

毕业设计（论文）

题目 基于前后端分离技术的

微信点餐系统开发

学生姓名 吴励寒 学号 2015112155

专业 计算机科学与技术 班级20151121

指导教师 彭代军

评阅教师

完成日期 年 月 日

目 录

[摘 要 1](#_Toc9421712)

[前 言 2](#_Toc9421713)

[1 绪论 4](#_Toc9421714)

[1.1 研究背景及意义 4](#_Toc9421715)

[1.2 国内外研究现状 4](#_Toc9421716)

[1.3 本文研究内容 6](#_Toc9421717)

[2 系统分析 7](#_Toc9421718)

[2.1 系统运行环境 7](#_Toc9421719)

[2.2 功能性需求分析 7](#_Toc9421720)

[2.3 非功能性需求分析 8](#_Toc9421721)

[2.4 可行性分析 9](#_Toc9421722)

[2.5 业务流程分析 9](#_Toc9421723)

[3 系统设计 12](#_Toc9421724)

[3.1系统架构设计 12](#_Toc9421725)

[3.2 功能结构设计 13](#_Toc9421726)

[3.3 接口设计 15](#_Toc9421727)

[3.4 数据库结构设计 36](#_Toc9421728)

[4 系统测试 41](#_Toc9421729)

[4.1 测试方法 42](#_Toc9421730)

[4.2 测试用例 42](#_Toc9421731)

[4.3 测试结论 42](#_Toc9421732)

[5 结论与总结 43](#_Toc9421733)

[致 谢 44](#_Toc9421734)

[参考文献 45](#_Toc9421735)

[附 录 46](#_Toc9421736)

基于前后端分离技术的微信点餐系统开发

学 生： 吴励寒

指导教师： 彭代军

（三峡大学 计算机与信息学院）

**摘 要：**

本文讲述的是一个基于前后端分离技术的微信点餐系统的设计及开发过程，使用了SpringBoot框架和vue框架。 手机上安装有微信的用户通过扫码进入点餐系统前端，而商家使用浏览器登录进入点餐系统后端。 买家可以使用产品视图，商家视图，购物车，订单，付款，评估等功能，卖家可以使用用户管理功能，角色管理功能，订单管理功能，资源管理功能，评论管理功能，类别管理功能，商品管理功能。本系统能大幅节省大量人力，节约大量物力，提高各个商家的出餐效率以及食客的用餐效率。

**关键词：**微信点餐系统，SpringBoot，Vue，前后端分离

**Abstract：**

This paper describes the design and development process of a WeChat ordering system based on front and rear end separation technology， using SpringBoot framework and Vue framework. Users with WeChat installed on their mobile phones can enter the front end of the ordering system by scanning the code， while merchants can log in the back end of the ordering system by using the browser. Buyers can use commodity view， merchants view， shopping cart， order placing， payment， evaluation and other functions， while sellers can use user management function， role management function， order management function， resource management function， comment management function， category management function and commodity management function. The system can greatly save a lot of manpower， save a lot of material resources， improve the efficiency of each business meal and diners dining efficiency.

**Keywords**：Wechat ordering system, SpringBoot，Vue，front and rear end separation

前 言

对于餐厅来说，在70年代的时候有算盘，在80年代的时候有人工记账，在90年代的时候有电脑，而在今天我们有互联网系统。大家的饮食观念、饮食习惯在变，而餐厅的管理和经营模式也在变，推动这种变化的正是现在全球信息化的飞速发展。

而餐饮信息化系统，在这种环境下应运而生。那么什么是餐饮信息化系统呢？其实从简单来讲，餐饮信息化系统就是利用现有的计算机技术，来代替人工去执行餐饮过程中的各个环节、管理各类信息数据，比如由传统的人工进行纸笔点餐改为电子点餐、传统的人工收银改为电子收银等等，而这种发展，大大提高了餐饮效率和用户体验。随着餐饮行业以及计算机技术的发展，餐饮信息化系统在餐饮企业中也变得越来越重要，许多餐饮企业通过餐饮信息化系统来降低成本、提高效率。而在餐饮信息化系统不断升级的过程中，无线点餐，将成为绕不过的一代系统。

从90年代到现今的发展来看，餐饮信息化系统可以大致被分为三代：第一代系统是在上世纪90年代出现的收银系统，它的核心功能是实现了使用电脑进行收银，能过解决在人工结账时存在的很多问题，比如人工舞弊、计算错误、交易金额错误等等问题；第二代系统则是出现于世纪初，手持点菜，然后厨房分单打印，其核心价值解决了第一代系统中存在的服务人员需要背菜单、记菜价、手工记录顾客点单并在餐桌、收银台、后厨间来回奔跑传递小票，收银台收银人员需再次录入顾客点单的低效问题；第三代系统则是当下出现的“互联网+”或“O2O”系统，除了实现第二代所需的必要功能，第三代系统的核心价值要解决的是：在当今这个互联网时代下，餐饮行业的销售渠道发生了很大变化，之前的渠道是纯门店，而现在则是线下和向线上多渠道，这些渠道的管理问题和餐饮核心资源：店铺资源和人力资源以及原材料资源价格不断上涨但资源效率仍然低的问题。

无线点菜系统的发展与日渐完善，标志着餐饮行业逐渐在向信息化发展,采用该系统能过增加商家效率、降低成本、提升企业服务形象,同时可以加强企业的日常经营管理，使企业由粗放型的管理逐渐转向集约型的转变。

而微信，正是当今最火的移动社交软件之一。

微信是中国知名计算机公司-腾讯公司在2011年1月21日推出的一个为手机、平板、电脑提供即时通讯的支持跨通信运营商(移动、联通、电信)的、跨操作系统(android、ios、macOS、linux、windows)的、免费的app，它可以发送文字信息进行聊天，可以进行语音聊天，也可以发送视频、图片，同时，也可以使用基于移动支付、地理位置信息、资讯媒体的功能插件，例如：“摇一摇”、“扫一扫”、“看一看”、“搜一搜”。微信不仅提供微信自己内部的功能例如朋友圈功能，用户也可以通过微信使用很多其他公司的小程序应用，比如京东、美团、新浪微博等等，也可以使用微信记录自己生活中的点点滴滴，或者阅读文章，并且分享给自己的好友或者朋友圈，也是很好的社交软件。微信的作用主要有：1、多平台、跨平台、不受平台及系统限制，使用ios系统的微信可以和使用android系统的微信聊天。2、降低了沟通的成本，只需要一点点流量就可以让天南地北的人保持联系。3、在传输视频、图片、音乐时，会自动进行压缩，节省了很多流量。4、在聊天时会实时显示对方状态，便于判断对方状态。

微信自5.0以后，重磅推出微信官方公众号，后演变成微信小程序，而基于微信10亿多的用户量、微信小程序6亿多的用户量，如果能将微信与餐饮点餐系统结合起来，无疑是一个O2O模式的入口，更是一个实现无线点餐的完美解决方案。所以微信点餐的魅力以及商机不言而喻。利用微信点餐未来可能是餐饮行业的一个不错的发展趋势，只要微信这个应用存在，以及它生态不断地扩大。将来不单单只是下单点餐，结合其本身更大的营销推广特点以及O2O的模式，将能给商家带来更多的利润。

本论文中的基于前后端分离技术的微信点餐系统的设计与开发正是在这种背景下诞生的。用微信小程序点餐，不仅能够提高餐厅的档次、节省人力成本、提高餐饮效率、体现与其他商家的差异性，而且也能通过微信这个10亿多用户的社交软件，免费为自家餐厅打广告，推送优惠信息，从而提高用户粘合度。

1 绪论

1.1 研究背景及意义

从2017年之后，餐饮行业正式迈入一个利润丰厚的时代，它的赚钱模式正在发生着天翻地覆的改变，尤其是信息化技术的发展对于餐饮业的改变尤为明显，而随着餐饮业和信息技术的进一步合作，开发出一套更为强大也更为便利的点餐系统显得尤为重要，它可以在使用更少成本、更少时间的情况下完成之前更多的工作，更好的促进餐饮业的发展、提高用户和商家的效率。

微信点餐，顾客只需在附近小程序点击小程序、或者在搜索栏搜索微信小程序名字、或者在餐厅内扫一扫该餐厅的微信小程序二维码，便可看到餐厅的菜肴、酒水的照片、简介以及价格，只要将自己中意的食物加入购物车，然后点击购物车进行结算，订单则会主动上传到点餐系统，厨房即可看到顾客的订单，不需要服务员在旁边等候点餐，极大提高了效率。点完菜后，订单上会显示菜品清单和待支付金额，顾客在线支付即可。餐厅管理员可以通过后台即时看到用户下单信息、支付信息，能很直观的统计收入，为餐饮行业提供高效的信息化服务。

1.2 国内外研究现状

随着世界计算机行业的发展，通过计算机进行内部控制成了餐饮行业的主要手段。以前的餐饮企业，想做大做强只需要味道好、服务好就行，而现在餐饮企业如果想要做大做强，就必须在计算机技术上投入，才能进行规划，才能更好地管理，首先实现标准化、流程化，进而为企业自身的发展铺平道路。

1.2.1 国内研究现状

互联网行业的飞速发展，已经对各行各业产生了影响，在国家“互联网+”政策的大力推行下，餐饮业也同样大力同互联网结合，形成餐饮业+互联网的组合，而国内餐厅常见的无线点餐方式大致有以下几种：

1. 使用点菜器进行点餐

服务员在顾客的指点下通过包含点菜、下单、结账以及查看菜品等功能的点菜器进行点单。这种方式相对降低了服务员工作量，节省了一定的人力成本，缺点是顾客无法在餐厅外部使用，缺乏预约和外卖功能，还是需要服务员花费一定时间去服务，功能较为单一。

1. 使用平板点餐

顾客到店后可以直接店家提供的平板上查看并选择菜品然后下单，而店家可以自由修改菜谱，支持多种浏览及管理功能。优缺点和点菜器类似。

1. 使用app点餐

顾客在手机上下载安装店家的app，在app里面进行浏览以及点餐等操作，其最大的优势是互动性强，店家可以使用app来做推广、活动、推送，但是缺点在于在前期的推广环节比较困难，顾客未必会乐意单独安装一个app为了在特定商家使用，大型连锁餐厅用户量较大，难度小一点；而小型餐厅则难度较高。

1. 使用微信扫码点餐

只要顾客手机上面安装了微信，就可以通过扫码点餐的方式自助点餐，这种方式既节省人力物力，也无须安装app，同时商家也能经常定制一些活动。

1.2.2 国外研究现状

相比于国内，欧美等一些西方的发达国家的信息化进步较早，餐饮业的信息化起步也较早，其发展历程大致有以下几个阶段：

1. 基于PC端的点餐系统

从20世纪80年代开始，一些餐厅开始打造自己的有线的PC端点餐系统。顾客前往前台点单，服务员在PC端进行下单，这种方式很大程度上解决了纸质点餐下单错误的问题。

1. 基于PDA的点餐系统

从二十世纪八十年代末期开始，随着无需电缆即可进行通信的Wireless network的出现以及嵌入式这一技术的发展，出现了基于PDA的点餐系统。服务员手持PDA按照顾客的需求进行下单，这种方式很大程度提高了点餐效率。

1. 基于移动端的点餐系统

从20世纪90年代开始，随着互联网的发展以及手机的普及，很多餐厅定制了自己的点餐app，顾客使用app进行浏览、点餐、支付等操作。

1.3 本文研究内容

本课题是设计与实现一个基于前后端分离的微信点餐系统，服务端采用SpringBoot框架，它集成了tomcat,不需要手动去部署war包，而且简化了maven的配置，很方便；前端则采用了Vue框架,以微信平台作为客户端，以mysql作为数据库。

2 系统分析

2.1 系统运行环境

2.1.1 硬件要求

处理器: Intel Xeon(Skylake) Platinum 8163

主频：2.5GHz

内存: 16 GiB

硬盘配置:50G SSD

带宽：2.5Gbps

电源:低功耗 300W 电源

2.1.2 软件要求

操作系统：centos7

数据库：mysql5.7,redis

Java环境：jdk1.8

Vue环境：node,cnpm

服务器：tomcat7, Nginx

2.2 功能性需求分析

2.2.1 卖家功能模块

1. 用户管理：包含用户登录、密码修改、用户添加、角色关联、用户修改、用户删除以及用户查找等功能。初始时提供一个系统管理员账号，店家可以使用账号登录系统进行一系列的管理，可自行添加用户并指派角色。
2. 角色管理：包含角色添加、角色查找、角色修改、资源分配、角色删除功能。系统管理员可以自由编辑角色的状态、名称以及可访问的资源，通过给用户指派角色控制用户的身份以及访问。
3. 资源管理：包含资源添加、资源查找、资源修改、资源删除。该模块主要用于定义各功能资源的key和url，系统根据资源的key做点击时的权限控制，根据资源的url做访问时的权限控制，保证系统的安全性。
4. 商品类目管理：包含类目添加、类目查询、类目修改、类目删除功能。管理员可对商品类别进行管理，便于顾客选购。
5. 商品管理：包含商品添加、商品查询、商品修改和商品删除功能。管理员可对商品进行管理，执行上架、下架、修改描述、修改价格等操作。
6. 订单管理：包含订单查看、新订单提醒、订单查找、订单完成和订单取消功能。接收小程序传来的订单并进行提示，店家可以根据实际情况修改订单状态。
7. 评论管理：包含评论添加、评论查询、评论修改和评论删除功能。有权限的管理员可对评论进行管理，便于及时调整商品、调整活动。

2.2.2 买家功能模块

1. 商品展示：包含商品类别的侧边栏展示以及商品详情的组合展示，方便用户能更直观选择商品。
2. 商家展示：包含商家信息、活动信息、评分信息等一系列信息在小程序的展示。
3. 购物车功能：用户在一次购买时可以将商品暂时存入购物车，点击购物车弹出所选商品，商品价格会自动进行累加，之后用户可以在选取完毕以后一次性进行结算。
4. 订单功能：点击下单则进入订单，卖家后台将收到新订单提醒，若取消订单则会在小程序界面显示订单取消；若订单完成则小程序界面跳转至支付界面。
5. 支付功能：调用后台的支付接口进行支付，支付结果将通知用户。
6. 评价功能：订单完成后用户可以对订单进行评价。

2.3 非功能性需求分析

1. 准确性需求：所有日期统一使用yyyy-MM-dd HH:mm:ss的日期格式；所以货币单位统一为元，统一采用浮点数进行货币计算，保留两位小数。
2. 性能需求：各项请求的响应时间、事务的处理时间都不应该超过3秒。
3. 易用性需求：系统操作应简单易懂，任何人都可以不需要过多思考轻易上手；界面美观，能一眼吸引住用户。
4. 稳定性需求：系统一年内故障次数不能超过10次，且日志功能健全，出现问题能很快定位，快速恢复；
5. 安全性需求：买家用户必须使用微信扫码才可进入小程序界面；卖家用户必须登录才可进入后台管理系统，且不同角色有不同权限控制；

2.4 可行性分析

1. 技术可行性分析：本系统是基于SpringBoot + Vue两大框架进行开发的，两大框架技术已经较为成熟且应用较为广泛；数据库则主要使用mysql，mysql是现在最通用的数据库之一，且是世界上最好的关系型数据库;开发工具主要使用IDEA，它是现在java开发者使用得最多同时也是最方便的java开发ide之一。
2. 社会可行性分析：本系统完成之后将带来以下好处：
3. 改变传统点餐方式，改善就餐高峰期的拥挤度，提高就餐效率。传统就餐方式在就餐高峰期时，其实大部分人浪费时间在排队上，而本系统就能解决这个问题。
4. 减少餐厅的人力投入。服务员不需要等着用户点完餐再下单，这样就不需要很多服务员等着客户点单，也无需服务员每次都花费很长时间介绍每种商品，这样也为餐厅减少了成本。
5. 用户可以更加自由地选择自己想吃的菜品，看到图片和评论会有更直观的感受和印象。
6. 商家可以随时查看后台数据，根据实际情况对商品做出调整。
7. 经济可行性分析：微信点餐系统完全由我自己开发，所以人力成本是免费的，开发设备也是用的是现有的设备，参考文件已经困难解决多由导师提供，唯一的消费主要产生于服务器的购买以及域名的购买，投入的人力、物力成本相对较少。而系统的价值远远超过投入。

2.5 业务流程分析

2.5.1 用户点餐流程

用户点餐流程如图2.1所示：

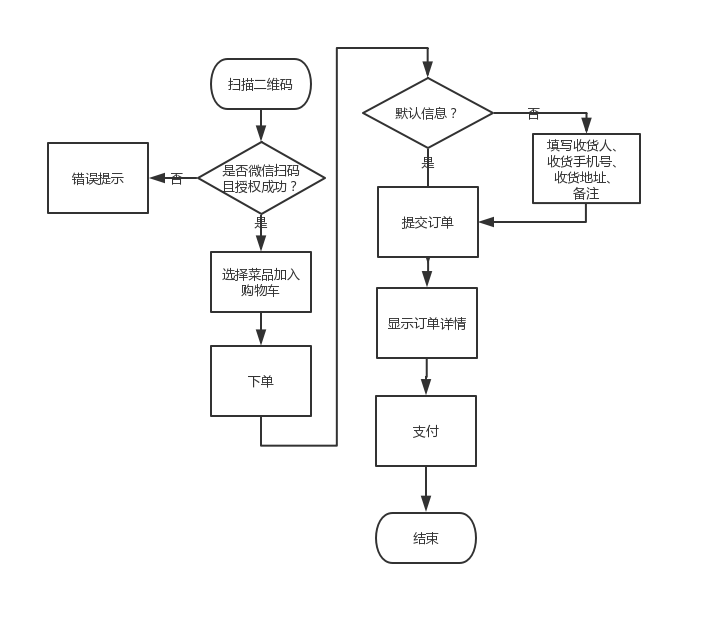


图2.1 用户点餐流程图

2.5.2 后台管理流程

后台管理流程如图2.2所示：

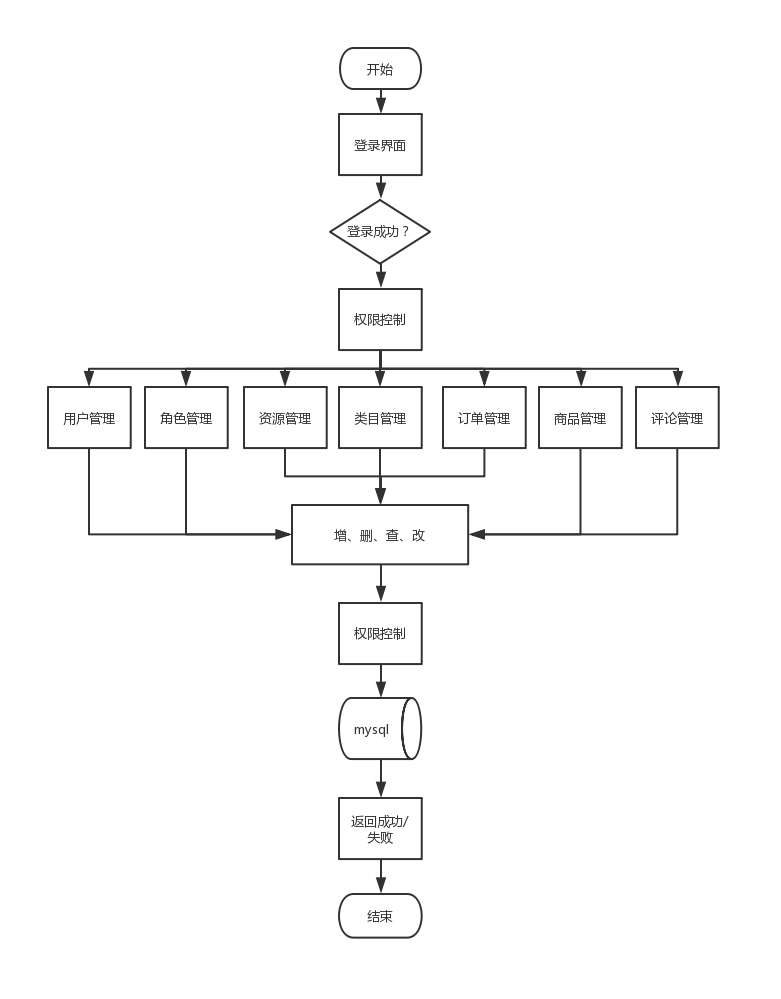


图2.2 后台管理流程图

3 系统设计

3.1系统架构设计

微信点餐系统是一个基于前后端分离的、使用SpringBoot框架、shiro框架、mysql、Vue框架开发的系统，其前后端分离的总体架构如图3.1、图3.2、图3.3所示：

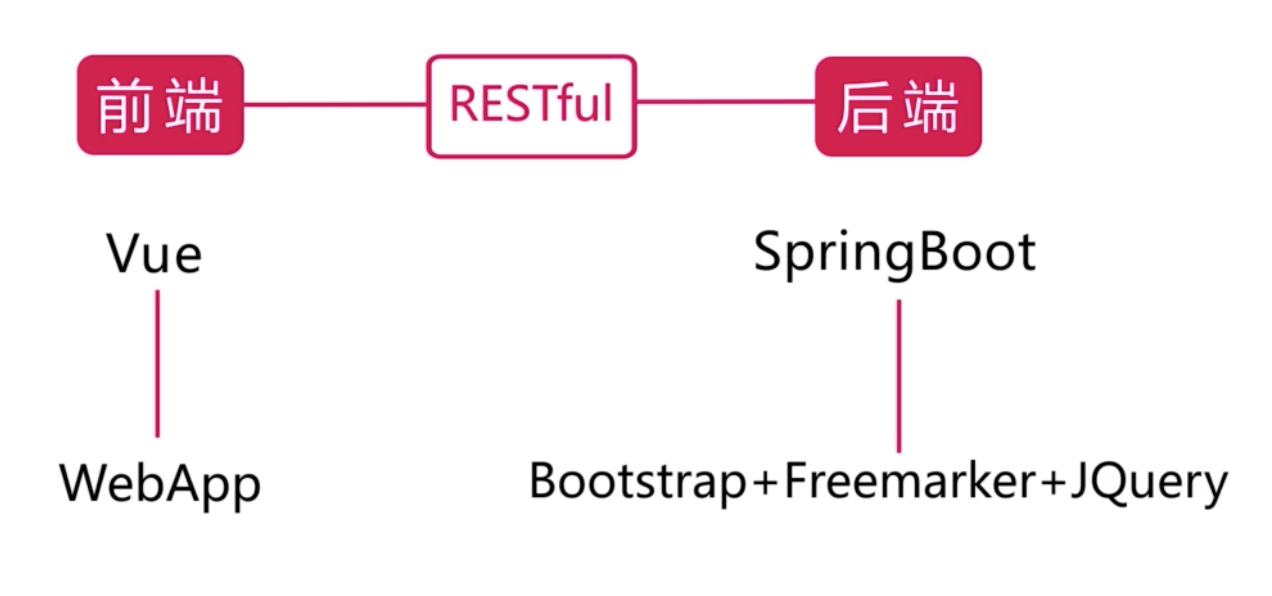


图3.1 前后端分离结构图

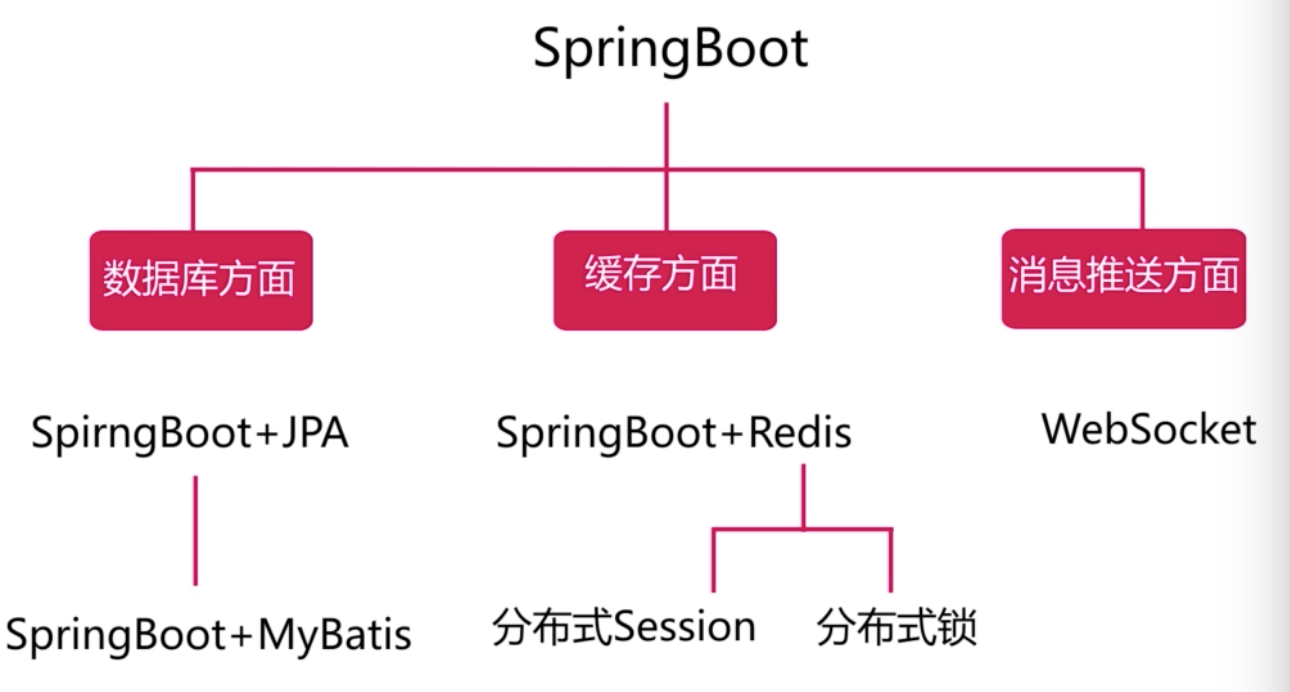


图3.2 SpringBoot功能图

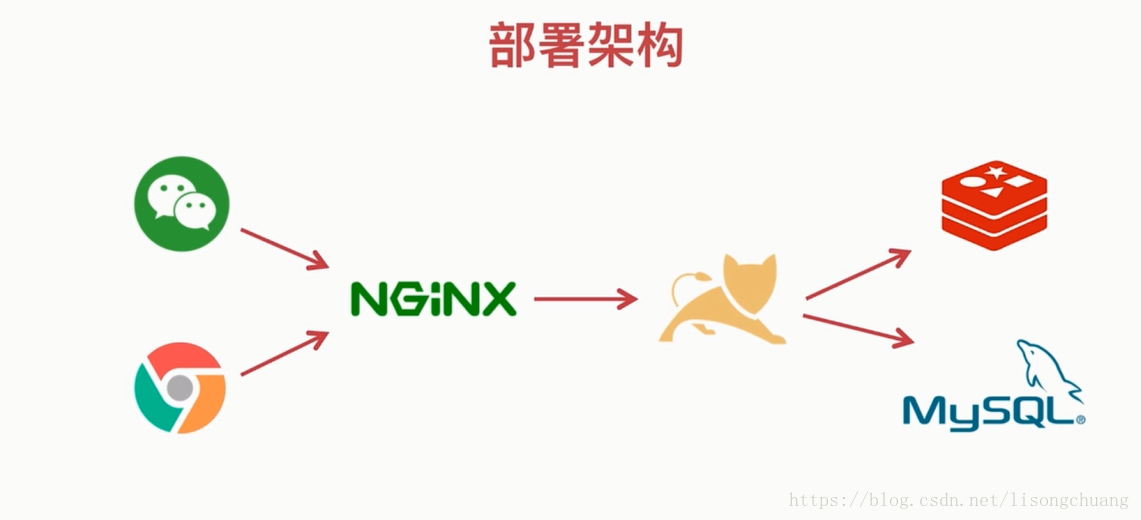


图3.3 部署架构图

3.2 功能结构设计

根据以上分析和设计的结果，从功能上可以可以分成前台功能结构图和后台功能结构图。

3.2.1 前台功能结构图

前台功能结构图如图3.4所示

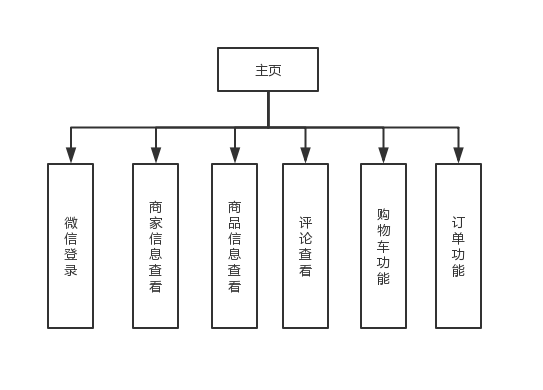


图3.4 前台功能结构图

3.2.2 后台功能结构图

后台功能结构图如图3.5所示：

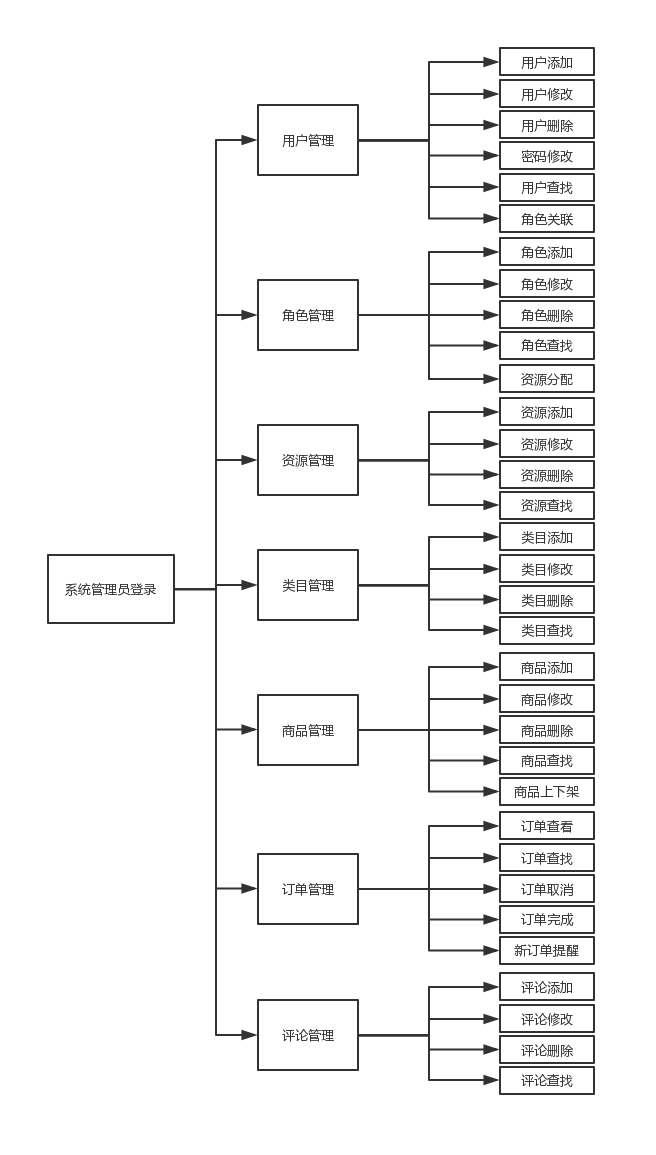


图3.5 后台功能结构图

3.3 接口设计

3.3.1 外部接口

3.3.1.1 用户界面接口

本系统无论是前端还是后端，都并不是给开发者而是提供给用户使用的，所以美观、简洁、实用以及易操作的UI界面是很有必要的,本着这个原则，我设计了以下各个模块的页面：

* **前台**
* 主页/商品浏览：如图3.5所示



图3.5 系统前端主页图

* 订单列表：如图3.6所示



图3.6 系统前端订单列表图

* 购物车：如图3.7所示



图3.7 系统前端购物车图

* 评论列表：如图3.8所示



图3.8 系统前端评论列表图

* 评论界面：如图3.9所示



图3.9 系统前端评论界面图

* 商家信息：如图3.10所示



图3.10 系统前端商家信息图

* 订单界面：如图3.11、3.12所示



图3.11 订单填写界面图



图3.12 订单界面图

* **后台**
* 登录界面：如图3.13所示

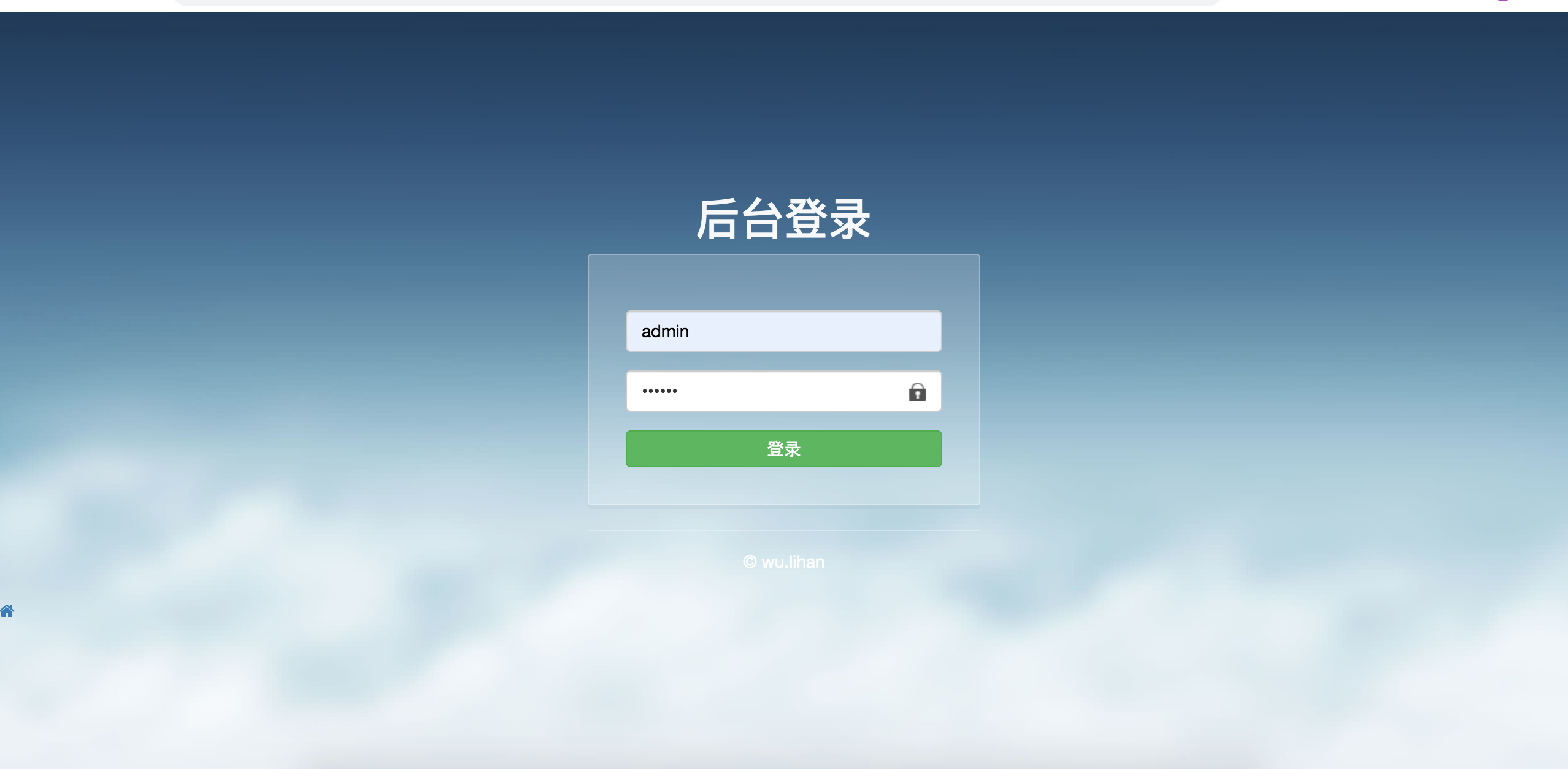


图3.13 后台登录界面图

* 用户管理：如图3.14所示

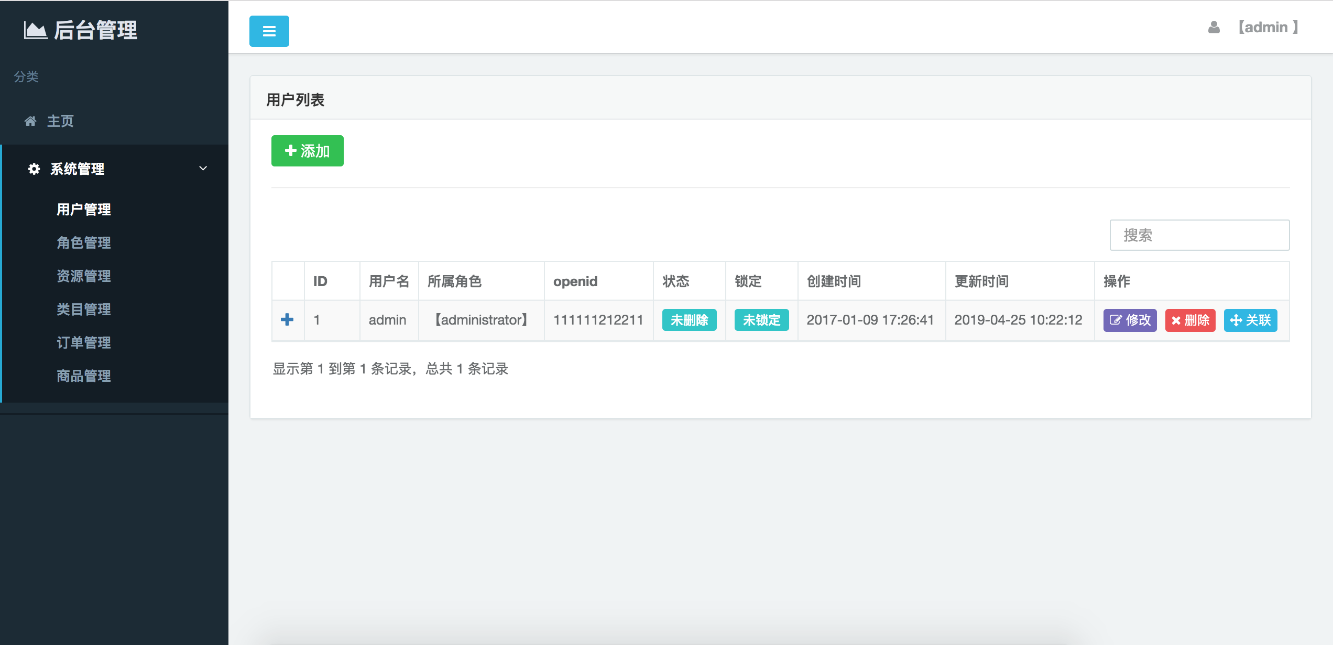


图3.14 用户管理界面图

* 角色管理：如图3.15所示

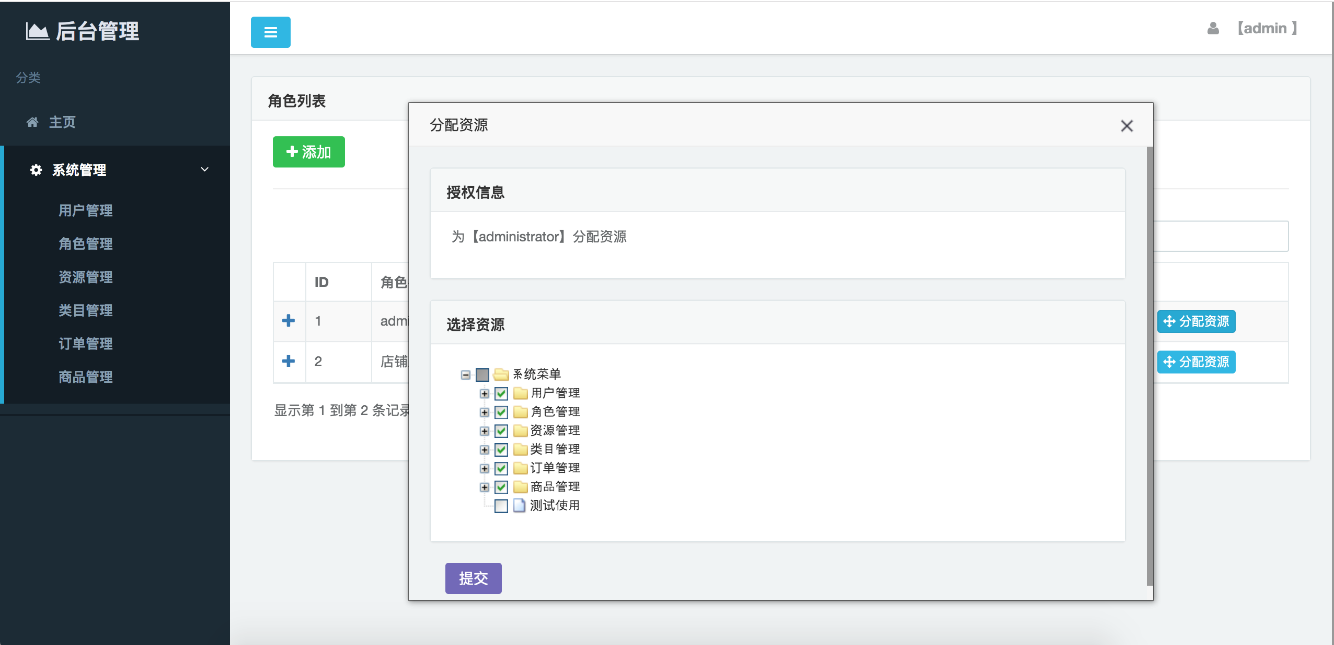


图3.15 角色管理界面图

* 资源管理：如图3.16所示

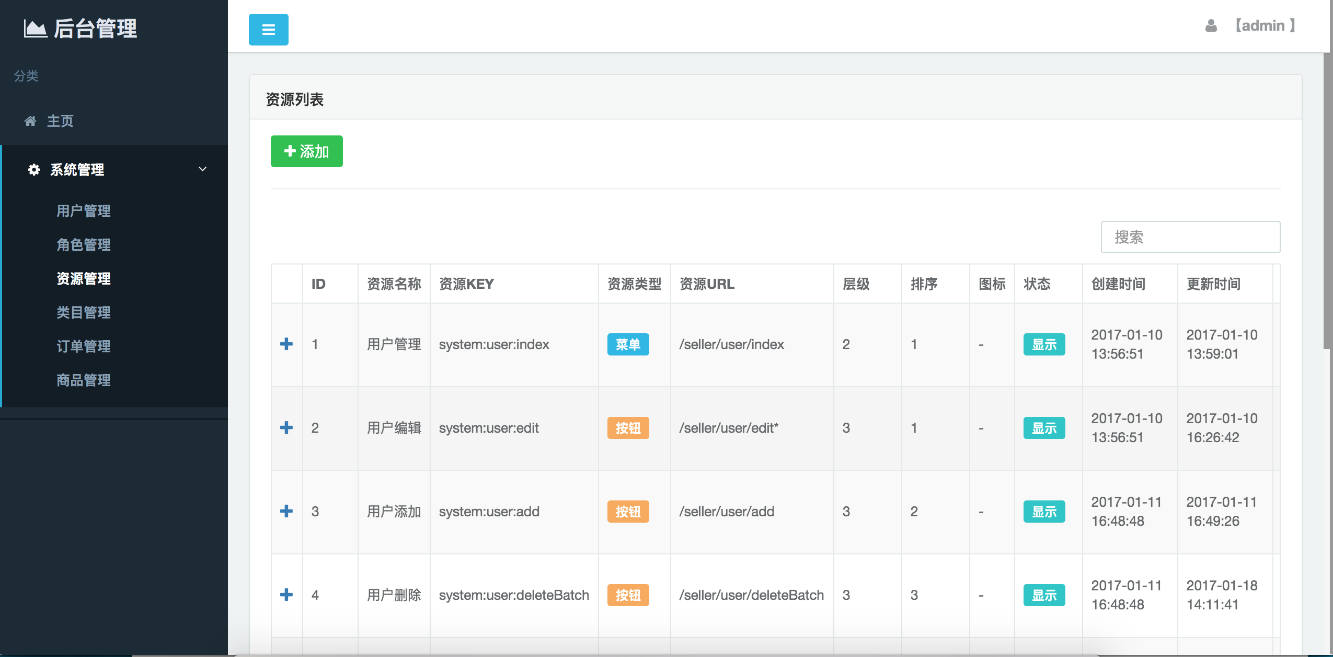


图3.16 资源管理界面图

* 类目管理：如图3.17所示

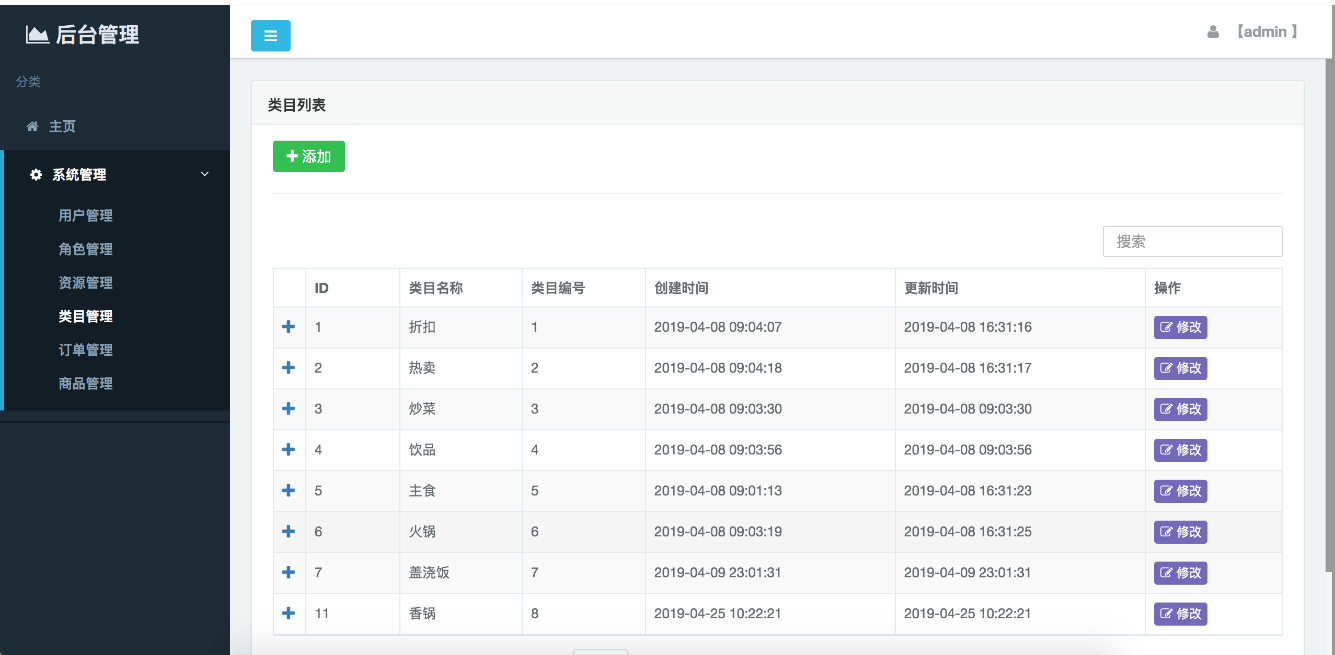


图3.17 类目管理界面图

* 订单管理：如图3.18所示

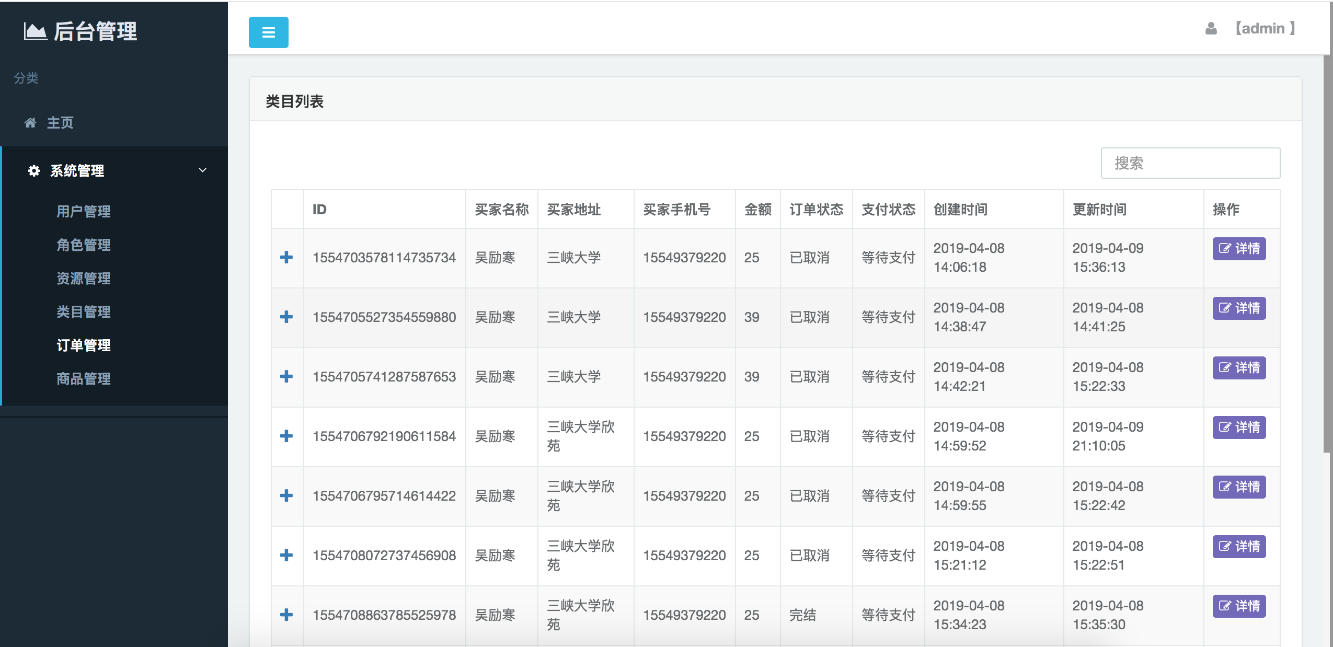


图3.18 订单管理界面图

* 商品管理：如图3.19所示

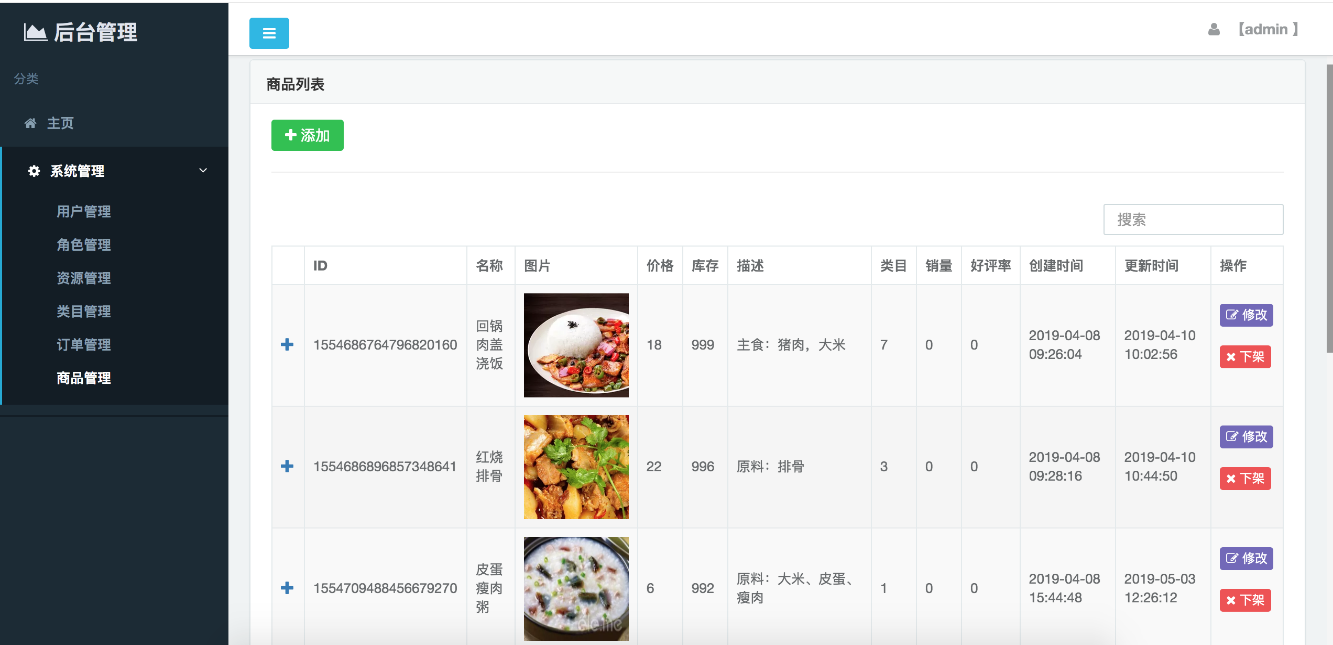


图3.19 商品管理界面图

3.3.1.2 访问接口

3.3.1.2.1 访问接口框图

前后端交互以及访问的接口，定义于controller文件夹下，如图3.20所示：

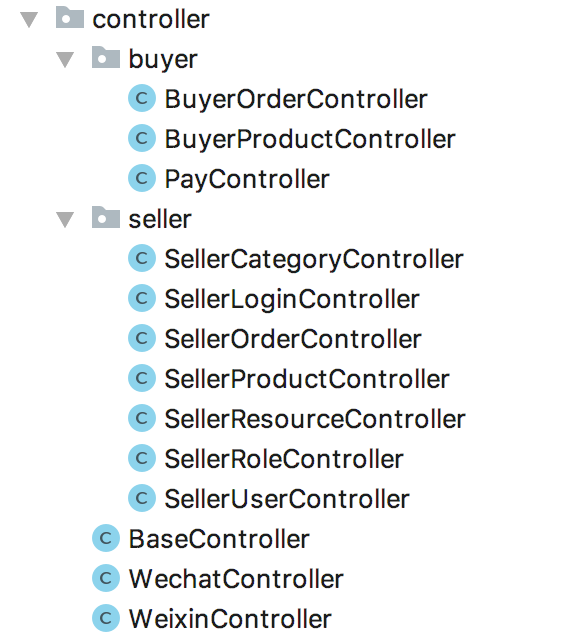


图3.20 访问接口框图

3.3.1.2.2 访问接口详情

访问接口详情如表3.21所示：

表 3.21 访问接口详情表

|  |  |  |  |
| --- | --- | --- | --- |
| 类名 | url前缀 | 功能 | 描述 |
| BaseController | 无 | 控制器的公共基类，定义了分页请求方法用以获取请求分页参数 | 无 |
| SellerLoginController | /seller | 后台登录控制器 | 包含登录接口以及后台未登录时的默认访问接口 |
| SellerCategoryController | /seller/category | 类目控制器 | 类目控制的所有操作 |
| SellerOrderController | /seller/order | 卖家端订单控制器 | 卖家端查询订单、完成订单、取消订单 |
| SellerProductController | /seller/product | 卖家端商品控制器 | 卖家端增删改、上下架商品 |
| SellerResourceController | /seller/resource | 资源控制器 | 资源控制的所有操作 |
| SellerRoleController | /seller/role | 角色控制器 | 角色控制的所有操作 |
| SellerUserController | /seller/user | 用户控制器 | 用户控制的所有操作 |
| BuyerOrderController | /buyer/order | 买家端订单控制器 | 买家端查询订单、取消订单、创建订单 |
| BuyerProductController | /buyer/order | 买家端商品控制器 | 买家端查询商品 |
| WechatController | /wechat | 微信开放平台控制器 | 调用微信开放平台接口进行微信支付授权、获取微信用户信息 |
| WeixinController | /weixn | 微信公众平台控制器 | 调用微信公众平台接口进行微信登录授权 |
| PayController | /pay | 微信支付控制器 | 创建微信支付订单、异步发送支付消息 |

3.3.1.3 mysql数据库接口

本系统主要使用Mysql数据库来进行数据存储，Spring Data JPA的方式来进行数据库访问。 Mysql是最常用的语言标准化数据库，mysql软件既适用于社区分离版本，也适用于商业策略版本，而且由于其规模小、速度快、成本较低，特别是相比于oracle它具有开放代码的这一特性，通常在网站开发中使用。除了Spring Data JPA默认的一些接口外，下面将展示各子模块创建的mysql数据库接口：

* 后台用户(User)模块：

1. **函数名**：findByUsername

**功 能：**在用户登录时以及用户主动更新密码时，会调用该方法查询用户信息，然后比对是否密码正确

**参 数**：username:用户名

**返 回**：User

1. **函数名**：findAllByUsernameContaining

**功 能：**根据部分用户名分页获取所有的用户信息

**参 数**：searchText：部分用户名

Pageable：分页信息

**返 回**：Page<User>

1. **函数名**：findByOpenid

**功 能：**根据用户微信openid分页获取所有匹配的用户

**参 数**：openid：微信openid

**返 回**：User

* 后台角色(Role)模块

1. **函数名**：findAllByNameContainingOrDescriptionContaining

**功 能：**根据角色名称或者描述分页获取所有匹配的角色

**参 数**：username:用户名

**返 回**：searchText1：查询文本

searchText2：查询文本

Pageable：分页信息

1. **函数名**：findById

**功 能：**根据角色id获取角色信息，用以显示权限树

**参 数**：id：角色id

**返 回**：Role

* 后台资源(Resource)模块

1. **函数名**：deleteGrant

**功 能：**删除对指定资源的授权功能

**参 数**：id:资源id

**返 回**：void

1. **函数名**：findAllByNameContaining

**功 能：**根据部分资源名分页获取所有的资源信息

**参 数**：searchText：部分用户名

Pageable：分页信息

**返 回**：Page<User>

1. **函数名**：findAllByOrderByParentAscIdAscSortAsc

**功 能：**获取所有资源信息按照parentid、sort升序，用以显示权限树

**参 数**：无

**返 回**：List<Resource>

* 订单(OrderMaster)模块

1. **函数名**：findByBuyerOpenidOrderByUpdateTimeDesc

**功 能：**根据买家用户的微信openid获取用户订单列表并按照更新时间倒序排序

**参 数**：buyerOpenid：买家微信openid

Pageable：分页信息

**返 回**：Page<OrderMaster>

1. **函数名**：findByOrderId

**功 能：**根据订单id获取订单信息

**参 数**：orderid：订单id

**返 回**：OrderMaster

1. **函数名**：findAllByBuyerNameContaining

**功 能：**根据买家用户名获取用户订单列表

**参 数**：searchText：部分用户名

Pageable：分页信息

**返 回**：Page<OrderMaster>

1. **函数名**：findByBuyerOpenidAndRatingStatusAndOrderIdInOrderByUpdateTimeDesc

**功 能：**根据买家用户的微信openid以及订单id列表获取相应评价状态的用户订单列表并按照更新时间倒序排序

**参 数**：buyerOpenid：买家微信openid

ratingStatus：评论状态，0未评价，1已评价

Pageable：分页信息

**返 回**：Page<OrderMaster>

* 订单商品(OrderDetail)模块

1. **函数名**：findByOrderId

**功 能：**根据订单id获取订单的所有商品信息

**参 数**：orderid：订单id

**返 回**：OrderDetail

1. **函数名**：findByProductId

**功 能：**根据商品id获取所有包含该商品的订单信息

**参 数**：productId：商品id

**返 回**：List<OrderDetail>

* 类目(ProductCategory)模块

1. **函数名**：findByCategoryTypeIn

**功 能：**根据商品类目的type字段查找满足条件的类目信息

**参 数**：categoryTypeList：要查询的类目的type列表

**返 回**：List<ProductCategory>

1. **函数名**：findByCategoryId

**功 能：**根据商品类目id查找满足条件的类目信息

**参 数**：categoryId：要查询的类目id

**返 回**：ProductCategory

1. **函数名**：findAllByCategoryNameContaining

**功 能：**根据买家用户名获取用户订单列表

**参 数**：searchText：部分用户名

Pageable：分页信息

**返 回**：Page<ProductCategory>

* 商品(ProductInfo)模块

1. **函数名**：findByProductStatus

**功 能：**根据商品状态筛选商品

**参 数**：productStatus： 商品状态

**返 回**：List<ProductInfo>

1. **函数名**：findByProductId

**功 能：**根据商品id查找满足条件的商品信息

**参 数**productId：要查询的商品id

**返 回**：ProductInfo

1. **函数名**：findAllByProductNameContaining

**功 能：**根据商品名获取商品列表

**参 数**：searchText：部分商品名

Pageable：分页信息

**返 回**：Page<ProductInfo>

3.3.2 内部接口

3.3.2.1 内部接口框图

内部接口框图如图3.22所示：

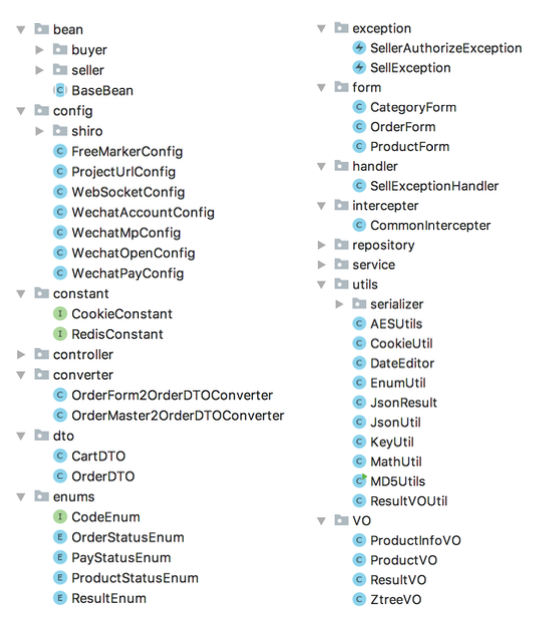


图3.22 内部接口框图

3.3.2.2 内部接口说明

1. **文件夹名**：bean

**功能**：此文件夹下主要存放数据实体及持久化对象,和数据库字段一一对应，对应MVC模式中的model,包含User、Role、Resource、OrderDetail、OrderMaster、ProductCategory、ProductInfo。

1. **文件夹名**：config

**功能**：此文件夹下主要存放配置信息，包含shiro配置（登录拦截）、freemaker配置、url配置、websocket配置、商户配置、公众平台配置、微信支付配置以及开放平台配置。

1. **文件夹名**：constant

**功能**：此文件夹下主要存放项目中的一些常量信息，包含cookie常量信息、redis常量信息。

1. **文件夹名**：converter

**功能**：此文件夹下对象主要用于对象转换，将前端传来的订单表格对象转化成订单数据传输对象、订单商品表格对象转化成订单商品数据传输对象，便于后端进行处理。

1. **文件夹名**：dto

**功能**：此文件夹下存放的是数据传输对象，用于前后端交互即展示层和服务层交互时的数据存储。包含订单数据传输对象dto、购物车数据传输对象dto。

1. **文件夹名**：enums

**功能**：此文件夹下存放的是枚举对象。包含订单状态枚举、支付状态枚举、商品状态枚举以及系统处理结果枚举。

1. **文件夹名**：exception

**功能**：用以拦截系统异常，加入异常信息提示。

1. **文件夹名**：form

**功能**：此文件夹下对象为表格对象，对应前端、后端的各项表格数据。包含订单表格对象、类目表格对象、商品表格对象。

1. **文件夹名**：handler

**功能**：此文件夹下对象用于拦截系统登录异常并进行处理，重定向到登录界面。

1. **文件夹名**：interceptor

**功能**：系统公共拦截器，在每次处理返回时设置属性信息。

1. **文件夹名**：utils

**功能**：此文件夹下对象为系统常用的工具类。例如日期转化工具类、枚举工具类、统一返回消息类、Json工具类、关键字工具类、加密工具类等等。

1. **文件夹名**：VO

**功能**：此文件夹下对象为视图对象，用于展示层，封装的是页面数据。包含商品视图对象、商品详情视图对象、http处理结果视图对象以及ztree树视图对象(用以展示权限树)。

3.4 数据库结构设计

3.4.1 用户表

用户表如表3.23所示：

表3.23 用户表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名字 | 类型 | 空 | 默认 | 注释 | 自增 |
| id | int(11) | 否 | 无 | 用户唯一id | 是 |
| username | varchar(255) | 是 | NULL | 用户名 | 否 |
| password | varchar(255) | 是 | NULL | md5加密后的密码 | 否 |
| openid | varchar(64) | 否 | 无 | 微信openid | 否 |
| description | varchar(255) | 是 | NULL | 用户描述 | 否 |
| locked | int(11) | 是 | NULL | 是否锁定:0未锁定,1锁定 | 否 |
| delete\_status | int(11) | 是 | NULL | 逻辑删除状态:0未删除,1删除 | 否 |
| create\_time | datetime | 是 | NULL | 创建时间 | 否 |
| update\_time | datetime | 是 | NULL | 更新时间 | 否 |

3.4.2 角色表

角色表如表3.24所示：

表3.24 角色表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名字 | 类型 | 空 | 默认 | 注释 | 自增 |
| id | int(11) | 否 | 无 | 角色唯一id | 是 |
| create\_time | datetime | 是 | NULL | 创建时间 | 否 |
| description | varchar(255) | 是 | NULL | 角色描述 | 否 |
| name | varchar(255) | 是 | NULL | 角色名称 | 否 |
| role\_key | varchar(255) | 是 | NULL | 角色唯一标识 | 否 |
| status | int(11) | 是 | NULL | 角色状态:0正常,1:删除 | 否 |
| update\_time | datetime | 是 | NULL | 更新时间 | 否 |

3.4.3 资源表

资源表如表3.25所示：

表3.25 资源表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名字 | 类型 | 空 | 默认 | 注释 | 自增 |
| id | int(11) | 否 | 无 | 资源表唯一id | 是 |
| description | varchar(255) | 是 | NULL | 资源描述 | 否 |
| icon | varchar(255) | 是 | NULL | 资源图标 | 否 |
| is\_hide | int(11) | 是 | NULL | 是否隐藏:0显示,1隐藏 | 否 |
| level | int(11) | 是 | NULL | 层级 | 否 |
| name | varchar(255) | 是 | NULL | 资源名称 | 否 |
| sort | int(11) | 是 | NULL | 排序 | 否 |
| source\_key | varchar(255) | 是 | NULL | 资源唯一标识 | 否 |
| source\_url | varchar(255) | 是 | NULL | 资源url | 否 |
| type | int(11) | 是 | NULL | 资源类型:0目录,1菜单,2按钮 | 否 |
| parent\_id | int(11) | 是 | NULL | 父资源id | 否 |
| create\_time | datetime | 是 | NULL | 创建时间 | 否 |
| update\_time | datetime | 是 | NULL | 更新时间 | 否 |

3.4.4 订单表

订单表如表3.26所示：

表3.26 订单表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名字 | 类型 | 空 | 默认 | 注释 | 自增 |
| id | int(11) | 否 | 无 | 行唯一标识 | 是 |
| order\_id | varchar(32) | 否 | 无 | 订单表唯一id | 否 |
| buyer\_name | varchar(32) | 否 | 无 | 买家名字 | 否 |
| buyer\_phone | varchar(32) | 否 | 无 | 买家电话 | 否 |
| buyer\_address | varchar(128) | 否 | 无 | 买家地址 | 否 |
| buyer\_openid | varchar(64) | 否 | 无 | 买家微信openid | 否 |
| order\_amount | decimal(8,2) | 否 | 无 | 订单总金额 | 否 |
| order\_status | tinyint(3) | 否 | 0 | 订单状态:默认为0新下单,1完结,2取消 | 否 |
| pay\_status | tinyint(3) | 否 | 0 | 支付状态: 默认为0未支付,1已支付 | 否 |
| rating\_status | tinyint(3) | 否 | 0 | 评价状态, 默认为0未评价,1已评价 | 否 |
| rating\_detail | varchar(255) | 是 | 无 | 评价 |  |
| create\_time | timestamp | 否 | CURRENT\_TIMESTAMP | 创建时间 | 否 |
| update\_time | timestamp | 否 | CURRENT\_TIMESTAMP | 修改时间 | 否 |

3.4.5 订单详情表

订单详情表如表3.27所示：

表3.27 订单详情表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名字 | 类型 | 空 | 默认 | 注释 | 自增 |
| id | int(11) | 否 | 无 | 行唯一标识 | 是 |
| detail\_id | varchar(32) | 否 | 无 | 订单详情唯一id | 否 |
| order\_id | varchar(32) | 否 | 无 | 订单id | 否 |
| product\_id | varchar(32) | 否 | 无 | 商品id | 否 |
| product\_name | varchar(64) | 否 | 无 | 商品名称 | 否 |
| product\_price | decimal(8,2) | 否 | 无 | 当前价格,单位分 | 否 |
| product\_quantity | int(11) | 否 | 无 | 数量 | 否 |
| product\_icon | varchar(512) | 是 | NULL | 小图 | 否 |
| product\_score | int(211) | 否 | 0 | 评价分数 | 否 |
| create\_time | timestamp | 否 | CURRENT\_TIMESTAMP | 创建时间 | 否 |
| update\_time | timestamp | 否 | CURRENT\_TIMESTAMP | 更新时间 | 否 |

3.4.6 商品信息表

商品信息表如表3.28所示：

表3.28 商品信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名字 | 类型 | 空 | 默认 | 注释 | 自增 |
| id | int(11) | 否 | 无 | 行唯一标识 | 是 |
| product\_id | varchar(32) | 否 | 无 | 商品唯一id | 否 |
| product\_name | varchar(64) | 否 | 无 | 商品名称 | 否 |
| product\_price | decimal(8,2) | 否 | 无 | 单价 | 否 |
| product\_stock | int(11) | 否 | 无 | 库存 | 否 |
| product\_description | varchar(64) | 是 | NULL | 描述 | 否 |
| product\_icon | varchar(512) | 是 | NULL | 小图 | 否 |
| product\_status | tinyint(3) | 是 | 0 | 商品状态:0正常1下架 | 否 |
| category\_type | int(11) | 否 | 无 | 类目编号 | 否 |
| create\_time | timestamp | 否 | CURRENT\_TIMESTAMP | 创建时间 | 否 |
| update\_time | timestamp | 否 | CURRENT\_TIMESTAMP | 修改时间 | 否 |
| sell\_count | decimal(8,2) | 否 | 0 | 销量 | 否 |
| rating | float | 否 | 0 | 好评率 | 否 |

3.4.7 商品类目表

商品类目表如表3.29所示：

表3.29 商品类目表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名字 | 类型 | 空 | 默认 | 注释 | 自增 |
| id | int(11) | 否 | 无 | 行唯一标识 | 是 |
| category\_id | int(11) | 否 | 无 | 类目唯一id | 否 |
| category\_name | varchar(64) | 否 | 无 | 类目名称 | 否 |
| category\_type | int(11) | 否 | 无 | 类目类型 | 否 |
| create\_time | timestamp | 否 | CURRENT\_TIMESTAMP | 创建时间 | 否 |
| update\_time | timestamp | 否 | CURRENT\_TIMESTAMP | 更新时间 | 否 |

3.4.8 用户角色关系表

用户角色关系表如表3.30所示：

表3.30 用户角色关系表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名字 | 类型 | 空 | 默认 | 注释 | 自增 |
| user\_id | int(11) | 否 | 无 | 用户id | 否 |
| role\_id | int(11) | 否 | 无 | 角色id | 否 |

3.4.9 角色资源关系表

角色资源关系表如表3.31所示：

表3.31 角色资源关系表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名字 | 类型 | 空 | 默认 | 注释 | 自增 |
| role\_id | int(11) | 否 | 无 | 角色id | 否 |
| resource\_id | int(11) | 否 | 无 | 资源id | 否 |

4 系统测试

4.1 测试方法

前后端分别模拟进行各项功能测试，每一步测试如果出现问题则需要修改系统重新进行再一次测试，后端对不同角色、不同权限用户的各项功能使用进行测试，前端主要针对界面以及和后端交互功能的测试，目标是力争不存在功能性问题，如果出现偶发性事件，出现概率在1%以下则可以忽略。

4.2 测试用例

* 后台测试

分别定义店主(所有权限)、服务员(登录功能、商品管理、商家管理、评论管理、类目管理)、前台(登录功能、订单管理)、游客(无权限)

1. 安全性测试：游客不可以登录，前台登录后只能看到订单管理，服务员登陆后看不到用户管理、角色管理以及资源管理，而店主登陆后可以看到所有功能模块，且登录时效30分钟，30分钟内访问网站无须再次登录。
2. 管理功能测试：商品管理、商家管理、评论管理、类目管理、用户管理、角色管理、资源管理均可进行正常的操作，分页显示、数据查询均未发现bug。

* 前端测试

1. 布局测试：后台对类目、商品、评论以及商家信息做任何修改，使用微信扫码访问，前端展示界面均正常显示。
2. 订单测试：使用微信进行下单，后端能及时收到新订单通知，后端修改订单状态后前端订单状态也会及时改变。

4.3 测试结论

通过上述测试，本系统已经满足设计的要求，达到预期目标，可以正常发布到服务器上进行使用，至于在高并发、海量数据的情况下是否能很好的工作还需要进一步进行测试。

5 结论与总结

本次毕业设计，时间长达三个月之久，从前期资料收集、技术储备，到中期设计思路完备、目标明确，再到后期编码完成、代码优化，一直到最后的论文整理、编写，这一路困难重重，自己踩的坑更是多不胜数，不管是初次接触的vue框架、shiro框架，还是已经学习过的springboot、springmvc、mysql等技术，或多或少地都犯了些错误，我深深体会到完成一个完成的项目的艰辛与困难，当然，也充分锻炼了我的思考能力、开发能力，进一步拓宽了我的技术栈。

在整个毕业设计的过程中，我学会了如何在最短时间内最有效地收集最有用的资料，提高了自己的系统分析能力、设计能力、分析能力、编码能力和测试能力。由于我所设计的系统是微信点餐系统，为了更好地了解功能需求，我体验了很多相似功能系统，例如饿了么、美团、美味不用等、二维火点餐等等一些点餐软件，虽然耗费了不少时间，但是相比纸上谈兵而言事半功倍。当然我也认识到了实力的重要性，自己还有许多地方需要提高、值得提高，在今后的开发生涯中，自己也会持续不断学习。

总而言之，本次毕业设计于我而言是一个很有学习价值、纪念意义的课程，让我收获了很多大学四年没学到的知识。

致 谢

在本文的设计过程中，之所以我能顺利完成，多亏了我的导师彭代军老师的耐心指导，给了我这次学习的机会，从前期的毕设选题、开题报告撰写、任务安排以及后面的编码、论文撰写，给了我详细的指导方针以及很多有建设性的意见和有价值的建议，老师对教育有严格的态度，所以我对他格外尊敬；当然，我的室友和同学，我们互相依靠，当他们遇到问题时主动和我讨论，我遇到问题时主动寻求帮助，是我可以和大家一起的努力让我顺利完成毕业设计。

参考文献

[1] 陈强.《精通Java开发技术》.清华大学出版社.2016

[2] 黄俊.《Java程序设计与应用开发》第2版.机械工业出版社.2016

[3] 李刚.《轻量级 Java EE企业应用实战》第5版. 电子工业出版社出版.2017

[4] 杨开振.《Java EE互联网轻量级框架整合开发》.电子工业出版社.2017

[5] 张剑飞.《Java EE开发技术》.哈尔滨工业大学出版社.2015

[6] 疯狂软件.《Spring+MyBatis企业应用实战》.电子工业出版社，2017

[7] 李燕飞.《MVC模式在J2EE中的应用》，电脑知识与技术，2017

[8] 向昌成.Java程序设计项目化教程.清华大学大学出版社.2016

[9] 林信良.《JSP & Servlet学习笔记》， 清华大学出版社, 2015年05月.

[10] Metsker SJ.《Java设计模式》.第二版.电子工业出版社.2015

[11] 魏艳《微信公众号深度解析》.第一版. 化学工业出版社.2017

[12] 刘勇军.《JavaWeb核心编程技术》.电子工业出版社.2015

[13] 林学良.《JSP&Servlet学习笔记》.清华大学出版社.2014

[14] Buell.Datastructures using Java. Jones&Bartlett Learning.2014.

[15] Martin NgobyeComputing Static Slice for Java Programs.2014

[16] Rachit Mohan Garg, YaminiSood, Balaji Kottana, Pallavi Totlani.AFramework Based Approach for the Development of Web Based Applications Waknaghat[J].Jaypee University of Information Technology,2014.

附 录

1. spring配置文件application.properties

**spring.jpa.hibernate.naming-strategy** = **org.hibernate.cfg.DefaultNamingStrategy***#避免分页查找时数据为空序列化失败，去掉后分页数据不够时会报错***spring.jackson.serialization.*FAIL\_ON\_EMPTY\_BEANS***=**false  
spring.jpa.show-sql**=**true**

2. 全局配置文件application.yml

**spring**:  
 **datasource**:  
 **driver-class-name**: com.mysql.jdbc.Driver  
 **username**: root  
 **password**: 123456  
 **url**: jdbc:mysql://127.0.0.1:3306/sell?characterEncoding=utf-8&useSSL=false&&autoReconnect=true  
 **jpa**:  
 **show-sql**: **true  
 jackson**:  
 **default-property-inclusion**: *non\_null* **redis**:  
 **host**: 127.0.0.1  
 **port**: 6379  
  
**wechat**:  
 **mpAppId**: wxc4b2697181351922  
 **mpAppSecret**: bf536aa94da1dff7d3a3583e098a9586  
 **openAppId**: wx6ad144e54af67d87  
 **openAppSecret**: 91a2ff6d38a2bbccfb7e9f9079108e2e  
 **mchId**: 1483469312  
 **mchKey**: 06C56A89949D617xxxxxxxxxxx  
 **keyPath**: /var/weixin\_cert/h5.p12  
 **notifyUrl**: http://ctguqmx.natapp1.cc/sell/pay/notify  
 **templateId**:  
 **orderStatus**: e-Cqq67QxD6YNI41iRiqawEYdFavW\_7pc7LyEMb-yeQ  
  
**url**:  
 **wechatMpAuthorize**: http://ctguqmx.natapp1.cc  
 **wechatOpenAuthorize**: http://ctguqmx.natapp1.cc  
 **sell**: http://ctguqmx.natapp1.cc  
  
**server**:  
 **context-path**: /sell  
 **port**: 9080

3. 日志配置文件logback-spring.xml

*<?***xml version="1.0" encoding="UTF-8"** *?>*<**configuration**>  
  
 <**appender name="consoleLog" class="ch.qos.logback.core.ConsoleAppender"**>  
 <**layout class="ch.qos.logback.classic.PatternLayout"**>  
 <**pattern**>  
 %d - %msg%n  
 </**pattern**>  
 </**layout**>  
 </**appender**>  
  
 <**appender name="fileInfoLog" class="ch.qos.logback.core.rolling.RollingFileAppender"**>  
 <**filter class="ch.qos.logback.classic.filter.LevelFilter"**>  
 <**level**>ERROR</**level**>  
 <**onMatch**>DENY</**onMatch**>  
 <**onMismatch**>ACCEPT</**onMismatch**>  
 </**filter**>  
 <**encoder**>  
 <**pattern**>  
 %msg%n  
 </**pattern**>  
 </**encoder**>  
 *<!--滚动策略-->* <**rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy"**>  
 *<!--路径-->* <**fileNamePattern**>~/log/sell/info.%d.log</**fileNamePattern**>  
 </**rollingPolicy**>  
 </**appender**>  
 <**appender name="fileErrorLog" class="ch.qos.logback.core.rolling.RollingFileAppender"**>  
 <**filter class="ch.qos.logback.classic.filter.ThresholdFilter"**>  
 <**level**>ERROR</**level**>  
 </**filter**>  
 <**encoder**>  
 <**pattern**>  
 %msg%n  
 </**pattern**>  
 </**encoder**>  
 *<!--滚动策略-->* <**rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy"**>  
 *<!--路径-->* <**fileNamePattern**>~/log/sell/error.%d.log</**fileNamePattern**>  
 </**rollingPolicy**>  
 </**appender**>  
 <**root level="debug"**>  
 <**appender-ref ref="consoleLog"** />  
 <**appender-ref ref="fileInfoLog"** />  
 <**appender-ref ref="fileErrorLog"** />  
 </**root**>  
  
</**configuration**>