需求

为保证产品的质量，产品质量需求如表 2-4 所示

表 2-4 产品质量需求表

Table 2-4 Product Quality Requirements

主要质量属性	详细要求
正确性	确保系统可以为点餐提供后台服务并可以进行后台管理
健壮性	确保系统故障容忍度高
可靠性	确保由于软件故障引起的点餐下单失败的概率不超过 1%
性能，效率	确保每个请求能在 10 秒内处理完毕
易用性	确保客户端能够方便使用本系统提供的接口，管理员能够在 10 分钟内学习后台管理系统的使用。
清晰性	确保各功能操作清晰易懂并有详细的说明
安全性	确保餐厅信息及密码安全，不泄露
可扩展性	确保整个系统可在 4 到 6 周内添加一个新的功能
兼容性	确保服务器接口可供任何可使用 http 协议的客户端使用，后台管理系统能够在 Android 各个版本的系统下正常运行
可移植性	确保整个系统便于迁移到别的平台

(二)系统设计

1. 系统功能设计

(1) 用户注册

客户端向服务器端请求注册新用户，详细流程如图 2-3 所示。

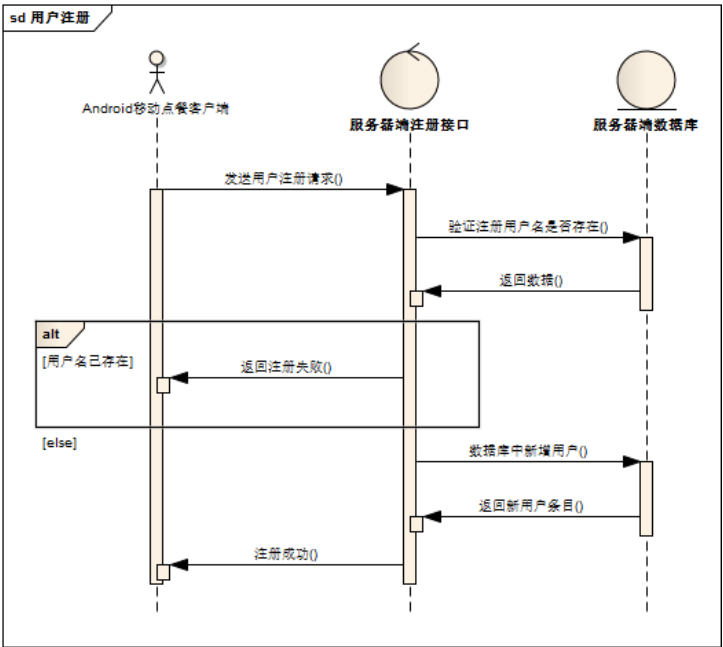


图 2-3 用户注册顺序图

Figure 2-3 User Registered Sequence Diagram

(2) 用户登录

客户端向服务器端验证用户或者通过后台管理系统本地数据库验证用户，详细流程如图 2-4 所示。

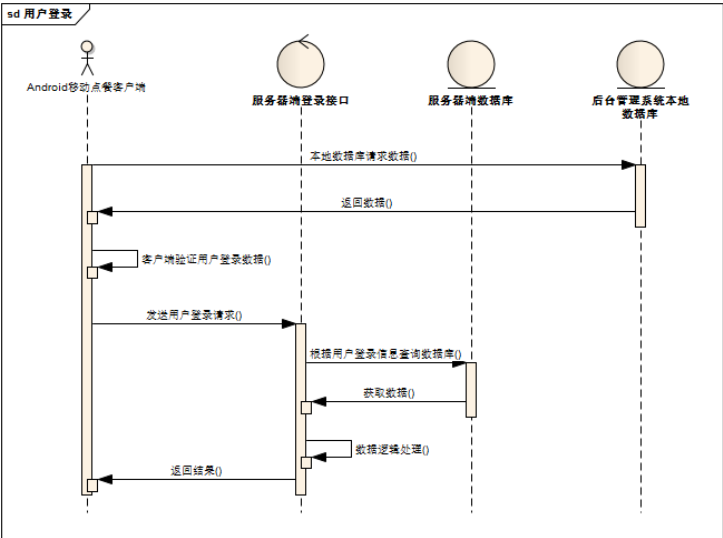


图 2-4 用户登录顺序图

Figure 2-4 User Log in Sequence Diagram

(3) 查看菜单

客户端向服务器请求菜单列表或通过后台管理系统本地数据库获取菜单数据，详细流程如图 2-5 所示。

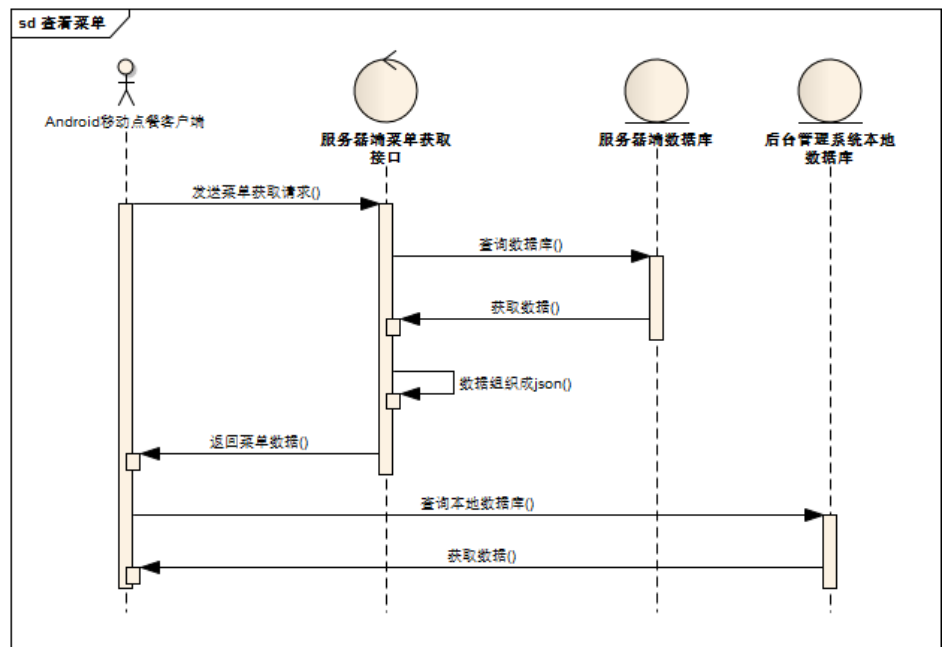


图 2-5 查看菜单顺序图

Figure 2-5 View Menu Sequence Diagram

(4) 点餐下单

客户端向服务器端请求点餐下单，详细流程如图 2-6 所示。

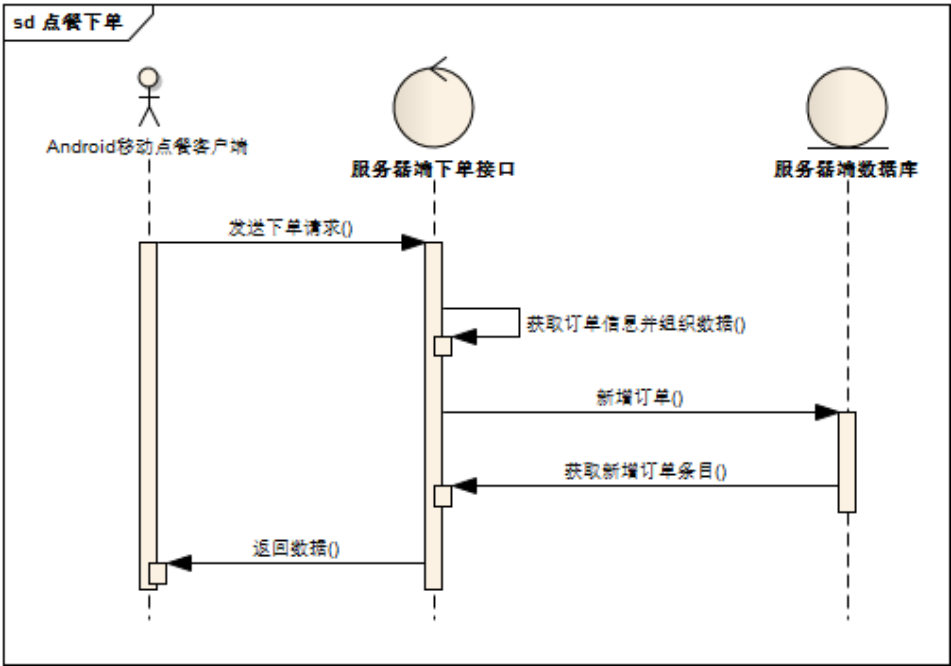


图 2-6 点餐下单流程图

Figure 2-6 Ordering Sequence Diagram

(5) 同步数据

后台管理系统客户端与服务器端数据进行同步，详细流程如图 2-7 所示。

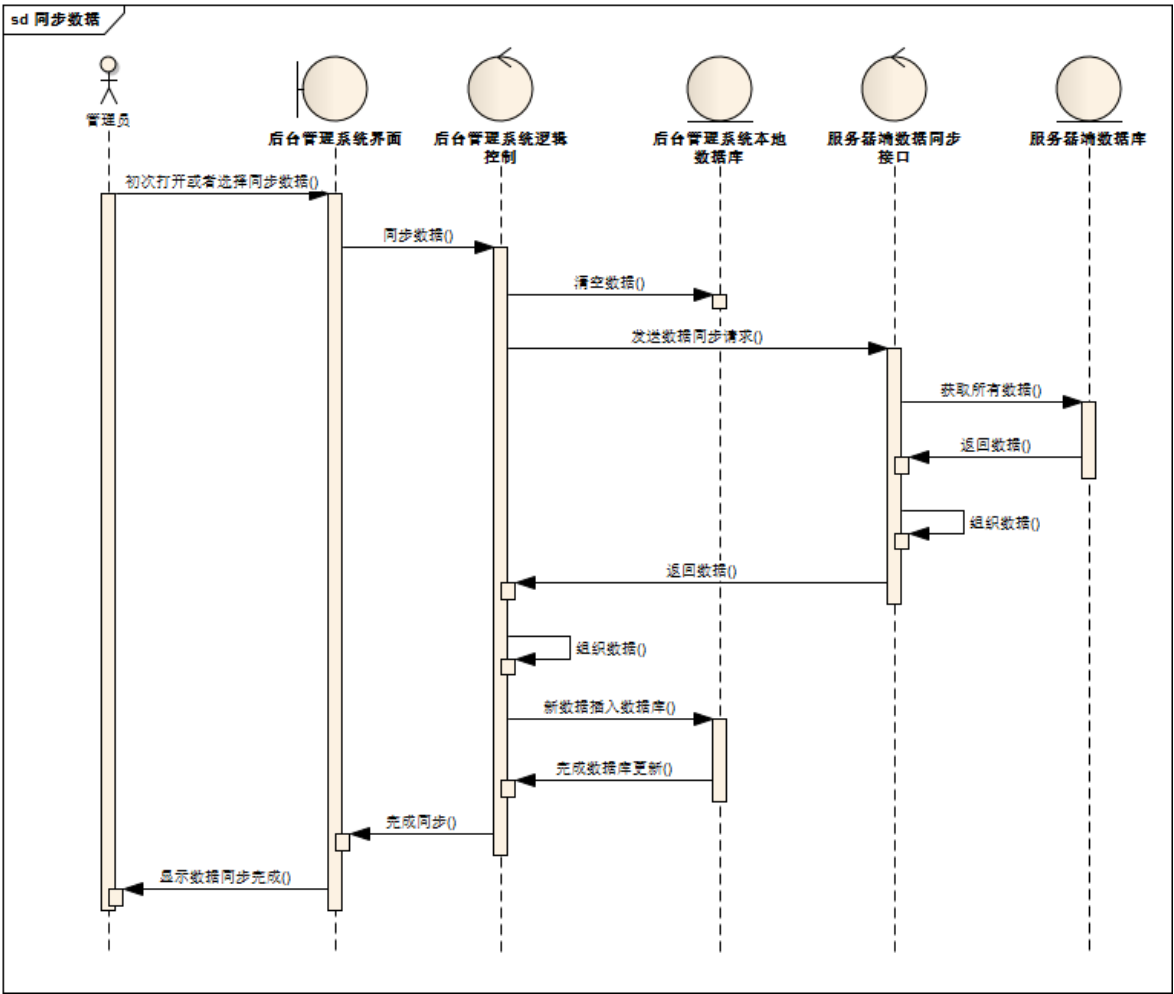


图 2-7 同步数据流程图

Figure 2-7 Data Synchronization Sequence Diagram

(6) 管理员登录

管理员登录后台管理系统，详细流程如图 2-8 所示。

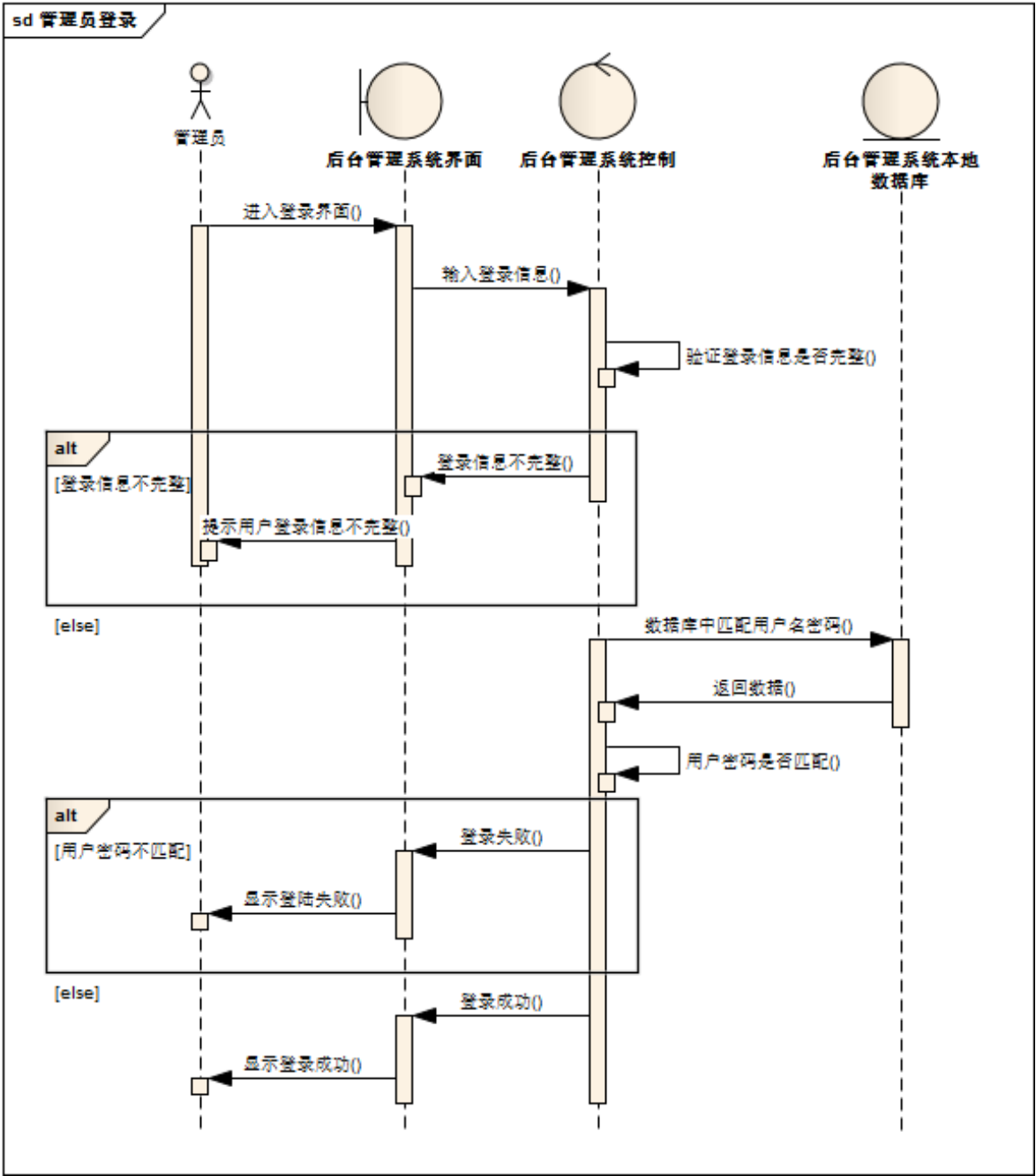


图 2-8 管理员登录流程图

Figure 2-8 Admin Log in Sequence Diagram

(7) 餐厅数据统计

后台管理系统对餐厅部分重要数据进行统计，详细流程如图 2-9 所示。

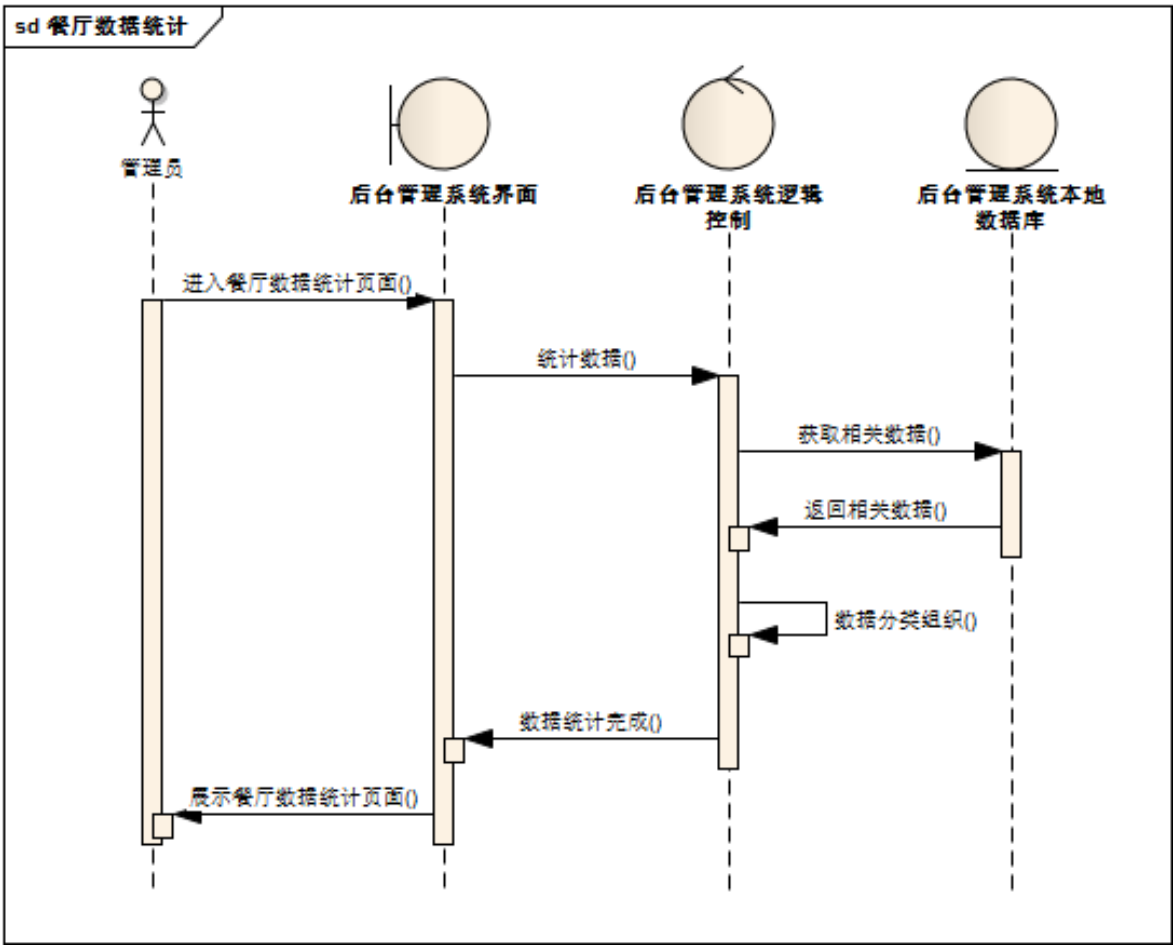


图 2-9 餐厅数据统计流程图

Figure 2-9 Restaurant Data Statistics Sequence Diagram

(8) 菜品添加

管理员添加新菜品，详细流程如图 2-10 所示。

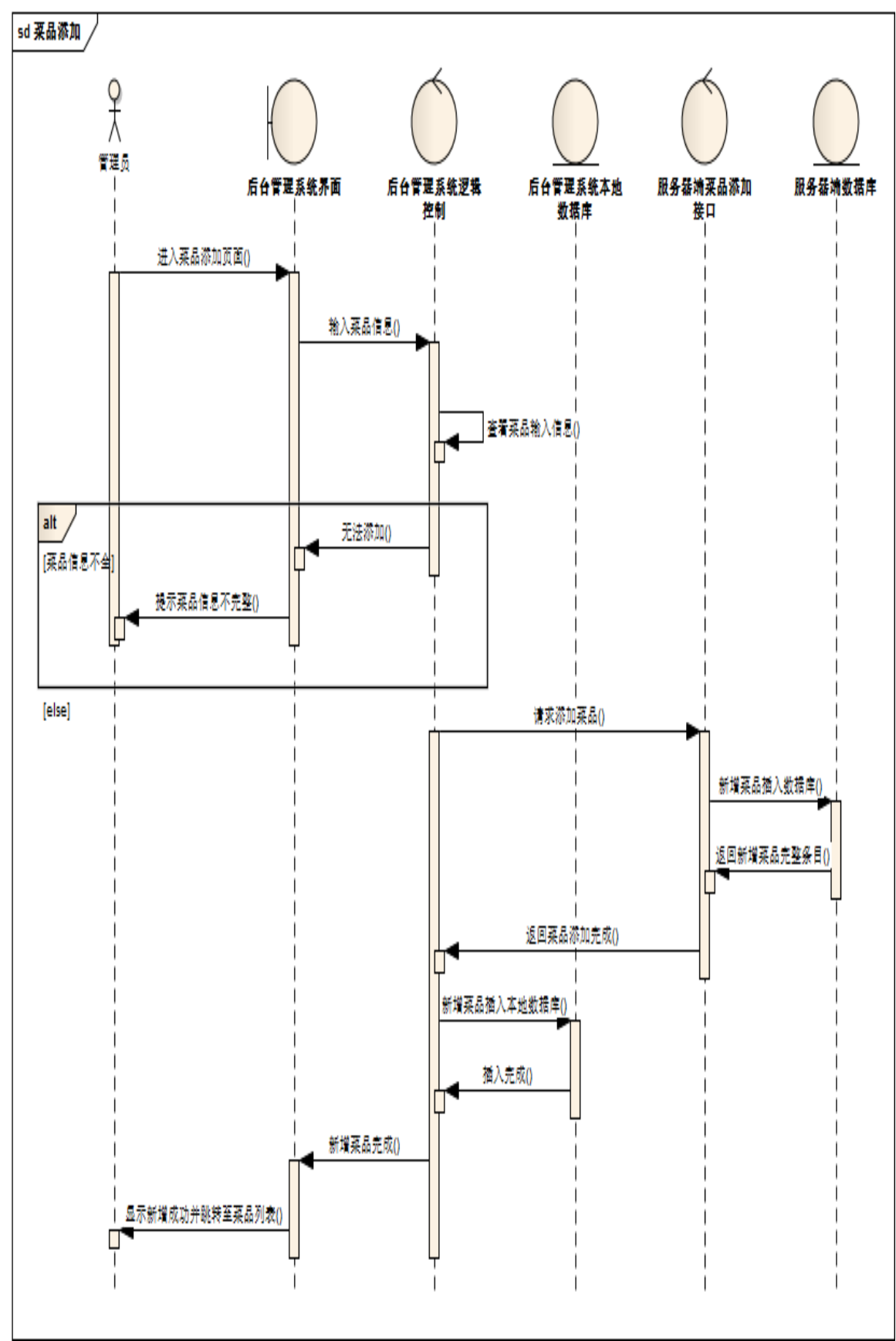


图 2-10 菜品添加流程图

Figure 2-10 Meal Adjuction Sequence Diagram

(9) 菜品修改

管理员对菜品的价格，介绍等进行修改，详细流程如图 2-11 所示。

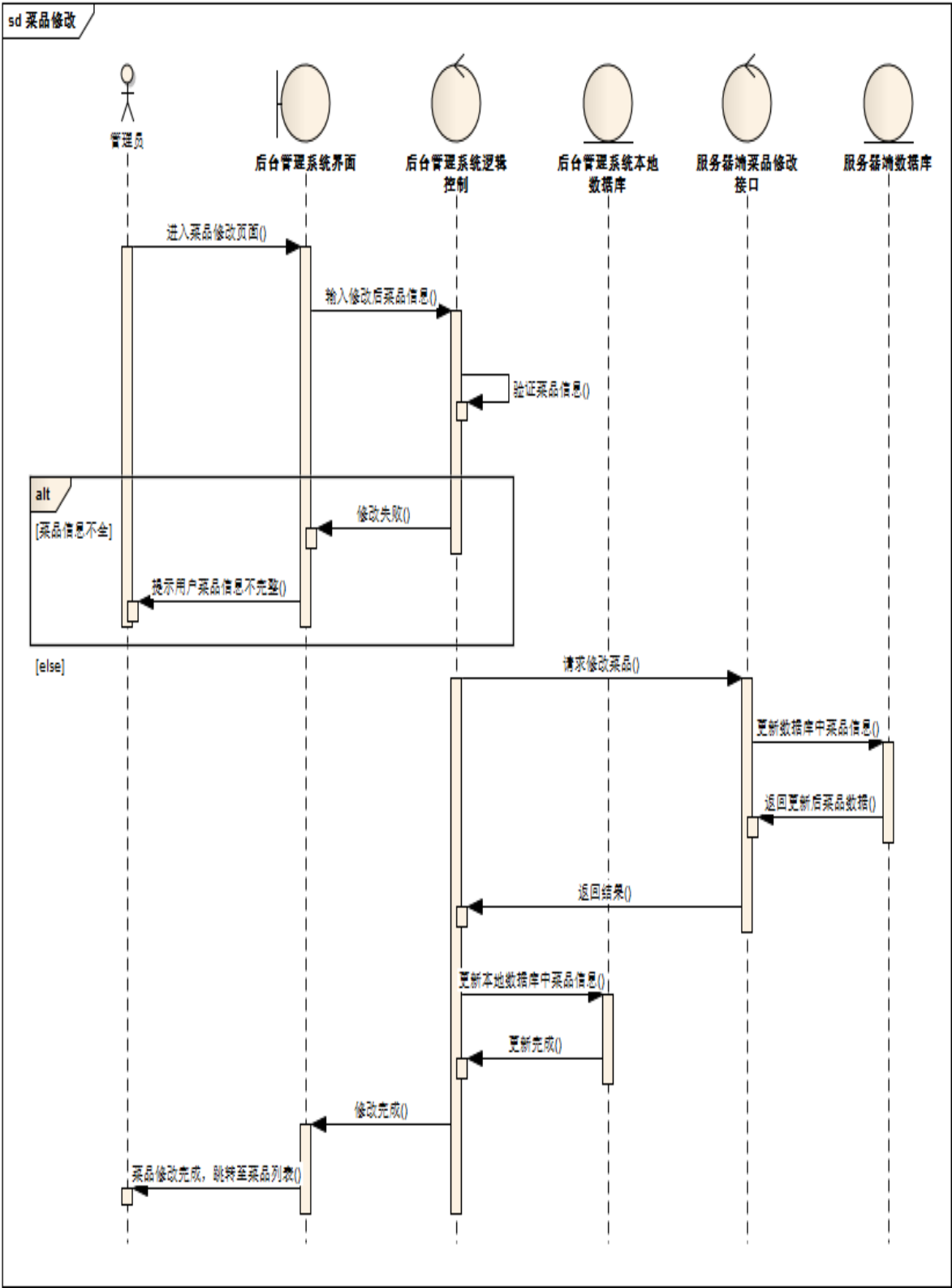


图 2-11 菜品修改流程图

Figure 2-11 Meal Modification Sequence Diagram

(10) 菜品删除

管理员在菜品列表中删除菜品，详细流程如图 2-12 所示。

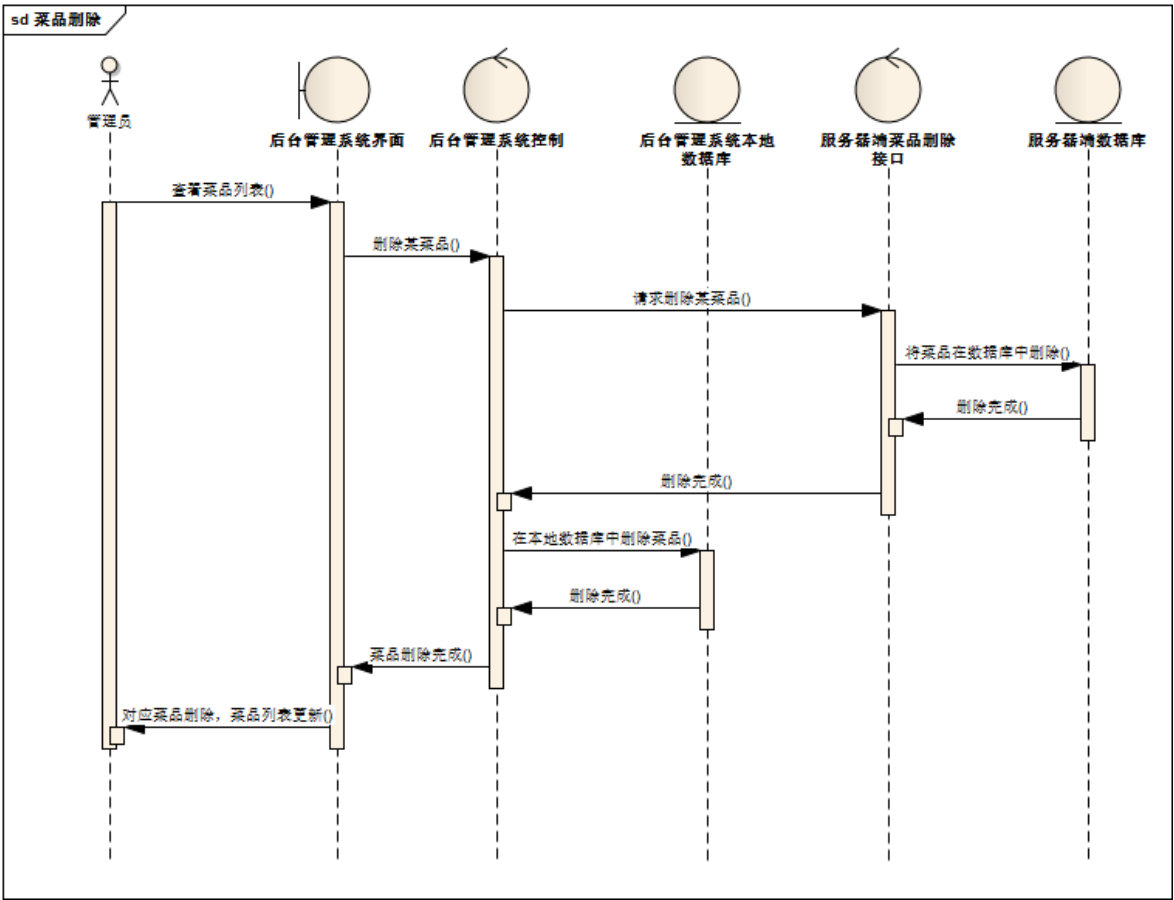


图 2-12 菜品删除流程图

Figure 2-12 Meal Deletion Sequence Diagram

2. 系统结构设计

(1) 服务器端结构设计

1) 总体设计

服务器端整体采用 MVC 框架，由于没有前台页面所以关键则在于 M(Model)和 C(Controllor)。为了更加方便控制于是将服务器端分为以下的结构：控制层，业务逻辑层，DAO(数据访问)层，整体结构如图 2-13 所示。

其中控制层负责接收 Android 客户端的 http 请求，客户端会对对应接口即 Servlet 发送请求，控制层接收请求后，调用业务逻辑层的对应 Service 进行逻辑处理，接着调用 DAO 层对数据库进行访问获得所需数据，接着业务逻辑层根据所得数据再进行逻辑判断处理，最后返回给控制层通过控制层将处理结果返回给客户端。服务器端整体结构设计就是这样的分层结构，这样的结构有利于业务职

责的分配，更有利于功能的添加以及修改，更加灵活地应对需求的变更。

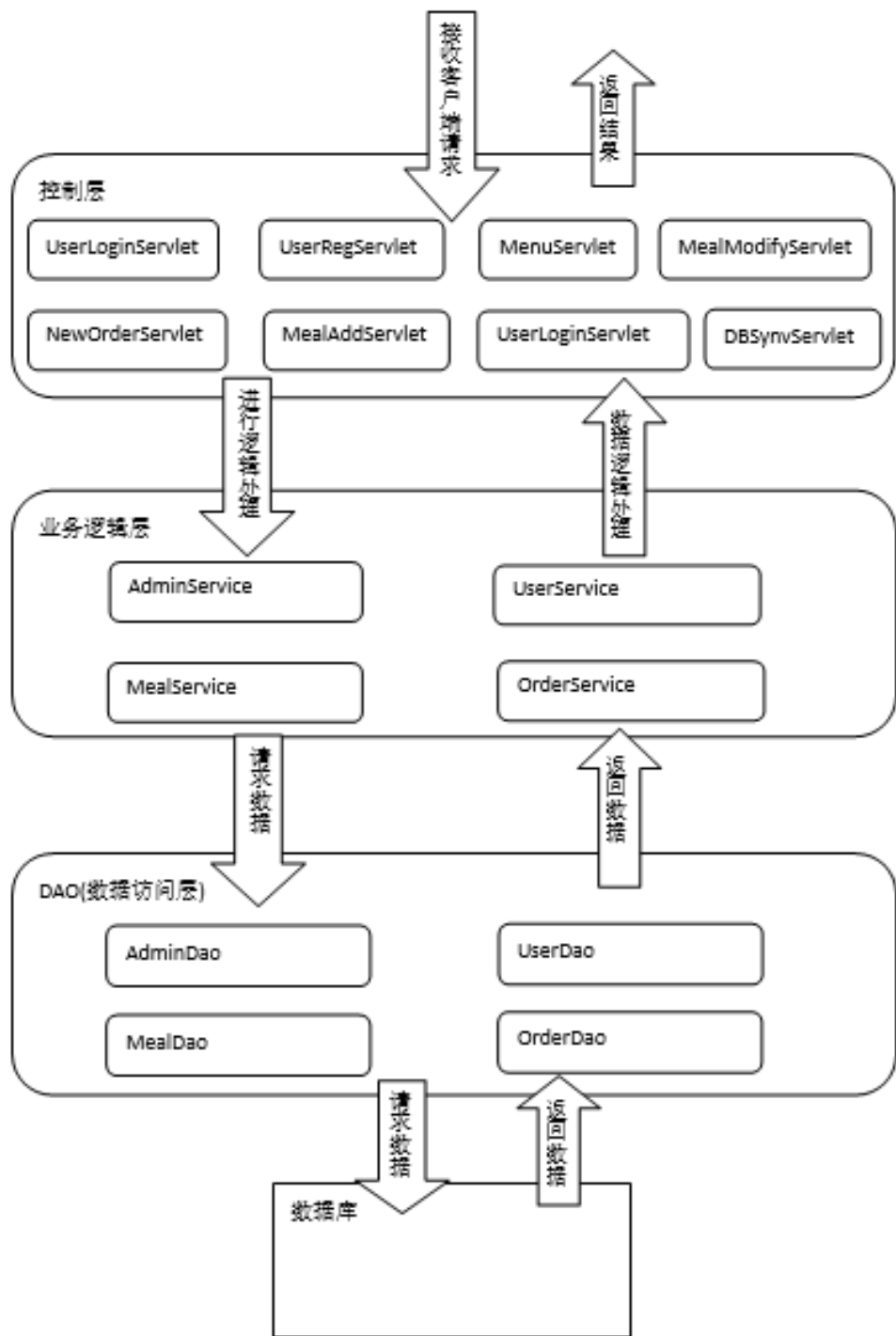


图 2-13 服务器端结构

Figure 2-13 Server Structure Diagram

2) 包设计

根据总体设计对服务器端包进行设计，并对职责进行严格划分，具体描述如表 2-5 所示。项目包设计图如图 2-14 所示。

表 2-5 服务器包设计表

Table 2-5 Server Package Design Table

包名	路径	职责
servlet	com.wirelessorder.servlet	接收客户端请求，并将任务分配给 service 处理
service	com.wirelessorder.service	具体业务逻辑的实现
dao	com.wirelessorder.dao	对数据库进行数据操作
po	com.wirelessorder.po	Java 对象，对应各个实体（如用户，管理员，菜品，订单等）
mapper	com.wirelessorder.mapper	数据管理类，将数据实例化
utils	com.wirelessorder.utils	辅助工具类

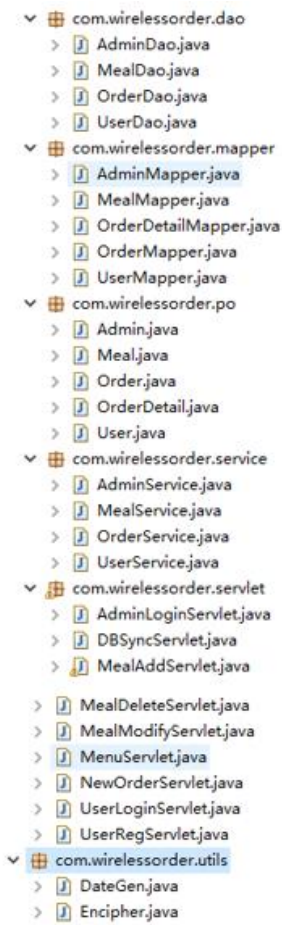


图 2-14 服务器包设计图

Figure 2-14 Server Package Design Diagram

(2) 后台管理系统客户端结构设计

后台管理系统同样建立本地 SQLite 数据库，总体结构与服务器端类似，Android 界面通过 Activity 展示页面与管理员用户进行交互，当涉及一些数据处理时，则通过 Service 进行相关处理并调用 dao 层接口访问数据库，最后返回给 activity 展示给用户。具体包设计如表 2-6 所示。

表 2-6 客户端包设计表

Table 2-6 Client Package Design Table

包名	路径	职责
func	com.wirelessorder.adminsystem.func	Android 页面与用户进行交互
service	com.wirelessorder.adminsystem.service	具体业务逻辑的实现
dao	com.wirelessorder.adminsystem.dao	对数据库进行数据操作
po	com.wirelessorder.adminsystem.po	Java 对象，对应各个实体（如用户，管理员，菜品，订单等）
utils	com.wirelessorder.adminsystem.utils	辅助工具类

(3) 系统总体结构设计

系统总体设计为多个客户端都可以通过 HTTP 请求与主服务器端进行通讯，客户端之间也可以通过 ContentProvider 进行数据访问。具体部署如图 2-15 所示。

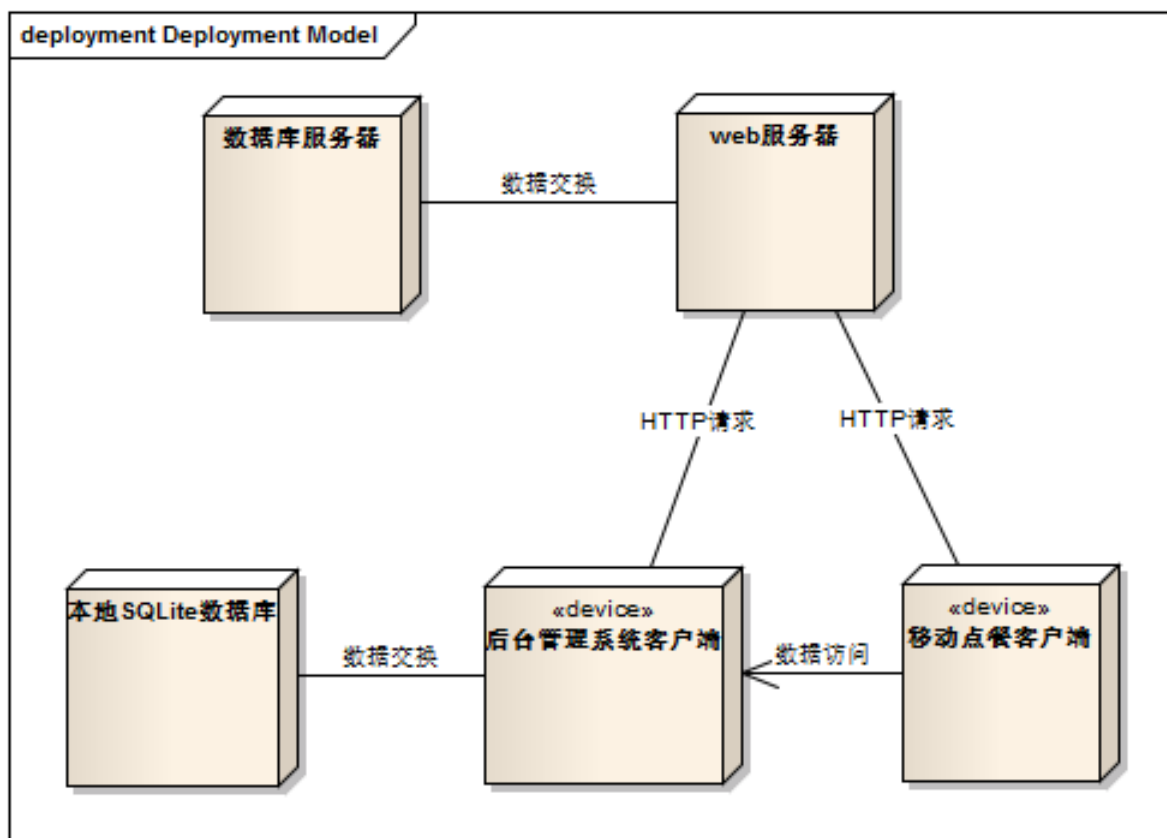


图 2-15 系统部署图

Figure 2-15 System Deployment Diagram

3. 系统数据库设计

(1) 数据库的选择

本系统服务器端数据库选择 MySQL 数据库，MySQL 数据库是现在比较流行的一款关系型数据库，因为它的很多优点如小，快，成本低而被很多企业所应用，为了节省本系统的成本^[11]，故在服务器端也选择了 MySQL 数据库。Android 后台管理系统客户端的数据库选择，则因为 SQLite 在 Android 中的应用便捷性被本系统选用，从而能为移动点餐客户端提供更简便的本地数据来源。

(2) 数据库 E-R 模型

针对本系统进行需求分析后，主要有以下几种实例对象：用户，管理员，菜品，订单。数据库 E-R 模型如图 2-16 所示。

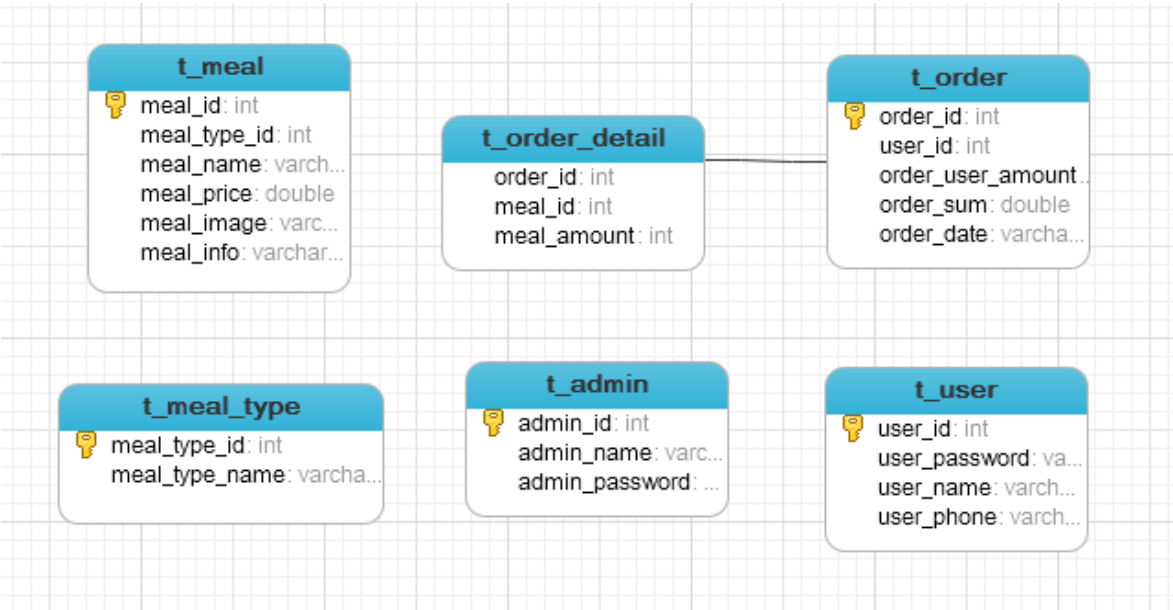


图 2-16 E-R 图

Figure 2-16 E-R Diagram

(3) 数据库表设计

用户表：存储用户的详细信息，如用户名，密码，手机号等，而用户 id 则用来唯一标识一个用户条目。如图 2-17 所示。

名	类型	长度	小数点	不是 null	
user_id	int	11	0	<input checked="" type="checkbox"/>	1
user_password	varchar	255	0	<input checked="" type="checkbox"/>	
user_name	varchar	255	0	<input checked="" type="checkbox"/>	
user_phone	varchar	255	0	<input checked="" type="checkbox"/>	

图 2-17 t_user 表设计

Figure 2-17 Database Table t_user Design

管理员表：主要存储管理员信息，主要包括用户名，密码。如图 2-18 所示。

名	类型	长度	小数点	不是 null	
admin_id	int	11	0	<input checked="" type="checkbox"/>	1
admin_name	varchar	255	0	<input checked="" type="checkbox"/>	
admin_password	varchar	255	0	<input checked="" type="checkbox"/>	

图 2-18 t_admin 表设计

Figure 2-18 Database Table t_admin Design

菜品表：主要存储菜品的相关信息，包括菜品的名称，价格等重要属性，具体设计如表 2-19 所示。


名	类型	长度	小数点	不是 null	
meal_id	int	11	0	<input checked="" type="checkbox"/>	 1
meal_type_id	int	11	0	<input checked="" type="checkbox"/>	
meal_name	varchar	255	0	<input checked="" type="checkbox"/>	
meal_price	double	0	0	<input checked="" type="checkbox"/>	
meal_image	varchar	255	0	<input type="checkbox"/>	
meal_info	varchar	255	0	<input type="checkbox"/>	

图 2-19 t_meal 表设计

Figure 2-19 Database Table t_meal Design

菜品类型表：存储菜品的类型，与菜品表关联，具体设计如图 2-20 所示。


名	类型	长度	小数点	不是 null	
meal_type_id	int	11	0	<input checked="" type="checkbox"/>	 1
meal_type_name	varchar	255	0	<input checked="" type="checkbox"/>	

图 2-20 t_meal_type 表设计

Figure 2-20 Database Table t_meal_type Design

订单表：存储某用户点餐下的订单，包括了一些主要信息，就餐人数金额等，具体设计如图 2-21 所示。


名	类型	长度	小数点	不是 null	
order_id	int	11	0	<input checked="" type="checkbox"/>	 1
user_id	int	11	0	<input checked="" type="checkbox"/>	
order_user_amount	int	11	0	<input type="checkbox"/>	
order_sum	double	0	0	<input checked="" type="checkbox"/>	
order_date	varchar	255	0	<input checked="" type="checkbox"/>	

图 2-21 t_order 表设计

Figure 2-21 Database Table t_user Design

订单详情表：详细记录某订单包含哪些菜品，具体设计如图 2-22 所示。

名	类型	长度	小数点	不是 null	
order_id	int	11	0	<input checked="" type="checkbox"/>	
meal_id	int	11	0	<input checked="" type="checkbox"/>	
meal_amount	int	11	0	<input checked="" type="checkbox"/>	

图 2-21 t_order_detail 表设计

Figure 2-21 Database Table t_order_detail Design

(4) 数据库连接

本系统服务器端与数据库的连接使用 Spring 框架中的数据库模块。通过 Spring 的 xml 配置文件对数据库连接的数据源进行配置从而连接数据库，最后通过 JdbcTemplate 类的相关方法对数据库进行诸多操作。

三、 系统实现

针对主要功能模板以及主要技术进行实现描述。

(一)系统开发环境及方法工具

用途	环境及方法工具
操作系统	Windows10, Ubuntu14.04
版本控制	git
开发语言	Java
服务器端开发环境	MyEclipse10
服务器端数据库	MySQL
Android 客户端开发环境	Android studio
服务器端	Tomcat
服务器部署硬件环境	阿里云服务器
建模工具	EA

(二)MYSQL 数据库连接

在本服务器端，使用 Spring 框架进行数据库连接，首先通过 Spring 的 xml 配置数据源，主要设置数据库的连接地址，用户名密码等信息。

```
<bean id="dataSource"
    class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="com.mysql.jdbc.Driver" />
    <property name="url"
        value="jdbc:mysql://56d6b10631055.sh.cdb.myqcloud.com:4003/wireless_order?characterEncoding=utf8" />
    <property name="username" value="cdb_outheroot" />
    <property name="password" value="metis123456" />
</bean>
```

在需要连接数据库的地方，新建JdbcTemplate的实例

```
JdbcTemplate jdbcTemplate = new JdbcTemplate(this.dataSource);
```

通过这个对象即可对数据库进行连接以及数据操作。

(三)Android 后台管理系统 SQLite 数据库建立

Android 客户端需要建立数据库则必须通过继承 SQLiteOpenHelper 实现, 本 android 应用在第一次打开时便会创建数据库, 通过调用重写其中的 onCreate() 方法, 实现数据库表的建立, 数据库表与服务器端保持一致^[17]。

本地数据库建立完成后, 由于 Android 开发的便利, 直接使用对应的 api 调用数据库即可进行数据库连接以及操作。

(四)ContentProvider 为其它应用提供数据访问接口

Android 后台管理系统客户端为了给移动点餐客户端提供一种本地的稳定的统一的数据访问方式, 通过 Android 重要组件 ContentProvider 对 SQLite 数据进行访问并以接口的形式提供给其它 app 使用, 根据 Android 的要求, 首先声明 ContentProvider 的 Authority, 之后使用接口 UriMatcher 对 Uri 进行匹配, 只有匹配了得 Uri 才能正确使用接口并获得数据, 本系统实现的 WirelessProvider 继承自 ContentPrvider 为移动点餐系统提供 query, insert 等接口, 使之能够获取到本地的数据。非常便捷。

(五)Volley 框架的使用

本系统使用的是 C/S 模式, 客户端与服务器端之间经常要进行网络通信, 客户端会经常发送 HTTP 请求, Android 的 HTTP 类库由于其实现的复杂以及需要另开线程实现请求的原因, 没有被我采用, 本系统使用的是一个较完备且方便易用的 Volley 框架, 使用他进行 HTTP 请求方便快捷容易上手, 为了使用该框架, 首先需要导入 volley 对应 jar 包, 成功后便可使用对应方法。

Volley 网络通信框架分为三层, 底层的为基础的 HTTP 网络通信的实现; 第二层负责对网络请求进行管理, 即按顺序请求; 最外层则为具体的网络请求的封装, 可以自定义各种请求^[15], 就是本系统主要使用到的一层。

本系统客户端对服务器端的请求主要包括一些数据的请求, 还有服务器端图片的请求, 而 Volley 全部帮我们封装好了工具类便于我们开发, 比如通过 StringRequest, ImageRequest 获取 String 类文本数据以及图片数据, 非常方便。

(六)用户注册功能的实现

UserRegServlet 接收移动点餐客户端的 post 请求, 并获取相应参数。

```
String userName = request.getParameter("userName");
```

```
String userPassword = request.getParameter("password");
```

```
String userPhone = request.getParameter("phone");
```

首先对用户名进行验证，通过 UserService 逻辑处理判断用户名是否存在。

`userService.isUserExists(userName)` 该函数通过对数据库进行查询是否有该用户名进行验证：

```
public boolean isUserExists(String userName) {  
    String sqlString = "SELECT count(*) FROM t_user WHERE user_name=?";  
    int count = jdbcTemplate.queryForInt(sqlString, new Object[]{userName});  
    return count > 0;  
}
```

当用户名存在时则返回添加失败的 json 信息，否则根据获得的注册参数（用户名，密码等）实例化 User 对象并新增用户：

```
public int addNewUser(final User user) {  
    final String sqlString = "INSERT INTO t_user (user_name, user_password,  
user_phone) VALUES (?, ?, ?)";  
    KeyHolder keyHolder = new GeneratedKeyHolder();  
    jdbcTemplate.update(new PreparedStatementCreator() {  
  
        @Override  
        public PreparedStatement createPreparedStatement(Connection conn)  
            throws SQLException {  
            PreparedStatement ps = conn.prepareStatement(sqlString,  
Statement.RETURN_GENERATED_KEYS);  
            ps.setString(1, user.getUserName());  
            ps.setString(2, Encipher.encrypt(user.getUserPassword()));  
            ps.setString(3, user.getUserPhone());  
            return ps;  
        }  
    }, keyHolder);  
    return keyHolder.getKey().intValue();  
}
```

用户添加完成则表示注册成功，获取用户 id 并返回给客户端。

(七)用户登录功能的实现

UserLoginServlet 接收用户 post 请求，获取对应用户名及密码信息，通过 UserService 对用户名及密码匹配情况进行逻辑验证，

```
public boolean isUserMatch(String userName, String userPassword) {  
    int count = userDao.getMatchUserCount(userName, userPassword);  
    return count > 0;  
}
```

UserDao 则对该用户密码在数据库中进行查询，查询是否有匹配的数据。

```
public int getMatchUserCount(String userName, String userPassword) {  
    String sqlStr = "SELECT count(*) FROM t_user WHERE user_name=? AND  
user_password=?";  
    return jdbcTemplate.queryForInt(sqlStr, new Object[] {userName,  
Encipher.encrypt(userPassword)});  
}
```

当查询数据中有对应条目时说明用户名密码匹配，用户登录时成功的，否则登录失败，于是在 UserLoginServlet 中将登陆结果以 json 的形式返回给客户端：

```
User user = userService.getUser(userName, password);  
response.setContentType("text/html;charset=UTF-8");  
PrintWriter out = response.getWriter();  
JSONObject result = new JSONObject();  
try {  
    result.put("loginSuccess", 1);  
    result.put("user_id", user.getUserId());  
    out.write(result.toString());  
    out.flush();  
    out.close();  
} catch (JSONException e) {  
    // TODO Auto-generated catch block  
    e.printStackTrace();  
}
```



```
}
```

(八)查看菜单功能的实现

MenuServlet 接收移动点餐客户端的 post 请求，并获得所需的菜品类型。

通过 MealService 根据菜品类型使用 MealDao 来获取对应菜品数据：

```
public List<Meal> getAllMealByType(int mealType) {  
    String sqlString = "SELECT * FROM t_meal WHERE meal_type_id=?";  
    List<Meal> mealList = null;  
    mealList = jdbcTemplate.query(sqlString, new Object[]{mealType}, new  
MealMapper());  
    return mealList;  
}
```

菜品数据获取后，再将数据整理封装为 json 返回给客户端：

```
List<Meal> mealList = mealService.getAllMealByType(mealType);  
for (Meal meal : mealList) {  
    JSONObject mealItem = new JSONObject();  
    mealItem.put("meal_id", meal.getMealId());  
    mealItem.put("meal_type", meal.getMealType());  
    mealItem.put("meal_name", meal.getMealName());  
    mealItem.put("meal_price", meal.getMealPrice());  
    mealItem.put("meal_image_url", meal.getMealImage());  
    mealItem.put("meal_info", meal.getMealInfo());  
    resultArray.put(mealItem);  
}
```

(九)点餐下单功能的实现

NewOrderServlet 接收对应 post 请求，首先获得订单的 json 数据并进行解析得到下单的用户 id，就餐人数，总金额，以及订单对应菜品等信息，然后通过 OrderService 对订单进行存储：

```
jsonOrder = new JSONObject(orderString);  
String timeString = DateGen.getTimeStr();  
int userId = jsonOrder.getInt("userId");
```

```
Order order = new Order(userId, jsonOrder.getInt("userAmount"),
    jsonOrder.getDouble("orderSum"), timeString);
int orderId = orderService.addOrder(order);

JSONArray mealList = jsonOrder.getJSONArray("mealList");
for (int i = 0; i < mealList.length(); i++) {
    JSONObject jsonMeal = mealList.getJSONObject(i);
    int mealId = jsonMeal.getInt("mealId");
    int mealAmount = jsonMeal.getInt("mealAmount");
    orderService.addMealToOrder(orderId, mealId, mealAmount);
}
```

其中 addOrder 函数和 addMealToOrder 函数分别将订单一般信息插入 t_order 表以及将订单详细菜品信息插入 t_order_detail 表。

(十)同步数据功能的实现

该功能在首次打开后台管理应用或者管理员选择都可以进行，首次打开应用在欢迎界面即同步数据，如图 3-1 所示。



图 3-1 数据同步界面

Figure 3-1 Data Synchronization Interface

应用通过 Volley 框架向服务器端发送请求获取所有数据，

```
private void getDataFromServer() {  
    String url = "http://www.zhouzezhou.site/WirelessOrder/servlet/DBSyncServlet";  
    RequestQueue requestQueue = Volley.newRequestQueue(mContext);  
    StringRequest stringRequest = new StringRequest(Request.Method.POST, url,  
        new Response.Listener<String>() {  
            @Override  
            public void onResponse(String s) {  
                Utils.syncData(mContext, s);  
                pDialog.dismiss();  
                mSp.edit().putBoolean("firstStartup", false).commit();  
            }  
        }, new Response.ErrorListener() {  
            @Override  
            public void onErrorResponse(VolleyError volleyError) {  
                Toast.makeText(mContext, "同步失败",  
                    Toast.LENGTH_SHORT).show();  
                finish();  
            }  
        });  
    requestQueue.add(stringRequest);  
}
```

而服务器端 DBSyncServlet 则接收到了请求，通过

UserService,MealService,AdminService,OrderService 获取到所有数据并组织成 json 数据返回给客户端。客户端成功拿到数据以后，对本地数据库进行操作，清空所有表并重新插入数据。

```
JSONArray userArray = resultJson.getJSONArray("t_user");  
JSONArray adminArray = resultJson.getJSONArray("t_admin");  
JSONArray mealArray = resultJson.getJSONArray("t_meal");  
JSONArray orderArray = resultJson.getJSONArray("t_order");
```

```
JSONArray orderDetailArray = resultJson.getJSONArray("t_order_detail");
```

```
userService.insertUsers(userArray);
```

```
adminService.insertAdmins(adminArray);
```

```
mealService.insertMeals(mealArray);
```

```
orderService.insertOrders(orderArray);
```

```
orderService.insertOrderDetails(orderDetailArray);
```

(十一) 管理员登录及餐厅数据统计的实现

管理员进入 app 的登录页面，输入用户名及密码，具体界面如图 3-2 所示。



图 3-2 管理员登录界面

Figure 3-2 Admin Log in Interface

用户名及密码填写完成后登录，通过 AdminService 进行业务逻辑处理，AdminDao 在数据库中查询对应管理员条目是否存在，如果不存在则弹出 Toast 提示管理员用户名或密码错误，否则登录成功进入系统。

进入系统后,系统自动统计餐厅信息,通过 OrderService, UserService, MealService 对用户数, 餐厅菜品数, 订单总数和金额进行统计, 主要通过 Dao 层获取所有数据, 对数据进行计数, 求和等操作得到统计信息。具体统计界面如图 3-3 所示。

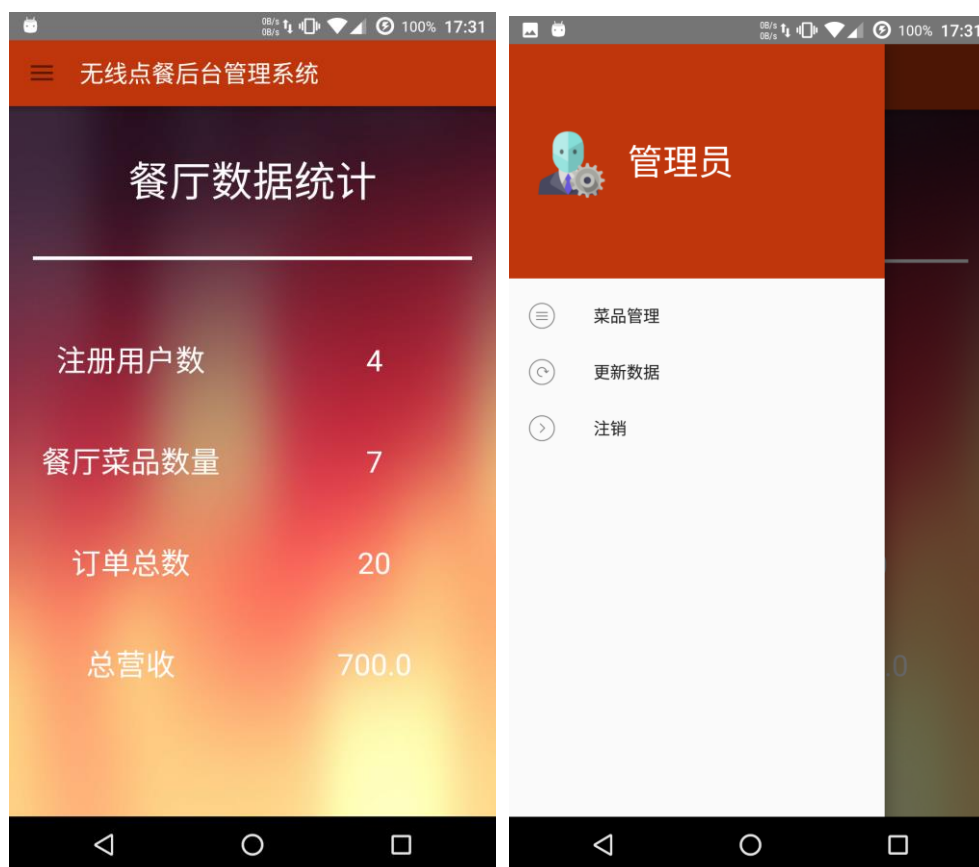


图 3-3 数据统计界面及菜单

Figure 3-3 Data Statistics Interface and Menu

(十二) 菜品修改功能的实现

管理员选择菜品管理即进入菜品管理页面, 菜品管理页面通过 android 的 RecyclerView 配合 CardView 进行使用, 进入页面首先通过 MealService 调用 MealDao 访问 SQLite 数据库获取所有菜品的数据, 然后将数据全部实例化为 Meal 对象存储在一个 list 当中 `mealList = (ArrayList) mealService.getAllMeals();`, RecyclerView 则将该菜品列表设为自定义的 Adapter, 实现菜品的列表视图, 通过 Volley 的控件 NetworkImageView 请求服务器端的图片, 并对每一个菜品条目显示相应信息, 设置监听事件等。具体界面如图 3-4 所示。

// 给 ViewHolder 设置元素

```
final Meal p = mealList.get(i);
```

```
RequestQueue requestQueue = Volley.newRequestQueue(mContext);
```

```
ImageLoader imageLoader = new ImageLoader(requestQueue, new  
BitmapCache());
```

```
viewHolder.mealName.setText(p.getMealName());
```

```
viewHolder.mealPrice.setText("¥ " + String.valueOf(p.getMealPrice()));
```

```
viewHolder.mealImage.setDefaultImageResId(R.mipmap.image_loading);
```

```
viewHolder.mealImage.setErrorImageResId(R.mipmap.error);
```

```
viewHolder.mealImage.setImageUrl(p.getMealImage(), imageLoader);
```



图 3-4 菜品管理界面

Figure 3-4 Meal Management Interface

点击菜品条目会弹出菜品编辑对话框如图 3-5 所示。

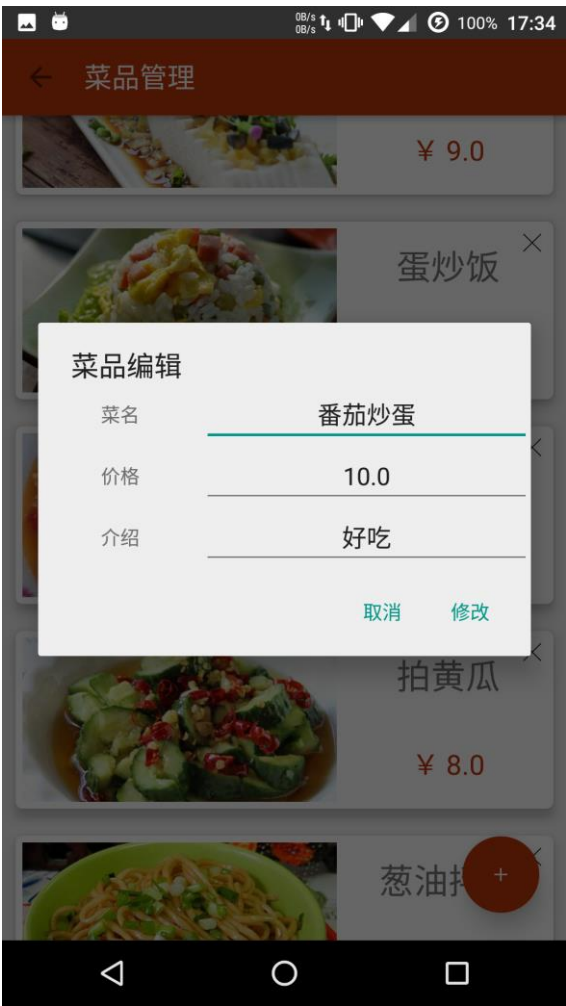


图 3-5 菜品编辑界面

Figure 3-5 Meal Edit Interface

管理员输入相应菜品信息后选择修改，客户端即通过 Volley 新建 StringRequest 发送 HTTP 请求，服务器端 MealModifyServlet 接收到请求并获取到修改后的菜品信息，在服务器端数据库中更新菜品，客户端接收到成功回调后，便通过 MealService 对本地数据也进行更新。

(十三) 菜品添加功能的实现

在菜品管理界面点击右下方的“+”号按钮即可进入菜品添加页面，页面如图 3-6 所示。点击选择图片，调用系统相册选取图片，再填写完整菜品信息，填写完成后点击添加，客户端便会通过 MultiPartEntity 存储信息向服务器端发送菜品添加请求：

```
MultipartEntity entity = new MultipartEntity();  
entity.addPart("mealType", new  
StringBody(String.valueOf(meal.getMealType())));
```

```
entity.addPart("mealName", new StringBody(meal.getMealName(),  
Charset.forName("UTF-8")));  
entity.addPart("mealPrice", new  
StringBody(String.valueOf(meal.getMealPrice())));  
entity.addPart("mealInfo", new StringBody(meal.getMealInfo(),  
Charset.forName("UTF-8")));  
entity.addPart("mealImage", new FileBody(new File(imagePath)));  
httpPost.setEntity(entity);
```

服务器端 MealAddServlet 接收到请求，接收到对应信息并根据数据类型进行分类处理（文本数据，图片数据），接着便将菜品条目插入到数据库中，同时将图片存储在服务器端目录。客户端接收到服务器端成功的 response 后，便将菜品存储至本地 SQLite 数据库，并返回菜品管理界面。



图 3-6 菜品添加界面

Figure 3-6 Meal Adjunction Interface

(十四) 菜品删除功能的实现

当点击菜品管理界面每个菜品条目右上角的“x”后，便会弹出确认删除对话框，如图 3-7 所示。



图 3-7 菜品删除界面

Figure 3-7 Meal Deletion Interface

当确定删除时，客户端会发送 HTTP 请求给服务器，服务器端 MealDeleteServlet 接收请求，获取到菜品的 ID，于是便通过 MealService 使用 MealDao 在数据库中进行菜品的删除。删除完成后，客户端得到回调，在本地 SQLite 数据库中也同时删除对应条目。

四、 系统测试

软件测试在本系统开发生命周期中是一个不可或缺的步骤,通过软件测试可以发现本系统存在的众多缺陷从而保证产品的质量^[13],在本文中针对本系统的几个重要功能进行黑盒测试,从用户的角度来确保本系统能够正常流畅地使用,提高用户体验。

(一)功能测试

本部分针对本系统的部分重要功能进行功能测试,确保功能的完整可用性。

针对服务器端功能,通过简单实现 jsp 页面对各模块进行功能测试。

1. 用户注册功能

测试用例	预期结果	实际结果
输入已存在的用户名,密码,手机号注册	注册失败	返回 regSuccess=0,注册失败,符合预期
输入不存在的用户名,密码,手机号注册	注册成功,数据库新增用户条目	返回 regSuccess=1 以及 user_id,注册成功,数据库中新增条目,符合预期

2. 用户登录功能

测试用例	预期结果	实际结果
输入用户名及错误密码登录	登录失败	返回 loginSuccess=0,登录失败,符合预期
输入正确的用户名及密码进行登录	登录成功	返回 loginSuccess=1 以及 user_id,登录成功,符合预期

3. 查看菜单功能

测试用例	预期结果	实际结果
菜单类型为 0 即全部	获取所有菜品数据	返回所有菜品数据,符合预期
菜单类型为 1 即冷菜	获取所有冷菜数据	返回菜品均为冷菜,符合预期

4. 点餐下单功能

测试用例	预期结果	实际结果
模型订单 json 数据进行下单	下单成功, 数据库中新增订单条目	返回一个包括 order_id 等字段的 json 数据, 表示下单成功, 数据库中新增了订单条目, 符合预期

5. 同步数据功能

测试用例	预期结果	实际结果
初次打开数据库	跳转至登录界面, 进入客户端在 Android 终端的 data 目录中新增了数据库以及数据	跳转至登录界面, 进入应用对应目录下查看数据库存在且有数据与服务器一致, 符合预期。
服务器端数据更新, 客户端手动选择同步数据	同步成功, 客户端数据与服务器端数据一致	弹出同步成功 toast, 查看两边数据一致。

6. 管理员登录

测试用例	预期结果	实际结果
用户名或密码不输入, 选择登录	无法登录, 需要完整信息	弹出“请输入完整信息” toast, 无法登录, 符合预期
用户名密码不匹配输入, 选择登录	无法登陆, 提示用户名或密码错误	弹出“用户名或密码错误” toast, 登录失败, 符合预期
输入正确用户名密码登录	登录成功, 跳转至餐厅数据统计页面	符合预期

7. 菜品添加功能

测试用例	预期结果	实际结果
菜品信息不填写完整, 选择添加	无法添加, 弹出“输入完整信息” toast	符合预期

输入完整菜品信息进行添加	添加成功，跳转至菜单列表，看到新增菜品，数据库中对新增对应菜品	符合预期
--------------	---------------------------------	------

8. 菜品修改功能

测试用例	预期结果	实际结果
菜品信息不填写完整，选择修改	无法修改，弹出“输入完整信息” toast	符合预期
输入完整菜品信息进行修改	修改成功，跳转至菜单列表，看到菜品信息已修改，数据库中对对应菜品条目信息修改	符合预期

9. 菜品删除功能

测试用例	预期结果	实际结果
点击删除按钮选择删除	菜品删除成功，列表中对对应菜品消失，数据库中对对应菜品条目删除	符合预期

(二)测试总结

在最后的测试阶段，主要使用了黑盒测试的方法针对系统的主要功能模块进行了功能性测试，通过这些测试验证了功能的完整可用性，保证用户在正常情况下使用本系统时能够没有 crash 没有 bug，正常流畅使用。

而针对系统的性能，代码并没有搭建测试框架进行完备的自动化测试，所以对系统的性能方面只从使用角度对其性能进行模糊的判断，保证不影响使用，但是没有对此进行量化。

所以后续测试在时间允许的情况下需要使用一些较为成熟的测试框架，对代码进行自动化测试，保证性能以及代码正确性。

五、 总结和展望

本论文通过对目前餐饮行业普遍面临的问题以及行业前景进行调查,再结合调查 Android 在近几年的市场份额的数据分析,了解到互联网化和移动化是今后的大势所趋,因此设计实现了这款基于 Android 的移动点餐系统服务器端,该系统可以满足客户端的网络请求,并且为餐厅管理员提供餐厅数据的可视化统计,菜品管理等功能,为普通餐厅顾客以及餐厅管理人员都提供了更加便捷新颖的就餐和管理方式,具有积极的意义。本系统的开发是一个分析客户需求,进行长远设计,发现并解决问题的过程,需求的变更为系统开发带来一定的难度,但通过长期的坚持最终能够完成系统并上手使用,本论文所做工作如下:

1. 收集客户需求,对客户需求进行建模并详细分析,严密设计出系统架构,开发的包设计,数据库表设计以及各个主要功能模块的流程。

2. 学习使用 Spring 部分框架功能并应用于 web 项目,购买阿里云服务器并成功将 web 项目部署于服务器,便于客户端进行访问。学习并进行服务器端开发,成功为移动点餐客户端提供用户注册,用户登录,菜单获取,点餐下单,菜品修改,删除,添加等接口。

3. 结合最新 Material Design 风格设计实现 Android 后台管理系统,对 Android 应用开发进行分层设计,便于开发和应对后续的需求变更。为餐厅管理员提供后台管理系统实现菜品管理,数据同步,数据统计等功能,并为移动点餐客户端提供数据的第二种获取方式。

4. 对本系统各大功能模块进行较为完备的黑盒测试,保证系统的正常使用,提高产品质量。

系统的整体进度及完成情况符合预期,能够投入到正常使用中。但还是有很多不足的地方需要改进,比如后台管理系统界面可以更加美化,服务器端响应时间可以更快,服务器端与后台管理系统客户端之间的数据同步应该更加方便智能。针对这些不足,在未来还有很多工作可以进行,可以从应用推送,UI 设计,系统优化等多个角度进行学习并应用修改于本系统,争取使本系统能够更加完美。

参考文献

- [1] 冯俊,刘倩洁. 我国餐饮研究现状分析[J]. 北京工商大学学报(社会科学版), 2012, 04:104.
- [2] 栗鑫林. 基于移动手机平台的智能点餐系统的设计与实现[D]. 电子科技大学, 2014:1-2
- [3] 徐光争. 无线点菜渐成主流[J]. 计算机世界, 2005, 211-214
- [4] Bob Hughes,Mike Cotterell. Software Project Management[M]. China Machine Press,2009
- [5] 蔡长安,王盈璞. C/S 和 B/S 的模式比较和选择[J]. 渭南师范学院学报, 2006, 21 (2) :47-48
- [6] 侯淑英. B/S 模式和 C/S 模式优势比较 [J]. 沈阳教育学院学报, 2007, 9 (2) :98-99
- [7] A Silberschatz,HF Korth,S Sudarshan. Database System Concepts,5th Edition. [M]. China Machine Press,2008
- [8] 池亚平, 方勇. Servlet 技术与应用方法 [J]. 北京邮电大学学报, 2003, 26:138-139
- [9] 吴晨清, 荣震华. 用 JSP/Servlet 技术构建 Web 应用 [J]. 计算机工程, 2001, 27 (1) :170
- [10] 樊振宇. 深入理解 SERVLET 和 JSP 原理 [J]. 电脑知识与技术, 2011, 7 (11) :2572
- [11] 张恒喜, 史争军. 基于 SQLite 的 Android 数据库编程 [J]. 电脑编程技巧与维护, 2011, 21:30
- [12] 倪凯, 夏海波, 魏建明, 程嘉昇, 李焱. 一种移动终端远程数据访问控制方法 [J]. 计算机应用与软件, 2012, 29 (6) :230-232
- [13] 王治国, 王捷. 精通 Android 应用开发 [M]. 清华大学出版社, 2014, 211-247
- [14] 魏松, 贺丹娜. 基于 MYSQL 的学生信息管理系统数据库设计 [J]. 软件设计开发, 2012, 14:207-209
- [15] 孟远. Android 网络通信框架 Volley 的解析和比较 [J]. 软件, 2014, 12: 66-68
- [16] 李宁, 李战怀. 基于黑盒测试的软件测试策略研究与实践 [J]. 计算机应用研究, 2009, 26 (03) :923-924
- [17] 倪红军. 基于 Android 系统的数据存储访问机制研究[J]. 计算机技术与发展, 2013, 23 (6) :90-93
- [18] 李小双. 国内 12 家餐饮软件服务商介绍[E]. <http://www.iyiou.com/p/4953>, 2014-05-04
- [19] Horstmann Cay S.,Gary Cornell. Core Java (Volume I--Fundamentals 9th Edition)[M]. Electronic Industry Press,2011
- [20] Bruce Eckle. Thinking in Java[M]. Upper Saddle River, New Jersey, USA: Prentice Hall, 2006
- [21] 杨文志. Google Android 程序设计指南 [M]. 电子工业出版社, 2009
- [22] 王飞, 张有志. 一种新型的电子点菜系统[J]. 电子技术应用, 2004 (06) :55-59

- [23] 彭宇雨. 无线自主点餐系统集成设计[D]. 北方工业大学, 2012:23-29
- [24] 李凡生, 戴小廷, 王洪伟. 餐饮企业管理系统的分析[J]. 电脑与信息技术, 2004(05) 22-28

附录

1. 服务器端 Spring，数据源配置

//配置数据库 url 以及用户名密码等信息，从而连接数据库

```
<bean id="dataSource"
    class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="com.mysql.jdbc.Driver" />
    <property name="url"

    value="jdbc:mysql://56d6b10631055.sh.cdb.myqcloud.com:4003/wireless_order?char
acterEncoding=utf8" />
    <property name="username" value="cdb_outerroot" />
    <property name="password" value="metis123456" />
</bean>

<bean id="userDao"
    class="com.wirelessorder.dao.UserDao">
    <property name="dataSource" ref="dataSource" />
</bean>

<bean id="mealDao"
    class="com.wirelessorder.dao.MealDao">
    <property name="dataSource" ref="dataSource" />
</bean>

<bean id="orderDao"
    class="com.wirelessorder.dao.OrderDao">
    <property name="dataSource" ref="dataSource" />
</bean>

<bean id="adminDao"
    class="com.wirelessorder.dao.AdminDao">
    <property name="dataSource" ref="dataSource" />
</bean>
```


2. 服务器端用户登录接口

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    request.setCharacterEncoding("UTF-8");
    String userName = request.getParameter("userName");//获取用户名
    String password = request.getParameter("password");//获取密码

    boolean isValid = userService.isUserMatch(userName, password);//数据库中
    匹配用户信息
    if (isValid) {
        /*登录成功*/
        User user = userService.getUser(userName, password);//获取用户完整信息
        response.setContentType("text/html;charset=UTF-8;");
        PrintWriter out = response.getWriter();
        JSONObject result = new JSONObject();
        try {
            result.put("loginSuccess", 1);
            result.put("user_id", user.getId());
            out.write(result.toString());
            out.flush();
            out.close();
        } catch (JSONException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    } else {
        /*登录失败*/
        response.setContentType("text/html;charset=UTF-8;");
        PrintWriter out = response.getWriter();
        JSONObject result = new JSONObject();
        try {
            result.put("loginSuccess", 0);
            out.write(result.toString());
            out.flush();
            out.close();
        } catch (JSONException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

3. 服务器端用户注册接口

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    request.setCharacterEncoding("UTF-8");
    /*获取注册用户相关信息*/
    String userName = request.getParameter("userName");
    String userPassword = request.getParameter("password");
    String userPhone = request.getParameter("phone");
    User user = new User(userName, userPassword, userPhone);
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    JSONObject result = new JSONObject();

    try {
        if (userService.isUserExists(userName)) { //判断用户名是否已存在
            result.put("regSuccess", 0);
        } else {
            int userId = userService.addUser(user); //添加用户
            result.put("regSuccess", 1);
            result.put("user_id", userId);
        }
        out.write(result.toString());
        out.flush();
        out.close();
    } catch (JSONException e) {
    }
}
```

4. 服务器端菜单获取接口

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    String type = request.getParameter("type"); //获取所需菜品的类型 “0”为所有菜
    品
    int mealType = Integer.parseInt(type);
    response.setContentType("text/html;charset=UTF-8;");
    PrintWriter out = response.getWriter();
    JSONArray resultArray = new JSONArray();

    try {
        if (mealType == TYPE_ALL) {
            List<Meal> mealList = mealService.getAllMeal(); //从数据库中获取菜
            品

            for (Meal meal : mealList) {
                JSONObject mealItem = new JSONObject();
                mealItem.put("meal_id", meal.getMealId());
                mealItem.put("meal_type", meal.getMealType());
                mealItem.put("meal_name", meal.getMealName());
                mealItem.put("meal_price", meal.getMealPrice());
                mealItem.put("meal_image_url", meal.getMealImage());
                mealItem.put("meal_info", meal.getMealInfo());
                resultArray.put(mealItem); //将数据放入 json 数组中
            }
        } else {
            List<Meal> mealList = mealService.getAllMealByType(mealType);
            for (Meal meal : mealList) {
                JSONObject mealItem = new JSONObject();
                mealItem.put("meal_id", meal.getMealId());
                mealItem.put("meal_type", meal.getMealType());
                mealItem.put("meal_name", meal.getMealName());
                mealItem.put("meal_price", meal.getMealPrice());
                mealItem.put("meal_image_url", meal.getMealImage());
                mealItem.put("meal_info", meal.getMealInfo());
                resultArray.put(mealItem);
            }
        }
        out.write(resultArray.toString());
        out.flush();
        out.close();
    } catch (JSONException e) {
        // TODO: handle exception
    }
}
```

5. 服务器端点餐下单接口

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    String orderString = request.getParameter("order");
    JSONObject jsonOrder;
    try {
        /*将获得的菜品数据转化为 json 处理*/
        jsonOrder = new JSONObject(orderString);
        String timeString = DateGen.getTimeStr();
        int userId = jsonOrder.getInt("userId");
        Order order = new Order(userId, jsonOrder.getInt("userAmount"),
            jsonOrder.getDouble("orderSum"), timeString);
        int orderId = orderService.addOrder(order); // 新建订单并获得订单 id

        JSONArray mealList = jsonOrder.getJSONArray("mealList");
        for (int i = 0; i < mealList.length(); i++) {
            JSONObject jsonMeal = mealList.getJSONObject(i);
            int mealId = jsonMeal.getInt("mealId");
            int mealAmount = jsonMeal.getInt("mealAmount");
            orderService.addMealToOrder(orderId, mealId, mealAmount); //将订单
中的详情插入数据库中
        }
        orderService.updateOrderSum(orderService.getOrderById(orderId)); //更新订
单的总金额

        response.setContentType("text/html;charset=UTF-8;");
        PrintWriter out = response.getWriter();
        JSONObject jsonResponse = new JSONObject();
        jsonResponse.put("order_id", orderId);
        jsonResponse.put("user_id", userId);
        jsonResponse.put("user_amount", jsonOrder.getInt("userAmount"));
        jsonResponse.put("order_date", timeString);
        out.write(jsonResponse.toString());
        out.flush();
        out.close();
    } catch (JSONException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

6. 后台管理客户端欢迎界面

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    mContext = this;
    setContentView(R.layout.activity_welcome);
    mSp = getSharedPreferences("app_info", MODE_PRIVATE);
    final boolean isFirstTime = mSp.getBoolean("firstStartup", true);
    /*在 2500 毫秒以后执行操作*/
    new Handler().postDelayed(new Runnable() {

        @Override
        public void run() {
            if (isFirstTime) { //判断是否是第一次启动应用
                if (!Utils.hasNetwork(mContext)) {
                    Toast.makeText(mContext,
getString(R.string.network_error),
                                Toast.LENGTH_SHORT).show();
                    finish();
                }
                progressDialog = ProgressDialog.show(mContext, null, "同步数据库",
true, true);

                progressDialog.setCancelable(false);
                progressDialog.setOnDismissListener(new
DialogInterface.OnDismissListener() {
                    @Override
                    public void onDismiss(DialogInterface dialog) {
                        loadMainActivity(); //对话框消失后载入登录界面
                    }
                });
                getDataFromServer();
            } else {
                loadMainActivity();
            }
        }

    }, 2500);
}

/*通过 Volley 发送 HTTP 请求从服务器端获取数据*/
private void getDataFromServer() {
    String url = "http://www.zhouzezhou.site/WirelessOrder/servlet/DBSyncServlet";
    RequestQueue requestQueue = Volley.newRequestQueue(mContext);
    StringRequest stringRequest = new StringRequest(Request.Method.POST, url,
        new Response.Listener<String>() {
            @Override
```

```
        public void onResponse(String s) {
            Utils.syncData(mContext, s);
            pDialog.dismiss();
            mSp.edit().putBoolean("firstStartup", false).commit();
        }
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError volleyError) {
            Toast.makeText(mContext, "    同    步    失    败    ",
Toast.LENGTH_SHORT).show();
            finish();
        }
    });
    requestQueue.add(stringRequest);
}
```

7. 本地 SQLite 数据库初始化

//在应用第一次使用数据库时调用 onCreate() 方法，执行相关表的建立的 SQL 语句

```
public static final String DB_NAME = "mydb.db";
public static final int DB_VERSION = 1;

public static final String CREATE_TABLE_ADMIN = "CREATE TABLE t_admin ("
+
    " admin_id integer primary key autoincrement," +
    " admin_name varchar(255)," +
    " admin_password varchar(255)" +
    ");";

public static final String CREATE_TABLE_MEAL = "CREATE TABLE t_meal (" +
    " meal_id integer primary key autoincrement," +
    " meal_type_id integer ," +
    " meal_name varchar(255) ," +
    " meal_price double ," +
    " meal_image varchar(255) ," +
    " meal_info varchar(255)" +
    ");";

public static final String CREATE_TABLE_MEAL_TYPE = "CREATE TABLE
t_meal_type (" +
    " meal_type_id integer primary key autoincrement," +
    " meal_type_name varchar(255) " +
    ");";

public static final String CREATE_TABLE_ORDER = "CREATE TABLE t_order ("
+
    " order_id integer primary key autoincrement," +
    " user_id integer ," +
    " order_user_amount integer ," +
    " order_sum double ," +
    " order_date varchar(255) " +
    ");";

public static final String CREATE_TABLE_ORDER_DETAIL = "CREATE TABLE
t_order_detail (" +
    " order_id integer ," +
    " meal_id integer ," +
    " meal_amount integer" +
    ");";

public static final String CREATE_TABLE_USER = "CREATE TABLE t_user (" +
    " user_id integer primary key autoincrement," +
```

```
        " user_password varchar(255) ," +
        " user_name varchar(255) ," +
        " user_phone varchar(255)" +
        ");";

public DataBaseHelper(Context context) {
    super(context, DB_NAME, null, DB_VERSION);
}

@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL(CREATE_TABLE_ADMIN);
    db.execSQL(CREATE_TABLE_MEAL);
    db.execSQL(CREATE_TABLE_MEAL_TYPE);
    db.execSQL(CREATE_TABLE_ORDER);
    db.execSQL(CREATE_TABLE_ORDER_DETAIL);
    db.execSQL(CREATE_TABLE_USER);
}
```


8. 服务器端数据同步接口

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8;");
    /*从数据库获取所有数据*/
    List<User> userList = userService.getAllUsers();
    List<Admin> adminList = adminService.getAllAdmins();
    List<Meal> mealList = mealService.getAllMeal();
    List<Order> orderList = orderService.getAllOrders();
    List<OrderDetail> orderDetailList = orderService.getAllOrderDetails();
    JSONArray userArray = new JSONArray();
    JSONArray adminArray = new JSONArray();
    JSONArray mealArray = new JSONArray();
    JSONArray orderArray = new JSONArray();
    JSONArray orderDetailArray = new JSONArray();
    PrintWriter out = response.getWriter();
    /*将数据都以 json 的形式返回给客户端*/
    try {
        for(User user : userList) {
            JSONObject userJson = new JSONObject();
            userJson.put("user_id", user.getUserId());
            userJson.put("user_password", user.getUserPassword());
            userJson.put("user_name", user.getUserName());
            userJson.put("user_phone", user.getUserPhone());
            userArray.put(userJson);
        }
        for(Admin admin : adminList) {
            JSONObject adminJson = new JSONObject();
            adminJson.put("admin_id", admin.getAdminId());
            adminJson.put("admin_password", admin.getAdminPassword());
            adminJson.put("admin_name", admin.getAdminName());
            adminArray.put(adminJson);
        }
        for(Meal meal : mealList) {
            JSONObject mealJson = new JSONObject();
            mealJson.put("meal_id", meal.getMealId());
            mealJson.put("meal_type_id", meal.getMealType());
            mealJson.put("meal_name", meal.getMealName());
            mealJson.put("meal_price", meal.getMealPrice());
            mealJson.put("meal_image", meal.getMealImage());
            mealJson.put("meal_info", meal.getMealInfo());
            mealArray.put(mealJson);
        }
        for(Order order : orderList) {
            JSONObject orderJson = new JSONObject();
```

```
        orderJson.put("order_id", order.getOrderid());
        orderJson.put("user_id", order.getUserid());
        orderJson.put("order_user_amount", order.getUserAmount());
        orderJson.put("order_sum", order.getOrderSum());
        orderJson.put("order_date", order.getOrderDate());
        orderArray.put(orderJson);
    }
    for(OrderDetail orderDetail : orderDetailList) {
        JSONObject orderDetailJson = new JSONObject();
        orderDetailJson.put("order_id", orderDetail.getOrderid());
        orderDetailJson.put("meal_id", orderDetail.getMealid());
        orderDetailJson.put("meal_amount", orderDetail.getMealAmount());
        orderDetailArray.put(orderDetailJson);
    }
    JSONObject dbJsonObject = new JSONObject();
    dbJsonObject.put("t_user", userArray);
    dbJsonObject.put("t_admin", adminArray);
    dbJsonObject.put("t_meal", mealArray);
    dbJsonObject.put("t_order", orderArray);
    dbJsonObject.put("t_order_detail", orderDetailArray);
    out.write(dbJsonObject.toString());
    out.flush();
    out.close();
} catch (JSONException e) {
    // TODO: handle exception
}
}
```

9. 后台管理系统管理员登录界面

@Override

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_login);
    getSupportActionBar().setDisplayHomeAsUpEnabled(false);
    setActionBarTitle(R.string.log_in);
    initView();
}

private void initView() {
    final EditText editUser = (EditText) findViewById(R.id.userName);
    final EditText editPassword = (EditText) findViewById(R.id.password);
    Button loginBtn = (Button) findViewById(R.id.login);
    loginBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if (!Utils.hasNetwork(LoginActivity.this)) {
                return;
            }
            String userName = editUser.getText().toString(); //获取用户名
            String password = editPassword.getText().toString(); //获取密码
            if(userName.isEmpty() || password.isEmpty()) {
                Toast.makeText(LoginActivity.this,
getString(R.string.login_mention),
                Toast.LENGTH_SHORT).show();
            } else {
                AdminService adminService = new
AdminService(LoginActivity.this);
                if (adminService.isAdminMatch(userName, password)) { //判断
用户名及密码是否匹配
                    Intent intent = new Intent(LoginActivity.this,
MainActivity.class);
                    startActivity(intent);
                    LoginActivity.this.finish();
                } else {
                    Toast.makeText(LoginActivity.this,
getString(R.string.login_fail),
                    Toast.LENGTH_SHORT).show();
                }
            }
        }
    });
}
```

10. 后台管理系统餐厅统计界面

```
private void initInfoView() {
    LinearLayout infoView = (LinearLayout) this.findViewById(R.id.infoPage);
    TextView userAmount = (TextView) infoView.findViewById(R.id.user_amount);
    TextView mealAmount = (TextView) infoView.findViewById(R.id.meal_amount);
    TextView orderAmount = (TextView) infoView.findViewById(R.id.order_amount);
    TextView orderSum = (TextView) infoView.findViewById(R.id.order_sum);
    /*通过各个 Service 获取有用数据并显示在页面中*/
    UserService userService = new UserService(mContext);
    MealService mealService = new MealService(mContext);
    OrderService orderService = new OrderService(mContext);
    userAmount.setText(String.valueOf(userService.getUserAmount()));
    mealAmount.setText(String.valueOf(mealService.getMealAmount()));
    orderAmount.setText(String.valueOf(orderService.getOrderAmount()));
    orderSum.setText(String.valueOf(orderService.getAllOrderSum()));
    userService.closeDB();
    mealService.closeDB();
    orderService.closeDB();
}
```

10. 数据同步模块

//通过 Volley 请求服务器端获取所有数据

```
private void syncData() {
    final ProgressDialog pDialog = ProgressDialog.show(mContext, null, "同步数据库", true, true);
    pDialog.setCancelable(false);
    String url = "http://www.zhouzezhou.site/WirelessOrder/servlet/DBSyncServlet";
    RequestQueue requestQueue = Volley.newRequestQueue(mContext);
    StringRequest stringRequest = new StringRequest(Request.Method.POST, url,
        new Response.Listener<String>() {
            @Override
            public void onResponse(String s) {
                Utils.syncData(mContext, s); //将数据插入 SQLite 中
                pDialog.dismiss();
                Toast.makeText(mContext, "同步成功",
                    Toast.LENGTH_SHORT).show();
            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError volleyError) {
                Toast.makeText(mContext, "同步失败",
                    Toast.LENGTH_SHORT).show();
                finish();
            }
        });
    requestQueue.add(stringRequest);
}
```

/*将从数据库中获取的数据分别插入数据库对应的表中*/

```
public static void syncData(Context mContext, String result) {
    try {
        JSONObject resultJson = new JSONObject(result);

        JSONArray userArray = resultJson.getJSONArray("t_user");
        JSONArray adminArray = resultJson.getJSONArray("t_admin");
        JSONArray mealArray = resultJson.getJSONArray("t_meal");
        JSONArray orderArray = resultJson.getJSONArray("t_order");
        JSONArray orderDetailArray = resultJson.getJSONArray("t_order_detail");

        UserService userService = new UserService(mContext);
        AdminService adminService = new AdminService(mContext);
        MealService mealService = new MealService(mContext);
        OrderService orderService = new OrderService(mContext);

        userService.insertUsers(userArray);
```

```
        adminService.insertAdmins(adminArray);
        mealService.insertMeals(mealArray);
        orderService.insertOrders(orderArray);
        orderService.insertOrderDetails(orderDetailArray);

        userService.closeDB();
        adminService.closeDB();
        mealService.closeDB();
        orderService.closeDB();
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
```

11. 加载菜品列表

```

private void initMenu() {
    initMealList(); //从数据库中获取菜品数据并存入 List 中
    mRecyclerView = (RecyclerView) findViewById(R.id.list);
    mRecyclerView.setLayoutManager(new LinearLayoutManager(mContext));
    mRecyclerView.setItemAnimator(new DefaultItemAnimator());
    mRecyclerView.setHasFixedSize(true);
    mealAdapter = new MealAdapter(mContext, mealList);
    mRecyclerView.setAdapter(mealAdapter);
}
//通过 RecyclerView 设置自定义的 Adapter 生成菜品列表
public class MealAdapter
    extends RecyclerView.Adapter<MealAdapter.ViewHolder>
    {

        private ArrayList<Meal> mealList;

        private Context mContext;

        public MealAdapter( Context context , ArrayList<Meal> mealList)
        {
            this.mContext = context;
            this.mealList = mealList;
        }

        @Override
        public ViewHolder onCreateViewHolder( ViewGroup viewGroup, int i )
        {
            // 给 ViewHolder 设置布局文件
            View v =
LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.card_meal, viewGroup,
false);
            return new ViewHolder(v);
        }

        @Override
        public void onBindViewHolder( ViewHolder viewHolder, int i )
        {
            // 给 ViewHolder 设置元素
            final Meal p = mealList.get(i);
            RequestQueue requestQueue = Volley.newRequestQueue(mContext);
            ImageLoader imageLoader = new ImageLoader(requestQueue, new
BitmapCache());
            viewHolder.mealName.setText(p.getMealName());
            viewHolder.mealPrice.setText("¥ " + String.valueOf(p.getMealPrice()));
        }
    }

```

```
viewHolder.mealImage.setDefaultImageResId(R.mipmap.image_loading);
viewHolder.mealImage.setErrorImageResId(R.mipmap.error);
viewHolder.mealImage.setImageUrl(p.getMealImage(), imageLoader);
viewHolder.rootView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        showMealEditDialog(p);
    }
});
viewHolder.deleteBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        showDeleteDialog(p);
    }
});
}

@Override
public int getItemCount()
{
    // 返回数据总数
    return mealList == null ? 0 : mealList.size();
}

// 重写的自定义 ViewHolder
public class ViewHolder
    extends RecyclerView.ViewHolder
{
    public TextView mealName, mealPrice;
    public NetworkImageView mealImage;
    public ImageView deleteBtn;
    public View rootView;

    public ViewHolder( View v )
    {
        super(v);
        rootView = v;
        mealImage = (NetworkImageView) v.findViewById(R.id.mealImage);
        mealName = (TextView) v.findViewById(R.id.mealName);
        mealPrice = (TextView) v.findViewById(R.id.mealPrice);
        deleteBtn = (ImageView) v.findViewById(R.id.delete);
    }
}
}
```


12. 菜品删除模块

//点击删除按钮时弹出对话框提示是否要删除

```
private void showDeleteDialog(final Meal meal) {
    AlertDialog deleteDialog = new AlertDialog.Builder(mContext)
        .setTitle(getString(R.string.meal_manage))
        .setMessage("确定删除" + meal.getMealName() + "?")
        .setPositiveButton(R.string.ok, new DialogInterface.OnClickListener()
        {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                deleteMeal(meal); //删除菜品
            }
        })
        .setNegativeButton(R.string.cancel, null)
        .create();
    deleteDialog.show();
}
```

//通过 Volley 向服务器端发送菜品删除请求

```
private void deleteMeal(final Meal meal) {
    final ProgressDialog pDialog = ProgressDialog.show(mContext, null, "删除菜品
中...", true, true);
    pDialog.setCancelable(false);
    String url =
"http://www.zhouzezhou.site/WirelessOrder/servlet/MealDeleteServlet";
    RequestQueue requestQueue = Volley.newRequestQueue(mContext);
    StringRequest stringRequest = new StringRequest(Request.Method.POST, url,
        new Response.Listener<String>() {
            @Override
            public void onResponse(String s) {
                pDialog.dismiss();
                Toast.makeText(mContext, " 删 除 成 功 ",
Toast.LENGTH_SHORT).show();
                //菜品删除成功后，本地数据库也删除对应菜品
                mealList.remove(meal);
                MealService mealService = new MealService(mContext);
                mealService.deleteMeal(meal.getMealId());
                mealService.closeDB();
                mealAdapter.notifyDataSetChanged();
            }
        }, new Response.ErrorListener() {
```

13. 菜品编辑模块

//点击菜品条目会弹出菜品编辑对话框，修改对应菜品的信息

```
private void showMealEditDialog(final Meal meal) {
    LayoutInflater inflater = LayoutInflater.from(mContext);
    View markView = inflater.inflate(R.layout.meal_edit, null);
    final EditText editMealName = (EditText)
markView.findViewById(R.id.m_meal_name);
    final EditText editMealPrice = (EditText)
markView.findViewById(R.id.m_meal_price);
    final EditText editMealInfo = (EditText)
markView.findViewById(R.id.m_meal_info);
    editMealName.setText(meal.getMealName());
    editMealPrice.setText(String.valueOf(meal.getMealPrice()));
    editMealInfo.setText(meal.getMealInfo());

    AlertDialog editDialog = new AlertDialog.Builder(mContext)
        .setTitle(getString(R.string.title_meal_edit))
        .setPositiveButton(getString(R.string.modify), new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                meal.setMealName(editMealName.getText().toString());

                meal.setMealPrice(Double.parseDouble(editMealPrice.getText().toString()));
                meal.setMealInfo(editMealInfo.getText().toString());
                updateMeal(meal); //修改菜品信息
            }
        })
        .setNegativeButton(getString(R.string.cancel), null)
        .setView(markView)
        .create();
    editDialog.show();
}
```

//通过 Volley 向服务器端发送菜品修改请求

```
private void updateMeal(final Meal meal) {
    final ProgressDialog pDialog = ProgressDialog.show(mContext, null, "修改菜品", true, true);
    pDialog.setCancelable(false);
    String url =
"http://www.zhouzezhou.site/WirelessOrder/servlet/MealModifyServlet";
    RequestQueue requestQueue = Volley.newRequestQueue(mContext);
    StringRequest stringRequest = new StringRequest(Request.Method.POST, url,
        new Response.Listener<String>() {
            @Override
```

```
        public void onResponse(String s) {
            pDialog.dismiss();
            Toast.makeText(mContext, "修改成功",
Toast.LENGTH_SHORT).show();
            //本地数据库中也更新对应菜品信息
            mealAdapter.notifyDataSetChanged();
            MealService mealService = new MealService(mContext);
            mealService.updateMeal(meal);
            mealService.closeDB();
        }
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError volleyError) {
            pDialog.dismiss();
            Toast.makeText(mContext, "修改失败",
Toast.LENGTH_SHORT).show();
        }
    }) {
        @Override
        protected Map<String, String> getParams() throws AuthFailureError {
            Map<String, String> map = new HashMap<String, String>();
            //将菜品更新参数发送给服务器端
            map.put("meal_id", String.valueOf(meal.getMealId()));
            map.put("meal_name", meal.getMealName());
            map.put("meal_price", String.valueOf(meal.getMealPrice()));
            map.put("meal_info", meal.getMealInfo());
            return map;
        }
    };
    requestQueue.add(stringRequest);
}
```

14. 菜品添加模块

//核心代码将菜品图片，菜品名等数据上传至服务器，因涉及到图片文本上传，使用 MultipartEntity 传送数据

```
private void doPost(Meal meal, String imagePath) {
    final ProgressDialog pDialog = ProgressDialog.show(mContext, null, "添加上传
菜品中...", true, true);
    pDialog.setCancelable(false);
    String url =
"http://www.zhouzezhou.site/WirelessOrder/servlet/MealAddServlet";
    HttpContext localContext = new BasicHttpContext();
    HttpClient httpclient = new DefaultHttpClient();
    HttpPost httpPost = new HttpPost(url);
    MultipartEntity entity = new MultipartEntity();
    try {
        //将图片，菜品名等数据添加到实体当中
        entity.addPart("mealType", new
StringBody(String.valueOf(meal.getMealType())));
        entity.addPart("mealName", new StringBody(meal.getMealName(),
Charset.forName("UTF-8")));
        entity.addPart("mealPrice", new
StringBody(String.valueOf(meal.getMealPrice())));
        entity.addPart("mealInfo", new StringBody(meal.getMealInfo(),
Charset.forName("UTF-8")));
        entity.addPart("mealImage", new FileBody(new File(imagePath)));
        httpPost.setEntity(entity);

        HttpResponse response = httpclient.execute(httpPost, localContext);
        HttpEntity resEntity = response.getEntity();
        if (resEntity != null) {
            String result = EntityUtils.toString(resEntity, HTTP.UTF_8);
            //上传成功得到数据插入到本地数据库中
            JSONObject mealJson = new JSONObject(result);
            Meal newMeal = new Meal(mealJson.getInt("mealType"),
mealJson.getString("mealName"),
mealJson.getDouble("mealPrice"),
mealJson.getString("mealImage"),
mealJson.getString("mealInfo"));
            newMeal.setMealId(mealJson.getInt("mealId"));
            MealService mealService = new MealService(mContext);
            mealService.insertMeal(newMeal);
            pDialog.dismiss();
            Toast.makeText(mContext, " 上 传 成 功 ",
Toast.LENGTH_SHORT).show();
            Intent intent = new Intent();
            intent.putExtra("newMeal", newMeal);
```

```

        MealAddActivity.this.setResult(RESULT_OK, intent);
        finish();
    }
} catch (Exception e) {
    Toast.makeText(mContext, "上传失败", Toast.LENGTH_SHORT).show();
    e.printStackTrace();
}
}

//服务器端菜品添加接口
public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    Meal meal = new Meal();
    DiskFileItemFactory dfif = new DiskFileItemFactory();
    ServletFileUpload sfu = new ServletFileUpload(dfif);
    sfu.setHeaderEncoding("utf-8");

    try {
        List<FileItem> items = sfu.parseRequest(request);
        for (int i = 0; i < items.size(); i++) {
            FileItem item = items.get(i);
            //根据数据类型不同，图片数据，文本数据分类处理
            if (item.isFormField()) {
                String name = item.getFieldName();
                String value = new String(item.getString("utf-8"));
                if (name.equalsIgnoreCase("mealType")) {
                    meal.setMealType(Integer.parseInt(value));
                } else if (name.equalsIgnoreCase("mealName")) {
                    meal.setMealName(value);
                } else if (name.equalsIgnoreCase("mealPrice")) {
                    meal.setMealPrice(Double.parseDouble(value));
                } else if (name.equalsIgnoreCase("mealInfo")) {
                    meal.setMealInfo(value);
                }
            } else {
                ServletContext sc = this.getServletContext();
                String path = sc.getRealPath("menu");
                String fileName = item.getName();
                String imagePath =
                    "http://www.zhouzezhou.site/WirelessOrder/menu/" + fileName;
                meal.setMealImage(imagePath);
                File file = new File(path + "/" + fileName);
                //将图片存到服务器相关目录中
                item.write(file);
            }
        }
    }
}

```

```
    }  
    } catch (Exception e) {  
        // TODO: handle exception  
    }  
  
    response.setContentType("text/html;charset=UTF-8");  
    PrintWriter out = response.getWriter();  
  
    int mealId = mealService.addMeal(meal); //数据库中插入相关菜品  
    JSONObject mealJson = new JSONObject();  
    try {  
        mealJson.put("mealId", mealId);  
        mealJson.put("mealType", meal.getMealType());  
        mealJson.put("mealName", meal.getMealName());  
        mealJson.put("mealPrice", meal.getMealPrice());  
        mealJson.put("mealImage", meal.getMealImage());  
        mealJson.put("mealInfo", meal.getMealInfo());  
    } catch (JSONException e) {  
        // TODO Auto-generated catch block  
        e.printStackTrace();  
    }  
    out.write(mealJson.toString());  
    out.flush();  
    out.close();  
}
```

15. ContentProvider 数据共享模块

为了向外界应用提供数据，通过继承 ContentProvider 实现

```
public static final Uri CONTENT_URI =
```

```
Uri.parse("content://com.wirelessorder.adminsystem.wirelessprovider/meals");
```

//匹配对应 Uri,返回匹配标识，让代码知道是操作用户还是菜品

```
static {
```

```
    uriMatcher.addURI(AUTHORITY, "users", USER_HANDLE);
```

```
    uriMatcher.addURI(AUTHORITY, "meals", MEAL_HANDLE);
```

```
}
```

```
@Override
```

```
public boolean onCreate() {
```

```
    return false;
```

```
}
```

```
@Override
```

```
public Cursor query(Uri uri, String[] projection, String selection, String[]
```

```
selectionArgs, String sortOrder) {
```

```
    Cursor cursor = null;
```

```
    switch (uriMatcher.match(uri)) {
```

```
        //根据标识进行对应操作，如用户查询，菜品查询
```

```
        case USER_HANDLE:
```

```
            userService = new UserService(getContext());
```

```
            String userName = selectionArgs[0];
```

```
            String password = selectionArgs[1];
```

```
            cursor = userService.getMatchUser(userName, password);
```

```
            break;
```

```
        case MEAL_HANDLE:
```

```
            mealService = new MealService(getContext());
```

```
            if (selectionArgs == null) {
```

```
        cursor = mealService.getAllMealsCursor();
    } else {
        String type = selectionArgs[0];
        cursor = mealService.getMealsByType(type);
    }
    break;
default:
    break;
}
return cursor;
}
```


致谢

植此四年，今昔一念，在华东师范大学的生活学习让我收获良多。借此毕业论文完成之际，特向四年来以及毕业设计完成过程中给予我巨大帮助的人们说一声谢谢。

首先我要感谢我的毕业论文指导老师全红艳老师对我的毕业设计的引导以及在毕业设计开发过程中，一些开发方向以及方法的指导，让我能够顺利完成毕业设计并且能够从中学习到相关技术知识。另外，在毕业论文的撰写方面，也非常感谢全红艳老师的悉心指导，对论文的整体结构以及写作技巧上面的耐心提示让我能够顺利写完论文。

其次我要非常感谢我的同学陈羽琪，在毕业设计的开发过程中，碰到了诸多问题并且影响到了自己的心情，她都能对我进行安慰引导，并督促我及时开发，开发过程中也给予我鼓励让我能够顺利开发出本系统。在毕业论文撰写过程中，陈羽琪同学也授予我大量参考资料以及写作心得，感谢她的陪伴她的帮助。

另外我还要感谢我的室友，大学四年来同住屋檐下，浓浓室友情不必多言，在学习生活中室友给我提供了大量的帮助让我能够顺利完成学业并让我的大学更加丰富多彩。

最后我要感谢我的父母，父母一直在我的背后给我提供最有力的支持，一个幸福的家庭使我能够健康积极地成长，使我的大学我的一生都阳光向上，谢谢父母为我做的一切，感谢他们的栽培。

毕业论文就此完结，感谢大学一路走来的帮助过我的老师，同学，父母，朋友，定不负你们的期望，走向更美好的未来。