**目录**

[**1 绪论** 1](#_Toc21407)

[1.1 研究的背景以及意义 1](#_Toc27939)

[1.1.1 选题的背景 1](#_Toc18702)

[1.1.2 国内外研究现状 1](#_Toc16472)

[1.1.3研究的意义 1](#_Toc27652)

[1.2 系统目标 2](#_Toc23638)

[**2 需求分析** 4](#_Toc31122)

[2.1 业务描述 4](#_Toc16053)

[2.1.1项目描述 4](#_Toc8825)

[2.1.2系统需求优先级 5](#_Toc152)

[2.1.3业务需求 6](#_Toc8488)

[2.2功能需求 18](#_Toc18725)

[2.2.1 系统总用例图 18](#_Toc17624)

[2.2.2 系统中角色分析 19](#_Toc1197)

[2.2.3 功能描述 19](#_Toc17038)

[2.2.4详细功能分析 20](#_Toc3642)

[2.3非功能需求 46](#_Toc25340)

[2.3.1 外部接口需求 46](#_Toc9814)

[2.3.2 性能需求 46](#_Toc7322)

[2.3.3 安全需求 47](#_Toc10570)

[2.3.4 系统运行需求 47](#_Toc25462)

[2.4.5设计与实现约束 48](#_Toc18477)

[**3 总体设计** 50](#_Toc1158)

[3.1体系结构设计 50](#_Toc25456)

[3.2 系统功能架构设计 51](#_Toc20609)

[3.2.1 用户管理子系统 52](#_Toc23248)

[3.2.2 商品管理子系统 52](#_Toc14988)

[3.2.3 咖啡厅前台子系统 53](#_Toc1492)

[3.2.4后台管理独立模块 53](#_Toc28985)

[**4 数据库设计** 55](#_Toc24897)

[4.1 概念结构设计 55](#_Toc27826)

[4.1.1 设计思路 55](#_Toc8492)

[4.1.2 E-R图 55](#_Toc775)

[4.2 逻辑结构设计 63](#_Toc5261)

[4.2.1 设计思路 63](#_Toc23528)

[4.2.2 逻辑模型 64](#_Toc28291)

[4.3 物理结构设计 68](#_Toc10080)

[4.3.1 存取方式 68](#_Toc27308)

[4.3.2 存储结构 69](#_Toc22352)

[**5 界面设计** 74](#_Toc11577)

[5.1 界面关系图或工作流图 74](#_Toc22959)

[5.2 界面设计成果 75](#_Toc1193)

[5.2.1 主界面 75](#_Toc477)

[5.2.2 子界面 80](#_Toc8780)

[**6 详细设计** 85](#_Toc399)

[6.1 系统主要功能模块介绍 85](#_Toc18435)

[6.2用户管理模块 85](#_Toc5663)

[6.2.1功能结构设计 85](#_Toc7381)

[6.2.2功能类设计 86](#_Toc30014)

[6.2.3时序图设计 88](#_Toc10066)

[6.2.4核心处理流程设计 89](#_Toc7261)

[6.3 发票管理 90](#_Toc9174)

[6.3.1功能结构设计 90](#_Toc78)

[6.3.2功能类设计 91](#_Toc24109)

[6.3.3时序图设计 93](#_Toc18149)

[6.3.4核心处理流程设计 93](#_Toc12594)

[6.4 购物车管理模块 94](#_Toc17054)

[6.4.1功能结构设计 94](#_Toc9292)

[6.4.2功能类设计 95](#_Toc26784)

[6.4.3时序图设计 97](#_Toc7677)

[6.4.4核心处理流程设计 97](#_Toc31625)

[6.5商品管理模块 98](#_Toc7146)

[6.5.1功能结构设计 98](#_Toc14825)

[6.5.2功能类设计 99](#_Toc9206)

[6.5.3时序图设计 100](#_Toc5469)

[6.5.4核心处理流程设计 100](#_Toc32621)

[6.6博客管理模块 101](#_Toc24479)

[6.6.1功能结构设计 101](#_Toc6226)

[6.6.2功能类设计 102](#_Toc1295)

[6.6.3时序图设计 103](#_Toc21065)

[6.6.4核心处理流程设计 104](#_Toc26745)

[6.7订单管理模块 105](#_Toc17340)

[6.7.1功能结构设计 105](#_Toc21037)

[6.7.2功能类设计 106](#_Toc6434)

[6.7.3时序图设计 107](#_Toc13466)

[6.7.4核心处理流程设计 108](#_Toc31641)

[6.8公告板管理模块 109](#_Toc23315)

[6.8.1功能结构设计 109](#_Toc23857)

[6.8.2功能类设计 110](#_Toc31590)

[6.8.3时序图设计 111](#_Toc24720)

[6.8.4核心处理流程设计 112](#_Toc10594)

[6.9评论管理模块 113](#_Toc32488)

[6.9.1功能结构设计 113](#_Toc25491)

[6.9.2功能类设计 114](#_Toc26186)

[6.9.3时序图设计 116](#_Toc27310)

[6.9.4核心处理流程设计 116](#_Toc7129)

[6.10 门店管理模块 117](#_Toc24963)

[6.10.1功能结构设计 117](#_Toc2166)

[6.10.2功能类设计 118](#_Toc24951)

[6.10.3时序图设计 119](#_Toc3028)

[6.10.4核心处理流程设计 120](#_Toc20433)

[**7 编码** 122](#_Toc20753)

[7.1 代码实现与核心算法 122](#_Toc25179)

[7.1.1图片上传 122](#_Toc14671)

[7.1.2购物车实现 124](#_Toc30250)

[7.2 代码优化分析 125](#_Toc12681)

[**8 测试** 127](#_Toc22795)

[8.1 测试方案设计 127](#_Toc29278)

[8.1.1 测试策略 127](#_Toc846)

[8.1.2 测试进度安排 127](#_Toc21625)

[8.1.3 测试资源 127](#_Toc17653)

[8.1.4 关键测试点 128](#_Toc14262)

[8.2 测试用例构建 128](#_Toc579)

[8.2.1 测试用例编写约定 128](#_Toc11227)

[8.2.2 测试用例设计 128](#_Toc32331)

[8.2.3 关键测试用例 129](#_Toc6281)

[8.2.4 测试用例维护 129](#_Toc22421)

[**9 总结与展望** 130](#_Toc27085)

[9.1 设计工作总结 130](#_Toc6911)

[9.2 未来工作展望 130](#_Toc3594)

[**谢辞** 131](#_Toc9842)

[**参考文献** 132](#_Toc14416)

[附录A 外文翻译—原文部分 133](#_Toc8800)

[附录B 外文翻译—译文部分 140](#_Toc12900)

[附录C 软件使用说明书 146](#_Toc20527)

[附录D 主要源代码 147](#_Toc9213)

[附录E 相关图表 150](#_Toc7415)

# 1 绪论

## 1.1研究的背景以及意义

### 1.1.1选题的背景

信息时代迅速发展，互联网在不知不觉中已经进入了我们的生活，在生活的每个环节我们都能找到互联网的身影。因此，为了方便人们的生活、提高人们的生活质量、满足人们的生活需求，一些互联网的新兴技术也跟随着人们的需求而被发明出来。在软件的层面，如何运用一些高效、快捷、易用的编程语言开发出美观、简洁、高效的软件成为时代主流。在今天经济与科技快速发展，人们的生活质量有了显著地提升，“咖啡文化”悄无声息地进入了日常生活，一杯咖啡可以在倍感压力的生活中缓解紧张的神经，可以在闲情逸致的时候增添一份享受，也可以在错综复杂的社交中彰显出独特魅力。咖啡逐渐与时尚、现代生活相互联系，形成一股新的潮流。咖啡屋渐渐成为人们交谈、休闲的地方。随着咖啡文化在中国的快速传播，一套好的咖啡厅订单管理系统也正被需要。为了缓解竞争激烈的公司环境，开发出一款线上咖啡厅应用提供点餐服务势在必行。

### 1.1.2国内外研究现状

在快速发展的中国，现如今的人们生活压力越来越大，在未来的几年内，缓解压力、提供休闲的需求会越来越多。现在市场上大多数咖啡厅、甜品店都是以传统的运营方式为主，主要是在首先在服务前台进行点单、然后顾客等待取餐，这种运营模式并不能满足每个人的点餐需求，加之现如今配送服务较为完善，越来越多人选择在网上点单。随着服务的不断完善，线下+线上运营方式替代传统的线下运营方式将会成为必然。目前市场上仅仅只有少数几家较为出名的咖啡品牌采取线上+线下运营方式，造成该现象的主要原因是传统的软件开发模式耗时耗力，会极大增加中小咖啡厅的运营成本。

### 1.1.3研究的意义

为了缓解中小咖啡品牌的运营成本、增加营业额，采用快速、维护容易的开发模式与技术可以很好地解决这个难题与困扰。开发出线上的应用也不仅仅是为了增加一种点餐方式，更是数据可视化、公开的一种体现，它的优点也不仅仅是减轻了服务员的工作量，也为解决与用户之间的交易争议提供了一种有效的手段，解决了传统的口头叫号取餐方式，服务员可以将取餐号投放在前台的显示屏上并显示状态，方便用户取餐。用户在自身消费权益受到侵犯时，采用此种点餐方式可以为用户提供一份详细的凭证。

## 1.2系统目标

1、设计一套完整且界面优美的咖啡屋订单管理系统，使用前沿技术对该系统进行实现。

2、后台管理设计包括两个子系统与五个独立模块：

（1）用户管理子系统包括普通用户管理、管理员管理、权限分配管理三个模块。权限分配管理模块包括用户登录、注册功能、管理员权限管理功能，管理员可以在该模块进行登录注册也可以对用户权限进行修改，维护用户信息。在普通用户管理模块可以查看用户的订单信息，并可以对用户的信息进行维护。一级管理员可以对管理员管理模块进行操作，可以增加二级管理员、修改管理员权限、删除管理员等操作。

（2）商品管理子系统包括饮品管理、品类管理、周边商品管理三个子模块。管理员可以通过饮品管理对饮品信息进行修改，增加饮品，修改饮品销售状态，删除已经停产的饮品。通过品类管理模块，管理员可以增加饮料品类、修改饮料品类信息、显示相关饮料品类下的饮品、删除饮料品类。管理员也可以通过周边商品管理模块对周边商品进行增删改查操作。

（3）门店管理模块，管理员在此模块管理门店信息，并对新加入的门店信息进行添加，对已经关闭的门店进行消除信息操作实时更新该品牌下所有门店信息。

（4）订单管理模块，管理员可以查看该模块下的所有订单信息，根据需要对其进行操作，对应客户的需求，可以为客户修改订单信息。并为因系统出现错误未添加的订单进行填补订单操作。管理员可以通过订单编号查找到需要操作的订单。

（5）博客管理模块，管理员在此模块对前台用户提交的博客信息进行审批，并将审批的博客呈现在前台界面上，也可以通过该模块通知博客状态，并提示用户修改敏感字。

（6）公告板管理模块，管理员在此模块通知用户相关通知，管理员可以将通知分为三个等级，分别是系统紧急通知、通知、通告。发出的通知，用户可以通过前台界面访问。

（7）评论管理模块，管理可以在此模块对相应的用户进行选择禁言时间，进行禁言操作。

3、前台界面包括个人中心、购物车管理、订单系统、门店模块、登录注册模块、博客模块，通知栏模块，也包含主页对饮品与周边商品的呈现，为用户购买商品提供推荐。

（1）用户可以通过个人中心查询个人信息、查询订单信息、退出登录。

（2）购物车包括用户添加的饮品与周边商品，并可以在购物车对商品进行增删操作。

（3）订单系统包括两个子模块，一个是饮品订单子模块，另一个是周边商品子模块。饮品订单子模块包括两个功能自提与配送，自提需要选择相关门店自提，配送需要提供配送地址、配送人信息。

（4）用户可以在门店模块，查看所有门店的信息，并可以致电咨询情况。

（5）用户可以根据自己的信息登录前台，如若没有账户，可以通过注册模块进行信息填写完成注册。用户忘记密码可以进入找回密码页面输入相关信息修改密码并找回自己的密码。

（6）用户可以通过该模块查看自己与他人发表的博文，也可以通过该模块发表自己的博文。

（7）用户可以在通知栏模块及时查看管理员发出的通告。

（8）在前台的主页用户可以查看饮品新品发布、限时折扣商品、销售冠军等信息。

（9）用户可以根据菜单来查找自己想要购买的商品，并且选择相应的规格，添加购物车操作。

4、用户前端界面主要采用美观的前端模板进行渲染，力求使用户感觉到咖啡屋的专业与态度，前端基于bootstrap+thymeleaf+jquery+layui实现前端界面的显示与数据交互，力求提升界面完美程度以及数据传输时间。

5、管理员前端界面采用比较直观界面，以对每位顾客的信息以及需求有一个更好的认识，界面要简洁明了，简化顾客操作。

# 2 需求分析

## 2.1 业务描述

### 2.1.1项目描述

系统分为前台系统与后台管理系统。前台有购物车管理、取餐方式管理、发票管理、评论管理、订单管理。后台管理有权限管理模块、用户管理模块、订单管理模块、饮料管理模块、公告板管理模块、品类管理模块、周边商品管理、门店管理、博客管理。

1、权限管理模块，包括登录、注册、权限分配三个子模块，所有用户都可以通过该子系统进行登录、注册操作，管理员可以通过该模块进行权限分配。

2、用户管理，包括用户信息列表，该菜单只有系统管理员可见，系统管理员可以维护用户信息。

3、订单管理，包括用户下单和管理员接单。根据客户需求，顾客选择指定饮料，提交订单，并通知制作员调制指定饮料。订单信息包括饮料名称、金额、桌号（具体桌号或打包）、支付方式（现金、刷卡等）、是否结算，订单序号。制作员接收到订单提醒后，根据订单时间顺序，开始调制饮料，调好后通知服务员，管理员接收通知后，接收饮料并结束订单。

4、饮料管理，包括饮料信息列表，可新增、修改饮料品类，新增后发布到公告板。饮料品类信息包括：饮料名称、图片、主要配料、类别（大小杯）、价格。列表可上架、下架饮料，也需要列出优惠饮料、人气饮料。

5、公告板管理，可展示新增饮料产品、店长强力推荐产品、销售冠军产品。

6、品类管理，包括对各个品类进行管理，有增加品类、删除品类、修改品类、查看品类操作，操作对象为管理员。

7、购物车管理，操作对象是顾客，顾客可以通过该模块，进行把商品加入购物车，删除商品，修改商品数量，查看购物车内容操作。

8、取餐方式管理，操作对象是顾客，顾客可以通过该模块，选择三种取餐方式分别是外送、自提、预约。外送需要顾客提供外送地址，自提需要在30分钟内取餐，预约需要提供预约人的号码、姓名。

9、周边商品管理，操作对象是管理眼，管理员可以通过该模块，对周边商品进行分类，分为食品和物品，并对它们进行增、删、改、查操作。

10、发票管理，操作对象是顾客，顾客可以通过该模块对开发票方式进行选择，但是只能自身开通的支付方式中选择开票方式。

11、评论管理，操作对象是管理员，他们可以通过该模块对评论主题管理与禁言管理进行操作，评论主题管理包括对主题的增删改查操作，禁言管理包括禁言对象选择与禁言时间选择两个子模块。

12、门店管理，操作对象是管理员，管理员可以通过该模块对门店进行增删改查操作，也包括推荐优质门店，人气门店，最佳匹配门店推荐这些模块。

13、博客管理，操作对象是管理员，管理员可以通过该模块，审核用户提交的博客，也可以通过该模块删除用户提交的不合格博客。

项目开发的方式以B/S架构为主，采用MVC三层架构，保证项目架构设计的合理有效，具有较强的可维护性，力求项目投入生产环境后运行高效稳定。

### 2.1.2系统需求优先级

系统需求优先级如表2-1所示：

表2-1系统需求优先级

|  |  |  |  |
| --- | --- | --- | --- |
| **需求编号** | **功能名称** | **优先级** | **说明** |
| 0001 | 用户管理 | 1 | 本系统优先开发模块 |
| 0002 | 饮料管理模块 | 2 | 对饮料进行管理模块 |
| 0003 | 订单管理模块 | 3 | 用户订单的管理 |
| 0004 | 公告板管理模块 | 4 | 相关信息显示 |
| 0005 | 登录注册 | 1 | 用户登录、注册 |
| 0006 | 品类管理 | 2 | 对品类进行管理 |
| 0007 | 购物车管理 | 5 | 对购物车进行操作 |
| 0008 | 取餐方式管理 | 5 | 取餐方式选择 |
| 0009 | 周边商品管理 | 5 | 周边商品的管理 |
| 0010 | 发票管理 | 5 | 对开票方式的选择 |
| 0011 | 评论管理 | 5 | 对商品评论管理 |
| 0012 | 门店管理 | 5 | 对门店信息进行管理 |

### 2.1.3业务需求

（一）订单管理

（1）业务需求描述

顾客根据菜单选择饮品，随后选择是否继续选择，如果继续选择，则将选择好的物品加入购物车，若结束选择，则生成相应订单，提交至服务员，服务员接收订单，根据订单提交给制作员，并制作出相应饮品，制作完成后，制作员修改订单状态将正在制作改为制作完成，随后服务员进行叫号，叫到号的顾客根据自己的订单号拿到商品，随后对该订单做出评价，订单结束。如图2-2所示

（2）业务流程如图2-2所示

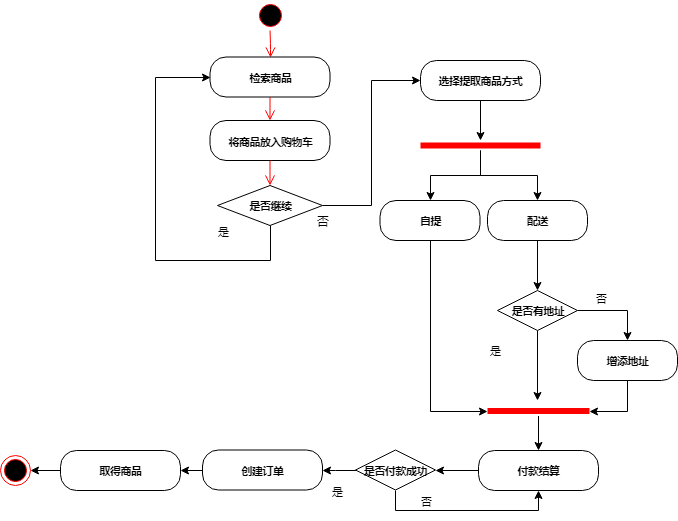


图2-2顾客订单业务流程图

（二）公告板展示

（1）业务需求描述

管理员以提高销售业绩为目的，通过编辑公告板信息展示给顾客近一段时间内畅销的产品并且通过公告板推出最新产品，服务员可以对公告板进行增加、删除、修改操作，对于已经过时的公告板进行删除操作，然后可以对新增加的产品增加至公告板，方便新品的推广与宣传，对店长推荐产品与销售冠军产品进行修改至公告板，提升店内产品的热度，增加销量。

（2）业务流程如图2-3所示

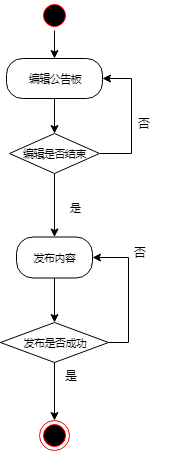


图2-3 公告板展示业务流程图

（三）用户管理

（1） 业务需求概述

管理员对用户进行分类，将用户分成几种不同的角色，分别是服务员、制作员、顾客、VIP顾客四个角色，每个角色都有不同的权限，对应不同的功能，并由此执行不同的功能与任务。管理员还需要对用户进行信息维护，保证用户信息的及时更新，对用户信息具有增加信息条目、修改某些信息、删除某个用户的信息。对有销号意愿的服务员、制作员、顾客进行销号处理，即把信息从数据库中整个删除。

（2）业务流程如图2-4所示

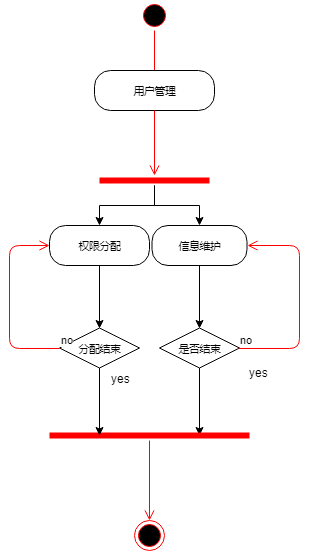


图2-4 用户管理业务流程图

（四）饮料管理

（1）业务需求概述

管理员需要对新产品进行添加，可以把新产品放到公告板进行展示，达到宣传的目的，并考虑调整相关饮料价格、更新饮料图片，修改一些饮料的配料等等操作，或者对某些失败的产品进行删除，下架处理。

（2）业务流程如图2-5所示

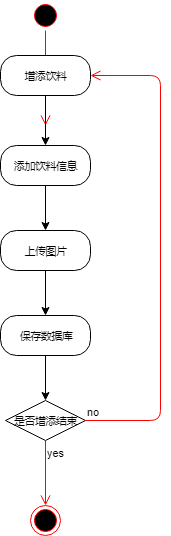


图2-5饮料管理业务流程图

（五）品类管理

（1）业务需求描述

服务员对品类进行管理，选择品类增加、删除、查看操作或者修改品类信息。首先可以选择操作，选择增加品类，如果操作完成则选择是否加入公告板，随后结束操作。也可以选择修改、删除、查看品类操作，操作完成后是否继续操作，如果否则结束流程。

（2）业务流程如图2-6所示

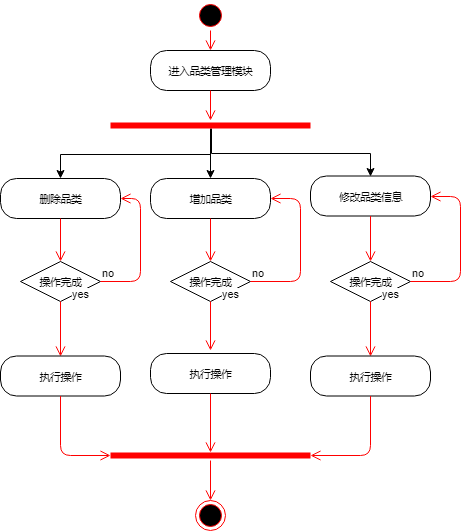


图2-6 品类管理流程图

（六）购物车管理

（1）业务需求描述

顾客对购物车进行操作管理，用户可以选择加入商品、删除商品操作，也可以选择修改商品数量，包括增加数量、减少数量，当数量减少至0时，则系统删除商品。操作完成后，选择是否继续操作，如果否，则结束流程。

（2）业务流程如图2-7所示

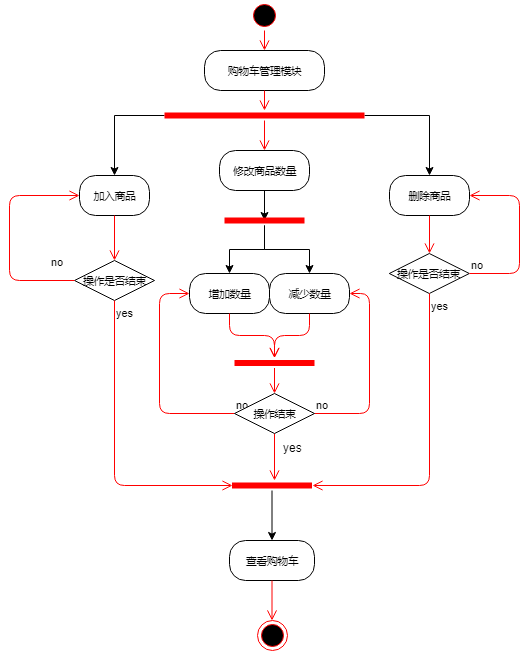


图2-7 购物车管理流程图

（七）取餐方式管理

（1）业务需求描述

顾客可以在此谋爱选择三种配送方式，分别是外送、自提、预约。选择外送时，如果用户没有提供地址，则增加地址或者选择地址，之后再选择并确认配送地址。自提需要用户提供取餐号进行操作，用户也可以预约服务，如果用户没有预留预约信息，系统提示增加预约信息。

（2）业务流程如图2-8所示

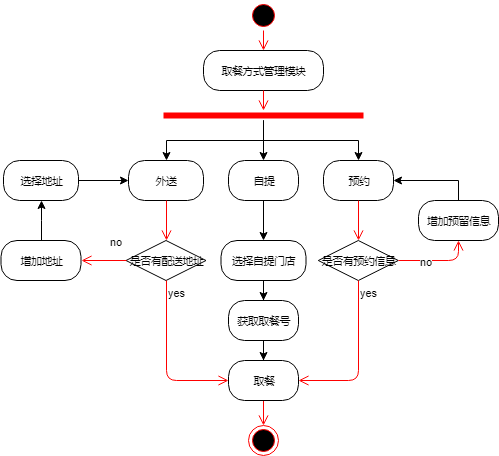


图2-8 取餐方式管理流程图

（八）周边商品管理

（1）业务需求描述

服务员在此模块，分别对物品管理与食品管理两个子模块进行管理。两个字模块都有增删改查操作，服务员选择操作，操作完毕后，假如不继续操作，则结束流程。

（2）业务流程如2-9所示

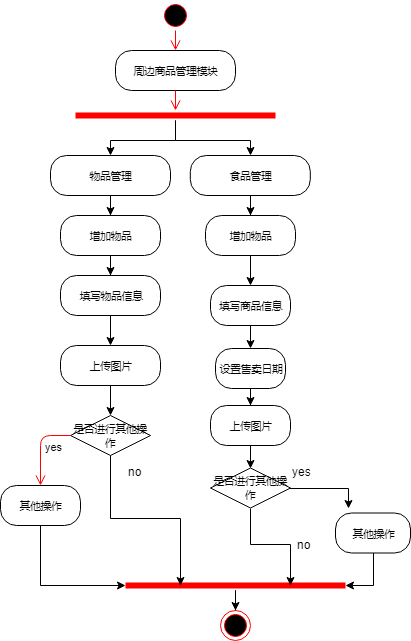


图2-9 周边商品管理流程图

（九）发票管理

（1）业务需求描述

顾客可以在此模块，选择开具发票方式，如果用户没有开通某种支付方式，系统自动提示用户，如果开通选定的开具发票方式，则结束操作和流程。

（2）业务流程如2-10所示

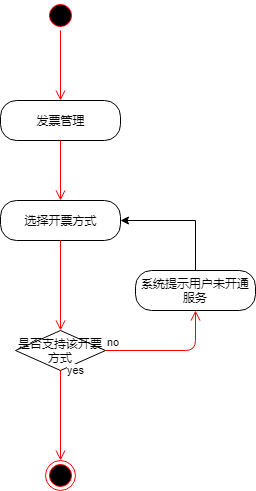


图2-10发票管理流程图

（十）评论管理

（1）业务需求描述

服务员和管理员可以选择禁言管理和主题管理两个模块，进入禁言管理模块，可以选择发言不当的用户进行禁言管理，并选择禁言时间。也可以对主题进行管理，选择修改主题，如果完成操作，则结束流程。

（2）业务流程如2-11所示

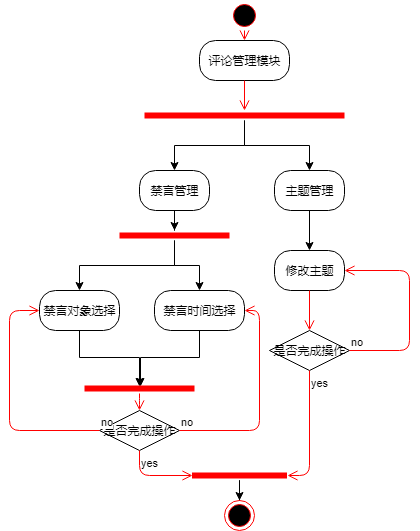


图2-11 评论管理流程图

（十一）门店管理

（1）业务需求描述

服务员在此模块对门店进行管理，如果选择修改门店信息操作，操作完成后选择其他操作或者结束流程，如果选择流出优质门店操作，则将目标门店加入优质门店块并提示是否继续操作，如果否则选择其他操作或者结束流程，也可以选择匹配最佳门店操作，用户也可以根据此模块由系统自动提供离自己最近的门店，匹配成功后，结束流程。

（2）业务流程如图2-12所示：

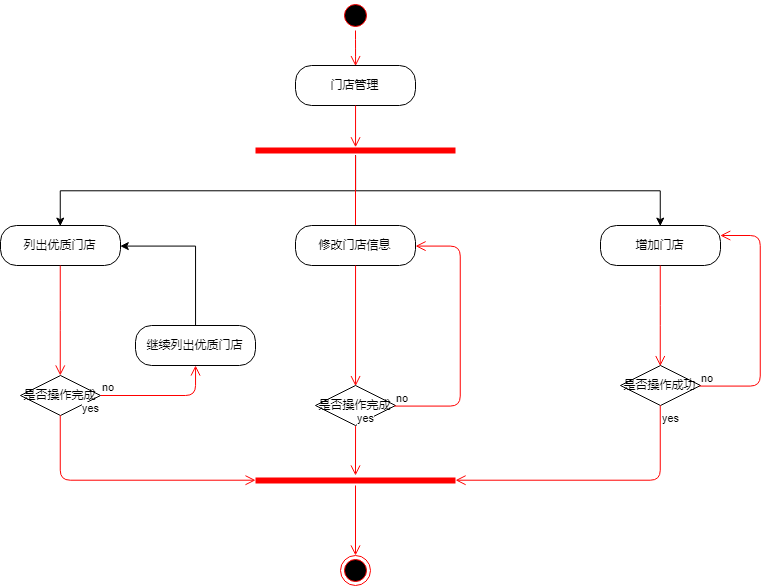


图2-12 门店管理流程图

（十二）登录注册

（1）业务需求描述

如果是曾经注册过账号的用户，则直接登录模块，如果账号、密码、验证码错误，系统提示重新输入或者选择找回密码操作。如果是未注册过的用户，则进入注册页面输入基本信息完成注册，如果填写的信息格式错误或者信息不充分，系统提示相关错误，注册完成后，流程结束。

（2）业务流程如图2-13所示

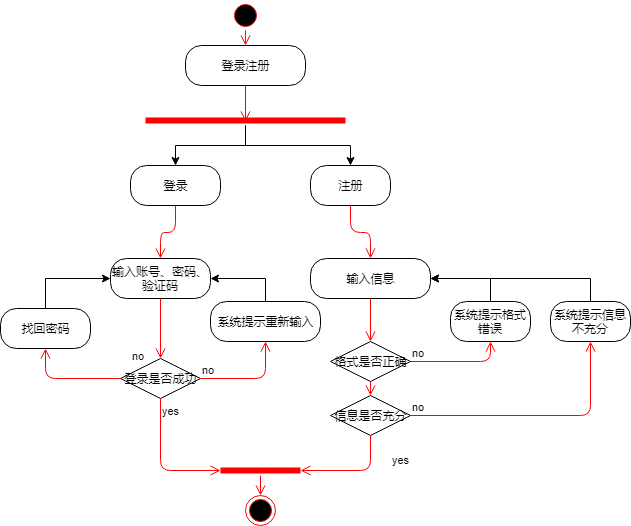


图2-13 登录注册流程图

（十三）博客管理

（1）业务需求描述

用户可以在前台进行编写博文，并上传图片后提交至后台，管理员在后台审核完成后才可以显示在前台界面。

（2）业务流程如图2-14所示

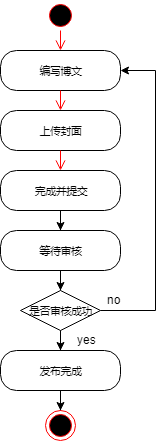


图2-14 博客管理业务流程图

## 2.2功能需求

### 2.2.1 系统总用例图

系统总用例图如2-14所示：

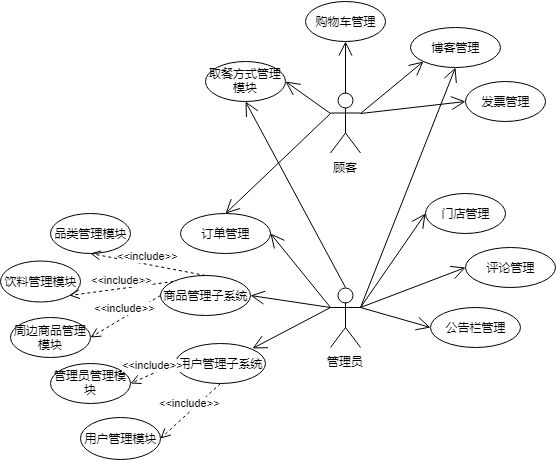


图2-14 系统总用例图

### 2.2.2 系统中角色分析

1、 管理员有评论管理、订单管理、周边商品管理、公告板管理、饮料管理、品类管理、门店管理、用户管理、博客管理九个模块。评论管理是包括评论主题管理与禁言操作两个子模块，用户管理包括对用户信息的增、删、改、查，订单管理包括对订单的增、删、改、查操作，周边商品管理包括食品周边管理、物品周边管理两个模块，公告板管理、饮料管理、品类管理都是由增、删、改、查操作组成，门店管理包括增、删、改、查、优质门店、匹配最佳门店六个模块，博客管理是对博客信息的维护，包括新增博客、修改博客审核状态、删除博客，查看博客列表。

2、 顾客有取餐方式管理、购物车管理、发票管理三个模块。取餐方式包括外送、自提、预约三个管理模块，购物车管理包括对购物车增、删、改、查操作，发票管理由开票方式选择组成。

### 2.2.3 功能描述

管理员把用户分为制作员、服务员、顾客三个不同的角色，并让他们负责执行不同的功能。随着成员信息的变动，管理员需要对用户信息进行增删改查操作。管理员还需要对公告板的内容做出改变，因此公告板管理也需要有公告信息的增删改操作，以便提高公告板信息的及时更新效率。订单管理中需要对用户提供的订单进行归总，根据先来先服务的原则对订单进行排序，并根据此顺序把订单递交给制作员制作出相应的饮品，在这个过程中用户可以在订单提交至制作员之前修改订单信息，随后，完成的订单将由服务员根据订单中的编号叫号，顾客根据自己的订单编号可以拿到饮品。店员们根据市场的需求做出创新，创造出新的产品，并在公告板上显示出新的产品，也可以对当下的销售冠军产品与店长推荐产品进行显示。评论管理是包括评论主题管理与禁言操作两个子模块，积分商城管理包括积分商品管理与积分管理，订单管理包括对订单的增、删、改、查操作，周边商品管理包括食品周边管理、物品周边管理两个模块，公告板管理、饮料管理、品类管理都是由增、删、改、查操作组成，门店管理包括增、删、改、查、优质门店、匹配最佳门店六个模块。取餐方式包括外送、自提、预约三个管理模块，购物车管理包括对购物车增、删、改、查操作，发票管理由开票方式选择组成。信息管理包括对用户信息增、删、改、查操作，权限分配包括对服务员、顾客、VIP顾客的增加，删除操作。

### 2.2.4详细功能分析

（一）登录注册用例图如图2-15所示：

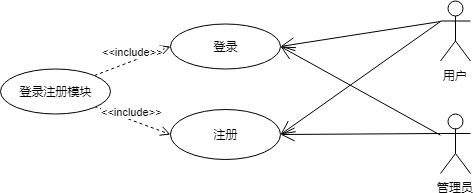


图2-15 登录注册用例图

用户与管理员可以在登录界面点击登录按钮完成登录，登录用例分析表如下表2-2所示

表2-2登录用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 登录 |
| 标识符 | 001 |
| 说明 | 用户需要拥有用户，才能登录 |
| 参与者 | 管理员、用户 |
| 状态 | 通过 |
| 前置条件 | 用户已经拥有账户 |
| 后置条件 | 进入后台首页 |
| 基本操作流程 | 1.用户输入账号、密码  2.点击登录按钮  3.提示登录成功  4.进入后台管理页面 |

用户与管理员可以在注册界面点击注册按钮完成注册，然后跳转至登录界面登录，注册用例分析表如下表2-3所示

表2-3注册用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 注册 |
| 标识符 | 002 |
| 说明 | 如果用户没有账户可以在此用例中创建账户 |
| 参与者 | 管理员、用户 |
| 状态 | 通过 |
| 前置条件 | 用户未有账户 |
| 后置条件 | 跳转至登陆界面 |
| 基本操作流程 | 1.点击登陆界面的注册按钮  2.跳转至注册界面  3.输入相关信息，点击注册按钮  4.跳转至登陆页面 |

（二）用户管理用例图如图2-16所示：

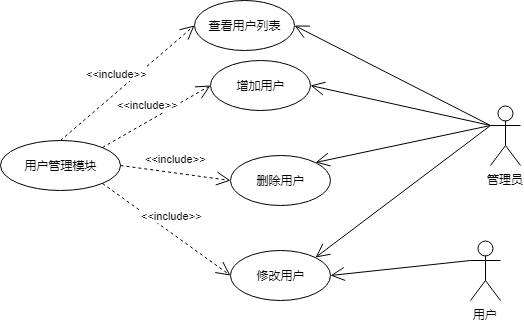


图2-16 用户管理用例图

在后台管理系统的用户管理模块，管理员可以查看用户列表，查看用户列表如表2-4所示

表2-4查看用户列表用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 查看用户列表 |
| 标识符 | 003 |
| 说明 | 只有用户管理员权限才可以查看用户列表 |
| 参与者 | 管理员 |
| 状态 | 通过 |
| 前置条件 | 管理员已经登录成功进入首页 |
| 后置条件 | 选择其他操作 |
| 基本操作流程 | 1.点击用户管理子系统导航栏标签  2.点击点击用户管理标签  3.点击用户列表  4.打开用户列表后，即可查看用户列表信息 |

在后台管理系统的用户管理模块，管理员可以增加用户，增加用户用例如表2-5所示

表2-5增加用户用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |

续表2-5

|  |  |
| --- | --- |
| 描述对象 | 增加用户 |
| 标识符 | 004 |
| 说明 | 拥有管理员权限才可以在后台管理系统增加用户 |
| 参与者 | 管理员 |
| 状态 | 通过 |
| 前置条件 | 管理员已经成功进入用户列表页面 |
| 后置条件 | 添加完成后跳转至用户列表 |
| 基本操作流程 | 1.管理员在用户列表点击增加按钮  2.填写相关用户信息  3.提交后，提示添加成功  4.跳转至用户列表 |

在后台管理系统的用户管理模块，管理员可以删除用户，删除用户用例如表2-6所示

表2-6删除用户用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 删除用户 |
| 标识符 | 005 |
| 说明 | 拥有管理员权限才可以在后台管理系统删除用户 |
| 参与者 | 管理员 |
| 状态 | 通过 |
| 前置条件 | 管理员已经成功进入用户列表页面 |
| 后置条件 | 删除完成后跳转至用户列表 |
| 基本操作流程 | 1.管理员在用户列表选择对象后点击删除按钮  2.显示删除成功  3.跳转至用户列表 |

在后台管理系统的用户管理模块，管理员可以修改用户信息，用户也可以在前台系统的个人中心中修改个人信息，修改用户用例如表2-7所示。

表2-7修改用户用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 修改用户 |

续表2-7

|  |  |
| --- | --- |
| 标识符 | 006 |
| 说明 | 管理员和用户都能修改信息，但是用户只能修改自己的信息 |
| 参与者 | 管理员、用户 |
| 状态 | 通过 |
| 前置条件 | 管理员已经成功进入用户列表页面，用户登录进入个人中心 |
| 后置条件 | 管理员修改完成后跳转至用户列表，用户修改完成后跳转个人中心 |
| 基本操作流程 | 1.管理员在用户列表点击修改按钮  2.在修改页面选择需要修改的信息  3.提交后，提示修改成功  4.跳转至用户列表  5.用户进入个人中心点击修改信息按钮  6.修改完成后提交信息 |

（三）博客管理用例图如图2-17所示：

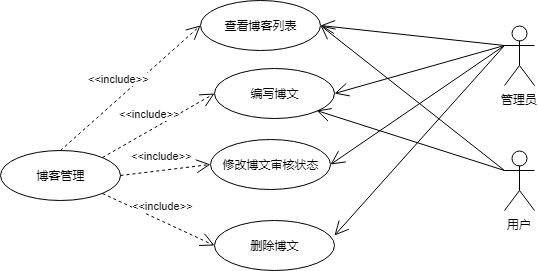


图2-17博客管理用例图

在后台管理系统，管理员可以在博客管理模块查看所有博客列表，而用户也可以在前台系统查看已审核的博客列表，查看博客列表的用例分析表如表2-8所示

表2-8查看博客列表用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |

续表2-8

|  |  |
| --- | --- |
| 描述对象 | 查看博客列表 |
| 标识符 | 007 |
| 说明 | 管理员和用户都能查看博客列表信息 |
| 参与者 | 管理员、用户 |
| 状态 | 通过 |
| 前置条件 | 管理员已经成功进入后台管理系统首页，用户登录进入前台系统首页 |
| 后置条件 | 管理员选择其他操作，用户选择其他操作 |
| 基本操作流程 | 1.管理员点击博客管理模块标签  2.管理员点击博客列表标签，查看博客列表信息  3.用户进入咖啡博客模块  4.用户可以选择进入咖啡博客详情信息 |

在后台管理系统，管理员可以在博客管理模块新增博客，而用户也可以在前台系统编写博客，编写博客的用例分析表如表2-9所示

表2-9编写博客用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 编写博客 |
| 标识符 | 008 |
| 说明 | 管理员和用户都能新增博客 |
| 参与者 | 管理员、用户 |
| 状态 | 通过 |
| 前置条件 | 管理员已经成功进入博客管理模块中的博客列表，用户进入前台系统中博客模块首页 |
| 后置条件 | 跳转至博客列表 |
| 基本操作流程 | 1.管理员进入博客列表  2.管理员点击新增博客按钮  3.填写相关信息后，显示新增成功  4. 跳转至博客列表  5.用户点击编写博客按钮  6.填写相关信息后，显示新增成功  7.跳转至博客列表 |

在后台管理系统，管理员可以在博客管理模块修改博客审核状态，修改博客审核状态的用例分析表如表2-10所示

表2-10修改博客审核状态用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 修改博客审核状态 |
| 标识符 | 009 |
| 说明 | 用户编写后的博客需要管理员审核，审核完成后才能显示 |
| 参与者 | 管理员 |
| 状态 | 通过 |
| 前置条件 | 管理员已经成功进入博客管理模块中的博客列表 |
| 后置条件 | 跳转至博客列表 |
| 基本操作流程 | 1.管理员进入博客列表  2.管理员点击修改按钮  3.修改审核状态后  4. 跳转至博客列表 |

在后台管理系统，对于不合规范的博文，管理员可以在博客管理模块删除博客，删除博客的用例分析表如表2-11所示

表2-11删除博客用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 删除博客 |
| 标识符 | 010 |
| 说明 | 对于不合规范的博文，管理员拥有权限删除 |
| 参与者 | 管理员 |
| 状态 | 通过 |
| 前置条件 | 管理员已经成功进入博客管理模块中的博客列表 |
| 后置条件 | 跳转至博客列表 |
| 基本操作流程 | 1.管理员进入博客列表  2.管理员选择删除对象后，点击删除按钮  3.提示删除成功，跳转至博客列表 |

（四）购物车管理用例图如图2-18所示：

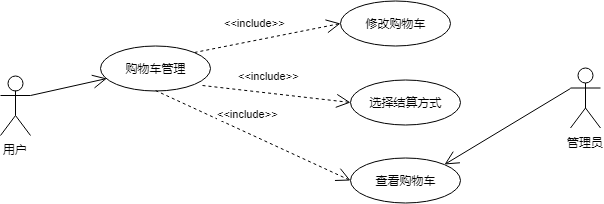


图2-18购物车管理用例图

管理员可以在后台查看用户的购物车列表，并凭借其推荐相关商品给用户，用户可以查看自己的购物车，查看购物车用例分析表如表2-12所示

表2-12查看购物车用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 查看购物车 |
| 标识符 | 011 |
| 说明 | 用户在前台购车模块查看购物车，管理员在后台查看购物车信息 |
| 参与者 | 管理员、用户 |
| 状态 | 通过 |
| 前置条件 | 用户登录进入购物车模块，管理员登录进入后台管理系统首页 |
| 后置条件 | 如果该用例成功，可以对购物车进行操作 |
| 基本操作流程 | 1.管理员进入后台管理系统，查看购物车信息  2.用户点击购物车，可以查看自己的购物车信息 |

用户可以在购物车内修改商品，查看购物车用例分析表如表2-13所示

表2-13修改购物车用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 修改购物车 |
| 标识符 | 012 |
| 说明 | 用户在前台系统，进入购物车模块修改购物车 |
| 参与者 | 用户 |

续表2-13

|  |  |
| --- | --- |
| 状态 | 通过 |
| 前置条件 | 用户登录进入购物车模块 |
| 后置条件 | 如果该用例成功，可以进行其他操作 |
| 基本操作流程 | 1.用户进入购物车模块  2.对相关商品进行修改操作 |

用户进入购物车模块选择结算方式，选择结算方式用例分析表如表2-14所示

表2-14选择结算方式用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 选择结算方式 |
| 标识符 | 013 |
| 说明 | 用户在前台系统，进入购物车模块选择结算方式 |
| 参与者 | 用户 |
| 状态 | 通过 |
| 前置条件 | 用户登录进入购物车模块 |
| 后置条件 | 如果该用例成功，进入创建订单页面 |
| 基本操作流程 | 1.用户进入购物车模块  2.选择结算方式后，跳转至创建订单页面 |

（五）取餐方式管理用例图如图2-19所示：

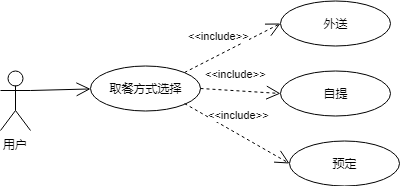


图2-19 取餐方式选择用例图

用户创建订单后，选择取餐方式并选择外送，外送的用例分析表如表2-15所示

表2-15外送用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 外送 |
| 标识符 | 014 |
| 说明 | 用户在前台系统，创建订单后选择取餐方式 |
| 参与者 | 用户 |
| 状态 | 通过 |
| 前置条件 | 用户创建订单后 |
| 后置条件 | 选择付款方式，并选择配送地址 |
| 基本操作流程 | 1.用户在创建订单后选择取餐方式  2. 选择外送 |

用户创建订单后，选择取餐方式并选择预约，预约的用例分析表如表2-16所示

表2-16预约用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 预约 |
| 标识符 | 015 |
| 说明 | 用户在前台系统，创建订单后选择取餐方式 |
| 参与者 | 用户 |
| 状态 | 通过 |
| 前置条件 | 用户创建订单后 |
| 后置条件 | 选择付款方式 |
| 基本操作流程 | 1.用户在创建订单后选择取餐方式  2.选择预约后，凭借个人信息获得预约内容 |

用户创建订单后，选择取餐方式并选择自提，自提的用例分析表如表2-17所示

表2-17自提用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 自提 |

续表2-17

|  |  |
| --- | --- |
| 标识符 | 016 |
| 说明 | 用户在前台系统，创建订单后选择取餐方式 |
| 参与者 | 用户 |
| 状态 | 通过 |
| 前置条件 | 用户创建订单后 |
| 后置条件 | 选择付款方式 |
| 基本操作流程 | 1.用户在创建订单后选择取餐方式  2.选择自提后，凭借自提信息前往门店获取商品 |

（六）周边商品管理用例图如图2-20所示：

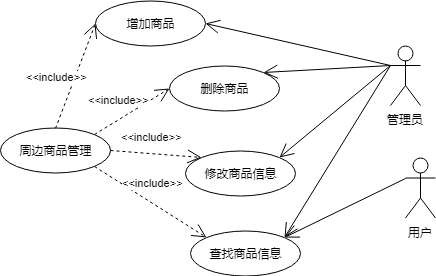


图2-20 周边商品管理用例图

管理员进入周边商品管理模块，在商品列表点击增加按钮，增加商品用例分析表如表2-18所示：

表2-18增加商品用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 增加商品 |
| 标识符 | 017 |

续表2-18

|  |  |
| --- | --- |
| 说明 | 管理员才有权限增加商品 |
| 参与者 | 管理员 |
| 状态 | 通过 |
| 前置条件 | 管理员进入周边商品管理模块 |
| 后置条件 | 跳转至商品列表 |
| 基本操作流程 | 1.管理员点击进入周边商品管理模块  2.管理员在周边商品管理模块点击新增商品按钮  3.填写商品信息后，提交  4.显示新增成功，跳转商品列表 |

管理员进入周边商品管理模块，在商品列表点击修改按钮，修改商品信息用例分析表如表2-19所示：

表2-19修改商品信息用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 修改商品信息 |
| 标识符 | 018 |
| 说明 | 管理员才有权限修改商品信息 |
| 参与者 | 管理员 |
| 状态 | 通过 |
| 前置条件 | 管理员进入周边商品管理模块 |
| 后置条件 | 跳转至商品列表 |
| 基本操作流程 | 1.管理员点击进入周边商品管理模块  2.管理员在周边商品管理模块点击修改商品按钮  3.填写商品信息后，提交  4.显示修改成功，跳转商品列表 |

管理员进入周边商品管理模块，进入商品列表查看商品，用户可以在前台系统菜单查看商品，查看商品用例分析表如表2-20所示

表2-20查看商品用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 查看商品 |
| 标识符 | 019 |

续表2-20

|  |  |
| --- | --- |
| 说明 | 管理员在后台管理系统商品列表查看商品，用户在前台系统菜单查看商品 |
| 参与者 | 管理员、用户 |
| 状态 | 通过 |
| 前置条件 | 管理员进入周边商品管理模块，用户登录进入商品菜单 |
| 后置条件 | 进行其他操作 |
| 基本操作流程 | 1.管理员点击进入周边商品管理模块  2.管理员进入商品列表  3.用户进入商品菜单，浏览商品 |

管理员进入周边商品管理模块，在商品列表对相关商品点击删除按钮，删除商品用例分析表如表2-21所示

表2-21删除商品用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 删除商品 |
| 标识符 | 020 |
| 说明 | 管理员才有权限删除商品 |
| 参与者 | 管理员 |
| 状态 | 通过 |
| 前置条件 | 管理员进入周边商品管理模块 |
| 后置条件 | 跳转至商品列表 |
| 基本操作流程 | 1.管理员点击进入周边商品管理模块  2.管理员在周边商品管理模块点击删除商品按钮  3.显示删除成功，跳转商品列表 |

（七）发票管理用例图如图2-21所示：

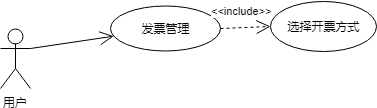


图2-21发票管理用例图

用户进入前台系统后，在支付订单完毕后，可以选择开票方式，选择开票方式如表2-22所示

表2-22选择开票方式用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 选择开票方式 |
| 标识符 | 021 |
| 说明 | 用户在支付订单后才可以选择开票方式 |
| 参与者 | 用户 |
| 状态 | 通过 |
| 前置条件 | 用户支付订单后 |
| 后置条件 | 进行其他操作 |
| 基本操作流程 | 1.用户选择开票方式  2.生成word文件，添加至下载列表  3.下载完毕后，跳转至前台首页 |

（八）订单管理用例图如图2-22所示：

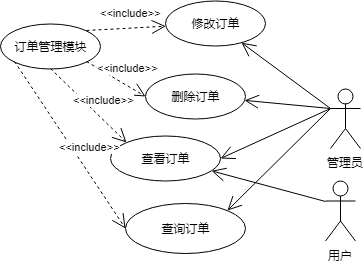


图2-22 订单管理用例图

用户进入前台系统后，可以在个人中心查看订单，管理员可以在订单管理模块查看订单列表，查看订单如表2-23所示

表2-23查看订单用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 查看订单 |

续表2-23

|  |  |
| --- | --- |
| 标识符 | 022 |
| 说明 | 用户在个人中心查看订单，管理员在订单管理模块查看订单列表 |
| 参与者 | 用户、管理员 |
| 状态 | 通过 |
| 前置条件 | 用户登录进入个人中心，管理员进入订单管理模块 |
| 后置条件 | 进行其他操作 |
| 基本操作流程 | 1.用户点击查看订单  2.跳转至个人订单列表  3.管理员点击订单管理  4.点击订单列表，跳转至订单列表 |

管理员可以在订单管理模块查看订单列表后，可以点击修改按钮对相应订单信息进行修改，修改订单如表2-24所示

表2-24修改订单用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 修改订单 |
| 标识符 | 023 |
| 说明 | 拥有管理员权限才能对订单进行修改 |
| 参与者 | 管理员 |
| 状态 | 通过 |
| 前置条件 | 管理员进入订单管理模块 |
| 后置条件 | 跳转至订单列表 |
| 基本操作流程 | 1.管理员点击订单管理  2.点击订单列表，跳转至订单列表  3.对指定订单点击修改按钮  4.修改完成后，跳转至订单列表 |

管理员可以在订单管理模块查看订单列表后，可以点击删除按钮对相应订单信息进行删除，删除订单如表2-25所示

表2-25删除订单用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 删除订单 |
| 标识符 | 024 |
| 说明 | 拥有管理员权限才能对订单进行删除 |
| 参与者 | 管理员 |
| 状态 | 通过 |
| 前置条件 | 管理员进入订单管理模块 |
| 后置条件 | 跳转至订单列表 |
| 基本操作流程 | 1.管理员点击订单管理  2.点击订单列表，跳转至订单列表  3.对指定订单点击删除按钮  4.删除完成后，跳转至订单列表 |

管理员可以在订单管理模块查看订单列表后，可以点击填补按钮对相应订单信息进行填补，填补订单如表2-26所示

表2-26填补订单用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 修改订单 |
| 标识符 | 025 |
| 说明 | 拥有管理员权限才能对订单进行填补，填补订单的目的是为了避免因为网络原因，漏加某些订单 |
| 参与者 | 管理员 |
| 状态 | 通过 |
| 前置条件 | 管理员进入订单管理模块 |
| 后置条件 | 跳转至订单列表 |
| 基本操作流程 | 1.管理员点击订单管理  2.点击订单列表，跳转至订单列表  3.点击填补订单按钮  4.填补完成后，跳转至订单列表 |

（九）公告板管理用例图如图2-23所示：

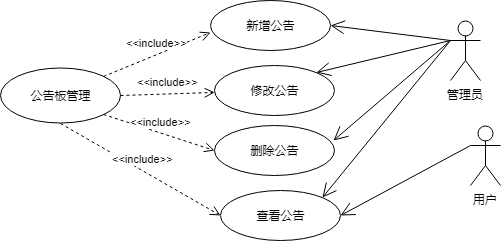


图2-23 公告板管理用例图

管理员进入公告板管理模块后，点击新增按钮新增公告信息，新增公告如表2-27所示

表2-27新增公告用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 新增公告 |
| 标识符 | 026 |
| 说明 | 拥有管理员权限才能对公告进行新增，新增公告提醒用户注意事项 |
| 参与者 | 管理员 |
| 状态 | 通过 |
| 前置条件 | 管理员进入公告板管理模块 |
| 后置条件 | 跳转至公告板列表 |
| 基本操作流程 | 1.管理员点击公告板管理  2.点击公告板列表，进入公告板列表  3.点击新增公告按钮  4.新增完成后，跳转至公告板列表 |

管理员进入公告板管理模块后，对应相关公告信息点击修改按钮，修改公告如表2-28所示

表2-28修改公告用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 修改公告 |
| 标识符 | 027 |

续表2-28

|  |  |
| --- | --- |
| 说明 | 拥有管理员权限才能对公告进行修改 |
| 参与者 | 管理员 |
| 状态 | 通过 |
| 前置条件 | 管理员进入公告板管理模块 |
| 后置条件 | 跳转至公告板列表 |
| 基本操作流程 | 1.管理员点击公告板管理  2.点击公告板列表，进入公告板列表  3.对应相关公告板信息选择修改信息  4.修改完成后，跳转至公告板列表 |

管理员进入公告板管理模块后，对应相关公告信息点击删除按钮，删除公告如表2-29所示

表2-29删除公告用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 删除公告 |
| 标识符 | 028 |
| 说明 | 拥有管理员权限才能对公告进行删除 |
| 参与者 | 管理员 |
| 状态 | 通过 |
| 前置条件 | 管理员进入公告板管理模块 |
| 后置条件 | 跳转至公告板列表 |
| 基本操作流程 | 1.管理员点击公告板管理  2.点击公告板列表，进入公告板列表  3.对相应的公告点击删除按钮  4.删除完成后，跳转至公告板列表 |

管理员进入公告板管理模块后，点击公告板列表查看所有公告信息，用户在前台导航栏的通知栏选择查看公告，查看公告如表2-30所示

表2-30查看公告用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 查看公告 |

续表2-30

|  |  |
| --- | --- |
| 标识符 | 029 |
| 说明 | 管理员和用户都可以在相关界面查看所有通告信息 |
| 参与者 | 管理员、用户 |
| 状态 | 通过 |
| 前置条件 | 管理员进入公告板管理模块，用户进入前台首页 |
| 后置条件 | 进行其他操作 |
| 基本操作流程 | 1.管理员点击公告板管理  2.点击公告板列表，进入公告板列表，查看公告  3.用户登录进入前台首页，点击通知栏  4.查看所有公告信息 |

（十）评论管理用例图如图2-24所示：

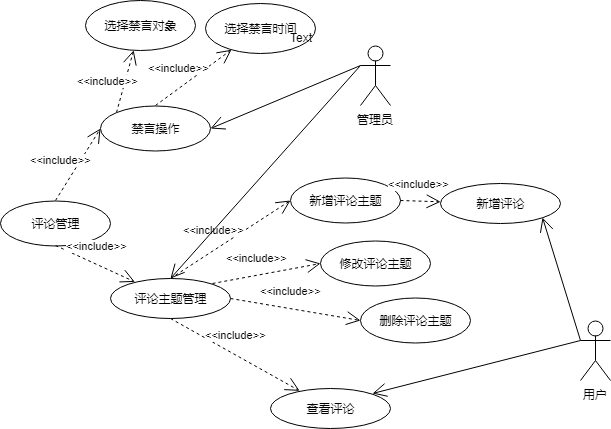


图2-24 评论管理用例图

管理员进入评论管理模块后，点击禁言子模块，选择其他操作，禁言如表2-31所示

表2-31禁言用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 禁言 |

续表2-31

|  |  |
| --- | --- |
| 标识符 | 030 |
| 说明 | 管理员才拥有禁言权限，对相关的用户进行操作 |
| 参与者 | 管理员 |
| 状态 | 通过 |
| 前置条件 | 管理员进入评论管理模块 |
| 后置条件 | 跳转禁言对象列表 |
| 基本操作流程 | 1.管理员点击评论管理模块  2.点击禁言列表，进入禁言列表  3.选择禁言对象与禁言时间  4.跳转至禁言列表 |

管理员进入评论管理模块后，点击主题管理子模块，选择其他操作，用户可以在前台查看主题与发表评论、查看评论，主题管理如表2-32所示

表2-32主题管理用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 主题管理 |
| 标识符 | 031 |
| 说明 | 管理员可以在主题管理对主题进行增删改查操作、用户可以在前台发表评论、查看评论 |
| 参与者 | 管理员、用户 |
| 状态 | 通过 |
| 前置条件 | 管理员进入评论管理模块、用户进入前台 |
| 后置条件 | 进行其他操作 |
| 基本操作流程 | 1.管理员点击评论管理模块  2.进入主题管理模块，进行其他操作  3.用户进入商品详情页面，查看评论，购买过商品也可以发表商品评论 |

（十一）饮料管理用例图如图2-25所示：

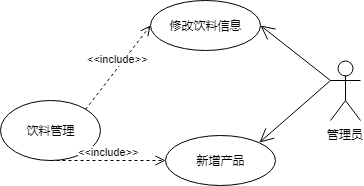


图2-25 饮料管理用例图

管理员进入饮料管理模块后，点击修改饮料信息，对相关信息进行修改，修改饮料信息如表2-33所示

表2-33修改饮料信息用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 修改饮料信息 |
| 标识符 | 032 |
| 说明 | 管理员才能对饮料信息进行修改 |
| 参与者 | 管理员 |
| 状态 | 通过 |
| 前置条件 | 管理员进入饮料管理模块 |
| 后置条件 | 跳转饮料列表 |
| 基本操作流程 | 1.管理员点击饮料管理模块  2.点击饮料列表，进入饮料列表  3.选择相关饮料后，点击修改按钮  4.修改信息完成后，提示修改完成 |

管理员进入饮料管理模块后，点击修改饮料信息，对相关信息进行修改，修改饮料信息如表2-34所示

表2-34新增产品用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 新增产品 |
| 标识符 | 033 |
| 说明 | 管理员才能新增产品 |

续表2-34

|  |  |
| --- | --- |
| 参与者 | 管理员 |
| 状态 | 通过 |
| 前置条件 | 管理员进入饮料管理模块 |
| 后置条件 | 跳转饮料列表 |
| 基本操作流程 | 1.管理员点击饮料管理模块  2.点击饮料列表，进入饮料列表  3.点击新增按钮，提交信息  4.提示新增成功后，跳转至饮料列表 |

（十二）品类管理用例图如图2-26所示：

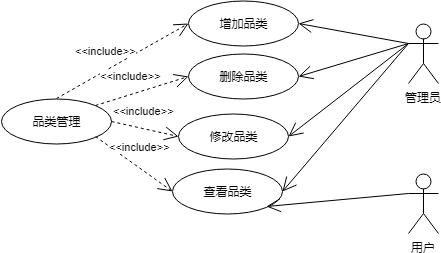


图2-26 品类管理用例图

管理员进入品类管理模块后，点击新增品类信息，新增品类信息如表2-35所示

表2-35新增品类用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 新增品类 |
| 标识符 | 034 |
| 说明 | 管理员才能新增品类 |
| 参与者 | 管理员 |
| 状态 | 通过 |

续表2-35

|  |  |
| --- | --- |
| 前置条件 | 管理员进入品类管理模块 |
| 后置条件 | 跳转品类列表 |
| 基本操作流程 | 1.管理员点击品类管理模块  2.点击品类列表，进入品类列表  3.点击新增按钮，提交信息  4.提示新增成功后，跳转至品类列表 |

管理员进入品类管理模块后，点击修改品类信息，修改品类信息如表2-36所示

表2-36修改品类信息用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 修改品类 |
| 标识符 | 035 |
| 说明 | 管理员才能修改品类信息 |
| 参与者 | 管理员 |
| 状态 | 通过 |
| 前置条件 | 管理员进入品类管理模块 |
| 后置条件 | 跳转品类列表 |
| 基本操作流程 | 1.管理员点击品类管理模块  2.点击品类列表，进入品类列表  3.对相应的品类点击修改按钮，提交信息  4.提示修改成功 |

管理员进入品类管理模块后，点击删除品类信息，删除品类信息如表2-37所示

表2-37删除品类信息用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 删除品类 |
| 标识符 | 036 |
| 说明 | 管理员才能删除品类信息 |
| 参与者 | 管理员 |

续表2-37

|  |  |
| --- | --- |
| 状态 | 通过 |
| 前置条件 | 管理员进入品类管理模块 |
| 后置条件 | 跳转品类列表 |
| 基本操作流程 | 1.管理员点击品类管理模块  2.点击品类列表，进入品类列表  3.对相应的品类点击删除按钮  4.提示删除成功 |

管理员进入品类管理模块后，进入品类信息列表即可查看所有品类信息，用户进入前台菜单，即可查看所有品类信息，查看品类信息如表2-38所示

表2-38查看品类信息用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 查看品类 |
| 标识符 | 037 |
| 说明 | 管理员、用户都可以在相关页面查看品类信息 |
| 参与者 | 管理员、用户 |
| 状态 | 通过 |
| 前置条件 | 管理员进入品类管理模块、用户进入前台首页 |
| 后置条件 | 进行其他操作 |
| 基本操作流程 | 1.管理员点击品类管理模块  2.点击品类列表，进入品类列表  3.用户进入前台菜单页面 |

（十三）门店管理用例图如图2-27所示：



图2-27 门店管理用例图

管理员进入门店管理模块后，点击新增门店，新增门店如表2-39所示

表2-39新增门店用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 新增门店 |
| 标识符 | 038 |
| 说明 | 管理员才能新增门店 |
| 参与者 | 管理员 |
| 状态 | 通过 |
| 前置条件 | 管理员进入门店管理模块 |
| 后置条件 | 跳转门店列表 |
| 基本操作流程 | 1.管理员点击门店管理模块  2.点击门店列表，进入门店列表  3.点击新增按钮，提交信息  4.提示新增成功后，跳转至门店列表 |

管理员进入门店管理模块后，点击修改门店信息，修改门店信息如表2-40所示

表2-40修改门店信息用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 修改门店 |
| 标识符 | 039 |
| 说明 | 管理员才能修改门店信息 |
| 参与者 | 管理员 |

续表2-40

|  |  |
| --- | --- |
| 状态 | 通过 |
| 前置条件 | 管理员进入门店管理模块 |
| 后置条件 | 跳转门店列表 |
| 基本操作流程 | 1.管理员点击门店管理模块  2.点击门店列表，进入门店列表  3.对相应的门店点击修改按钮，提交信息  4.提示修改成功 |

管理员进入品类管理模块后，点击删除门店信息，删除门店信息如表2-41所示

表2-41删除门店用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 删除门店 |
| 标识符 | 040 |
| 说明 | 管理员才能删除门店信息 |
| 参与者 | 管理员 |
| 状态 | 通过 |
| 前置条件 | 管理员进入门店管理模块 |
| 后置条件 | 跳转门店列表 |
| 基本操作流程 | 1.管理员点击门店管理模块  2.点击门店列表，进入门店列表  3.对相应的门店点击删除按钮  4.提示删除成功 |

管理员进入门店管理模块后，进入门店信息列表即可查看所有门店信息，用户进入前台门店，即可查看所有门店信息，查看门店信息如表2-42所示

表2-42查看门店信息用例分析表

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 描述对象 | 查看门店 |
| 标识符 | 041 |
| 说明 | 管理员、用户都可以在相关页面查看门店信息 |

续表2-42

|  |  |
| --- | --- |
| 参与者 | 管理员、用户 |
| 状态 | 通过 |
| 前置条件 | 管理员进入门店管理模块、用户进入前台首页 |
| 后置条件 | 进行其他操作 |
| 基本操作流程 | 1.管理员点击门店管理模块  2.点击门店列表，进入门店列表  3.用户进入前台门店页面 |

## 2.3非功能需求

### 2.3.1 外部接口需求

用户接口：用户界面反应速度应当<=1s，保持设计的界面简洁明了、保持界面元素一致、界面元素的位置应当合理放置、页面应当具有吸引力。

软件接口：此软件需要百度地图开放平台API用于获取经纬度。

### 2.3.2 性能需求

1、响应时间：操作平均时间应当小于2.00秒，页面响应时间<=50ms，事务处理时间<=0.50s，查询等待时间<=1.00s，修改操作时间<=0.50s。

2、灵活性：该系统采用B/S模式，可以确保各种常用操作系统客户端的正常使用，并支持IE7.0/IE8.0及更高版本的客户端。

3、稳定性：在系统出现卡死或者出现停电现象，系统会自动退出，并保存信息，下次正常运行可以保证数据不丢失。假如进行错误操作时，系统会明确提示错误信息并指导用户按照系统错误处理手册进行处理。

4、可扩展性：在系统设计时，应当为以后该项目的新增模块以及优化功能提供扩展接口，保证系统具有一定的扩展能力。

5、用户友好性：本系统的界面设计应该简洁、操作简单，界面风格应该遵循当前主流的方向设计。

6、可靠性：所有数据库写操作都必须使用事务处理模式来支持事务的提交和回滚，并且数据可以同时写入两个独立存储（可以与系统软件配合完成）。

### 2.3.3 安全需求

系统应对数据和应用的安全性作充分考虑，提供较完善的数据加密及访问控制，提供日志跟踪与统计分析功能，保证管理员与服务员进行的每一个对管理的操作记录，确保系统正常运行。

按照访问控制可以将系统对不同角色分配在不同的组内，每个组的对于系统的数据的权限都不一致。这样比较方便对用户数据的安全进行管理，以防数据遭到随意的篡改或者盗取。确保系统安全可靠，也可以将系统的数据上传至云端，让第三方数据安全管理公司代为管理，这样能最大确保系统的安全性。

### 2.3.4 系统运行需求

（一）软件需求

客户端：系统运行环境采用window10旗舰版（简体中文版）

服务器端：系统运行环境采用Window10旗舰版

（二）硬件需求

服务器端如表2-43所示：

表2-43服务器端要求

|  |  |
| --- | --- |
| **硬件** | **要求** |
| CPU | 最低要求2.3GHz Intel i5处理器（推荐2.5GHz Intel i7处理器） |
| 内存 | 最低RAM要求1G（推荐2G内存） |
| 硬盘 | 需要5GB以上的可用空间 |
| 驱动器 | 需要CD-ROM或DVD-ROM驱动器 |
| 显示 | Super VGA (800x600)或更高分辨率的显示器（颜色设置为32位真彩色） |
| 鼠标 | Microsoft鼠标或兼容的指点设备 |

客户端如表2-44所示：

表2-44客户器端要求

|  |  |
| --- | --- |
| **硬件** | **要求** |
| CPU | 最低要求2.3GHz Intel i5处理器（推荐2.5GHz Intel i7处理器） |
| 内存 | 最低RAM要求512MB（推荐1G内存） |
| 硬盘 | 需要5GB以上的可用空间 |
| 驱动器 | 需要CD-ROM或DVD-ROM驱动器 |
| 显示 | Super VGA (800x600)或更高分辨率的显示器（颜色设置为32位真彩色） |
| 鼠标 | Microsoft鼠标或兼容的指点设备 |

（三）环境需求

本系统基于Java技术SpringBoot框架开发，能够使得本系统可以直接在Windows、Linux、MacOS三大主流操作系统上进行部署及运行得益于JVM的跨平台性，无需大量修改代码以适应不同平台，即可实现良好的可扩展性和跨平台。同时，系统整体上也使用开源框架和技术来实现，并且对操作环境的配置要求不高，再之不需要花费更多的人力成本于软件维护上，这使得它具备良好的经济可行性，如表2-45所示：

表2-45环境需求

|  |  |
| --- | --- |
| **要求** | **内容** |
| 操作系统 | Windows 7及以上版本 |
| JDK | 1.8及其以上 |
| 应用服务器 | Apache Tomcat 8.0及以上 |
| 应用服务器 | Mysql 5.7及以上版本 |
| 开发框架 | SpringBoot、Layui、Bootstrap。 |

### 2.4.5设计与实现约束

（一）需求约束

本系统为咖啡厅与奶茶店所量身定制，价格与配料内容都应该遵守《中华人民共和国食品安全法》与《中华人民共和国价格法》，获取了百度地图获取经纬度的API，应当遵守《百度地图API规范》。

响应时间需要根据用户的运行环境保证在一个合理的范围，各个咖啡厅在需求上要求不一致，这就要求在开发软件时，应当赋予其灵活性，使其具有扩展性，因此本系统并不能一次性开发完成，需要根据具体情况做定夺。

本系统虽然只是为咖啡厅用户开发，但就整个项目涉猎的范围而言，是可以面向整个服务行业的系统，无论是奶茶店、餐饮店还是甜品店都可以基于咖啡厅订单系统进行二次开发，开发出适应自身的系统。因此，需求不能仅仅只局限在咖啡厅订单，也需要对其他服务行业的营业模式进行分析。可以基于咖啡厅的数据对比其他服务业的数据，以完成系统更好的优化与改进。在此过程中，应该首先以咖啡厅作为出发点，随后逐步实践与探索，并不能一次性解决所有需求，也很难做到一次定型。

（二）技术约束

考虑到本系统实际应用背景以及开发业务背景，并从安全性和可维护性两个方向和角度出发，我们必须采用开发的技术体系。因此，本系统采用JavaWeb体系为最佳。基于Java的SpringBoot技术框架具有开放的源代码，Java本身是开源的，基于Java的开发工具、其他开发框架等都是开源技术，且这些技术是目前业内（含国内）应用最为成熟、最为广泛的技术产品。另外，基于Java的其他开源框架和组件也多为开源。开源意味着可以获取并使用其源代码。一方面有利于进行技术对接，另外一方面更有利于客户方对系统深入审查，确保系统内部没有安插非法的插件等，以此确保系统本身和系统中的数据不被恶意代码所侵害。

（三）环境约束

本系统开发采用SpringBoot技术框架开发，JDK版本为JDK1.8，服务器采用的是SpringBoot集成的服务器类SpringBootApplication，数据库为Mysql5.7。

（四）隐含约束

1、咖啡厅的服务员，制作员必须能熟练掌握系统的使用与解决基础问题。

2、在意外情况发生时，系统能安全退出并保证数据保存与避免数据丢失。

# 3 总体设计

## 3.1体系结构设计

本系统主要采用的体系结构是B/S结构，由三层结构组成，分别是表现层、逻辑层、数据层。表现层的主要能力表现为用户和后台的数据交互以及输出最终查询结果，逻辑层主要功能表现为利用服务器用于完成客户端的应用逻辑功能，数据层主要表现为在接收客户端请求之后独立进行各种数据运算。

B/S具有以下优点与缺点：

优点：

客户端使用统一的浏览器。由于浏览器具有统一性的特点，所以它能够有效屏蔽不同服务商提供给用户使用服务的差异性，因此它不需要特殊的配置以及网络连接。此外，它的简便性来源于浏览器的交互特性这使得用户能非常简便地使用它，且用户行为的可继承性非常强，简单来讲就是用户只要学会了上网，不管是使用哪个应用，在使用其他互联网服务时同样有使用经验，因为他们都基于同样的浏览器操作界面。

服务器基于统一的HTTP协议。使用统一的HTTP协议也为服务提供商简化了开发模式，使得服务器开发者可以采用相对规范的开发模式，这样可以大大节省开发成本。

升级服务时，不需要和C/S架构一样更新客户端，只需要部署服务器即可，这样的作法可以减低安装带来的麻烦和大量的人力成本，而且采用这种结构更新速度特别快，因为只需要更新服务器的应用程序，客户端就可以使用更新后的应用程序。

缺点：

客户端服务器端的交互时请求-响应模式，通常动态刷新页面，响应速度明显降低，无法实现分页显示，给数据库访问造成较大的压力。

在跨浏览器上，B/S架构不尽人意。

个性化特点明显降低，无法实现具有个性化的功能要求。

在速度和安全性上需要花费巨大的设计成本。

B/S架构解释图如图3-1所示：

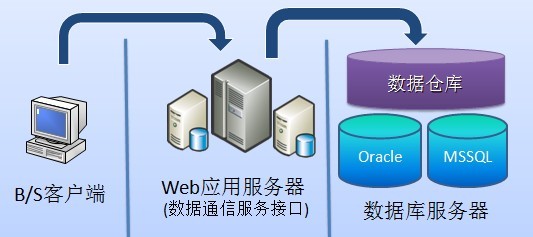


图3-1 B/S架构体系结构图

## 3.2 系统功能架构设计

系统功能架构设计如图3-2所示：

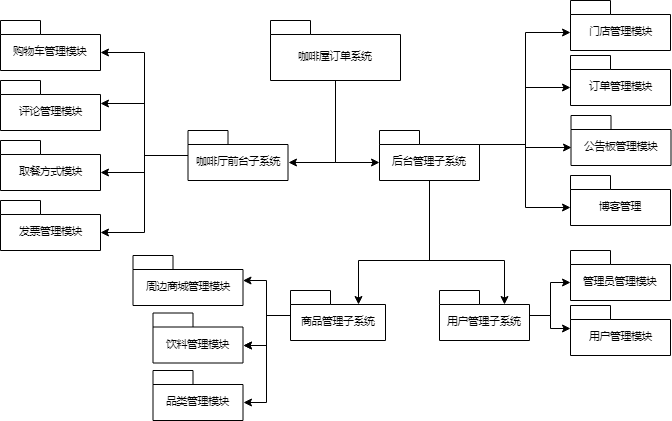


图3-2 系统功能架构设计图

咖啡屋订单系统由四个子系统、三个独立模块构成，分别是用户管理子系统、商品管理子系统、咖啡厅子系统、后台管理子系统，后台管理四个独立模块包括了订单管理模块、门店管理模块、公告板管理模块以及博客管理模块。用户管理子系统的操作对象是管理员，包括四个子模块：用户管理模块、权限分配模块、管理员管理模块、登录注册模块，为了完成对系统用户进行权限分配与信息维护。商品管理子系统包括：饮料管理模块、品类管理模块、周边商城管理模块，管理员分别对饮料与品类进行管理，包括增、删、改、查操作。咖啡厅前台子系统包括评论管理模块、购物车管理模块、取餐管理模块、发票管理模块。发票管理模块的操作对象是顾客，顾客可以通过该模块进行4种方式进行开票：按订单开票、咖啡钱包购买开票、礼品卡购买开票、充值开票，也可以查看发票记录，发票抬头，购物车管理模块的操作对象是顾客，顾客可以在此模块对购物车内的信息进行修改，并选择结算方式，

咖啡厅前台子系统操作对象是顾客。后台管理子系统操作对象是管理员，后台管理子系统包括门店管理模块、订单管理模块、公告板管理。公告板管理模块的操作对象是管理员，管理员通过该子系统对公告板进行增加、删除、修改操作。订单管理模块操作对象是管理员，通过该模块可以对订单进行增加、删除、修改、查找操作。评论管理模块的操作对象是管理员，管理员与服务员可以通过禁言操作选择禁言对象与禁言时间，也可以对主题进行管理。门店管理的操作对象是管理员，管理员通过该模块对门店进行增删改查操作，也可以列出优质门店，给顾客匹配最佳门店。

### 3.2.1 用户管理子系统

用户管理子系统，如图3-3所示

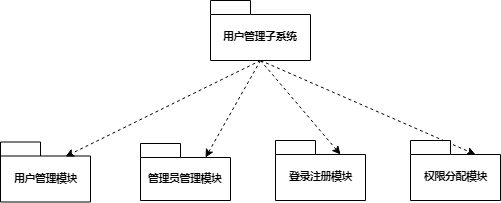


图3-3 用户管理子系统结构设计图

用户管理子系统概述：用户管理子系统包括权限分配模块、登录注册模块、用户管理、管理员管理四个子模块。权限分配模块有增加用户、删除用户、修改用户权限、查看用户信息功能，登录模块、注册模块包括管理员 、顾客、员工的登录注册功能。

### 3.2.2 商品管理子系统

商品管理子系统，如图3-4所示

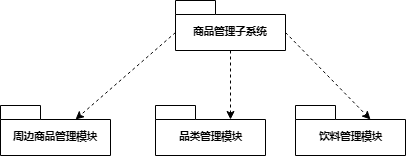


图3-4 商品管理子系统结构设计图

商品管理子系统概述：商品管理子系统包括周边商品管理模块、品类管理模块、饮料管理模块。周边商品管理模块包括周边食品管理、周边物品管理两个子功能。饮料管理模块、品类管理只有一个功能模块。

### 3.2.3 咖啡厅前台子系统

咖啡厅前台子系统，如图3-5所示

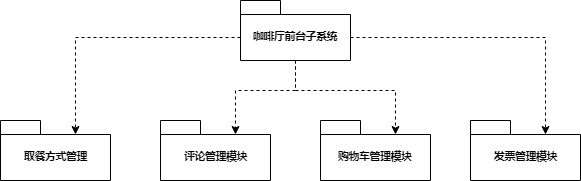


图3-5 咖啡厅前台子系统结构设计图

咖啡厅前台子系统概述：咖啡厅前台子系统包括评论管理模块、取餐方式管理模块、购物车管理模块、发票管理模块四个模块，评论管理模块包括禁言操作与评论主题管理两个子功能模块。

### 3.2.4后台管理独立模块

后台管理独立模块，如图3-6所示

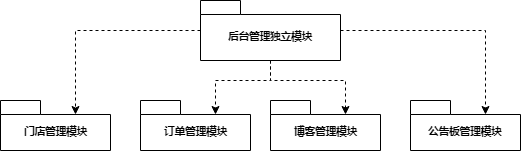


图3-6 咖啡厅后台管理独立模块结构设计图

后台管理子系统独立模块概述：后台管理子系统独立模块包括订单管理模块、门店管理模块、公告板管理模块、博客管理模块四个模块。

# 4 数据库设计

## 4.1 概念结构设计

### 4.1.1 设计思路

概念设计是用来反映现实世界中各个实体、属性、关联的数据形式，建立数据库的用户视图。

设计步骤分三步依次进行：首先是完成对局部E-R模式的设计，然后再连接每个局部的E-R模式组成一个相互关联的整体，最后优化全局的E-R模式，从而得到最后的E-R模式。

### 4.1.2 E-R图

E-R模型的“联系”用于描述实体之间的关联。一种完整的方式是对局部结构中任意两个实体类型，依据需求分析的结果，考察局部结构中任意两个实体类型之间是否存在联系。若有联系，进一步确定是1:1、1:N、M:N的关系。还需要对一个实体内部是否存在联系进行考察，以及两个实体之间是否存在联系，多个实体类型之间是否存在联系等，本系统实体关系如下：

（1）本系统的实体关系图，如图4-1所示。



图4-1 总体E-R图

（2）用户信息实体图，如图4-2所示。

实体：用户

属性：用户id，用户名，性别，手机号，用户类型，地址，头像地址，用户姓名，邮箱地址，生日，用户密码。

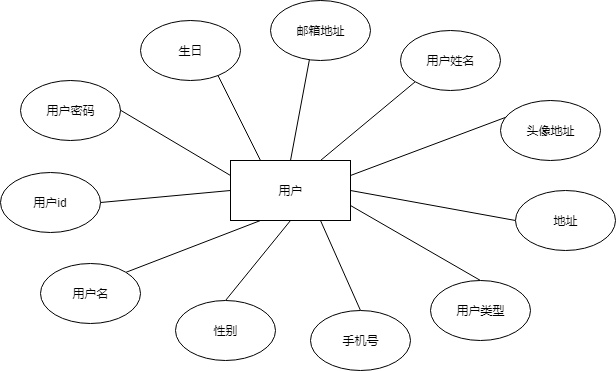


图4-2 用户实体图

（3）订单实体图，如图4-3所示

实体：订单

属性：取单号，订单内容，商品总价，优惠金额，支付方式，消费金额，订单状态，订单id，取餐方式，订单时间

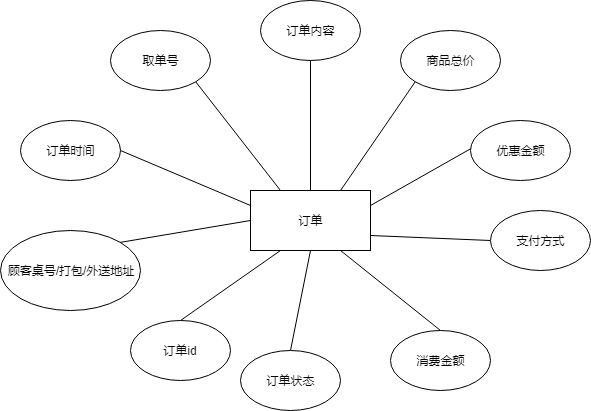


图4-3 订单实体图

（4）评论实体图，如图4-4所示

实体：评论

属性：评论id，类型，发布时间，发布者，地理位置，备注

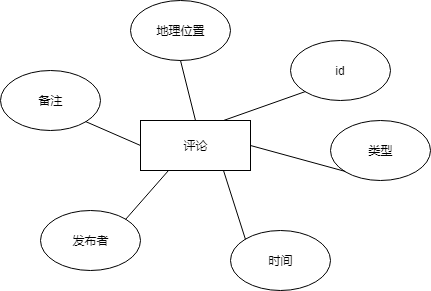


图4-4 评论实体图

（5）饮料实体类，如图4-5所示

实体：饮料

属性：饮料id，类别，上架否，主要配料，价格，糖量，额外料，温度，饮料名称，图片

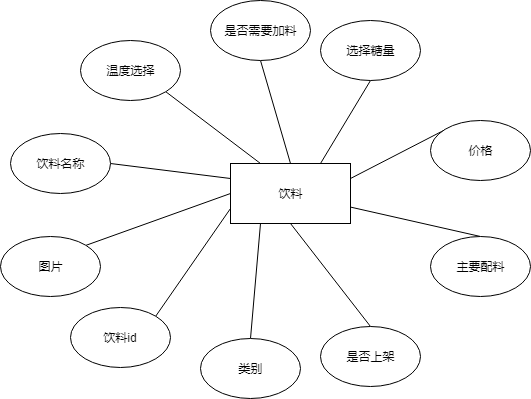


图4-5 饮料实体图

（6）品类实体类，如图4-6所示

实体：品类

属性：名称，品类id，包含饮料

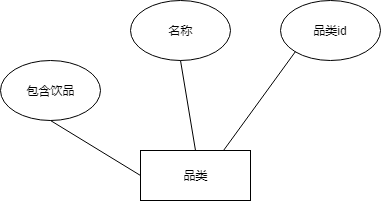


图4-6 品类实体图

（7）公告板实体图，如图4-7所示

实体：公告板

属性：公告板id，图片，公告类型，发布时间，公告内容，发布者

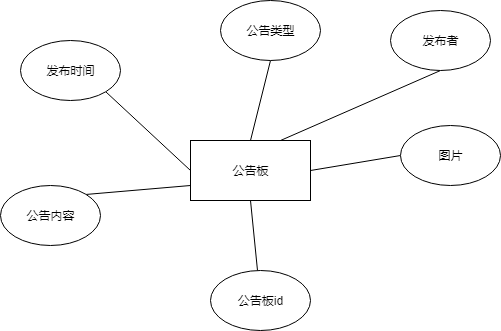


图4-7 公告板实体图

（8）发票实体图，如图4-8所示

实体：发票

属性：发票id，名称，用户邮箱地址，银行账户，手机号，地址，税务号

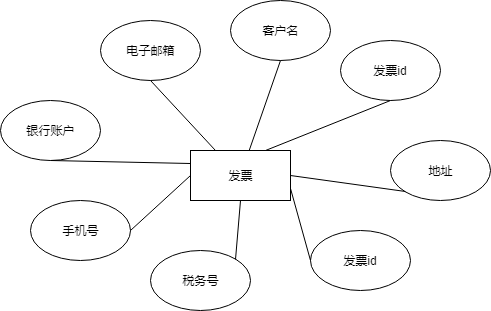


图4-8 发票实体图

（9）购物车实体图，如图4-9所示

实体：购物车

属性：id，商品数量，总价，结算方式

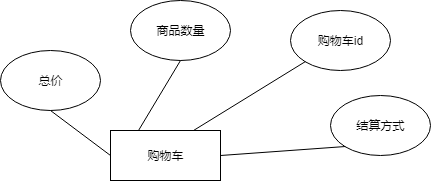


图4-9 购物车实体图

（10）积分商品实体图，如图4-10所示

实体：积分商品

属性：id，名称，所需积分，评论，类型，图片

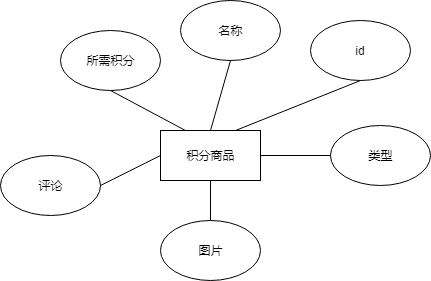


图4-10 积分商品实体图

（11）门店实体图，如图4-11所示

实体：门店

属性：id，店名，地址，特色，电话，营业时间

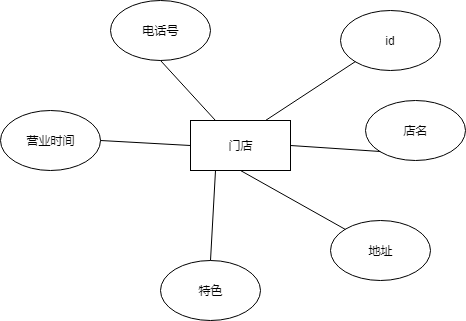


图4-11 门店实体图

（12）周边商品实体图，如图4-12所示

实体：周边商品

属性：id，名称，价格，评价，图片，类型，规格，额外备注

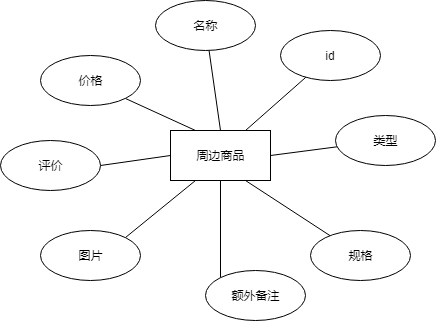


图4-12 周边商品实体图

（13）饮品订单明细实体图，如图4-13所示

实体：饮品订单明细

属性：id，折扣，商品价格，订单id，饮品id，商品数量

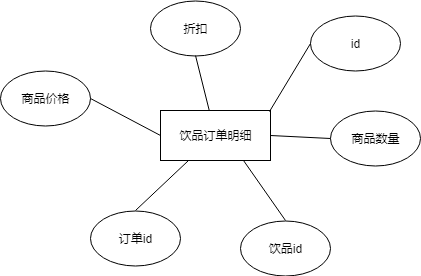


图4-13 饮品订单明细实体图

（14）周边商品订单明细实体图，如图4-14所示

实体：周边商品订单明细

属性：id，折扣，商品价格，订单id，周边商品id，商品数量

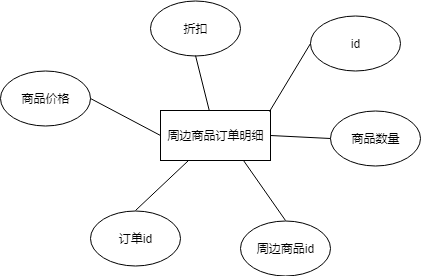


图4-14 周边商品订单明细实体图

（15）博客实体图，如图4-15所示

实体：博客

属性：id，封面，发布者，内容，时间，是否审核，主题，名称

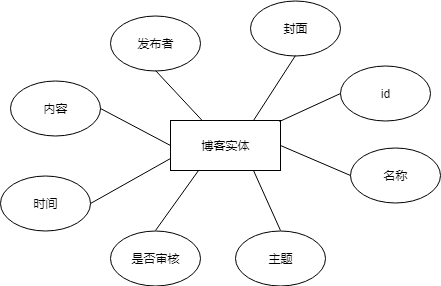


图4-15 博客实体图

（16）配送地址实体图，如图4-16所示

实体：配送地址

属性：id，省份，城市，地区，街道，详细地址，用户id，收货人，收货人电话

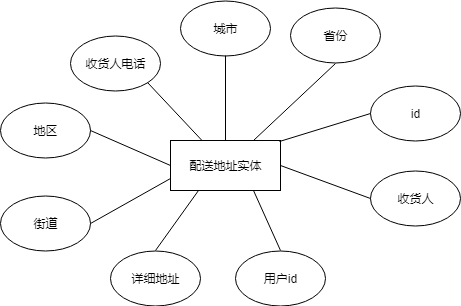


图4-16 配送地址实体图

## 4.2 逻辑结构设计

### 4.2.1 设计思路

（1）将概念结构转化为一般的关系、网状、层次模型

（2）对数据模型进行优化

### 4.2.2 逻辑模型

用户表：用户id，用户名，性别，手机号，用户类型，地址，头像地址，用户姓名，邮箱地址，生日，用户密码。

订单表：取单号，订单内容，商品总价，优惠金额，支付方式，消费金额，订单状态，订单id，取餐方式，订单时间

评论表：评论id，类型，发布时间，发布者，地理位置，备注

饮料表：饮料id，类别，上架否，主要配料，价格，糖量，额外料，温度，饮料名称，图片

品类表：名称，品类id，包含饮料

公告板表：公告板id，图片，公告类型，发布时间，公告内容，发布者

发票表：发票id，名称，用户邮箱地址，银行账户，手机号，地址，税务号

购物车表：id，商品数量，总价，结算方式

积分商品表：id，名称，所需积分，评论，类型，图片

门店表：id，店名，地址，特色，电话，营业时间

周边商品表：id，名称，价格，评价，图片，类型，规格，额外备注

饮品订单明细表：id，折扣，商品价格，订单id，饮品id，商品数量

周边商品订单明细表：id，折扣，商品价格，订单id，周边商品id，商品数量

博客表：id，封面，发布者，内容，时间，是否审核，主题，名称

配送地址表：id，省份，城市，地区，街道，详细地址，用户id，收货人，收货人电话

1. 用户表，如表4-1所示

表4-1 用户表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 类型 | 长度 | 备注 |
| 1 | 用户id | 整型 | 11 | 主键 |
| 2 | 姓名 | 字符型 | 50 |  |
| 3 | 用户名 | 字符型 | 50 |  |
| 4 | 密码 | 字符型 | 32 |  |
| 5 | 性别 | 整型 | 1 |  |
| 6 | 头像 | 字符型 | 128 |  |
| 7 | 手机号 | 字符型 | 15 |  |
| 8 | 邮箱 | 字符型 | 100 |  |
| 9 | 角色 | 整型 | 1 |  |
| 10 | 生日 | 日期 | 0 |  |
| 11 | 地址 | 字符型 | 255 |  |

1. 订单表，如表4-2所示

表4-2 订单表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 类型 | 长度 | 备注 |
| 1 | 订单id | 整型 | 11 | 主键 |
| 2 | 取单号 | 字符型 | 50 |  |
| 3 | 订单内容 | 字符型 | 50 |  |
| 4 | 商品总价 | 整型 | 20 |  |
| 5 | 优惠价格 | 整型 | 20 |  |
| 6 | 支付方式 | 字符型 | 128 |  |
| 7 | 消费金额 | 整型 | 20 |  |
| 8 | 订单状态 | 字符型 | 100 |  |
| 9 | 取餐方式 | 字符型 | 30 |  |
| 10 | 订单时间 | 日期 | 0 |  |

1. 评论表，如表4-3所示

表4-3 评论表

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | | 字段名称 | | 类型 | | 长度 | | 备注 |
| 1 | 评论id | | 整型 | | 11 | | 主键 | |
| 2 | 类型 | | 字符型 | | 50 | |  | |
| 3 | 发布时间 | | 日期 | | 0 | |  | |
| 4 | 发布者 | | 字符型 | | 20 | |  | |
| 5 | 地理位置 | | 字符型 | | 200 | |  | |
| 6 | 备注 | | 字符型 | | 128 | |  | |

1. 饮料表，如表4-4所示

表4-4 饮料表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 类型 | 长度 | 备注 |
| 1 | 饮料id | 整型 | 11 | 主键 |
| 2 | 类别 | 字符型 | 50 |  |
| 3 | 上架否 | 整型 | 11 |  |
| 4 | 主要配料 | 字符型 | 200 |  |
| 5 | 价格 | 整型 | 20 |  |
| 6 | 糖量 | 字符型 | 128 |  |
| 7 | 额外料 | 字符型 | 120 |  |
| 8 | 温度 | 字符型 | 100 |  |
| 9 | 饮料名称 | 字符型 | 100 |  |
| 10 | 图片 | 字符型 | 200 |  |

1. 品类表，如表4-5所示

表4-5 品类表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 类型 | 长度 | 备注 |

续表4-5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 品类id | 整型 | 11 | 主键 |
| 2 | 名称 | 字符型 | 50 |  |
| 3 | 包含饮料 | 字符型 | 200 |  |

1. 公告板表，如表4-6所示

表4-6 公告板表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 类型 | 长度 | 备注 |
| 1 | 公告板id | 整型 | 11 | 主键 |
| 2 | 图片 | 字符型 | 50 |  |
| 3 | 公告类型 | 字符型 | 100 |  |
| 4 | 发布时间 | 整型 | 0 |  |
| 5 | 发布内容 | 字符型 | 200 |  |
| 6 | 发布者 | 字符型 | 128 |  |

1. 发票表，如表4-7所示

表4-7 发票表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 类型 | 长度 | 备注 |
| 1 | 发票id | 整型 | 11 | 主键 |
| 2 | 名称 | 字符型 | 50 |  |
| 3 | 用户邮箱地址 | 字符型 | 100 |  |
| 4 | 银行账户 | 字符型 | 20 |  |
| 5 | 手机号 | 字符型 | 200 |  |
| 6 | 地址 | 字符型 | 128 |  |
| 7 | 税务号 | 字符型 | 128 |  |

1. 购物车表，如表4-8所示

表4-8 购物车表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 类型 | 长度 | 备注 |
| 1 | 购物车id | 整型 | 11 | 主键 |
| 2 | 商品数量 | 字符型 | 50 |  |
| 3 | 总价 | 整型 | 10 |  |
| 4 | 结算方式 | 字符型 | 120 |  |

1. 积分商品表，如表4-9所示

表4-9 积分商品表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 类型 | 长度 | 备注 |
| 1 | 商品id | 整型 | 11 | 主键 |
| 2 | 名称 | 字符型 | 50 |  |
| 3 | 所需积分 | 整型 | 20 |  |
| 4 | 评论 | 字符型 | 200 |  |

续表4-9

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | 类型 | 字符型 | 200 |  |
| 6 | 图片 | 字符型 | 128 |  |

1. 门店表，如表4-10所示

表4-10 门店表

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | | 字段名称 | | 类型 | | 长度 | | 备注 |
| 1 | | 门店id | | 整型 | | 11 | | 主键 |
| 2 | | 店名 | | 字符型 | | 50 | |  |
| 3 | | 地址 | | 字符型 | | 20 | |  |
| 4 | | 特色 | | 字符型 | | 50 | |  |
| 5 | | 电话 | | 字符型 | | 100 | |  |
| 6 | 营业时间 | | 字符型 | | 128 | |  | |

1. 周边商品表，如表4-11所示

表4-11 周边商品表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 类型 | 长度 | 备注 |
| 1 | 商品id | 整型 | 11 | 主键 |
| 2 | 名称 | 字符型 | 50 |  |
| 3 | 价格 | 整型 | 10 |  |
| 4 | 评价 | 字符型 | 200 |  |
| 5 | 图片 | 字符型 | 100 |  |
| 6 | 类型 | 字符型 | 28 |  |
| 7 | 规格 | 字符型 | 30 |  |
| 8 | 额外备注 | 字符型 | 200 |  |

1. 饮品订单明细表，如表4-12所示

表4-12 饮品订单明细表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 类型 | 长度 | 备注 |
| 1 | id | 整型 | 11 | 主键 |
| 2 | 饮品id | 整型 | 50 |  |
| 3 | 数量 | 整型 | 10 |  |
| 4 | 订单id | 字符型 | 200 |  |
| 5 | 价格 | 整型 | 11 |  |
| 6 | 折扣 | 字符型 | 28 |  |

1. 周边商品订单明细表，如表4-13所示

表4-13 周边商品订单表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 类型 | 长度 | 备注 |
| 1 | id | 整型 | 11 | 主键 |

续表4-13

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 商品id | 整型 | 50 |  |
| 3 | 数量 | 整型 | 10 |  |
| 4 | 订单id | 字符型 | 200 |  |
| 5 | 价格 | 整型 | 11 |  |
| 6 | 折扣 | 字符型 | 28 |  |

1. 博客表，如表4-14所示

表4-14 博客表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 类型 | 长度 | 备注 |
| 1 | id | 整型 | 11 | 主键 |
| 2 | 名称 | 字符型 | 50 |  |
| 3 | 主题 | 字符型 | 50 |  |
| 4 | 是否审核 | 整型 | 11 |  |
| 5 | 封面 | 字符型 | 100 |  |
| 6 | 内容 | 字符型 | 255 |  |
| 7 | 作者 | 字符型 | 30 |  |
| 8 | 时间 | Date | 0 |  |

1. 配送地址表，如表4-15所示

表4-15 配送地址表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 类型 | 长度 | 备注 |
| 1 | id | 整型 | 11 | 主键 |
| 2 | 省份 | 字符型 | 50 |  |
| 3 | 城市 | 字符型 | 10 |  |
| 4 | 地区 | 字符型 | 200 |  |
| 5 | 街道 | 字符型 | 100 |  |
| 6 | 详细信息 | 字符型 | 28 |  |
| 7 | 用户id | 整型 | 30 |  |
| 8 | 收货人 | 字符型 | 200 |  |
| 9 | 收货人电话 | 字符型 | 200 |  |

## 4.3 物理结构设计

### 4.3.1 存取方式

本文研究采用MYSQL数据库进行数据的存储。

当前DBMS通常提供多种访问方法，例如索引方法，HASH方法等。 本文使用了索引方法。 在数据库中，索引使应用程序可以快速查找表中的数据，而不必扫描整个数据库。 索引是表中的数据和相应存储位置的列表。 索引的使用可以大大减少数据的查询时间。

### 4.3.2 存储结构

（1）用户表，如表4-16所示

表4-16 用户表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 英文名 | 类型 | 长度 | 备注 | |
| 用户id | Id | Int | 11 | 主键 | |
| 姓名 | Name | Varchar | 50 |  | |
| 用户名 | Username | Varchar | 50 | | 唯一 |
| 密码 | Password | Varchar | 32 | |  |
| 性别 | Gender | int | 1 | | 1是男0是女 |
| 头像 | Picture | Varchar | 128 | |  |
| 手机号 | Tel | Varchar | 15 | | 唯一 |
| 邮箱 | Email | Varchar | 100 | | 唯一 |
| 角色 | Roles | Varchar | 1 | |  |
| 生日 | Birth | Date | 0 | |  |
| 地址 | Address | Varchar | 255 | |  |

（2）订单表，如表4-17所示

表4-17 订单表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 英文名 | 类型 | 长度 | 备注 |
| 订单id | Id | Int | 11 | 主键 |
| 取单号 | Orderid | Varchar | 50 | 唯一 |
| 订单内容 | Orderinfo | Varchar | 50 |  |
| 商品总价 | Totalprice | Int | 20 |  |
| 优惠价格 | Discount | Int | 20 |  |
| 支付方式 | Payway | Varchar | 128 |  |
| 消费金额 | Factpay | Varchar | 20 |  |
| 订单状态 | Orderstatus | Varchar | 100 |  |
| 取餐方式 | Catchway | Varchar | 30 |  |
| 订单时间 | Ordertime | Date | 0 |  |

（3）评论表，如表4-18所示

表4-18评论表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 英文名 | 类型 | 长度 | 备注 |
| 评论id | Id | Int | 11 | 主键 |
| 类型 | Type | Varchar | 50 |  |
| 发布时间 | Releasetime | Date | 0 |  |

续表4-18

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 发布者 | Author | Varchar | 20 |  |
| 地理位置 | Address | Varchar | 200 |  |
| 备注 | Info | Varchar | 128 |  |

（4）饮料表，如表4-19所示

表4-19饮料表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 英文名 | 类型 | 长度 | 备注 |
| 饮料id | Id | Int | 11 | 主键 |
| 类别 | Type | Varchar | 50 |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 上架否 | Ifsold | Int | 11 | 1在售0停售 |
| 主要配料 | Info | Varchar | 200 |  |
| 价格 | Price | Int | 20 |  |
| 糖量 | Suger | Varchar | 128 |  |
| 额外料 | Otherthing | Varchar | 120 |  |
| 温度 | Temperature | Varchar | 100 |  |
| 饮料名称 | Drinkname | Varchar | 100 |  |
| 图片 | Picture | Varchar | 200 |  |

（5）品类表，如表4-20所示

表4-20品类表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 英文名 | 类型 | 长度 | 备注 |
| 品类id | Id | Int | 11 | 主键 |
| 名称 | Typename | Varchar | 50 |  |
| 包含饮料 | Belongs | Varchar | 200 |  |

（6）公告板表，如表4-21所示

表4-21公告板表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 英文名 | 类型 | 长度 | 备注 |
| 公告板id | Id | Int | 11 | 主键 |
| 图片 | Picture | Varchar | 50 |  |
| 公告类型 | Type | Varchar | 100 |  |
| 发布时间 | Releasetime | Date | 0 |  |
| 发布内容 | Releaseinfo | Varchar | 200 |  |
| 发布者 | Author | Varchar | 128 |  |

（7）发票表，如表4-22所示

表4-22发票表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 英文名 | 类型 | 长度 | 备注 |
| 发票id | Id | Int | 11 | 主键 |

续表4-22

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | Name | Varchar | 50 |  |
| 用户邮箱地址 | Email | Varchar | 100 |  |
| 银行账户 | Account | Varchar | 20 |  |
| 手机号 | Tel | Varchar | 200 |  |
| 地址 | Address | Varchar | 128 |  |
| 税务号 | Billid | Varchar | 128 |  |

（8）购物车表，如表4-23所示

表4-23购物车表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段名称 | 英文名 | 类型 | 长度 | | 备注 | |
| 购物车id | Id | Int | | 11 | | 主键 |
| 商品数量 | Num | Varchar | | 50 | |  |
| 总价 | Totalprice | Int | | 10 | |  |
| 结算方式 | Payway | Varchar | | 120 | |  |

（9）积分商品表，如表4-24所示

表4-24 积分商品表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 英文名 | 类型 | 长度 | 备注 |
| 商品id | Id | Int | 11 | 主键 |
| 名称 | Name | Varchar | 50 |  |
| 所需积分 | Credit | Int | 20 |  |
| 评论 | Comment | Varchar | 200 |  |
| 类型 | Type | Varchar | 200 |  |
| 图片 | Picture | Varchar | 128 |  |

（10）门店表，如表4-25所示

表4-25 门店表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 英文名 | 类型 | 长度 | 备注 |
| 门店id | Id | Int | 11 | 主键 |
| 店名 | Storename | Varchar | 50 |  |
| 地址 | Address | Varchar | 20 |  |
| 特色 | Specility | Varchar | 50 |  |
| 电话 | Tel | Varchar | 100 |  |
| 营业时间 | Worktime | Varchar | 128 |  |

（11）周边商品表，如表4-26所示

表4-26 周边商品表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 英文名 | 类型 | 长度 | 备注 |

续表4-26

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 商品id | Id | Int | 11 | 主键 |
| 名称 | Name | Varchar | 50 |  |
| 价格 | Price | Int | 10 |  |
| 评价 | Comment | Varchar | 200 |  |
| 图片 | Picture | Varchar | 100 |  |
| 类型 | Type | Varchar | 28 |  |
| 规格 | Size | Varchar | 30 |  |
| 额外备注 | Extrainfo | Varchar | 200 |  |

（12）饮品订单明细表，如表4-27所示

表4-27 饮品订单明细表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 英文名 | 类型 | 长度 | | 备注 |
| Id | Id | Int | | 11 | 主键 |
| 饮品id | Drinkid | Int | | 50 |  |
| 数量 | Num | Int | | 10 |  |
| 订单id | Orderid | Varchar | | 200 |  |
| 价格 | Price | Int | | 11 |  |
| 折扣 | Discount | Varchar | | 28 |  |

(13)周边商品订单明细表，如表4-28所示

表4-28 周边商品订单明细表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 英文名 | 类型 | 长度 | 备注 |
| id | Id | Int | 11 | 主键 |
| 商品id | Productid | Int | 50 |  |
| 数量 | Num | Int | 10 |  |
| 订单id | Orderid | Varchar | 200 |  |
| 折扣 | Discount | Int | 11 |  |
| 单价 | Price | Varchar | 28 |  |

(14)博客表，如表4-29所示

表4-29 博客表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 英文名 | 类型 | 长度 | 备注 |
| id | Id | Int | 11 | 主键 |
| 名称 | Name | Varchar | 50 |  |
| 主题 | Type | Varchar | 50 |  |
| 是否审核 | Ifcheck | Int | 11 |  |
| 封面 | Picture | Varchar | 100 |  |
| 内容 | Content | Varchar | 255 |  |
| 作者 | Author | Varchar | 30 |  |

续表4-29

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 时间 | Time | Date | 0 |  |

(15)配送地址表，如表4-30所示

表4-30 配送地址表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 英文名 | 类型 | 长度 | 备注 |
| id | Id | Int | 11 | 主键 |
| 省份 | Province | Varchar | 50 |  |
| 城市 | City | Varchar | 10 |  |
| 地区 | Region | Varchar | 200 |  |
| 街道 | Street | Varchar | 100 |  |
| 详细信息 | Info | Varchar | 28 |  |
| 用户id | Userid | Int | 30 |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 收货人 | Catchman | Varchar | 200 |  |
| 收货人电话 | Catchmantel | Varchar | 200 |  |

# 5 界面设计

## 5.1 界面关系图或工作流图

（一）后台管理界面工作流程如图5-1所示

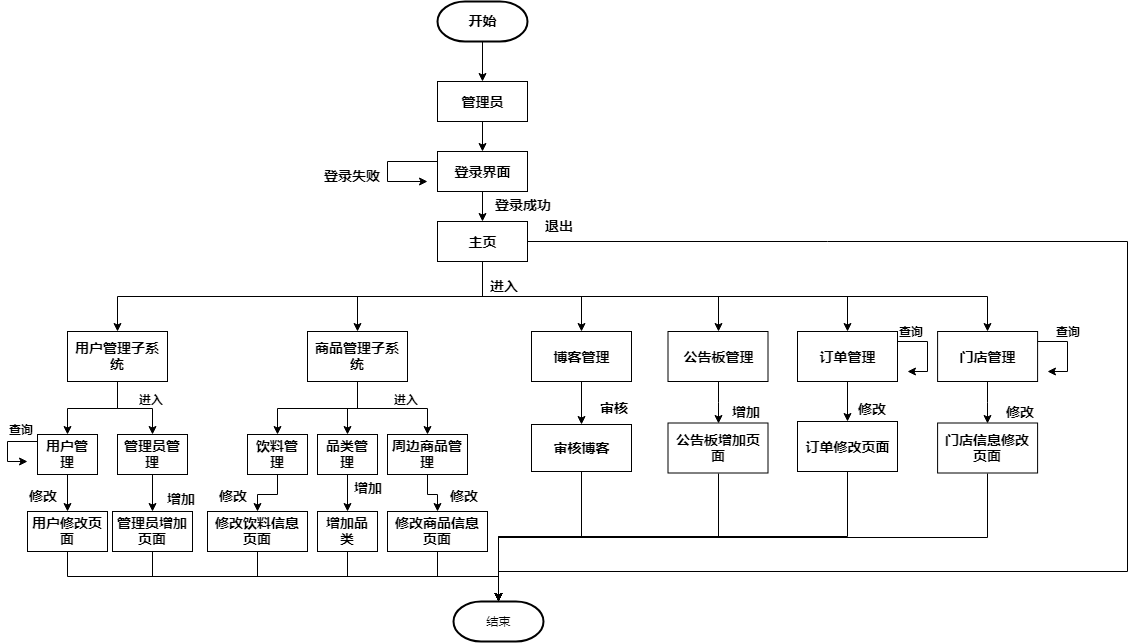


图5-1 后台管理界面工作流图

（二）前台界面工作流程如图5-2所示

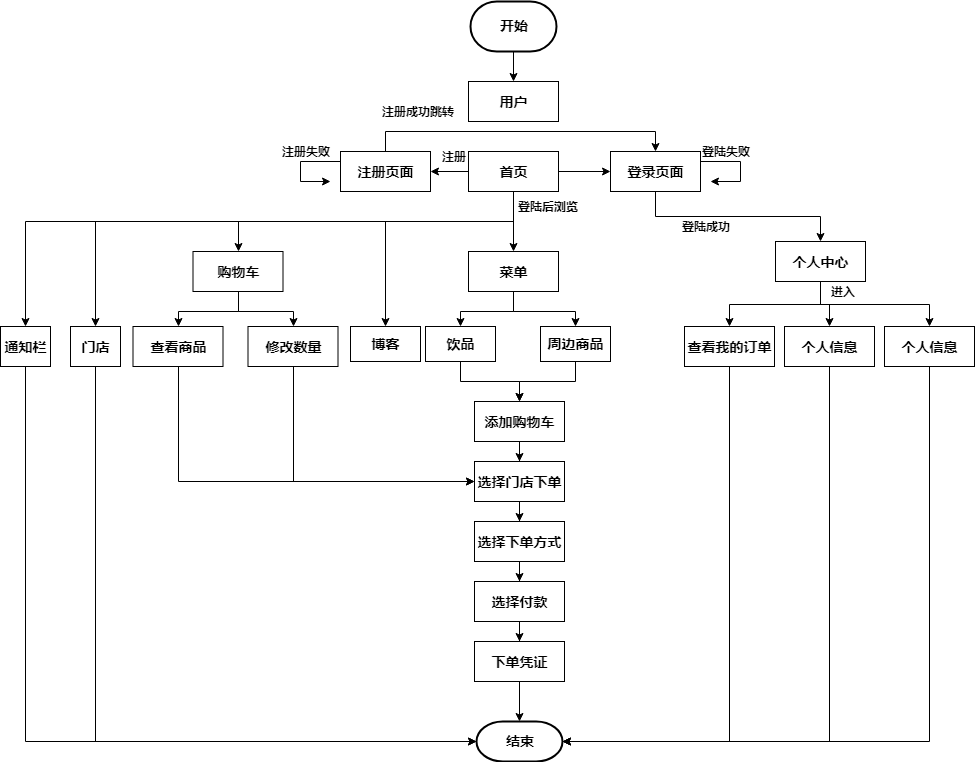


图5-2 前台界面工作流图

## 5.2 界面设计成果

### 5.2.1 主界面

(一) 后台管理

（1）首页，如图5-3所示

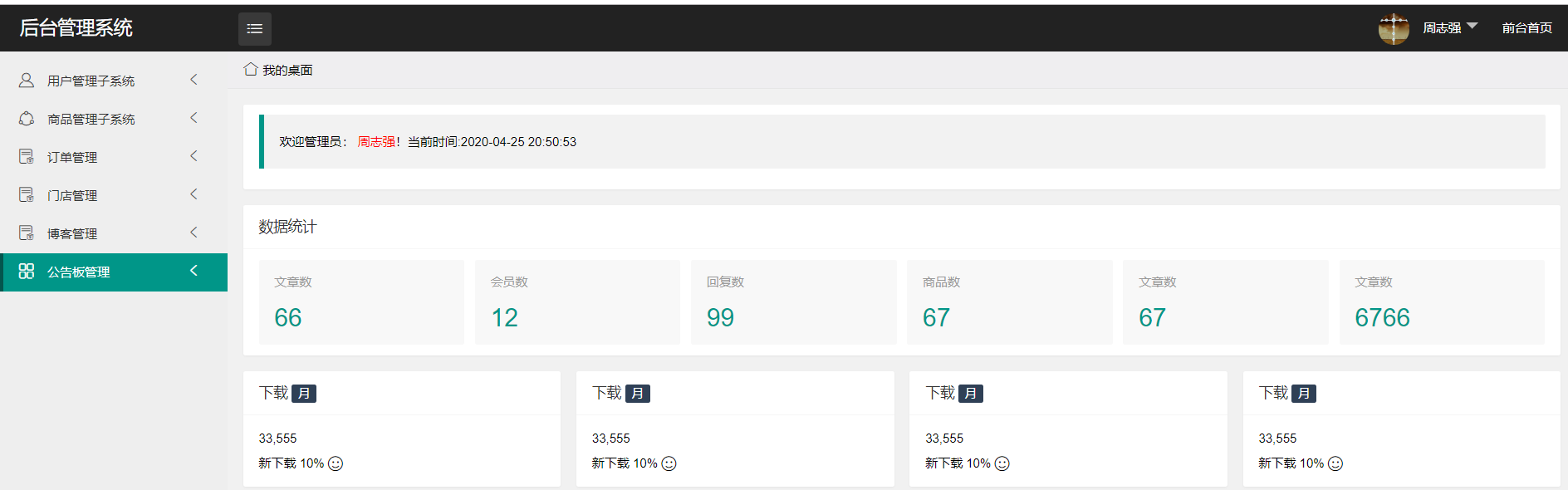


图5-3 后台管理首页

（2）用户管理子系统，如图5-4、5-5所示

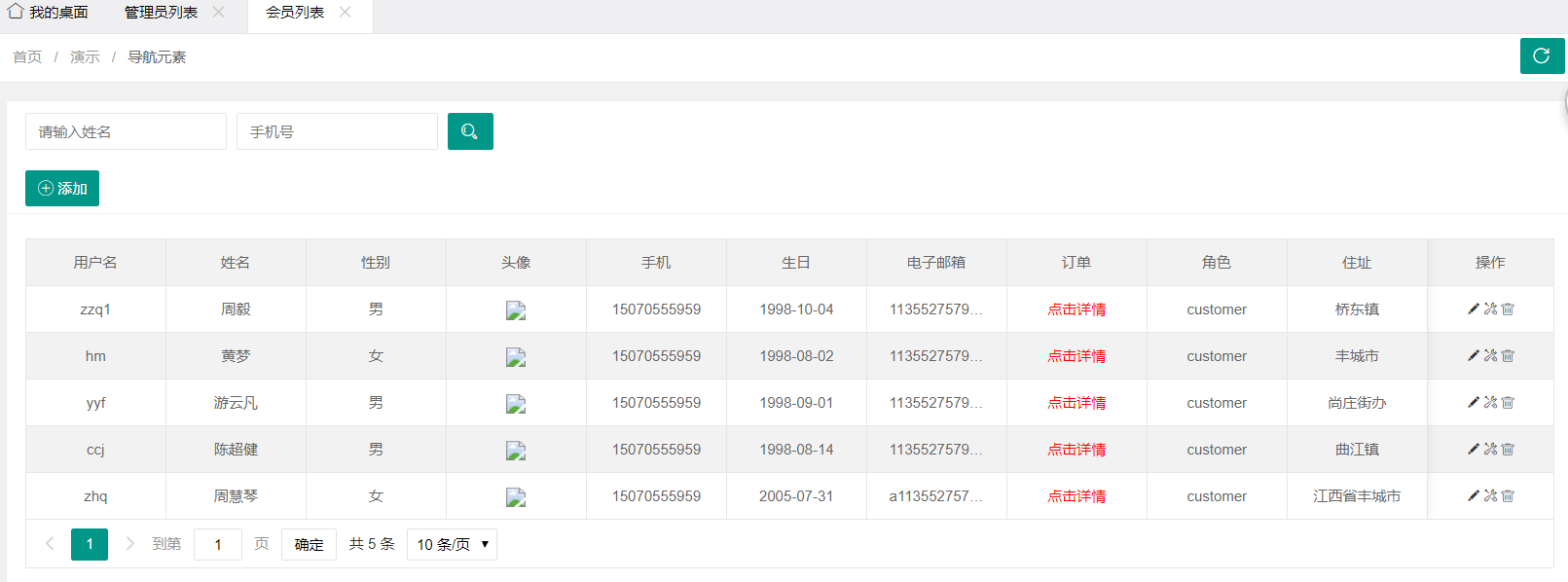


图5-4会员列表

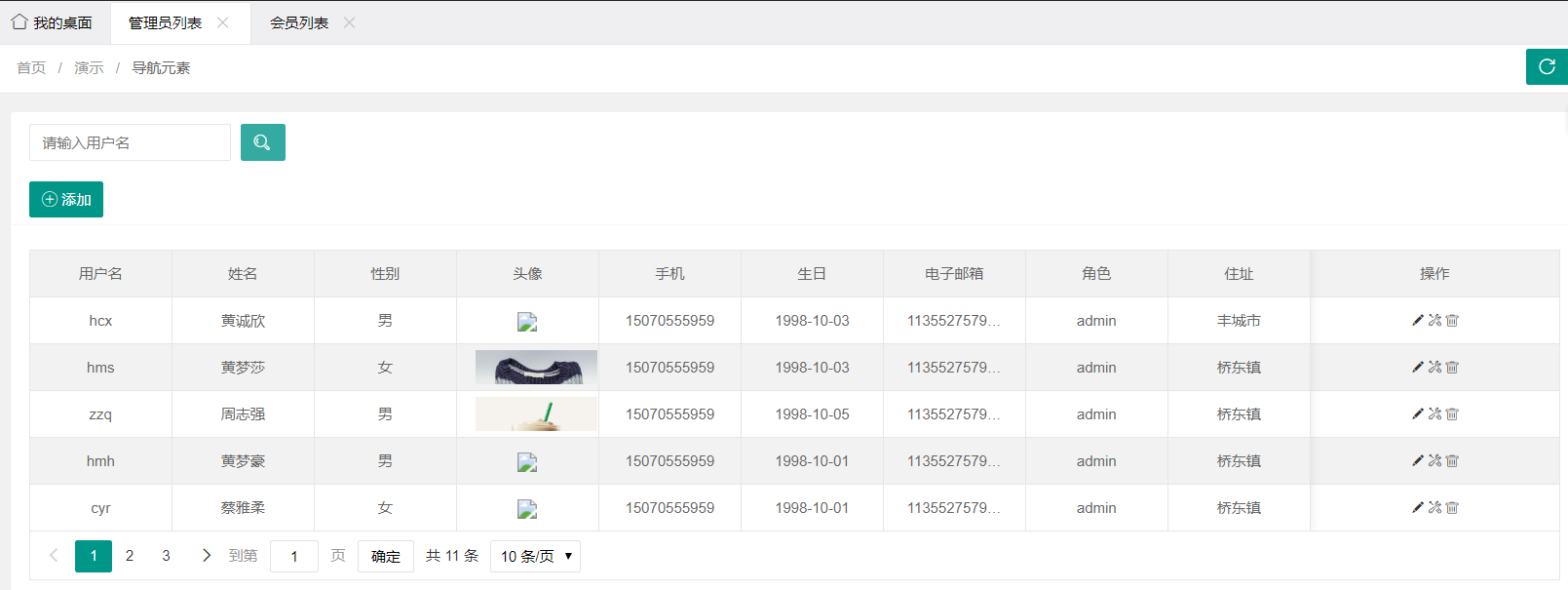


图5-5管理员列表

（3）商品管理子系统，如图5-6、5-7、5-8所示

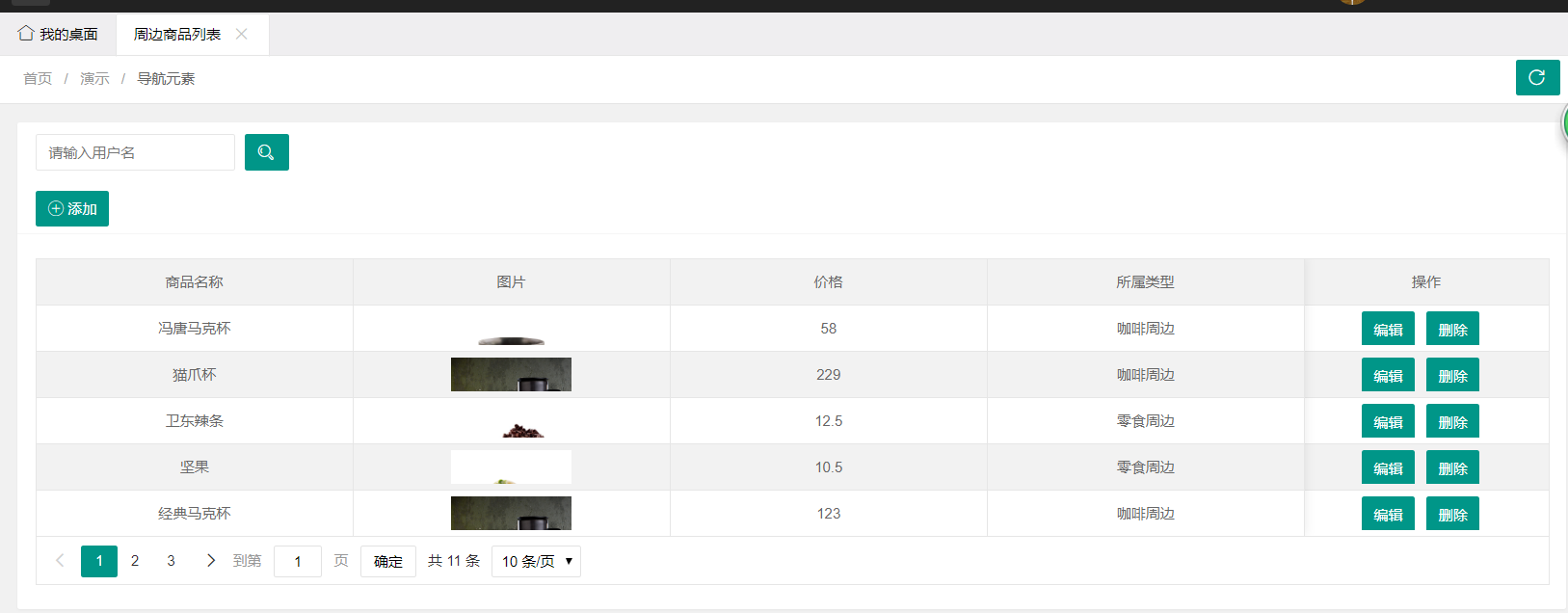


图5-6周边商品列表



图5-7饮品列表



图5-8品类列表

（4）门店管理，如图5-9、5-10所示

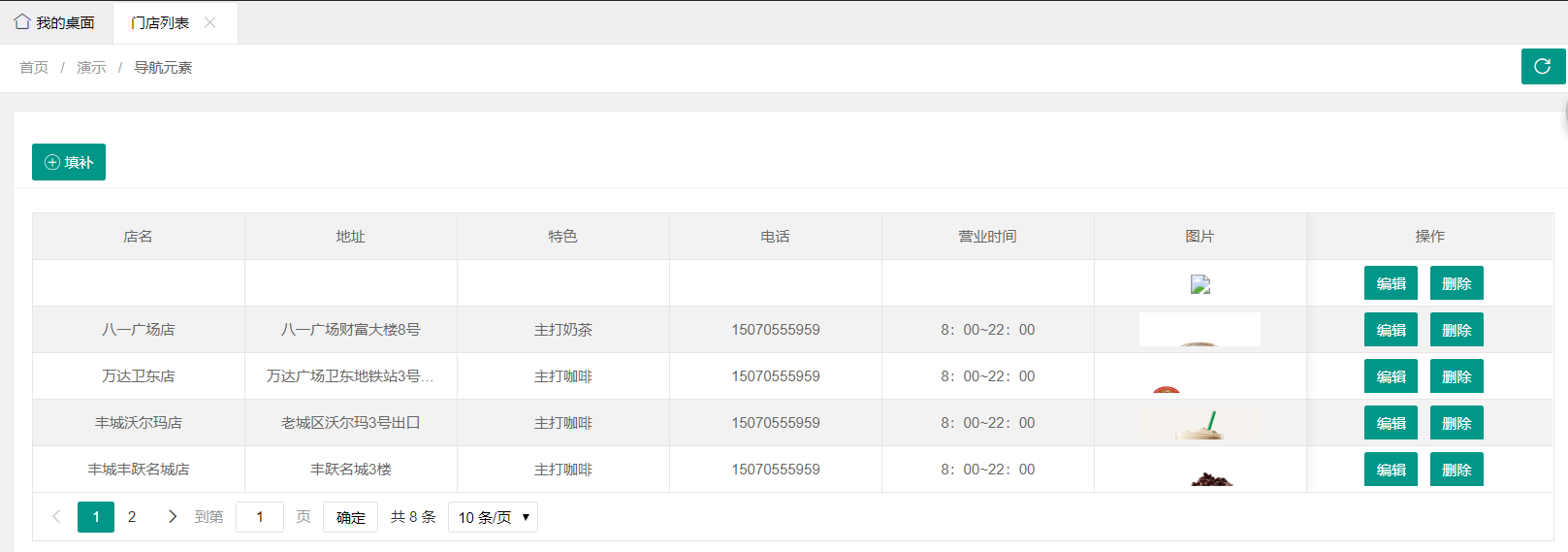


图5-9门店列表

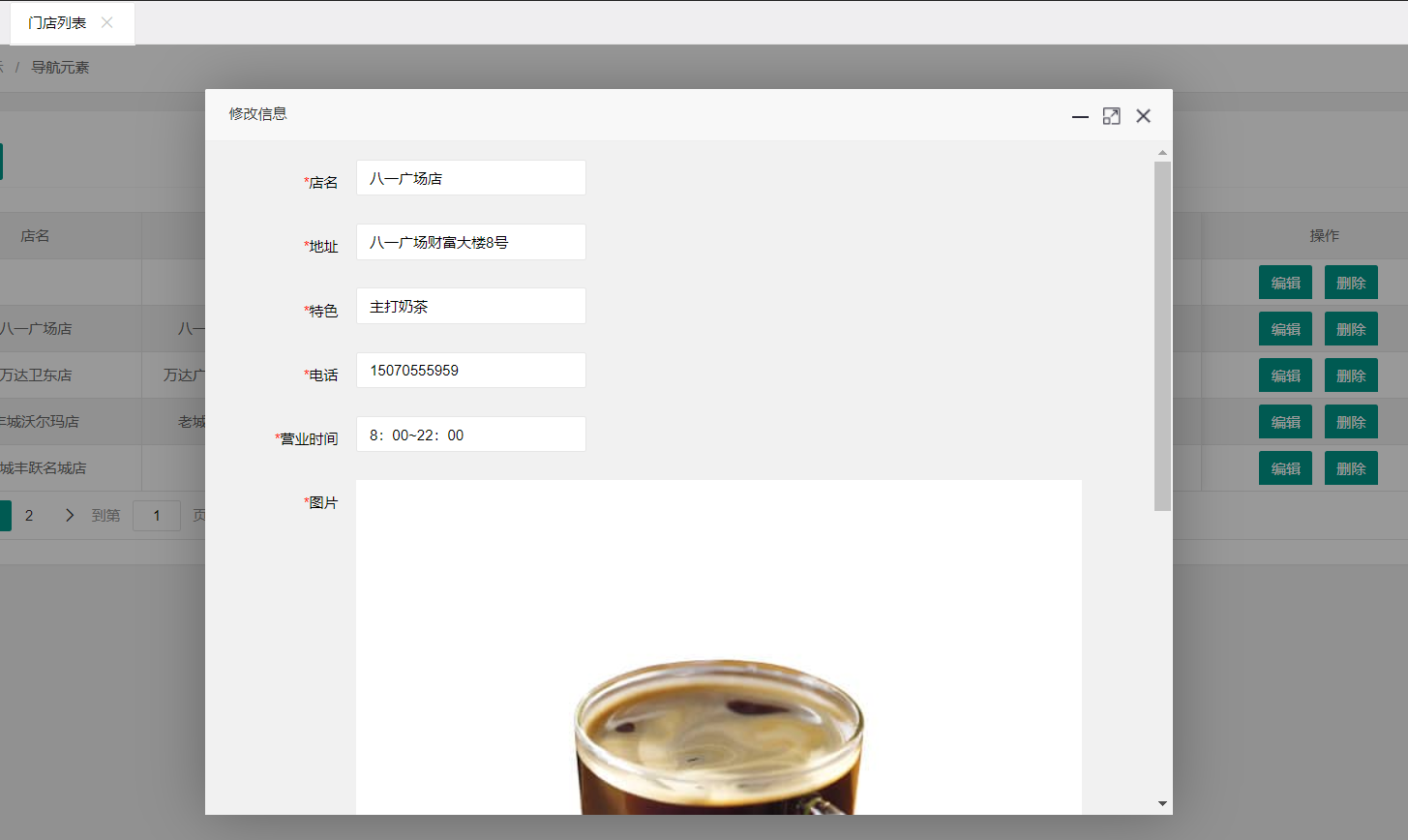


图5-10门店信息修改页面

（5）订单管理，如图5-11、5-12所示



图5-11饮品订单列表



图5-12周边商品订单列表

（6)博客管理，如图5-13所示



图5-13博客列表

(7)公告板管理，如图5-14所示



图5-14公告板列表

（二）前台页面

（1）首页，如图5-15、5-16、5-17所示

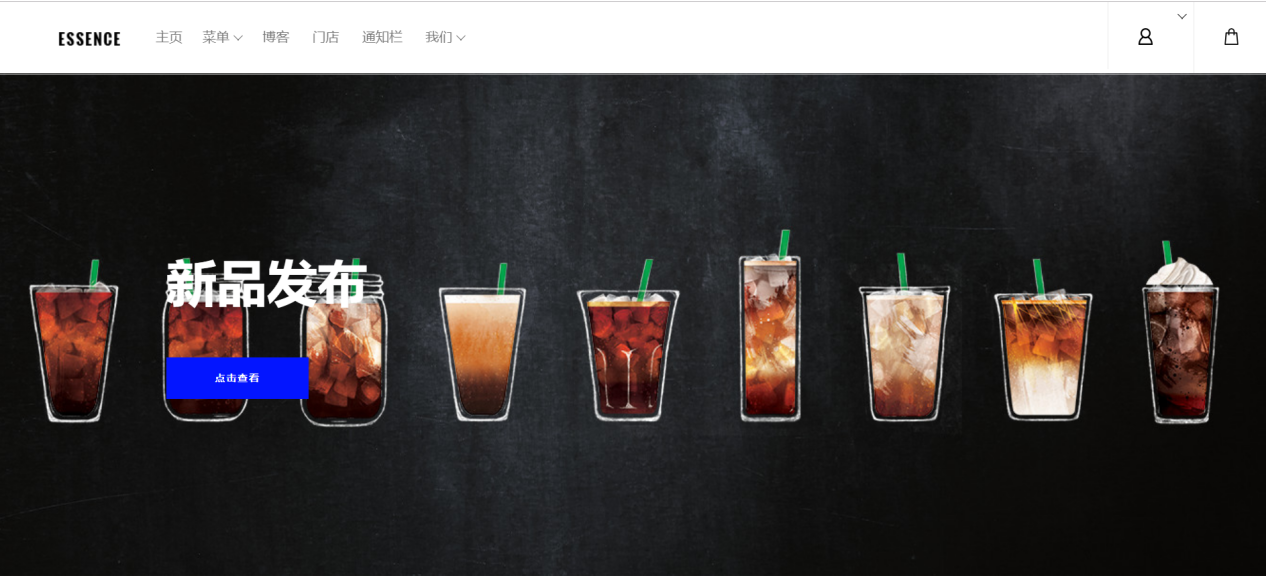


图5-15首页1



图5-16首页2



图5-17首页3

### 5.2.2 子界面

（一）后台管理子界面

(1)增加，如图5-18所示

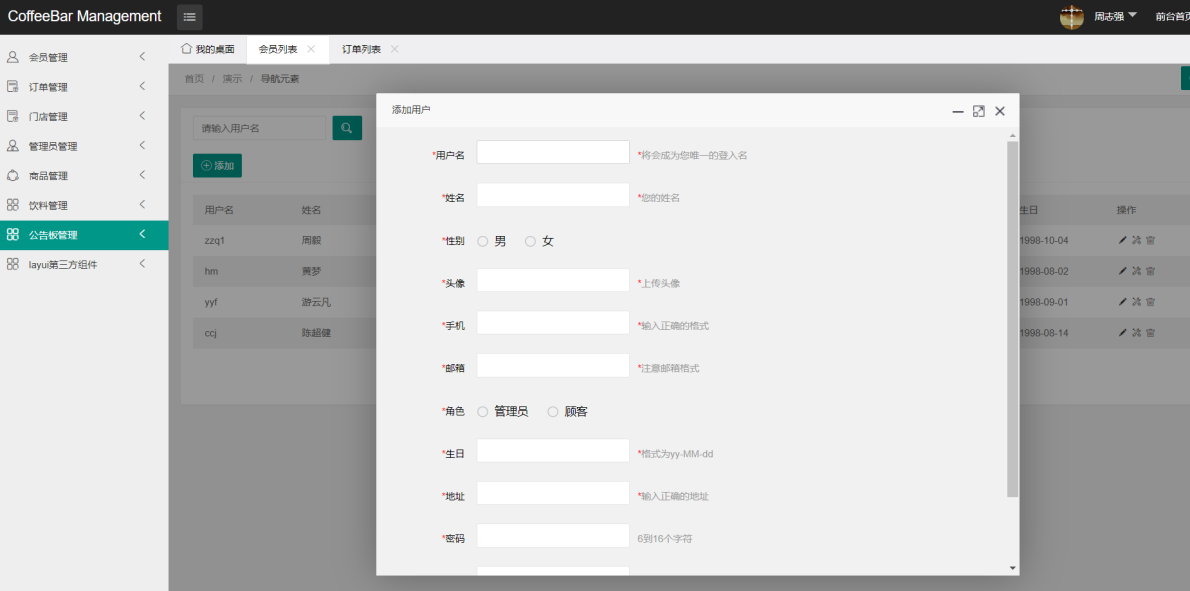


图5-18增加页面

（2）修改，如图5-19所示

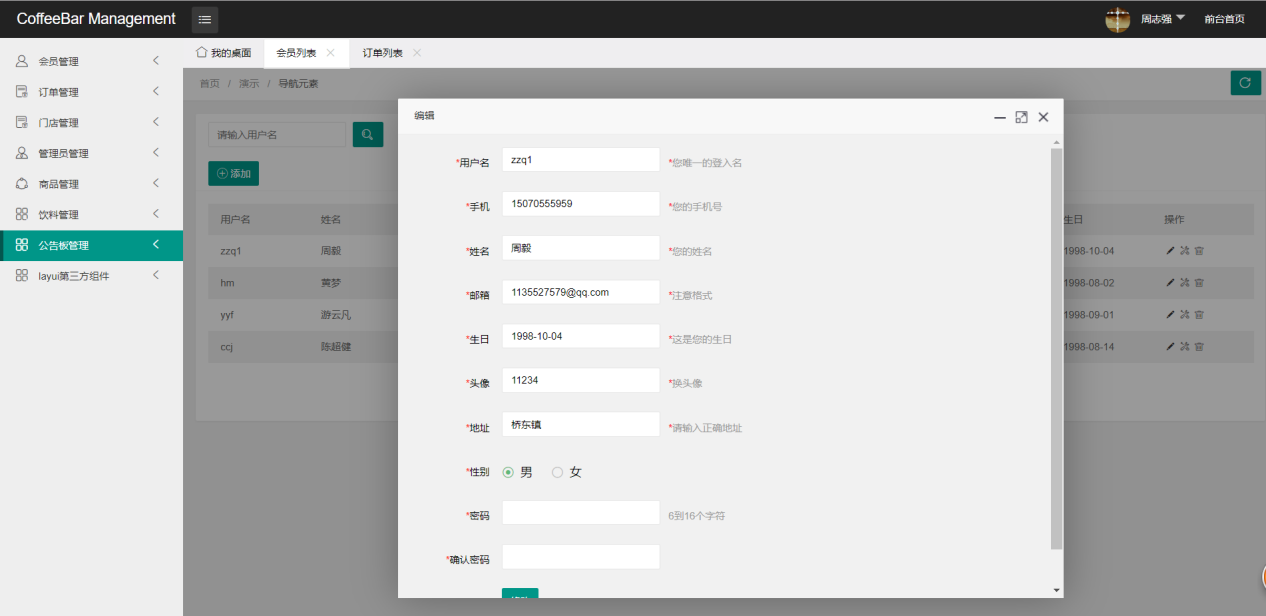


图5-19修改页面

（3）注册，如图5-20所示



图5-20注册页面

（4）登录，如图5-21所示

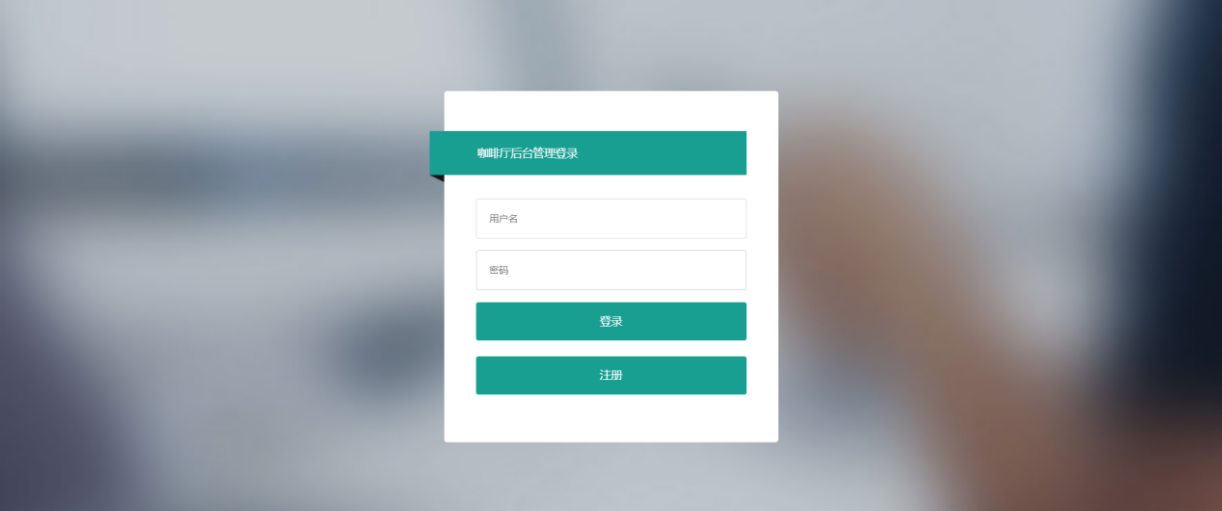


图5-21登录页面

（二）前台界面

（1）找回密码界面，如图5-22所示



图5-22找回密码页面

（2）饮品，如图5-23、5-24所示

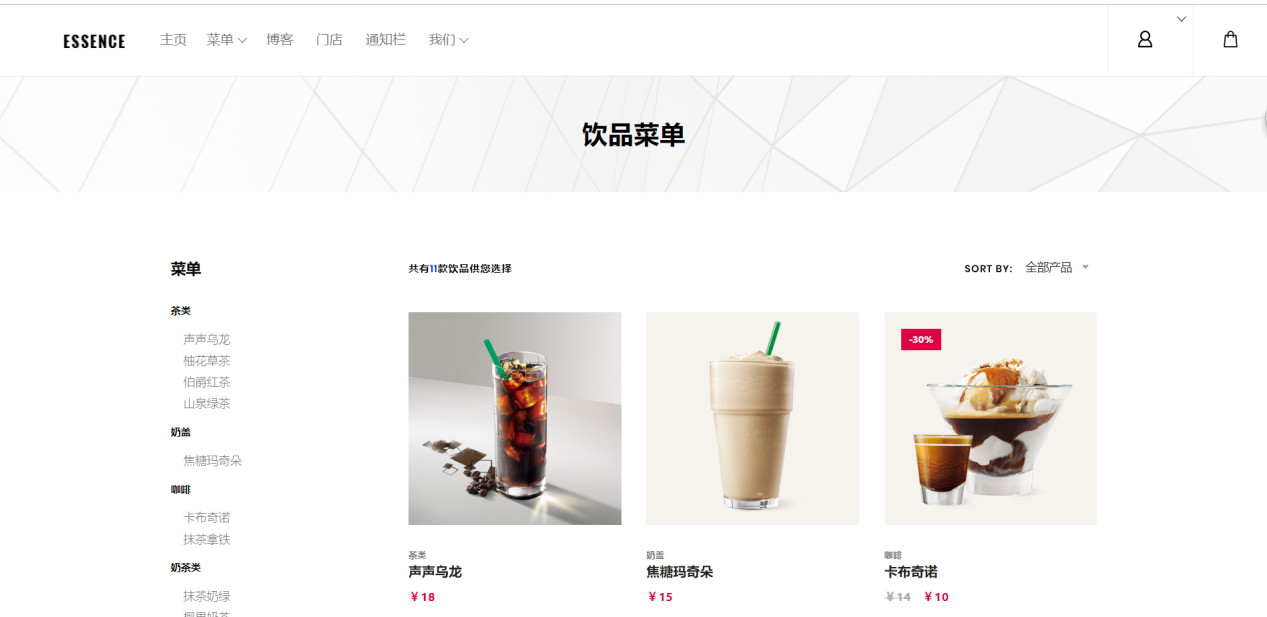


图5-23饮品列表

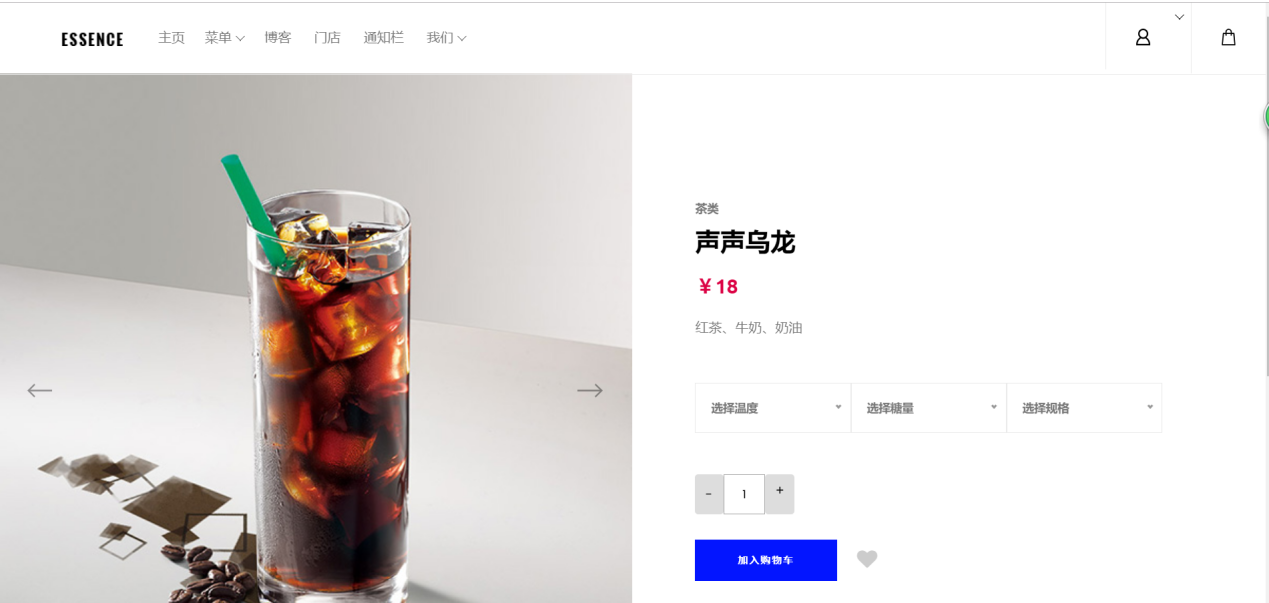


图5-24饮品详情页面

（3)咖啡博客界面，如图5-25所示



图5-25咖啡博客界面

(4)门店列表页面，如图5-26所示

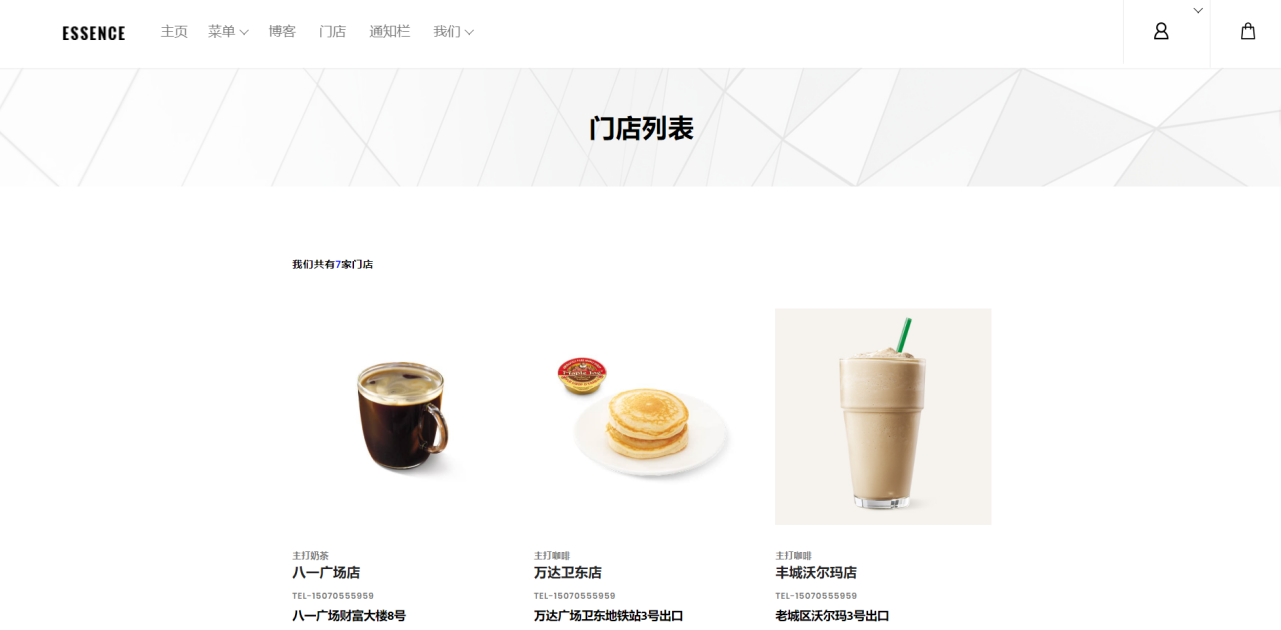


图5-26门店列表界面

(5)通知栏界面，如图5-27所示



图5-27通知栏界面

(6)查看订单界面，如图5-28所示



图5-28查看订单界面

# 6 详细设计

## 6.1 系统主要功能模块介绍

（一）项目结构概述

该项目分为前台与后台管理两个大分类，前台有评论管理模块、购物车管理模块、发票管理模块组成，后台由用户管理子系统、商品管理子系统、订单管理模块、门店管理模块、公告板管理模块构成。用户管理子系统包括管理员管理模块、用户管理模块、权限管理模块，商品管理子系统包括品类管理模块、饮料管理模块、周边商品管理模块、博客管理模块。

（二）项目结构图

项目结构图如图6-1所示：

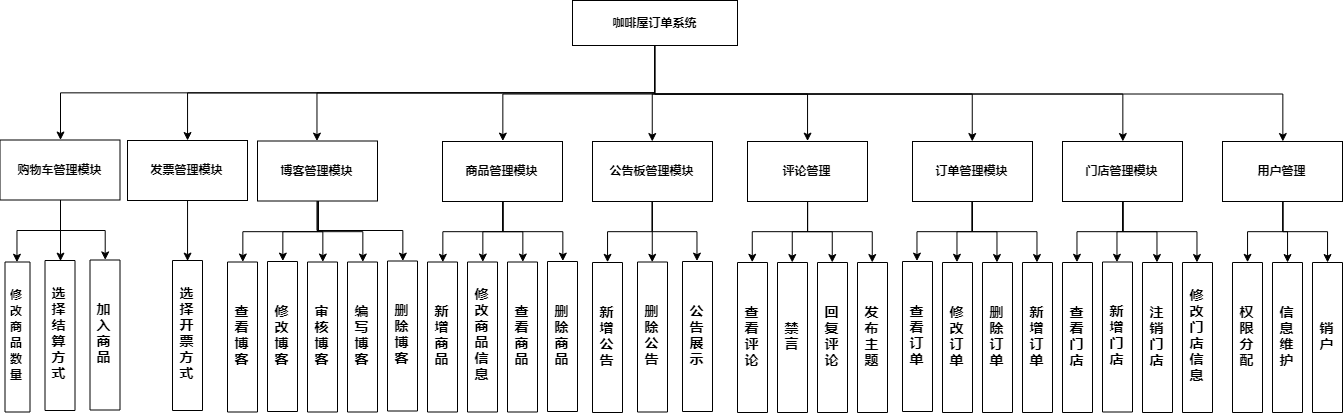


图6-1 系统功能结构图

## 6.2用户管理模块

### 6.2.1功能结构设计

用户管理模块概述：用户管理模块包括三个功能，分别是权限分配、信息维护、销户。管理员可以通过该功能模块对权限进行分配，起到分配角色的作用，维护所有用户的基本信息，以及对有销户意愿的用户进行销户处理，具体如下图6-2所示。

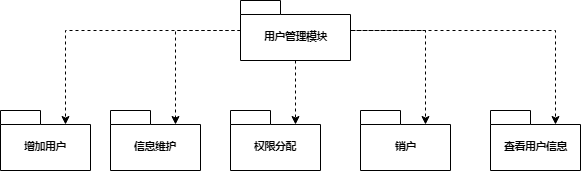


图6-2 用户管理模块功能结构设计图

### 6.2.2功能类设计

用户管理模块主要涉及的关键类有拍卖品提醒控制类、用户管理模块业务处理类、用户管理模块处理实现类、用户管理模块数据库处理类、用户管理模块实体类，其关系如类图6-3所示：

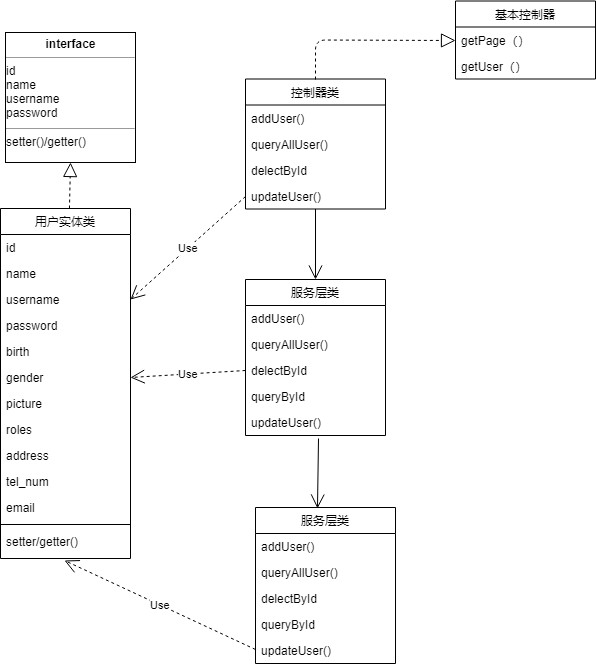


图6-3 用户管理模块类图

用户管理由5个类构成，分别是数据库层操作类、服务层类、控制器类、基本操作类、用户实体类。基本控制器详细如下表6-1所示：

表6-1 基本控制器类描述

|  |  |  |
| --- | --- | --- |
| **基本控制器类** | | |
| 类的描述：管理所有与控制器的公用功能和获取公共元素，以及获取shiro的用户信息 | | |
| 属性描述 | log | 系统日志 |
| user | 已登录的用户信息 |
| 操作描述 | getPage() | 组装分页对象 |
| getUser() | 从shiro中获取当前登录用户的对象 |

用户管理控制器类如表6-2所示：

表6-2 用户管理控制器类描述

|  |  |  |
| --- | --- | --- |
| **控制器类** | | |
| 类的描述：用于用户信息、用户权限的管理 | | |
| 操作描述 | queryAlluser() | 获取所有用户信息 |
| updateUser() | 更新用户信息 |
| deleteById | 删除用户 |
| addUser | 增加用户 |

用户管理服务层类描述如下表6-3所示：

表6-3 用户管理服务层类描述

|  |  |  |
| --- | --- | --- |
| **服务层类** | | |
| 类的描述：用于封装对数据库操作 | | |
| 操作描述 | queryAlluser() | 获取所有用户信息 |
| queryById() | 通过id获取用户信息 |
| updateUser() | 修改用户信息 |
| addUser() | 增加用户 |
| deleteById() | 通过用户id删除用户 |

用户管理数据库操作类描述如下表6-4所示：

表6-4 用户管理数据库操作类描述

|  |
| --- |
| **数据库操作类** |
| 类的描述：用户管理相关操作的数据库操作层方法封装 |

续表6-4

|  |  |  |
| --- | --- | --- |
| 操作描述 | addUser() | 插入用户数据 |
| updateUser() | 更新用户数据 |
| deleteById() | 根据用户id删除用户数据 |
| queryById() | 根据用户id查看用户数据 |
| queryAllUser() | 查询所有用户信息 |

用户实体类描述如下6-5所示：

表6-5 用户实体类描述

|  |  |  |
| --- | --- | --- |
| **实体类** | | |
| 类的描述：用户的实体对象类 | | |
| 属性描述 | id | 用户id |
| Username | 用户名 |
| Password | 密码 |
| Name | 姓名 |
| Gender | 性别 |
| Tel | 电话号码 |
| Roles | 角色 |
| Birth | 生日 |
| Address | 地址 |
| Email | 电子邮箱 |
| Picture | 头像 |
| 操作描述 | setter/getter() | 对问题属性的set和get方法 |

### 6.2.3时序图设计

用户管理模块的时序图如下图6-4所示：

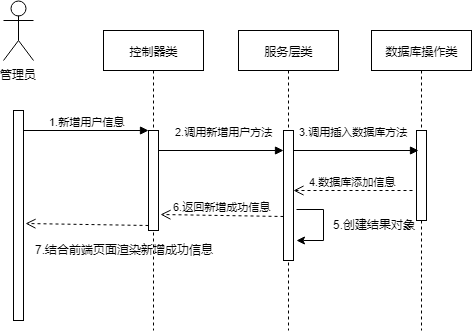


图6-4用户管理模块时序图

在用户管理模块中，管理员点击新增用户按钮，填写信息后，将数据传送至控制器类，随后在服务层类中进行数据封装处理，最后传至数据库操作类与数据库进行对接，返回得到的数据，在前端页面进行呈现。

### 6.2.4核心处理流程设计

用户管理模块的活动图如下图6-5所示：

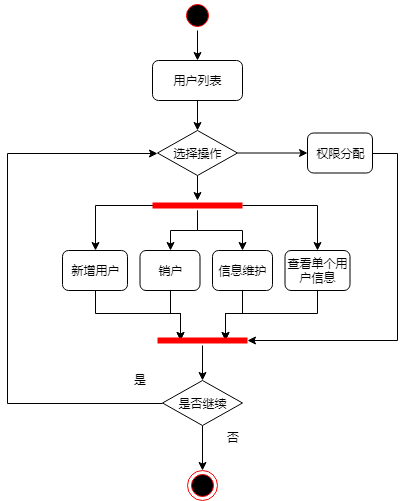


图6-5用户管理模块活动图

在用户管理模块活动图中，管理员可以选择新增用户、销户、维护用户信息、查看单个用户信息、权限分配操作，操作完成后可以选择继续操作或者是退出。

## 6.3 发票管理

### 6.3.1功能结构设计

发票管理模块包括选择开票方式与查看开票记录两个功能，如图6-6所示

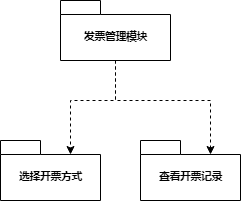


图6-6发票管理模块功能结构设计图

### 6.3.2功能类设计

发票管理模块主要涉及的关键类有拍卖品提醒控制类、发票管理模块业务处理类、发票管理模块处理实现类、发票管理模块数据库处理类、发票管理模块实体类，其关系如类图6-7所示：



图6-7 发票管理模块类图

发票管理由5个类构成，分别是数据库层操作类、服务层类、控制器类、基本操作类、发票实体类。发票管理控制器类详细如下表6-6所示：

表6-6 发票管理控制器类描述

|  |
| --- |
| **控制器类** |

续表6-6

|  |  |  |
| --- | --- | --- |
| 类的描述：用于发票操作 | | |
| 操作描述 | getBillWay() | 显示所有开票方式 |
| updateWay() | 修改开票方式 |
| deleteWay() | 删除开票方式 |
| addWay() | 增加开票方式 |

发票管理服务层类描述如下表6-7所示：

表6-7 发票管理服务层类描述

|  |  |  |
| --- | --- | --- |
| **服务层类** | | |
| 类的描述：用于封装数据 | | |
| 操作描述 | getBillWay() | 显示所有开票方式 |
| updateWay() | 修改开票方式 |
| deleteWay() | 删除开票方式 |
| addWay() | 增加开票方式 |

发票管理数据库层类描述如下表6-8所示：

表6-8 发票管理数据库层类描述

|  |  |  |
| --- | --- | --- |
| **数据库层类** | | |
| 类的描述：对数据库数据进行操作 | | |
| 操作描述 | findWay() | 显示所有开票方式 |
| delWay() | 删除开票方式 |
| addWay() | 增加开票方式 |
| updateWay() | 修改开票方式 |

发票实体类描述如下6-9所示：

表6-9 发票实体类描述

|  |  |  |
| --- | --- | --- |
| **实体类** | | |
| 类的描述：发票的实体对象类 | | |
| 属性描述 | id | 发票id |
| Name | 发票名称 |
| Email | 邮箱 |
|  | BankAccount | 银行账户 |
| Number | 电话号码 |
| Address | 地址 |
|  | TaxNumber | 税务号 |

续表6-9

|  |  |  |
| --- | --- | --- |
| 操作描述 | setter/getter() | 对问题属性的set和get方法 |

### 6.3.3时序图设计

发票管理模块的时序图如下图6-8所示：

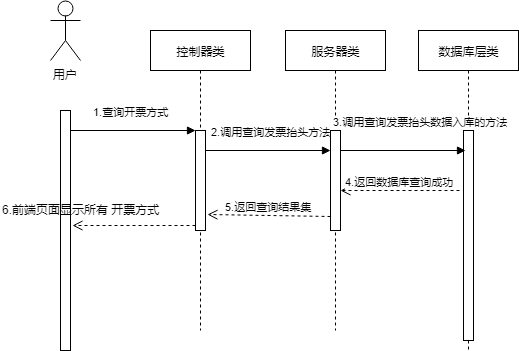


图6-8 发票管理模块时序图

在发票管理模块中，管理员点击查询开票方式按钮，将数据传送至控制器类，随后在服务层类中进行数据封装处理，最后传至数据库操作类与数据库进行对接，返回得到的数据，在前端页面显示所有开票方式。

### 6.3.4核心处理流程设计

发票管理模块的活动图如下图6-9所示：

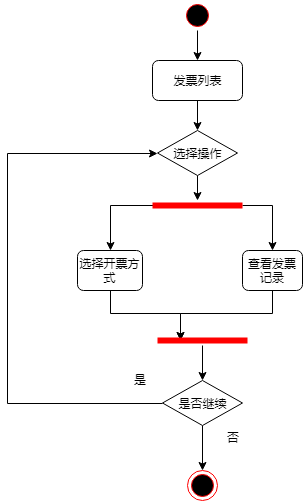


图6-9发票管理模块活动图

在发票管理模块活动图中，用户可以选择开票方式、查看发票记录操作，操作完成后可以选择继续操作或者是退出。

## 6.4 购物车管理模块

### 6.4.1功能结构设计

购物车管理模块的操作对象是用户，包括修改商品数量、加入商品、删除商品、选择结算方式、查看购物车五个部分，如图6-10所示。

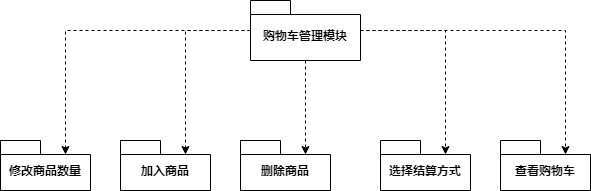


图6-10 购物车管理模块功能结构设计图

### 6.4.2功能类设计

购物车管理模块，包括购物车控制器类，基本控制器，购物车服务层类，购物车数据库层类，购物车实体类，商品实体类，如下图6-11所示：

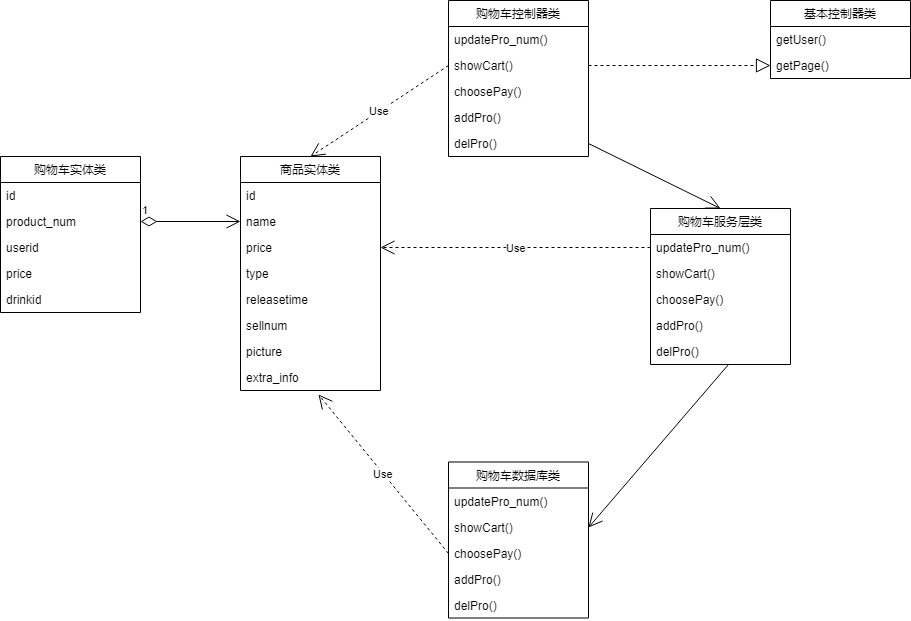


图6-11 购物车管理模块类图

购物车管理模块控制器类如表6-10所示：

表6-10 购物车管理控制器类描述

|  |
| --- |
| **控制器类** |
| 类的描述：用于购物车管理商品数量 |

续表6-10

|  |  |  |
| --- | --- | --- |
| 操作描述 | UpdatePro\_Num() | 修改商品数量 |
| ShowCart() | 显示购物车内容 |
| ChoosePay() | 选择支付方式 |
| addPro() | 增加商品加入购物车 |
| DelPro() | 删除商品从购物车 |

购物车实体类如表6-11所示：

表6-11 购物车实体类描述

|  |  |  |
| --- | --- | --- |
| **实体类** | | |
| 类的描述：购物车的实体对象类 | | |
| 属性描述 | id | 购物车id |
| Price | 商品单价 |
| Product\_num | 商品数量 |
| userId | 用户id |
| Drinkid | 商品id |
| 操作描述 | setter/getter() | 对问题属性的set和get方法 |

商品实体类如表6-12所示：

表6-12商品实体类描述

|  |  |  |
| --- | --- | --- |
| **实体类** | | |
| 类的描述：商品的实体对象类 | | |
| 属性描述 | id | 商品id |
| Name | 商品名称 |
| Price | 商品价格 |
| Type | 商品类型 |
| Releasetime | 发布时间 |
| Sellnum | 销售量 |
| Picture | 图片 |
|  | Extra\_info | 备注 |
| 操作描述 | setter/getter() | 对问题属性的set和get方法 |

### 6.4.3时序图设计

购物车管理模块的时序图如下图6-12所示：

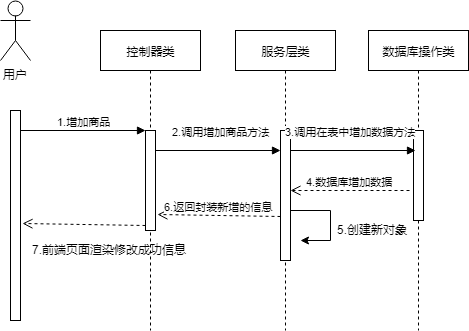


图6-12购物车管理模块时序图

在购物车管理模块活动图中，用户选择某一商品点击加入购物车按钮，随后将数据传至控制器类，在服务层类中封装数据，最后提交给数据库操作类并通过相关方法操作数据库信息，然后返回数据，并呈现在前端页面上。

### 6.4.4核心处理流程设计

购物车管理模块的活动图如下图6-13所示：

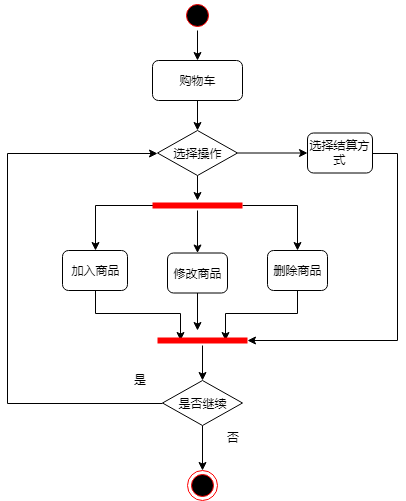


图6-13购物车管理模块活动图

在购物车管理模块活动图中，用户可以选择结算方式、加入商品、修改商品、删除商品操作，操作完成后可以选择继续操作或者是退出。

## 6.5商品管理模块

### 6.5.1功能结构设计

商品管理模块包括增加商品、删除商品、修改商品、查找商品四个部分，如图6-14所示：

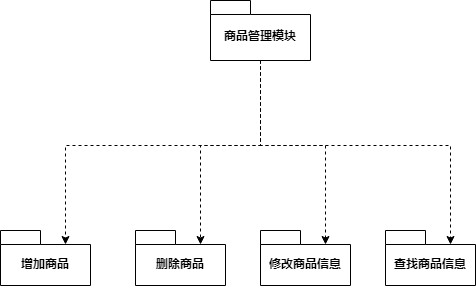


图6-14商品管理模块功能结构设计图

### 6.5.2功能类设计

商品管理模块包括商品实体类、基本控制器、控制器类、服务层类、数据库层类，如图6-15所示。

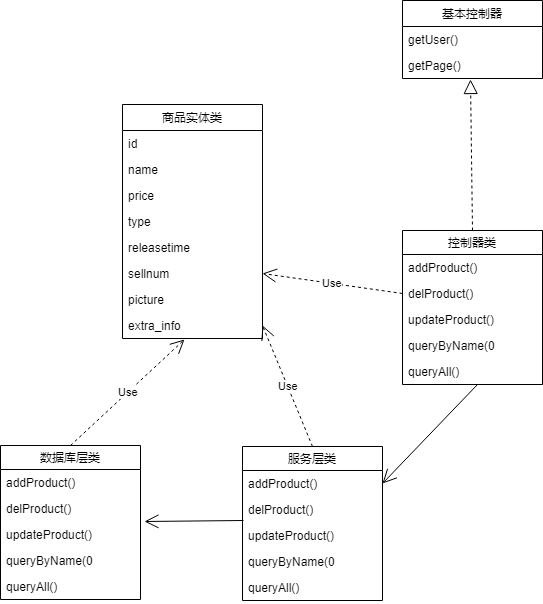


图6-15 商品管理模块类图

购物车管理模块控制器类如表6-13所示：

表6-13 商品管理控制器类描述

|  |  |  |
| --- | --- | --- |
| **控制器类** | | |
| 类的描述：用于商品信息操作管理 | | |
| 操作描述 | AddProduct() | 增加商品 |
| DelProduct() | 删除商品 |
| UpdateProduct() | 更新商品信息 |
| QueryByName() | 根据商品名称查找商品 |
| QueryAll() | 查找所有商品 |

### 6.5.3时序图设计

商品管理模块的时序图如下图6-16所示：



图6-16 周边商品管理模块时序图

在商品管理模块中，管理员点击删除按钮，把所需要删除的商品的id传递至控制器层，然后传至服务层，最后传至数据库操作层并在数据库进行数据擦欧总，最终把结果传递至前端页面。

### 6.5.4核心处理流程设计

商品管理模块的活动图如下图6-17所示：

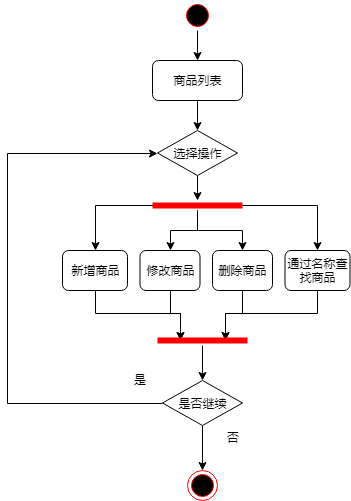


图6-17商品管理模块活动图

在商品管理模块活动图中，管理员可以选择新增商品、删除商品、修改商品、通过名称查找商品操作，操作完成后可以选择继续操作或者是退出。

## 6.6博客管理模块

### 6.6.1功能结构设计

博客管理模块拥有删除博客、审核博客、查看博客、编写博客四个部分，如下图6-18所示：



图6-18博客管理模块功能结构设计图

### 6.6.2功能类设计

博客管理模块拥有博客实体类、控制器类、基本控制器、服务层类、数据库层类，如下图6-19所示：

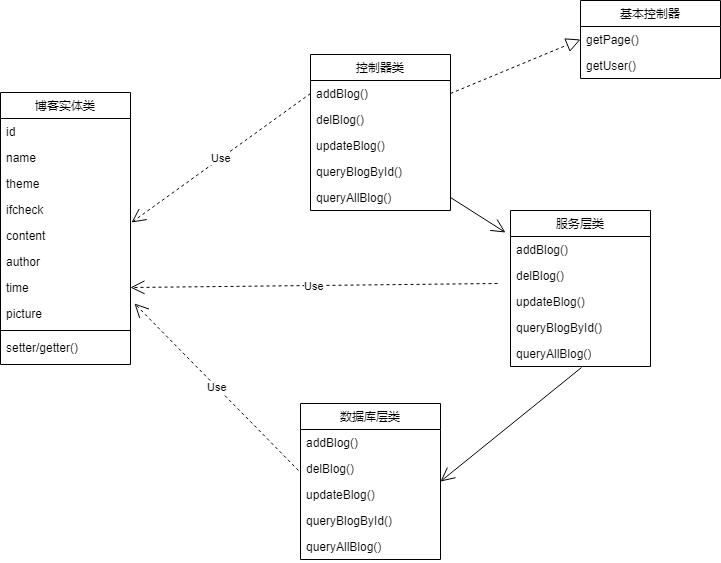


图6-19 博客管理模块类图

博客管理模块控制器类如表6-14所示：

表6-14 博客管理控制器类描述

|  |
| --- |
| **控制器类** |
| 类的描述：用于用户信息、用户权限的管理 |

续表6-14

|  |  |  |
| --- | --- | --- |
| 操作描述 | AddBlog() | 编写博客 |
| DelBlog() | 删除博客 |
| UpdateBlog() | 更新博客 |
| QueryBlogById() | 根据id查找博客 |
| QueryAllBlog() | 展示所有博客 |

博客实体类如表6-15所示：

表6-15 博客实体类描述

|  |  |  |
| --- | --- | --- |
| **实体类** | | |
| 类的描述：博客的实体对象类 | | |
| 属性描述 | id | 博客id |
| Name | 博客名称 |
| Theme | 主题 |
| Ifcheck | 是否审核 |
| Content | 内容 |
| Author | 作者 |
| Time | 发布时间 |
| Picture | 封面 |
| 操作描述 | setter/getter() | 对问题属性的set和get方法 |

### 6.6.3时序图设计

博客管理模块的时序图如下图6-20所示：

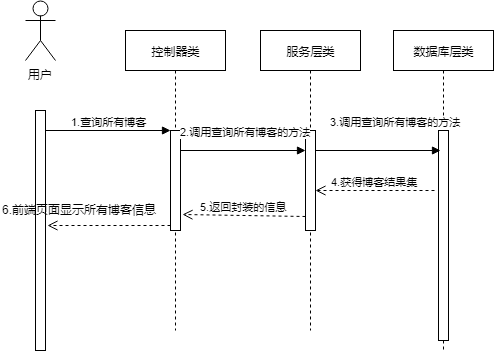


图6-20博客管理模块时序图

在博客管理模块活动图中，用户可以查询所有博客信息在前台博客模块，进入博客模块，便会向控制器发送查询所有博客信息请求，依次传递服务层类，最后在数据库层得到所有博客信息结果集，将信息返回至前端。

### 6.6.4核心处理流程设计

博客管理模块的活动图如下图6-21所示：

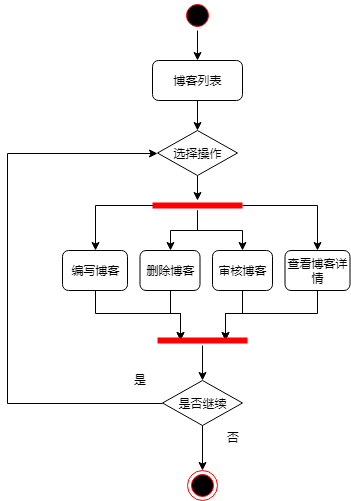


图6-21博客模块活动图

在博客管理模块活动图中，用户可以选择编写博客、查看博客详情信息，管理员可以选择审核博客、删除博客、编写博客、查看博客详情操作，在操作完成后可以选择继续操作或者是退出。

## 6.7订单管理模块

### 6.7.1功能结构设计

订单管理模块包括查询订单、查看单个订单信息、删除订单、修改订单、列出订单列表五个部分，如图6-22所示：

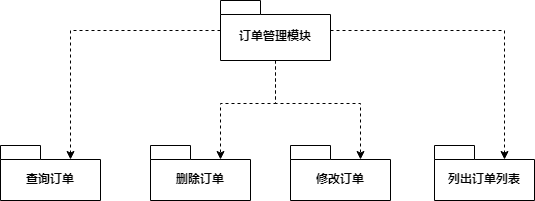


图6-22订单管理模块功能结构设计图

### 6.7.2功能类设计

订单管理模块由订单实体类、基本控制器、订单控制器类、订单服务层类、订单数据库层类组成，如图6-23所示：



图6-23 订单管理模块类图

订单管理模块控制器类如表6-16所示：

表6-16 订单管理控制器类描述

|  |  |  |
| --- | --- | --- |
| **控制器类** | | |
| 类的描述：用于订单操作管理 | | |
| 操作描述 | Amend\_Order() | 修改订单 |
| DelOrder() | 删除订单 |
| FindOrderById() | 通过订单id查找订单 |

续表6-16

|  |  |  |
| --- | --- | --- |
|  | QueryAllOrder) | 展示所有订单 |

订单实体类如表6-17所示：

表6-17 订单实体类描述

|  |  |  |
| --- | --- | --- |
| **实体类** | | |
| 类的描述：订单的实体对象类 | | |
| 属性描述 | id | 订单id |
| Order\_id | 订单编号 |
| OrderInfo | 订单备注 |
| Totalprice | 总价 |
| Discount | 折扣 |
| Payway | 支付方式 |
| OrderStatus | 订单状态 |
| Catchway | 取货方式 |
| Ordertime | 下单时间 |
| Uid | 用户id |
| Deliver | 配送地址 |
| 操作描述 | setter/getter() | 对问题属性的set和get方法 |

### 6.7.3时序图设计

订单管理模块的时序图如下图6-24所示：

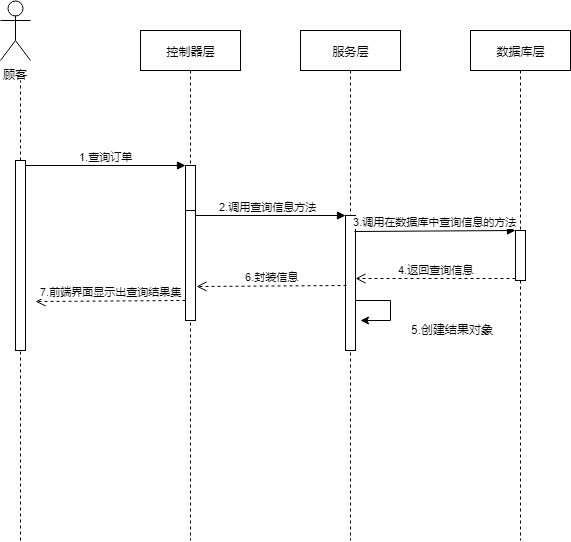


图6-24订单管理模块时序图

在订单管理模块中，管理员可以通过订单编号查询订单信息，将订单编号提交至控制器层、服务层、数据库层并在数据库中取得数据并返回前端显示。

### 6.7.4核心处理流程设计

订单管理模块的活动图如下图6-25所示：

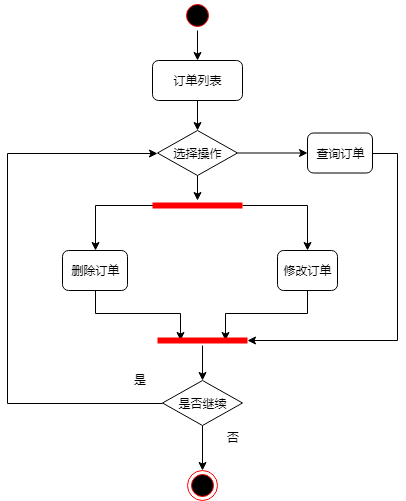


图6-25订单管理模块活动图

在订单管理模块活动图中，管理员可以选择删除订单、修改订单信息操作、查看所有订单信息，用户可以查看自己的订单信息，在完成后可以选择继续操作或者是退出。

## 6.8公告板管理模块

### 6.8.1功能结构设计

在公告板管理模块中包括新增公告、修改公告、删除公告、查看公告四部分，如图6-26所示：

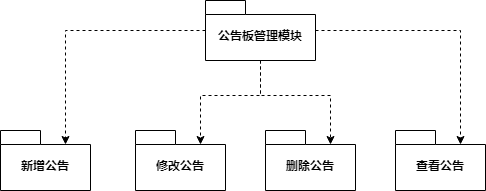


图6-26 公告板管理模块功能结构设计图

### 6.8.2功能类设计

在公告板管理模块中包括公告板实体类、公告板控制器类、公告板服务层类、基本控制器、公告板数据库层类，如图6-27所示：

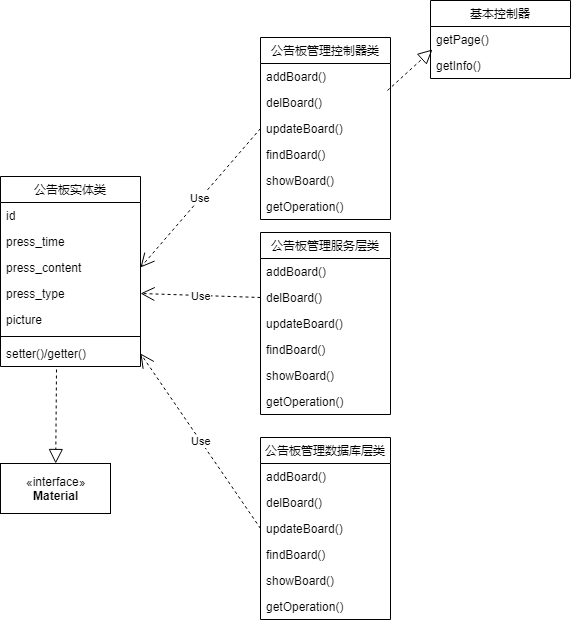


图6-27 公告板管理模块类图

公告板管理模块控制器类如表6-18所示：

表6-18 公告板管理控制器类描述

|  |  |  |
| --- | --- | --- |
| **控制器类** | | |
| 类的描述：用于公告板操作的管理 | | |
| 操作描述 | AddBoard() | 增加公告栏 |
| DelBoard() | 删除公告栏 |
| FindBoard() | 通过id查找公告栏 |
| ShowBoard() | 展示所有公告栏信息 |
| GetOperation() | 获取操作名称 |
| UpdateBoard() | 更新公告栏 |

公告栏实体类如表6-19所示：

表6-19 公告板实体类描述

|  |  |  |
| --- | --- | --- |
| **实体类** | | |
| 类的描述：公告板的实体对象类 | | |
| 属性描述 | id | 公告栏id |
| Press\_time | 发布时间 |
| Press\_content | 发布内容 |
| Press\_type | 发布类型 |
| picture | 图片 |
| 操作描述 | setter/getter() | 对问题属性的set和get方法 |

### 6.8.3时序图设计

公告板管理模块的时序图如下图6-28所示：

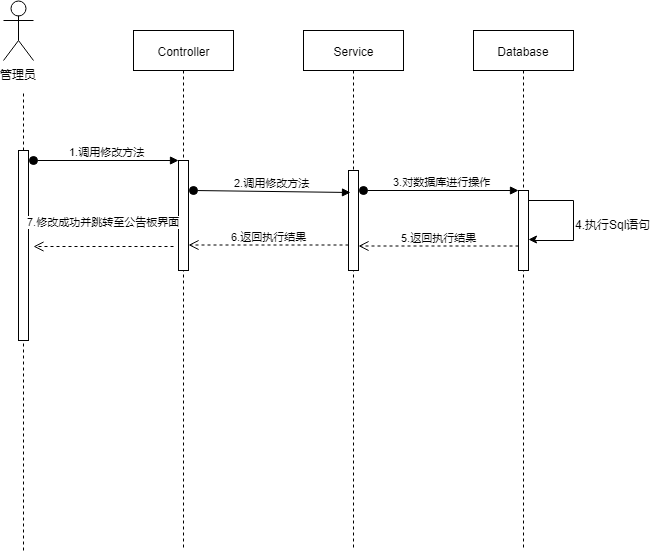


图6-28公告板管理模块时序图

在公告板管理模块中，管理员对应需要修改的公告点击修改按钮，跳转至修改界面，将提交的信息传至控制器类，随后传至服务层类进行封装，最后传至数据库操作类，对数据库信息进行修改并返回数据至前端页面。

### 6.8.4核心处理流程设计

公告板管理模块的活动图如下图6-29所示：

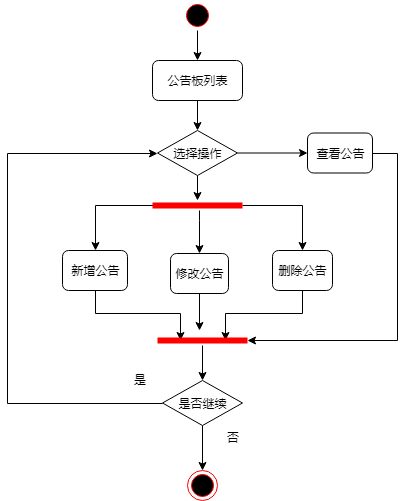


图6-29公告板管理模块活动图

在公告板管理模块活动图中，管理员可以选择新增公告、删除公告、修改公告、查看公告操作，操作完成后可以选择继续操作或者是退出。

## 6.9评论管理模块

### 6.9.1功能结构设计

评论管理模块由禁言、评论主题管理、查看评论，删除评论四大部分，其中禁言包括选择禁言对象、禁言时间，评论主题管理包括发布评论主题、修改评论主题、删除评论主题、查看评论主题，如图6-30所示：

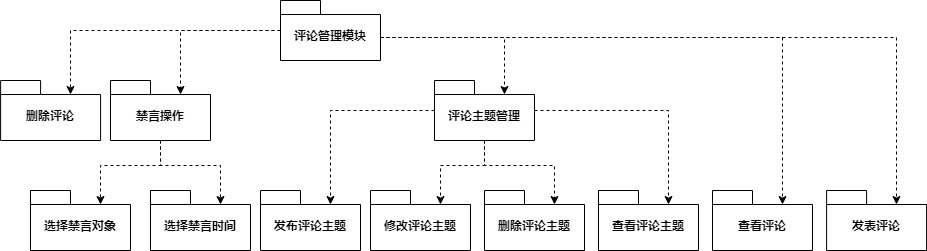


图6-30评论管理模块功能结构设计图

### 6.9.2功能类设计

评论管理模块由评论实体类、主题实体类、评论控制器类、评论服务层类、评论数据库操作类、基本控制器，如图6-31所示：

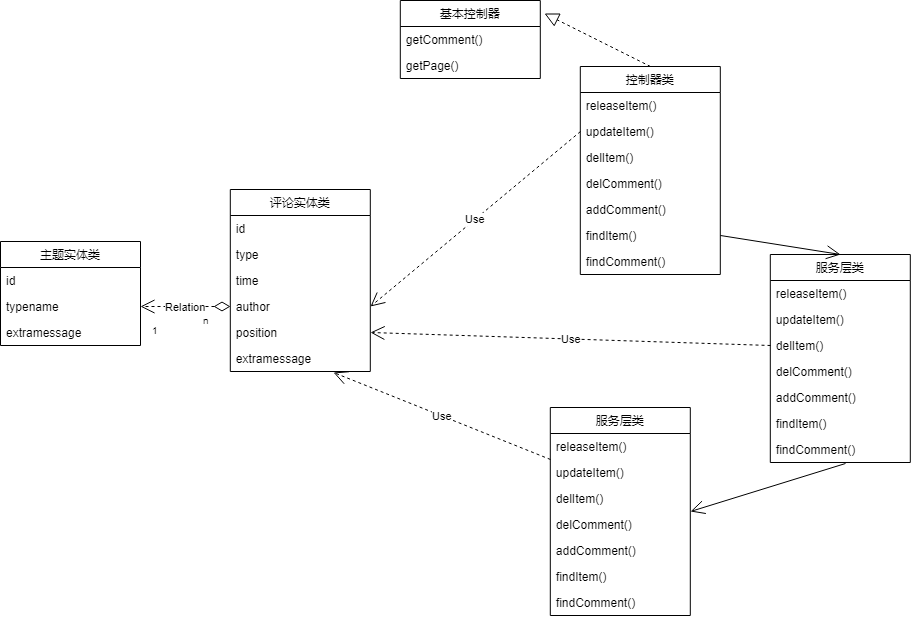


图6-31评论管理模块类图

公告板管理模块控制器类如表6-20所示：

表6-20 评论管理控制器类描述

|  |
| --- |
| **控制器类** |

续表6-20

|  |  |  |
| --- | --- | --- |
| 类的描述：用于评论操作的管理 | | |
| 操作描述 | Addcomment() | 发布评论 |
| DelComment() | 删除评论 |
| FindComment() | 查看评论 |
| ReleaseItem() | 发布评论主题 |
| FindItem() | 查看主题 |
| UpdateItem() | 更新主题 |

评论主题实体类如表6-21所示：

表6-21评论主题实体类描述

|  |  |  |
| --- | --- | --- |
| **实体类** | | |
| 类的描述：评论的实体对象类 | | |
| 属性描述 | id | 主题id |
| Typename | 主题名称 |
| Extramessage | 备注信息 |
| 操作描述 | setter/getter() | 对问题属性的set和get方法 |

评论实体类如表6-22所示：

表6-22 评论实体类描述

|  |  |  |
| --- | --- | --- |
| **实体类** | | |
| 类的描述：评论的实体对象类 | | |
| 属性描述 | id | 评论id |
| Type | 类型 |
| Extramessage | 备注信息 |
| Time | 时间 |
| Author | 作者 |
| Position | 地点 |
| 操作描述 | setter/getter() | 对问题属性的set和get方法 |

### 6.9.3时序图设计

评论管理模块的时序图如下图6-32所示：

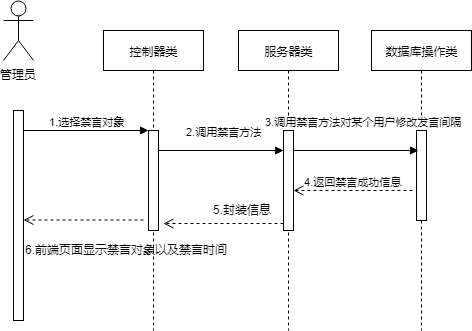


图6-32评论管理模块时序图

在评论管理模块中，管理员可以选择禁言操作，并选择禁言对象、禁言时间，操作完成后可以选择继续操作或者是退出。

### 6.9.4核心处理流程设计

用户管理模块的活动图如下图6-33所示：

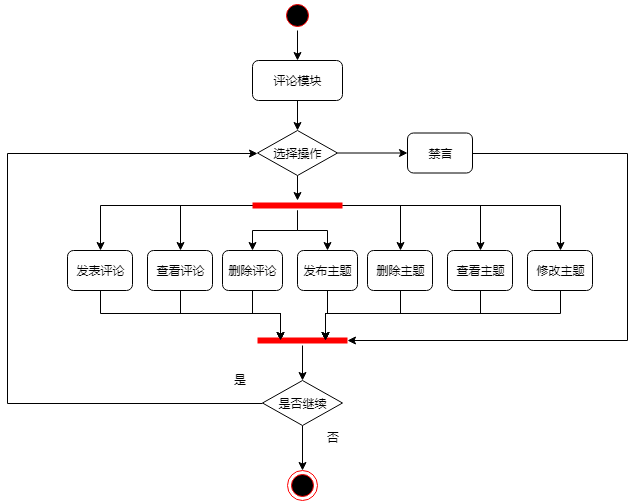


图6-33评论管理模块活动图

在评论管理模块活动图中，管理员可以选择禁言操作，对主题进行管理，用户、管理员可以在这个模块查看评论、删除评论、发表评论，操作完成后可以选择继续操作或者是退出。

## 6.10 门店管理模块

### 6.10.1功能结构设计

门店管理模块包括增加门店、删除门店、修改门店、查看门店、列出优质门店五个部分，如图6-34所示：

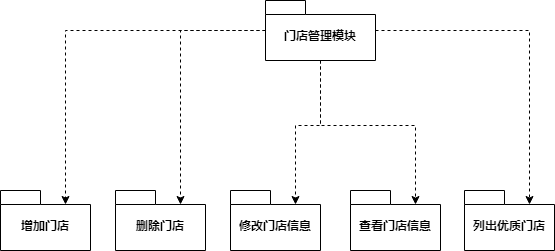


图6-34门店管理模块功能结构设计图

### 6.10.2功能类设计

门店管理模块由门店实体类、门店控制器类、门店服务层类、门店数据库操作类、基本控制器，如图6-35所示：

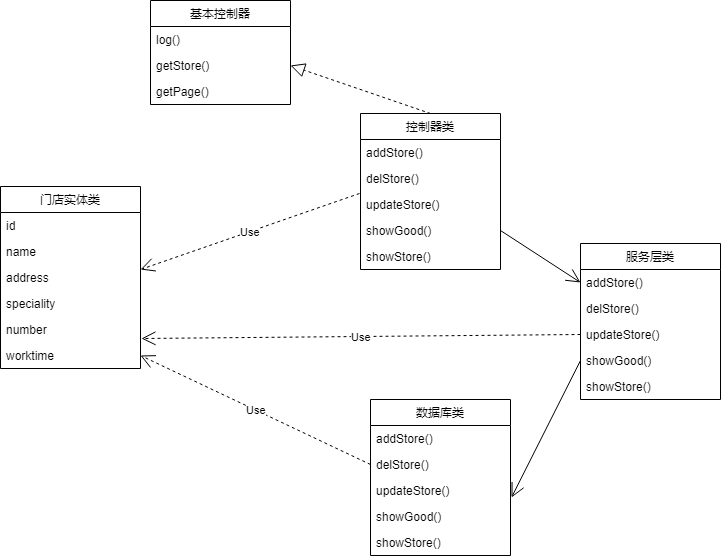


图6-35门店管理模块功能类图

%3CmxGraphModel%3E%3Croot%3E%3CmxCell%20id%3D%220%22%2F%3E%3CmxCell%20id%3D%221%22%20parent%3D%220%22%2F%3E%3CmxCell%20id%3D%222%22%20value%3D%22%E6%9F%A5%E7%9C%8B%E4%B8%BB%E9%A2%98%22%20style%3D%22rounded%3D1%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22580%22%20y%3D%22260%22%20width%3D%2265%22%20height%3D%2240%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3C%2Froot%3E%3C%2FmxGraphModel%3E%3CmxGraphModel%3E%3Croot%3E%3CmxCell%20id%3D%220%22%2F%3E%3CmxCell%20id%3D%221%22%20parent%3D%220%22%2F%3E%3CmxCell%20id%3D%222%22%20value%3D%22%E6%9F%A5%E7%9C%8B%E4%B8%BB%E9%A2%98%22%20style%3D%22rounded%3D1%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22580%22%20y%3D%22260%22%20width%3D%2265%22%20height%3D%2240%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3C%2Froot%3E%3C%2FmxGraphModel%3

门店管理模块控制器类如表6-23所示：

表6-23 门店管理控制器类描述

|  |
| --- |
| **控制器类** |

续表6-23

|  |  |  |
| --- | --- | --- |
| 类的描述：用于门店操作的管理 | | |
| 操作描述 | Addstore() | 增加门店 |
| Delstore() | 删除门店 |
| UpdateStore() | 更新门店 |
| ShowGood() | 列出优质门店 |
| ShowStore() | 显示所有门店 |

门店实体类如表6-24所示：

表6-24 门店实体类描述

|  |  |  |
| --- | --- | --- |
| **实体类** | | |
| 类的描述：门店的实体对象类 | | |
| 属性描述 | id | 门店id |
| Worktime | 工作时间 |
| Name | 店名 |
| Specility | 特色 |
|  | Address | 地址 |
| Number | 电话 |
| 操作描述 | setter/getter() | 对问题属性的set和get方法 |

### 6.10.3时序图设计

门店管理模块的时序图如下图6-36所示：

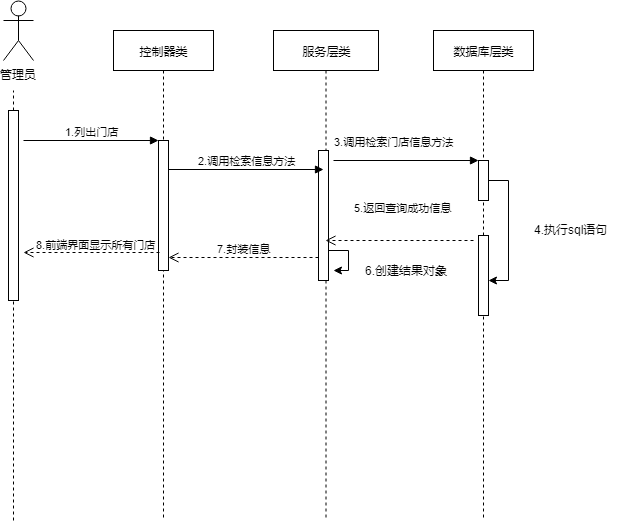


图6-36门店管理模块时序图

在门店管理模块中，管理员可以在后台选择优质门店，并列出优质门店呈现于前台系统。

### 6.10.4核心处理流程设计

门店管理模块的活动图如下图6-37所示：

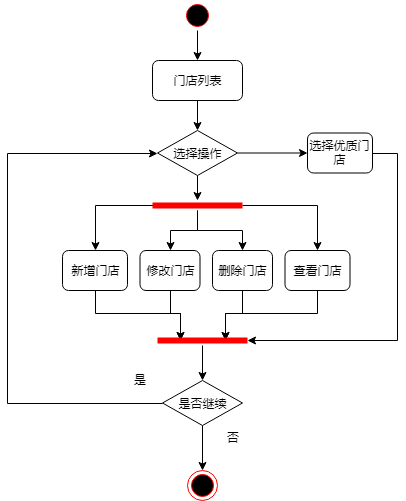


图6-37门店管理模块活动图

在门店管理模块活动图中，管理员可以选择新增门店、修改门店、删除门店、查看门店还可以选择优质门店，操作完成后可以选择继续操作或者是退出。

# 7 编码

## 7.1 代码实现与核心算法

### 7.1.1图片上传

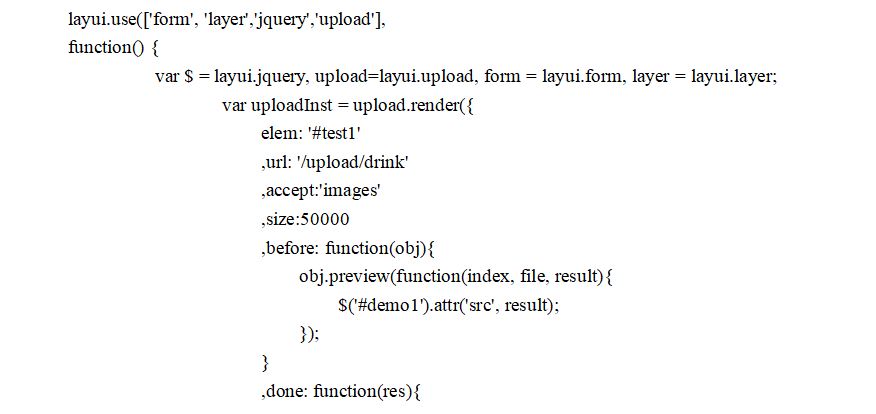
这段代码是用于商品图片以及上传博客封面等图片上传。

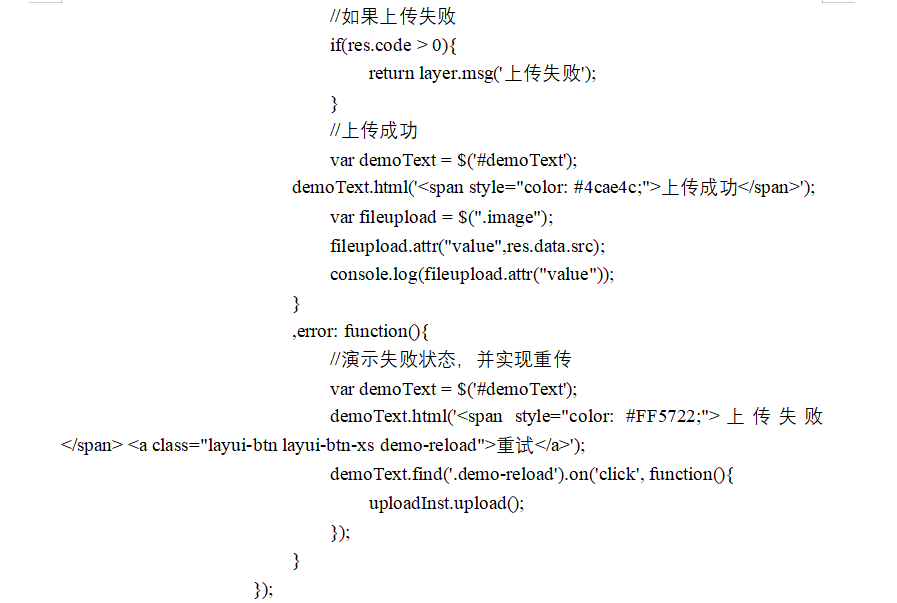
Controller类：





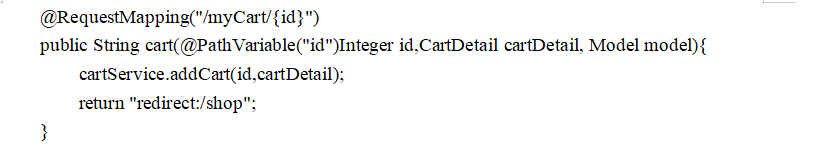
JavaScript代码：



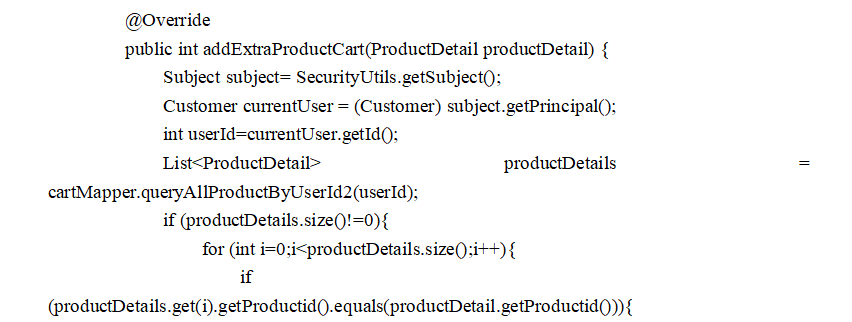


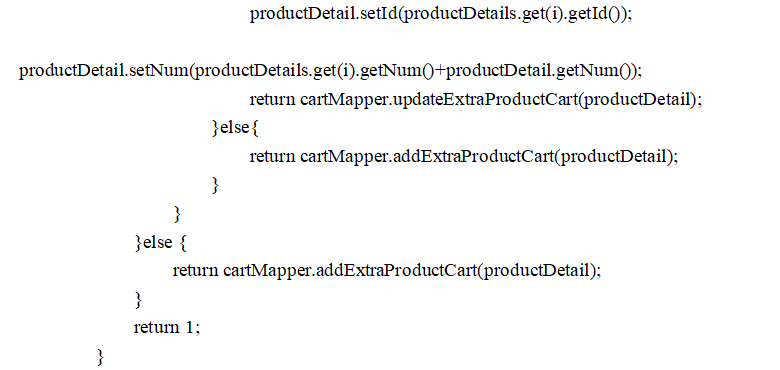
### 7.1.2购物车实现

Controller类：

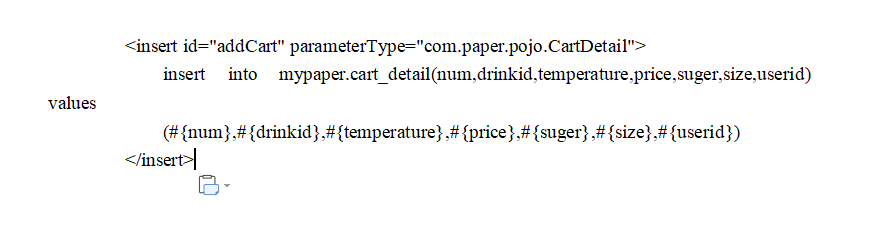


ServiceImpl类：



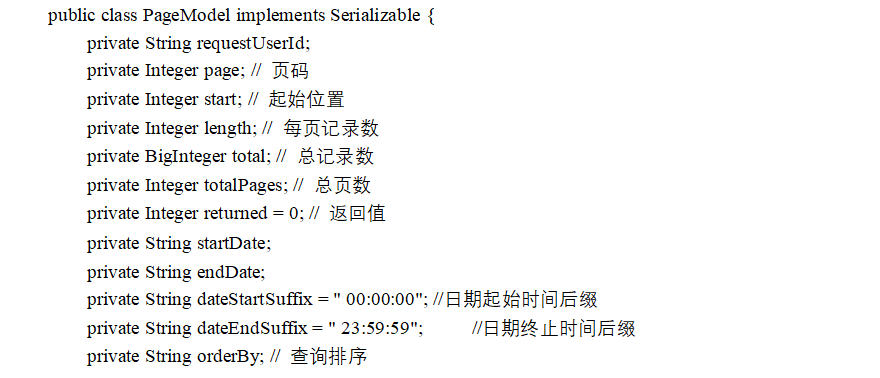


Mapper.Xml：

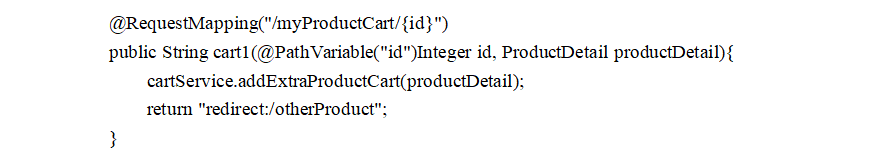


## 7.2 代码优化分析

实现分页功能时，创建了一个PageModel类为所有实体提供实现分页所需要的属性，提高了代码的复用。



为了防止controller类过于繁琐，数据处理全部交予Service层处理并封装数据返回给Controller。





# 8 测试

## 8.1 测试方案设计

本节主要是以介绍测试方案的设计过程为中心出发的，包括设计测试策略、安排测试进度、配置测试静态资源以及对几个关键测试点注意事项。

### 8.1.1 测试策略

本系统的测试按照四种测试原则进行，分别是单元测试、黑盒测试、白盒测试和系统集成测试。其中单元测试是对Java类功能进行检查和验证；黑盒测试是把程序看成一个不能打开的盒子而对系统每一个功能点的测试，返回测试结果；白盒测试是用来测试系统的业务逻辑是否正确的测试手段，主要用于测试本系统的验证问题。

### 8.1.2 测试进度安排

由于本系统业务逻辑比较复杂，所以应当分为几个部分进行，本系统的测试进度安排如下表8-1所示：

表8-1 测试进度安排表

|  |  |
| --- | --- |
| **测试阶段** | **测试任务** |
| 单元测试 | 在每个模块上执行功能测试，并在此阶段的测试任务与项目开发同时进行。 |
| 黑盒测试  （前台部分） | 在系统前台的用户界面上执行统一的黑盒测试，收集测试结果和反馈 |
| 黑盒测试  （后台部分） | 对系统后台管理员操作界面进行统一的黑盒测试，收集测试结果和反馈 |
| 白盒测试 | 对系统重要功能点进行白盒测试，收集系统运行信息，确保代码运行不出现逻辑问题。 |
| 系统集成测试 | 对系统总体功能进行最后的测试，兼顾功能、性能和用户体验。 |

### 8.1.3 测试资源

为了使本次测试数据的真实性，在测试资源的安排上，使用了最平常的测试环境，测试资料如下表8-2所示：

表8-2 测试资源表

|  |  |  |
| --- | --- | --- |
| **硬件环境** | | |
| **资源类型** | **资源描述** | **数量** |

续表8-2

|  |  |  |
| --- | --- | --- |
| 计算机硬件 | Intel I5 8G内存 ITG硬盘 | 1 |
| 操作系统 | Windows 10 1920X1080分辨率 | 1 |
| JDK | JDK 1.8 | 1 |
| 浏览器 | Chrome 浏览器 | 1 |
| **软件环境** | | |
| **资源类型** | **软件** | |
| 应用数据库 | Mysql 5.7 | |
| 应用服务器 | Tomcat 8.5.16 | |
| 缓存数据库 | Redis 3.1 | |

### 8.1.4 关键测试点

1. 用户是否能添加购物车并选择规格
2. 用户是否能浏览博客内容，并发表博客信息
3. 用户是否能在忘记密码的时候，根据自己信息修改密码
4. 管理员在上传商品图片模块，是否能正常上传
5. 管理员是否能正常进行权限管理
6. 用户是否能正常付款下单，并填写相应地址
7. 用户能否及时阅读通知栏信息
8. 用户能否修改自己信息

## 8.2 测试用例构建

### 8.2.1 测试用例编写约定

（1）对于每个功能节点，应该从不同的方向、角色切入并设计不同的测试用例

（2）对于数据库sql语句测试，应该先行将语句放入Navicat中测试，数据测试正确，再结合程序一并测试，这样可以避免程序进行多余的测试

（3）为了满足新增的系统功能需求，测试用例应当需要小范围的变动

### 8.2.2 测试用例设计

为了保证测试的正确性从而达到预想的目的，原则上应该选用多种测试方式设计测试用例，主要使用到的测试用例方法有：

（1）等价类划分法

采用此种测试方法，可以将本系统大多数类似的功能归并为一类，例如用户管理、商品管理、门店管理，少量的测试结果可以代表大多数的测试结果，减少测试数量。

（2）判定表法

对于本系统一些逻辑较为复杂的功能点，需要进行严密的测试用例设计，例如博客管理，用户和管理员拥有不同的权限对于博客的撰写。

（3）边界值法

对于本系统的输入框值得输入，经常会出现边界值情况，例如电话号码位数、生日、电子邮箱、密码长度，如果忽视这些边界值得测试，会造成严重的后果，例如内存溢出。

（4）正交实验法

对于本系统一些对数据格式以及内容要求严格的功能点，应当设计较为完善的测试数据进行测试，避免在运行的时候出现功能点设计的遗漏。

### 8.2.3 关键测试用例

针对本系统的所有测试用例，本文从中选取了部分关键的测试用例，如下表8-3所示：

表8-3 购物车管理模块测试用例

|  |  |  |  |
| --- | --- | --- | --- |
| **测试目的** | | 购物车管理模块的功能 | |
| **预置条件** | | 使用有效用户的账号和密码登录咖啡厅订单系统 | |
| **序号** | **测试步骤** | | **预期结果** |
| 1 | 点击“商品详情”按钮 | | 进入到商品详情的页面 |
| 2 | 在商品详情页面，选择饮品的规格以及参数，点击“加入购物车”按钮 | | 购物车模块新增该选择商品 |
| 3 | 再点击相同商品的详情页面按钮，选择相同的规格 | | 购物车该商品增加数量，未增加商品 |
| 4 | 点击相同商品的详情页面按钮，选择不一样的规格 | | 购物车模块新增该选择商品 |
| 5 | 在购物车内，点击“删除”按钮 | | 减少一个商品数量，如果商品数量为0，便删除该商品 |
| 6 | 点击“配送”按钮 | | 进入配送商品方式的详情页 |
| 7 | 点击“自提”按钮 | | 进入自提商品方式的详情页 |
| **测试状态** | | 上述操作均达到“测试”要求 | |
| **测试结果** | | 经过测试，该模块功能正常，能够对用户的正常操作进行处理，也可以对错误的操作进行阻止，并提示用户错误信息，达到了预期目标。 | |

### 8.2.4 测试用例维护

在本系统开发过程中，可能会有比较复杂的功能点，就会出现一些测试数据不够完善的情况，导致测试结果无效，此时应该重新设计测试数据完善测试结果。

# 9 总结与展望

## 9.1 设计工作总结

随着互联网时代的发展、技术不断创新，学会使用新兴技术解决当下的时代需求是我们立足于IT行业的奠基石。只有不断掌握新技术，才能保证自己高水平开发，本文就是采用了当下比较热门且新兴的开发技术以及思想。后台管理系统前端界面采用了Layui设计，前台系统前端界面采用了Bootstrap设计，后台数据管理运用了Java轻量级框架SpringBoot，中间的数据交互与实现采用了Thymeleaf模板引擎与Layui.js。系统通过了较为完善的系统测试达到了预期的目标。

本系统主要的工作体现在以下几个方面：  
（1）咖啡厅订单系统的功能需求与非功能性需求，其中包含了角色分析与业务需求分析。

（2）咖啡厅订单系统的详细设计、具体代码实现、体系结构设计、数据库设计完成了咖啡厅订单系统的主要模块设计实现。

（3）采用完善的测试策略与测试工具对本文的咖啡厅订单系统进行详细的功能测试。

## 9.2 未来工作展望

本文系统虽然达到了预期需求，但相较于商用系统还存在一定的差距，无论是功能模块实现还是用户体验感都有某种程度上的差异。因此，完善系统功能以及优化系统功能是必然的。系统在开发的过程中也为未来扩展与优化提供了接口，例如本系统可以在现有的基础上增加文化介绍模块，让更多人都认识与了解咖啡文化。也可以将本系统精简化设计成微信小程序，为周边社区进行线下门店服务，用户通过扫码点单，这种下单方式更加简便。在未来，不断为咖啡厅订单系统添砖加瓦就是系统更好的发展提供更有力的保证。

# 谢辞

# 参考文献

[1]Bootstrap3/4参考网站 [EB/OL].<https://www.bootcss.com/>.

[2]《Spring + Mybatis企业应用实战》[M].北京：清华大学出版社,2018.3:200-231.

[3]《Java Web项目开发实战入门》[M].北京人民邮电出版社，2018.3:123-168.

[4]《JAVA从入门到精通》[M].北京：清华大学出版社，2017.5:56-99.

[5]《Bootstrap实战从入门到精通》[M].北京：中国水利水电出版社，2017.10:67-134.

[6]《一步一步学Spring Boot2：微服务项目实战》[M].北京：清华大学出版社，2018.6:156-198.

[7]《轻量级JavaWeb企业应用实战》[M].北京：电子工业出版社，2017：88-105.

[8]《Spring Boot实战派》[M].北京：电子工业出版社，2018：44-69.

[9]《高性能Mysql》[M].北京：电子工业出版社，2019：152-169.

[10]《SpringBoot+Layui商城实战》[M].北京：清华大学出版社，2019：100-250.

[11]李宁《Java Web编程实战宝典》[M].北京：清华大学出版社，2016：77-123.

[12]LayUI官方文档[EB/OL].<https://www.layui.com/>

[13]《Spring实战》[M].北京：清华大学出版社，2018：123-154.

[14]《深入理解Java虚拟机》[M].北京：清华大学出版社，2017：110-163.

[15]《Spring Boot+Thymeleaf企业应用实战》[M].北京：清华大学出版社，2018：120-167.

[16]《SpringBoot响应式微服务》[M].成都：电子科技大学出版社，2019：32-69.

[17]《Spring原理探究》[M].成都：电子科技大学出版社，2014：11-63.

[18]《Mybatis应用实战》[M].成都：电子科技大学出版社，2015：55-125.

[19]Thymeleaf参考文档[EB/OL].https://www.Thymeleaf.org/

[20]SpringMVC原理分析[J].SpringMVC深入研究，2014，(5)：61-71.

[21]SpringBoot自动装配原理[J].SpringBoot原理分析，2017，(3)：12-22.

[22]SpringBoot注解解析过程[J].SpringBoot原理分析，2017，(8)：45-57.

[23]SpringCloud实战应用[J].SpringBoot应用实战，2016，(4)：11-68.

[24]Shiro一个管理安全的框架[J].Shiro安全管理，2017，(5)：13-31.

[25]SpringSecurity轻量级安全框架[J].SpringBoot安全篇，2018，(1)：1-32.

[26]《SpringBoot+Vue全栈开发实战》[M].北京：清华大学出版社，2016：33-121.

[27]《Java编程思想》[M].成都：电子科技大学出版社，2017：23-65.

[28]Michael Fogus，JavaScript functional programming[J]. 2017, Vol.124: 36-143.

[29]Dragos-Paul Pop,Adam Altar. Designing an MVC Model for Rapid Web Application Development[J]. Procedia Engineering,2014,69.

[30]Yunmei Shi,Xuhong Liu,Ning Li,Xia Hou. An MVC-based Intelligent Document Model Using UIML[J]. Information,2015,6(2).

[31]Xuhong Liu. Spring cloud microservice architecture practitioners[J].Microservice analysis,2016,7(1).

# 附录A 外文翻译—原文部分

原文出处：Martin Folwer-MicroServices

**Microservices-a definition of this new architectural term**

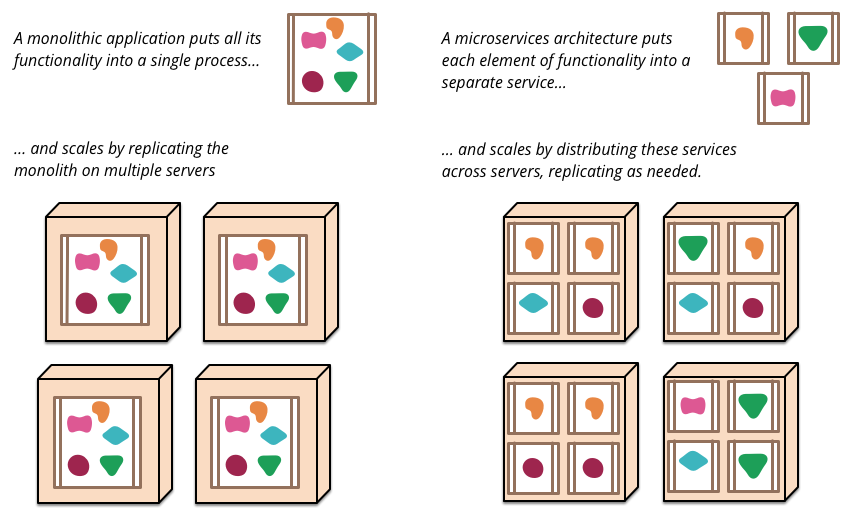
"Microservices" - yet another new term on the crowded streets of software architecture. Although our natural inclination is to pass such things by with a contemptuous glance, this bit of terminology describes a style of software systems that we are finding more and more appealing. We've seen many projects use this style in the last few years, and results so far have been positive, so much so that for many of our colleagues this is becoming the default style for building enterprise applications. Sadly, however, there's not much information that outlines what the microservice style is and how to do it.

In short, the microservice architectural style [[1]](https://martinfowler.com/articles/microservices.html#footnote-etymology) is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies.

To start explaining the microservice style it's useful to compare it to the monolithic style: a monolithic application built as a single unit. Enterprise Applications are often built in three main parts: a client-side user interface (consisting of HTML pages and javascript running in a browser on the user's machine) a database (consisting of many tables inserted into a common, and usually relational, database management system), and a server-side application. The server-side application will handle HTTP requests, execute domain logic, retrieve and update data from the database, and select and populate HTML views to be sent to the browser. This server-side application is a monolith - a single logical executable[[2]](https://martinfowler.com/articles/microservices.html#footnote-monolith). Any changes to the system involve building and deploying a new version of the server-side application.

Such a monolithic server is a natural way to approach building such a system. All your logic for handling a request runs in a single process, allowing you to use the basic features of your language to divide up the application into classes, functions, and namespaces. With some care, you can run and test the application on a developer's laptop, and use a deployment pipeline to ensure that changes are properly tested and deployed into production. You can horizontally scale the monolith by running many instances behind a load-balancer.

Monolithic applications can be successful, but increasingly people are feeling frustrations with them - especially as more applications are being deployed to the cloud . Change cycles are tied together - a change made to a small part of the application, requires the entire monolith to be rebuilt and deployed. Over time it's often hard to keep a good modular structure, making it harder to keep changes that ought to only affect one module within that module. Scaling requires scaling of the entire application rather than parts of it that require greater resource.



**Figure 1: Monoliths and Microservices**

These frustrations have led to the microservice architectural style: building applications as suites of services. As well as the fact that services are independently deployable and scalable, each service also provides a firm module boundary, even allowing for different services to be written in different programming languages. They can also be managed by different teams .

We do not claim that the microservice style is novel or innovative, its roots go back at least to the design principles of Unix. But we do think that not enough people consider a microservice architecture and that many software developments would be better off if they used it.

1. **Characteristics of a Microservice Architecture**

We cannot say there is a formal definition of the microservices architectural style, but we can attempt to describe what we see as common characteristics for architectures that fit the label. As with any definition that outlines common characteristics, not all microservice architectures have all the characteristics, but we do expect that most microservice architectures exhibit most characteristics. While we authors have been active members of this rather loose community, our intention is to attempt a description of what we see in our own work and in similar efforts by teams we know of. In particular we are not laying down some definition to conform to.

1. **Componentization via Services**

For as long as we've been involved in the software industry, there's been a desire to build systems by plugging together components, much in the way we see things are made in the physical world. During the last couple of decades we've seen considerable progress with large compendiums of common libraries that are part of most language platforms.

When talking about components we run into the difficult definition of what makes a component. Our definition is that a component is a unit of software that is independently replaceable and upgradeable.

Microservice architectures will use libraries, but their primary way of componentizing their own software is by breaking down into services. We define libraries as components that are linked into a program and called using in-memory function calls, while services are out-of-process components who communicate with a mechanism such as a web service request, or remote procedure call. (This is a different concept to that of a service object in many OO programs [[3]](https://martinfowler.com/articles/microservices.html#footnote-service-object).)

One main reason for using services as components (rather than libraries) is that services are independently deployable. If you have an application [[4]](https://martinfowler.com/articles/microservices.html#footnote-application) that consists of a multiple libraries in a single process, a change to any single component results in having to redeploy the entire application. But if that application is decomposed into multiple services, you can expect many single service changes to only require that service to be redeployed. That's not an absolute, some changes will change service interfaces resulting in some coordination, but the aim of a good microservice architecture is to minimize these through cohesive service boundaries and evolution mechanisms in the service contracts.

Another consequence of using services as components is a more explicit component interface. Most languages do not have a good mechanism for defining an explicit [Published Interface](https://martinfowler.com/bliki/PublishedInterface.html). Often it's only documentation and discipline that prevents clients breaking a component's encapsulation, leading to overly-tight coupling between components. Services make it easier to avoid this by using explicit remote call mechanisms.

Using services like this does have downsides. Remote calls are more expensive than in-process calls, and thus remote APIs need to be coarser-grained, which is often more awkward to use. If you need to change the allocation of responsibilities between components, such movements of behavior are harder to do when you're crossing process boundaries.

At a first approximation, we can observe that services map to runtime processes, but that is only a first approximation. A service may consist of multiple processes that will always be developed and deployed together, such as an application process and a database that's only used by that service.

1. **Organized around Business Capabilities**

When looking to split a large application into parts, often management focuses on the technology layer, leading to UI teams, server-side logic teams, and database teams. When teams are separated along these lines, even simple changes can lead to a cross-team project taking time and budgetary approval. A smart team will optimise around this and plump for the lesser of two evils - just force the logic into whichever application they have access to. Logic everywhere in other words. This is an example of Conway's Law[[5]](https://martinfowler.com/articles/microservices.html#footnote-conwayslaw) in action.

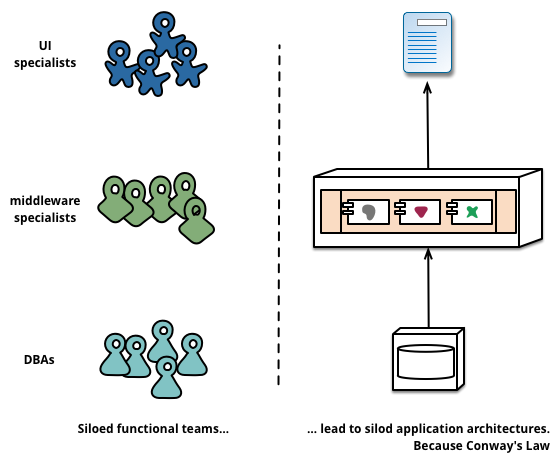


Figure 2: Conway's Law in action

The microservice approach to division is different, splitting up into services organized around business capability. Such services take a broad-stack implementation of software for that business area, including user-interface, persistant storage, and any external collaborations. Consequently the teams are cross-functional, including the full range of skills required for the development: user-experience, database, and project management.

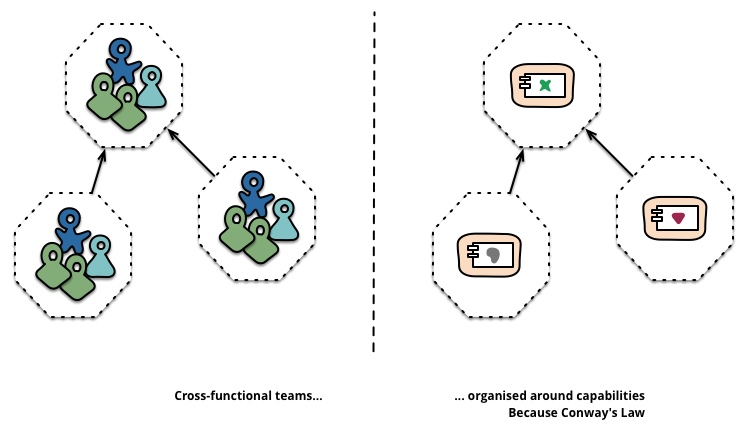


Figure 3: Service boundaries reinforced by team boundaries

One company organised in this way is [www.comparethemarket.com](http://www.comparethemarket.com/). Cross functional teams are responsible for building and operating each product and each product is split out into a number of individual services communicating via a message bus.

Large monolithic applications can always be modularized around business capabilities too, although that's not the common case. Certainly we would urge a large team building a monolithic application to divide itself along business lines. The main issue we have seen here, is that they tend to be organised around too many contexts. If the monolith spans many of these modular boundaries it can be difficult for individual members of a team to fit them into their short-term memory. Additionally we see that the modular lines require a great deal of discipline to enforce. The necessarily more explicit separation required by service components makes it easier to keep the team boundaries clear.

1. **Products not Projects**

Most application development efforts that we see use a project model: where the aim is to deliver some piece of software which is then considered to be completed. On completion the software is handed over to a maintenance organization and the project team that built it is disbanded.

Microservice proponents tend to avoid this model, preferring instead the notion that a team should own a product over its full lifetime. A common inspiration for this is Amazon's notion of ["you build, you run it"](https://queue.acm.org/detail.cfm?id=1142065) where a development team takes full responsibility for the software in production. This brings developers into day-to-day contact with how their software behaves in production and increases contact with their users, as they have to take on at least some of the support burden.

The product mentality, ties in with the linkage to business capabilities. Rather than looking at the software as a set of functionality to be completed, there is an on-going relationship where the question is how can software assist its users to enhance the business capability.

There's no reason why this same approach can't be taken with monolithic applications, but the smaller granularity of services can make it easier to create the personal relationships between service developers and their users.

1. **Smart endpoints and dumb pipes**

When building communication structures between different processes, we've seen many products and approaches that stress putting significant smarts into the communication mechanism itself. A good example of this is the Enterprise Service Bus (ESB), where ESB products often include sophisticated facilities for message routing, choreography, transformation, and applying business rules.

The microservice community favours an alternative approach: smart endpoints and dumb pipes. Applications built from microservices aim to be as decoupled and as cohesive as possible - they own their own domain logic and act more as filters in the classical Unix sense - receiving a request, applying logic as appropriate and producing a response. These are choreographed using simple RESTish protocols rather than complex protocols such as WS-Choreography or BPEL or orchestration by a central tool.

The two protocols used most commonly are HTTP request-response with resource API's and lightweight messaging[[8]](https://martinfowler.com/articles/microservices.html#footnote-protobufs). The best expression of the first is Microservice teams use the principles and protocols that the world wide web (and to a large extent, Unix) is built on. Often used resources can be cached with very little effort on the part of developers or operations folk.

The second approach in common use is messaging over a lightweight message bus. The infrastructure chosen is typically dumb (dumb as in acts as a message router only) - simple implementations such as RabbitMQ or ZeroMQ don't do much more than provide a reliable asynchronous fabric - the smarts still live in the end points that are producing and consuming messages; in the services.

In a monolith, the components are executing in-process and communication between them is via either method invocation or function call. The biggest issue in changing a monolith into microservices lies in changing the communication pattern. A naive conversion from in-memory method calls to RPC leads to chatty communications which don't perform well. Instead you need to replace the fine-grained communication with a coarser -grained approach.

1. **Decentralized Governance**

One of the consequences of centralised governance is the tendency to standardise on single technology platforms. Experience shows that this approach is constricting - not every problem is a nail and not every solution a hammer. We prefer using the right tool for the job and while monolithic applications can take advantage of different languages to a certain extent, it isn't that common.

Splitting the monolith's components out into services we have a choice when building each of them. You want to use Node.js to standup a simple reports page? Go for it. C++ for a particularly gnarly near-real-time component? Fine. You want to swap in a different flavour of database that better suits the read behaviour of one component? We have the technology to rebuild him.

Of course, just because you can do something, doesn't mean you should - but partitioning your system in this way means you have the option.

Teams building microservices prefer a different approach to standards too. Rather than use a set of defined standards written down somewhere on paper they prefer the idea of producing useful tools that other developers can use to solve similar problems to the ones they are facing. These tools are usually harvested from implementations and shared with a wider group, sometimes, but not exclusively using an internal open source model. Now that git and github have become the de facto version control system of choice, open source practices are becoming more and more common in-house .

Netflix is a good example of an organisation that follows this philosophy. Sharing useful and, above all, battle-tested code as libraries encourages other developers to solve similar problems in similar ways yet leaves the door open to picking a different approach if required. Shared libraries tend to be focused on common problems of data storage, inter-process communication and as we discuss further below, infrastructure automation.

For the microservice community, overheads are particularly unattractive. That isn't to say that the community doesn't value service contracts. Quite the opposite, since there tend to be many more of them. It's just that they are looking at different ways of managing those contracts. Patterns like [Tolerant Reader](https://martinfowler.com/bliki/TolerantReader.html) and [Consumer-Driven Contracts](https://martinfowler.com/articles/consumerDrivenContracts.html) are often applied to microservices. These aid service contracts in evolving independently. Executing consumer driven contracts as part of your build increases confidence and provides fast feedback on whether your services are functioning. Indeed we know of a team in Australia who drive the build of new services with consumer driven contracts. They use simple tools that allow them to define the contract for a service. This becomes part of the automated build before code for the new service is even written. The service is then built out only to the point where it satisfies the contract - an elegant approach to avoid the 'YAGNI'[[9]](https://martinfowler.com/articles/microservices.html#footnote-YAGNI) dilemma when building new software. These techniques and the tooling growing up around them, limit the need for central contract management by decreasing the temporal coupling between services.

Perhaps the apogee of decentralised governance is the build it / run it ethos popularised by Amazon. Teams are responsible for all aspects of the software they build including operating the software 24/7. Devolution of this level of responsibility is definitely not the norm but we do see more and more companies pushing responsibility to the development teams. Netflix is another organisation that has adopted this ethos[[11]](https://martinfowler.com/articles/microservices.html#footnote-netflix-flowcon). Being woken up at 3am every night by your pager is certainly a powerful incentive to focus on quality when writing your code. These ideas are about as far away from the traditional centralized governance model as it is possible to be.

# 附录B 外文翻译—译文部分

**微服务——一个新架构术语的定义**

在过去几年中，“微服务架构”这一术语如雨后春笋般涌现出来，它描述了一种将软件应用程序设计为一组可独立部署的服务的特定方式。虽然这种架构风格没有明确的定义，但在组织、业务能力上有一些共同的特征：自动化部署，端点智能化，语言和数据的去中心化控制。

“微服务” - 软件架构拥挤大街上的有一个新术语。虽然我们自然的倾向是轻蔑的一瞥将它一带而过，然而我们发现这一术语描述了一种越来越吸引人的软件系统风格。我们已看到，在过去的几年中有许多项目使用了这种风格，并且到目前为止结果都还不错，以致于这已变成了我们同事在构建企业级应用程序时默认使用的架构风格。然而，遗憾的是并没有太多的信息来概述什么是微服务风格以及怎样用这种风格。

简单来说，微服务架构风格是一种将一个单一应用程序开发为一组小型服务的方法，每个服务运行在自己的进程中，服务间通信采用轻量级通信机制(通常用HTTP资源API)。这些服务围绕业务能力构建并且可通过全自动部署机制独立部署。这些服务共用一个最小型的集中式的管理，服务可用不同的语言开发，使用不同的数据存储技术。

与单体风格作对比有助于开始解释微服务风格：单体应用程序被构建为单一单元。企业级应用程序通常由三部分组成：客户端侧用户接口(由运行于开发机上的浏览器里的HTML页面和Javascript组成)，数据库(由插入到通用关系型数据库管理系统中的许多数据表格组成)，服务端应用程序。服务端应用程序处理HTTP请求，执行领域逻辑，从数据库中检索、更新数据，选择、填充将要发送到浏览器的HTTP视图。服务端应用程序是一个单一的逻辑可执行单体。系统的任何改变都将牵涉到重新构建和部署服务端的一个新版本。

这样的单体服务器是构建这样一个系统最自然的方式。处理请求的所有逻辑都运行在一个单一进程中，允许你使用编程语言的基本特性将应用程序划分类、函数和命名空间。你认真的在开发机上运行测试应用程序，并使用部署管道来保证变更已被正确地测试并部署到生产环境中。该单体的水平扩展可以通过在负载均衡器后面运行多个实例来实现。

单体应用程序可以是成功的，但人们日益对他们感到挫败，尤其是随着更多的应用程序被部署在云上。变更周期被捆绑在一起 —— 即使只变更应用程序的一部分，也需要重新构建并部署整个单体。长此以往，通常将很难保持一个良好的模块架构，这使得很难变更只发生在需要变更的模块内。程序扩展要求进行整个应用程序的扩展而不是需要更多资源的应用程序部分的扩展。

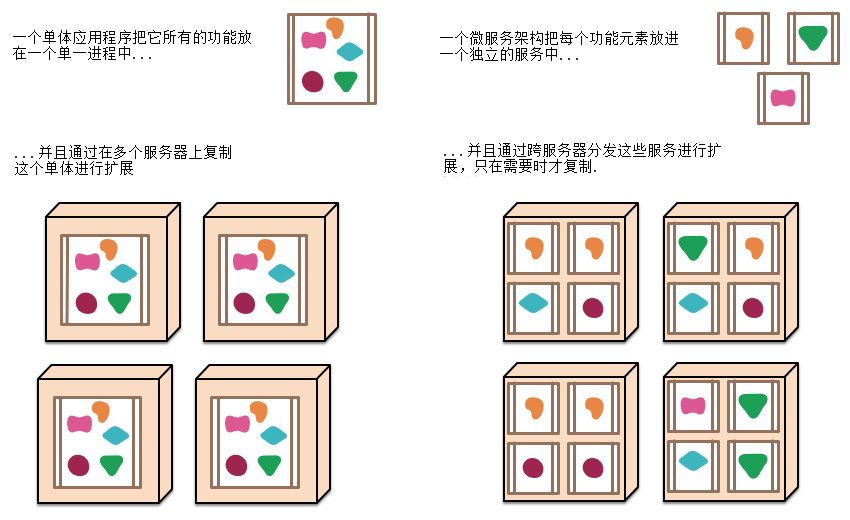


图1: 单体和微服务

这些挫败导向了微服务架构风格：构建应用程序为服务套件。除了服务是可独立部署、可独立扩展的之外，每个服务都提供一个固定的模块边界。甚至允许不同的服务用不同的的语言开发，由不同的团队管理。

我们不会声称微服务风格是新颖的、创新的，其本质至少可以回溯到Unix的设计哲学。但我们的确认为没有足够的人仔细考虑微服务架构，并且如果使用它很多软件实现将会更好。

**1.微服务架构的特征**

我们无法给出微服务架构风格的一个正式定义，但我们可以尝试去描述我们看到的符合该架构的一些共性。就概述共性的任何定义来说，并非所有的微服务架构风格都有这些共性，但我们期望大多数微服务架构风格展现出大多数特性。虽然本文作者一直是这个相当松散的社区的活跃用户，我们的目的是试图描述我们工作中和我们知道的一些团队的相似努力中的所见所闻。特别是我们不会制定一些可遵守的定义。

1.1 通过服务组件化

只要我们一直从事软件行业，一个愿望就是通过把组件插在一起构建系统，如同我们看到的现实世界中事物的构造方式一样。在最近的二十年中，我们看到作为大多数语言平台一部分的公共库的大量汇编工作取得了很大的进展。

当谈到组件时，我们遭遇困难的定义：组件是什么。我们的定义是：组件是一个可独立替换和独立升级的软件单元。

微服务架构将使用库，但组件化软件的主要方式是分解成服务。我们把库定义为链接到程序并使用内存函数调用来调用的组件，而服务是一种进程外的组件，它通过web服务请求或rpc(远程过程调用)机制通信(这和很多面向对象程序中的服务对象的概念是不同的。)

使用服务作为组件而不是使用库的一个主要原因是服务是可独立部署的。如果你有一个应用程序[[4]](http://blog.cuicc.com/blog/2015/07/22/microservices/" \l "fn4)是由单一进程里的多个库组成，任何一个组件的更改都导致必须重新部署整个应用程序。但如果应用程序可分解成多个服务，那么单个服务的变更只需要重新部署该服务即可。当然这也不是绝对的，一些变更将会改变服务接口导致一些协作，但一个好的微服务架构的目的是通过内聚服务边界和按合约演进机制来最小化这些协作。

使用服务作为组件的另一个结果是一个更加明确的组件接口。大多数语言没有一个好的机制来定义一个明确的[发布接口](http://martinfowler.com/bliki/PublishedInterface.html)。通常只有文档和规则来预防客户端打破组件的封装，这导致组件间过于紧耦合。服务通过明确的远程调用机制可以很容易的避免这些。

像这样使用服务确实有一些缺点，远程调用比进程内调用更昂贵，因此远程API被设计成粗粒度，这往往更不便于使用。如果你需要更改组件间的责任分配，当你跨进程边界时，这样的行为动作更难达成。

直观的估计，我们观察到服务与运行时进程一一映射，但这仅仅是直观的估计而已。一个服务可能由多进程组成，这些进程总是被一起开发和部署，比如只被这个服务使用的应用进程和数据库。

1.2 围绕业务能力组织

当想要把大型应用程序拆分成部件时，通常管理层聚焦在技术层面，导致UI团队、服务侧逻辑团队、数据库团队的划分。当团队按这些技术线路划分时，即使是简单的更改也会导致跨团队的时间和预算审批。一个聪明的团队将围绕这些优化，两害取其轻 - 只把业务逻辑强制放在它们会访问的应用程序中。换句话说，逻辑无处不在。这是Conway法则在起作用的一个例子。

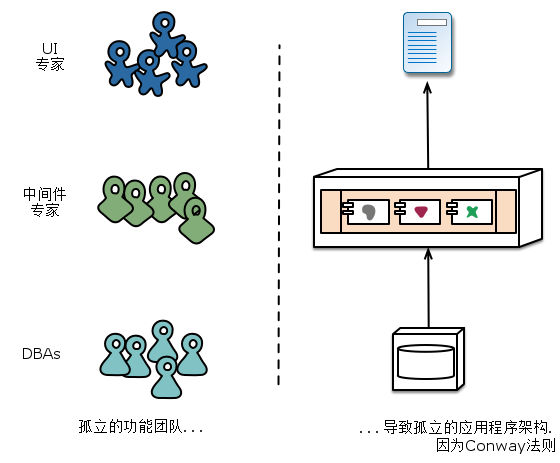


图2: Conway法则在起作用

微服务采用不同的分割方法，划分成围绕业务能力组织的服务。这些服务采取该业务领域软件的宽栈实现，包括用户接口、持久化存储和任何外部协作。因此，团队都是跨职能的，包括开发需要的全方位技能：用户体验、数据库、项目管理。

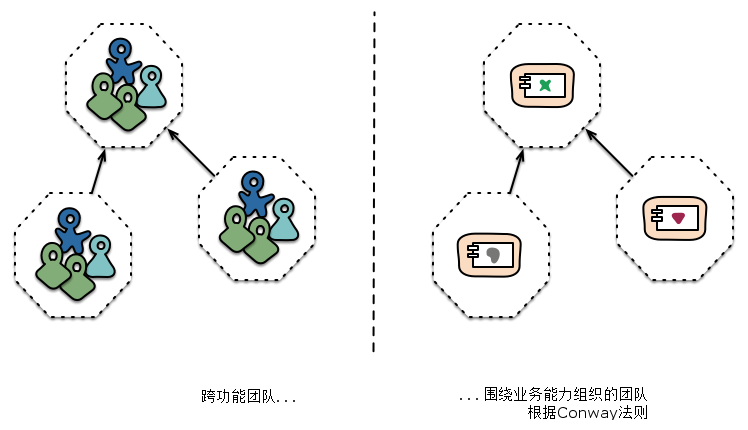


图3: 团队边界增强的服务边界

[www.comparethemarket.com](http://blog.cuicc.com/blog/2015/07/22/microservices/www.comparethemarket.com)是按这种方式组织的一个公司。跨职能团队负责创建和运营产品，产品被划分成若干个体服务，这些服务通过消息总线通信。

大型单体应用程序也总是可以围绕业务能力来模块化，虽然这不是常见的情况。当然，我们将敦促创建单体应用程序的大型团队将团队本身按业务线拆分。我们看到这种情况的主要问题是他们趋向于围绕太多的上下文进行组织。如果单体横跨了多个模块边界，对团队个体成员来说，很难把它们装进他们的短期记忆里。另外，我们看到模块化的路线需要大量的规则来强制实施。服务组件所要求的更加明确的分离，使得它更容易保持团队边界清晰。

1.3 是产品不是项目

我们看到大多数应用程序开发工作使用一个项目模式：目标是交付将要完成的一些软件。完成后的软件被交接给维护组织，然后它的构建团队就解散了。

微服务支持者倾向于避免这种模式，而是认为一个团队应该负责产品的整个生命周期。对此一个共同的启示是亚马逊的理念 [“you build, you run it”](https://queue.acm.org/detail.cfm?id=1142065)，开发团队负责软件的整个产品周期。这使开发者经常接触他们的软件在生产环境如何工作，并增加与他们的用户联系，因为他们必须承担至少部分的支持工作。

产品思想与业务能力紧紧联系在一起。要持续关注软件如何帮助用户提升业务能力，而不是把软件看成是将要完成的一组功能。

没有理由说为什么同样的方法不能用在单体应用程序上，但服务的粒度更小，使得它更容易在服务开发者和用户之间建立个人关系。

1.4 智能端点和哑管道

当在不同进程间创建通信结构时，我们已经看到了很多的产品和方法，把显著的智慧强压进通信机制本身。一个很好的例子就是企业服务总线(ESB)，在ESB产品中通常为消息路由、编排(choreography)、转化和应用业务规则引入先进的设施。

微服务社区主张另一种方法：智能端点和哑管道。基于微服务构建的应用程序的目标是尽可能的解耦和尽可能的内聚 - 他们拥有自己的领域逻辑，他们的行为更像经典UNIX理念中的过滤器 - 接收请求，应用适当的逻辑并产生响应。使用简单的REST风格的协议来编排他们，而不是使用像WS-Choreography或者BPEL或者通过中心工具编制(orchestration)等复杂的协议。

最常用的两种协议是使用资源API的HTTP请求-响应和轻量级消息传送。对第一种协议最好的表述是

微服务团队使用的规则和协议，正是构建万维网的规则和协议(在更大程度上，是UNIX的)。从开发者和运营人员的角度讲，通常使用的资源可以很容易的缓存。

第二种常用方法是在轻量级消息总线上传递消息。选择的基础设施是典型的哑的(哑在这里只充当消息路由器) - 像RabbitMQ或ZeroMQ这样简单的实现仅仅提供一个可靠的异步交换结构 - 在服务里，智能仍旧存活于端点中，生产和消费消息。

单体应用中，组件都在同一进程内执行，它们之间通过方法调用或函数调用通信。把单体变成微服务最大的问题在于通信模式的改变。一种幼稚的转换是从内存方法调用转变成RPC，这导致频繁通信且性能不好。相反，你需要用粗粒度通信代替细粒度通信。

1.5 去中心化治理

集中治理的一个后果是单一技术平台的标准化发展趋势。经验表明，这种方法正在收缩 - 不是每个问题都是钉子，不是每个问题都是锤子。我们更喜欢使用正确的工具来完成工作，而单体应用程序在一定程度上可以利用语言的优势，这是不常见的。

把单体的组件分裂成服务，在构建这些服务时可以有自己的选择。你想使用Node.js开发一个简单的报告页面？去吧。用C++实现一个特别粗糙的近乎实时的组件？好极了。你想换用一个更适合组件读操作数据的不同风格的数据库？我们有技术来重建它。

当然，仅仅因为你可以做些什么，而不意味着你应该这样做 - 但用这种方式划分系统意味着你可以选择。

团队在构建微服务时也更喜欢用不同的方法来达标。他们更喜欢生产有用的工具这种想法，而不是写在纸上的标准，这样其他开发者可以用这些工具解决他们所面临的相似的问题。有时，这些工具通常在实施中收获并与更广泛的群体共享，但不完全使用一个内部开源模型。现在git和github已经成为事实上的版本控制系统的选择，在内部开放源代码的实践也正变得越来越常见。

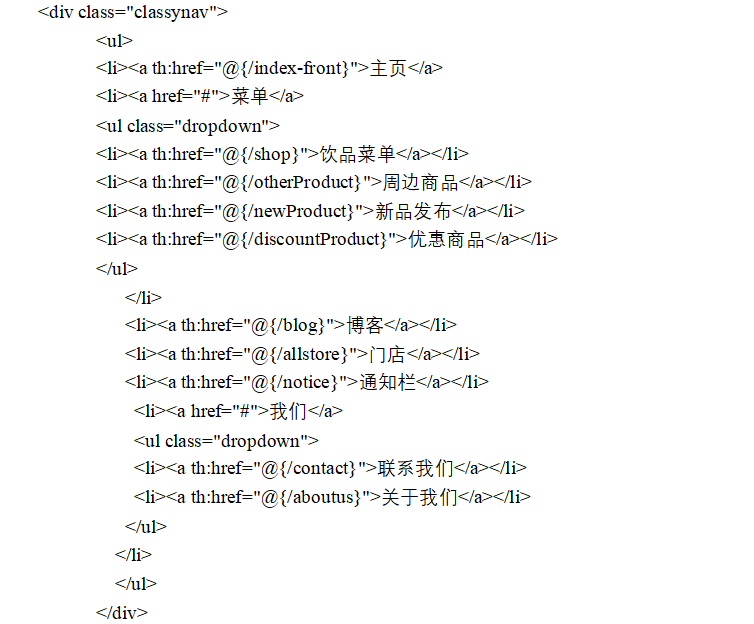
Netflix是遵守这一理念的很好的例子。尤其是，以库的形式分享有用的且经过市场检验的代码，这激励其他开发者用类似的方式解决相似的问题，同时还为采用不同方法敞开了大门。共享库倾向于聚焦在数据存储、进程间通信和我们接下来要深入讨论的基础设施自动化的共性问题。

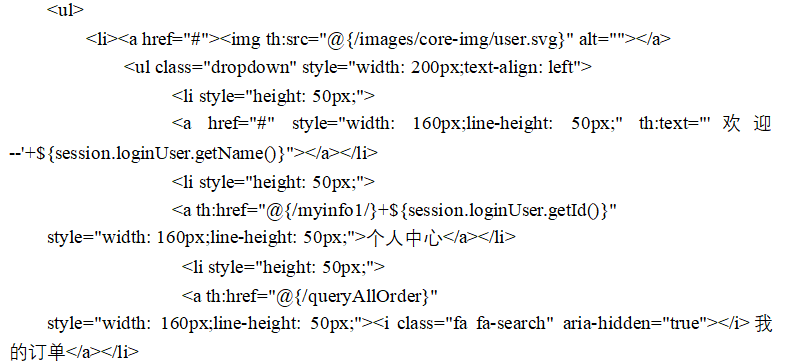
对为服务社区来说，开销特别缺乏吸引力。这并不是说社区不重视服务合约。恰恰相反，因为他们有更多的合约。只是他们正在寻找不同的方式来管理这些合约。像[Tolerant Reader](http://martinfowler.com/bliki/TolerantReader.html)和消费者驱动的契约([Consumer-Driven Contracts](http://martinfowler.com/articles/consumerDrivenContracts.html))这样的模式通常被用于微服务。  
 这些援助服务合约在独立进化。执行消费者驱动的合约作为构建的一部分，增加了信心并对服务是否在运作提供了更快的反馈。事实上，我们知道澳大利亚的一个团队用消费者驱动的合约这种模式来驱动新业务的构建。他们使用简单的工具定义服务的合约。这已变成自动构建的一部分，即使新服务的代码还没写。服务仅在满足合约的时候才被创建出来 - 这是在构建新软件时避免"YAGNI"[[9]](http://blog.cuicc.com/blog/2015/07/22/microservices/#fn9)困境的一个优雅的方法。围绕这些成长起来的技术和工具，通过减少服务间的临时耦合，限制了中心合约管理的需要。

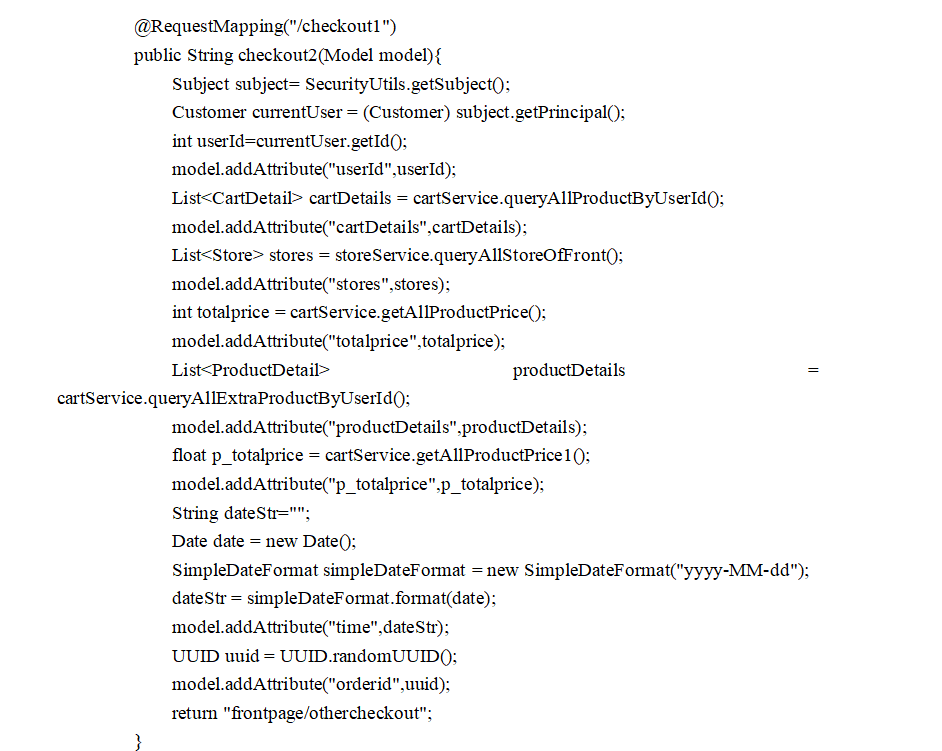
也许去中心化治理的最高境界就是亚马逊广为流传的build it/run it理念。团队要对他们构建的软件的各方面负责，包括7\*24小时的运营。这一级别的责任下放绝对是不规范的，但我们看到越来越多的公司让开发团队负起更多责任。Netflix是采用这一理念的另一家公司[[11]](http://blog.cuicc.com/blog/2015/07/22/microservices/#fn11)。每天凌晨3点被传呼机叫醒无疑是一个强有力的激励，使你在写代码时关注质量。这是关于尽可能远离传统的集中治理模式的一些想法。

# 附录C 软件使用说明书

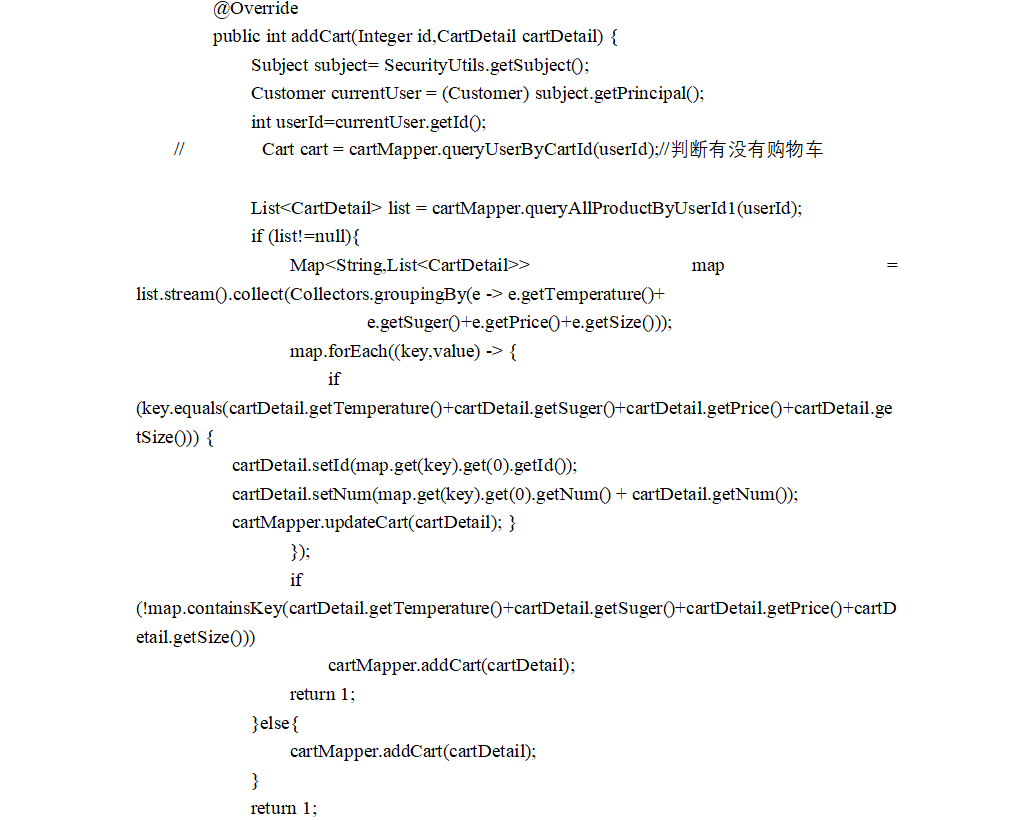
# 附录D 主要源代码











# 附录E 相关图表