届 别 2022届

学 号 201814160116





毕业设计（论文）

姓　　 名 谭佳利

系 别、 专 业 计算机与人工智能学院

网络工程

导 师 姓 名、职 称 张霞 副教授

完 成 时 间 2022年05月10日

目录

[摘　　要 3](#_Toc24015)

[ABSTRACT 4](#_Toc15033)

[第一章 概述 5](#_Toc17961)

[1.1. 研究背景与意义 5](#_Toc22658)

[1.2. 研究现状 5](#_Toc29165)

[1.3. 研究的主要内容 6](#_Toc10492)

[第二章 关键技术介绍 7](#_Toc23502)

[2.1. 后端技术介绍 7](#_Toc21894)

[2.2. 前端技术运用 9](#_Toc9031)

[2.3. 系统架构设计模型 9](#_Toc29950)

[第三章 系统分析 10](#_Toc19417)

[3.1. 系统可行性分析 10](#_Toc11381)

[3.2. 需求分析 11](#_Toc16262)

[3.3. 非功能性需求分析 12](#_Toc30161)

[3.4. 数据库分析 13](#_Toc10188)

[第四章 系统总体设计 13](#_Toc27656)

[4.1. 系统架构设计 13](#_Toc31352)

[4.2. 系统模块总体设计 14](#_Toc16557)

[4.3. 系统功能使用流程图 16](#_Toc20657)

[4.4. 数据库设计 18](#_Toc4905)

[第五章 系统详细设计 23](#_Toc25828)

[5.1. 系统开发环境 23](#_Toc4668)

[5.2. 主要功能模块实现 23](#_Toc6348)

[第六章 系统测试 43](#_Toc14018)

[6.1. 测试目的和分类 43](#_Toc28573)

[6.2. 主要核心功能的测试 44](#_Toc21956)

[第七章 结论 51](#_Toc22104)

[致谢 51](#_Toc19440)

[参考文献 52](#_Toc5085)

# 摘　　要

当下，网络购物体系已然日益庞大，越来越多的商家在网上建起商店，向消费者展示出一片新鲜的购物大地。市场中也不乏缺少一些各式各样的在线商城系统。但良好的购物环境往往只方便了消费者一方，大都没有考虑到商户的感受，传统的在线商城系统的管理端，往往只有简单的一些商品上架下架，设置商品信息等基础功能，这对管理商品十分不便，使之不得不人为维护商品。为此本论文设计了一个以商城可视化管理工具为主体的微商城系统，从而解决传统商城管理端功能不全，运营困难，统计不便所造成的人力资源白白浪费。

本系统采用前后端分离开发技术，利用Idea开发工具，结合MySQL数据库，以VUE为平台开发出了一个基于Java的管理系统。该系统包括前台在线商城以及后台可视化管理工具两个基本模块。综合运用Java Web开发的各项技术，深入研究使用Spring Boot、MyBatis和进行企业级开发的基本方法，设计实现了商城管理系统。

系统首先实现统计功能。对商城的日订单、月订单进行统计，经计算展示其折线图，更方便观察商城营业亏损状态。对商品的库存和上架下架数量进行统计展示，方便商户进行管理。对用户注册量进行统计，方便业务扩展。对订单进行状态跟踪，实时获取订单的交易状态，方便商品运维人员处理。其次，实现了商品分类管理，不仅实现商品上线功能，还针对商品分类，商品个体属性、品牌进行精确管理。订单管理功能则是对订单不同状态存在不同的功能，包括订单过期时间的管理，退货订单管理，产品退货原因管理。

关键词：网络购物；商品管理；Visualization tools；产品运维；Spring Boot；

# ABSTRACT

At present, the online shopping system has become increasingly large, and more and more merchants have built stores online to show consumers a fresh shopping land. There is also a lack of a variety of online mall systems in the market. But a good shopping environment is often only convenient for the consumer side, most of them do not take into account the feelings of merchants, the management end of the traditional online mall system, often only a simple number of goods on the shelves, set up basic functions such as commodity information, which is very inconvenient to manage goods, so that it has to artificially maintain goods. To this end, this paper designed a micro-mall system with the mall visual management tool as the main body, so as to solve the waste of human resources caused by the incomplete function of the traditional mall management end, operational difficulties, and statistical inconvenience.

This system adopts the front-end and back-end separation development technology, uses Idea development tools, combines MySQL database, and develops a Java-based management system based on VUE as the platform. The system includes two basic modules: the front-end online mall and the back-office visual management tool. Comprehensively using the various technologies of Java web development, in-depth study of the basic methods of using Spring Boot, MyBatis and enterprise-level development, the design and implementation of the mall management system.

The system first implements the statistical function. The daily orders and monthly orders of the mall are counted, and the line chart is displayed through calculation, which is more convenient to observe the operating loss status of the mall. The inventory of goods and the number of shelves and shelves are displayed statistically to facilitate merchant management. The number of user registrations is counted to facilitate business expansion. Track the status of the order, obtain the transaction status of the order in real time, and facilitate the processing of commodity operation and maintenance personnel. Secondly, the product classification management is realized, which not only realizes the function of commodity on-line, but also accurately manages the commodity classification, individual attributes and brands of commodities. The order management function is a function that has different functions for different statuses of the order, including the management of the order expiration time, the management of the return order, and the management of the reason for the return of the product.

**Keywords**: online shopping；product management；Visualization tools；product operation；maintenance,；Spring Boot；

# 概述

## 研究背景与意义

随着互联网技术的不断发展,特别是电子商务的不断发展,网上购物的需求也在不断的增加。根据《2017-2022年中国电子商务市场运行态势及投资战略研究报告》中指出2015年中国电子商务市场交易额规模16.4万亿元,增长22.7%。而其中网络购物增长36.2%,成为推动电子商务市场发展的重要力量。面对如此大的机遇,电子商务网站运营商必须不断的提高网站的服务质量,虽然商品质量和价钱是顾客很关注的,但是用户在网上购物时的体验也是很重要的。[[[1]](#endnote-0)]

网络购物已经融入到人们的日常生活。不仅在购物上使消费者享受足够的便利,也更好的适应了现代化市场的需求。但凡事有利皆有弊,网络购物在带给广大消费者便利的同时,也伴随着出现了许多问题。商品推荐不合理、商品质量不合格、用户信息安全问题一直得不到有效解决。致使消费者进入电子商城后被大量商品信息干扰,在挑选商品上花费大量时间。甚至到最后还购买了不需要的商品。与此同时,购买的商品质量还可能存在问题。针对此类问题,本文设计和实现了一种电子商城系统。[[[2]](#endnote-1)]

## 研究现状

现如今，我们正处在互联网风口浪尖之上，并非趋于饱和。相反的，网络的发展只会越来越快，人们的生活方式进入到了快时代，便捷已经不是一个新鲜的词汇，但永远不会过时。[电子](http://www.youerw.com/tongxin/" \t "http://www.youerw.com/yanjiu/_blank)商务的流行也是势在必行，消费者完全接受了这种购物方式和思想。对国外等一些信息化水平较高的国家而言，网上购物的方式以疾如雷电的速度发展起来。

调查[研究](http://www.youerw.com/yanjiu/" \t "http://www.youerw.com/yanjiu/_blank)表明，美国人已经越来越习惯于在家中购物了。据《中国商界》杂志报道，美国人[网络](http://www.youerw.com/jisuanji/" \t "http://www.youerw.com/yanjiu/_blank)购物成时尚，美国是世界上电子商务最发达的国家，也拥有全球最大的网上购物市场。美国人推崇时尚，追求效率，这是美国网上购物市场发展迅速的主要原因，确切的说，由于法制上的完善，越来越多消费者对在线购物都很放得下心。而且，美国人讲究实惠，而越来越多的网上商店千方百计不断满足消费者的需要，使网上购物成为一种时尚和乐趣。

当前，英国网上购物方兴未艾，势头正猛。英国几家机构前几年的调查结果显示，同欧洲其他国家和美国相比，英国网上购物者占人口比例最高，购买商品种类最多，购物增长速度最快。

法国人网上购物成习惯，据法国电子商务及远程销售联合会2008年1月24日公布的数据，去年法国网上购物继续保持强劲上涨势头，网上购物总额大幅增加。数据显示，法国去年网上购物交易总额达161亿欧元，增长了35%，而2004年网上购物总额还只有57亿欧元。该联合会预计，到2010年，法国网上购物交易总额将达到314亿欧元 24226。

我国的网络购物的历史不长，自1998年3月6日下午3:20国内的第一笔网上交易成功，中国的网络购物开始出现在人们的视线中。

1999年底，互联网开始沸腾，国内开始有300多家网络公司从事B2C这个行业。2000年，递增至700余家，但是，让人印象深刻的却仅三四家，那是一段漫长的谷底期。直至SARS的出现，人们不敢出门，便开始热爱上了足不出户的购物方式。于是，购物网站开始逐渐的成功上线，从而一步步到了今天。[[[3]](#endnote-2)]

## 研究的主要内容

本论文综合运用Java Web开发的各项技术，深入研究使用Spring Boot、MyBatis和进行企业级开发的基本方法，设计实现了商城管理系统。实现商城的登录注册、购物车、搜索商品、等基本功能。管理端则实现数据处理的功能，包括商品信息设置，用户权限、资源分配，统计，订单管理，消费券管理等。

统计功能：主要是对用户、订单、商品、营业额做统计，展示当月、当天的用户注册量，订单各状态数量，商品上下架数量，销售金额及对比其增长率。

商品管理功能：主要实现商品的添加删除，对商品库存和商品个体属性，分类进行统一管理。

用户授权功能：主要针对后台管理系统用户的授权，更方便的分区块管理后台，其主要内容有：菜单模块的显示管理，用户角色关系管理，资源管理。其中资源管理主要对客户端访问服务器进行路径限制。

订单管理功能：主要是处理订单状态，对订单进行动态管理，包括下单商品未付款时，规定时间后对订单进行关闭的功能。修改订单基本信息等。

营销模块消费券管理:主要是设置在线商城首页展示内容，管理商城减免券功能，设置消费券过期时间，获取方法等。

# 关键技术介绍

## 后端技术介绍

* **Spring Boot：**

Spring Boot可以轻松创建独立的、生产级的、基于Spring应用程序，更简洁的开发框架。

借助这一点，项目初期的开发环境搭建，通过Maven导入Spring Boot相关依赖，快速搭建起商城管理系统后端架构，不但能够保证项目的完整性，还能提高开发效率，减少开发成本。

* **MyBatis|MyBatis Plus：**

MyBatis 是一款优秀的持久层框架，它支持自定义 SQL、存储过程以及高级映射。项目中使用MyBatis取代JDBC 代码以及设置参数和获取结果集的工作。

同Spring一样，项目中完全只靠MyBatis完成数据库的操作，就难以避免繁杂的配置项，虽然可以通过简单的 XML 或注解来配置和映射原始类型、接口和 Java POJO（Plain Old Java Objects）为数据库中的操作记录，但效率依旧堪忧。而国人开发的Plus增强版（MyBatis Plus），又将这个框架再次集成封装，是一个 MyBatis (opens new window)的增强工具，在 MyBatis 的基础上只做增强不做改变，为简化开发、提高效率而生。拥有无侵入、损耗小、强大的 CRUD 操作、支持 Lambda 形式调用、支持主键自动生成、内置代码生成器、内置分页插件、内置性能分析插件、内置全局拦截插件等强大功能。

项目初期对Controller层、Server层、POJO层、Mapper层的代码生成，使用的就是MyBatis Plus工具的逆向工程，自动根据数据库中的表结构进行基本代码的生成。

而项目中比较简单的CRUD则使用MyBatis Plus的数据操作语句实现功能。进一步提高了开发效率。对于表中需要进行逻辑删除或者自动填充的字段，比如创建时间、更新时间、删除记录功能、回收站功能的实现，则利用MyBatis Plus的注解，即可轻松解决。

* **MySQL：**

最流行的关系型数据库管理系统，在 WEB 应用方面 MySQL 是最好的 RDBMS(Relational Database Management System：关系数据库管理系统)应用软件之一,且是一种完全免费的产品。

* **Redis：**

是一个由 Salvatore Sanfilippo 写的 key-value 存储系统，是跨平台的非关系型数据库。Redis 是一个开源的使用 ANSI 、C 语言编写、遵守 BSD 协议、支持网络可选持久性的键值对(Key-Value)存储数据库，并提供多种语言的 API。 Redis 通常被称为数据结构服务器，因为值（value）可以是字符串(String)、哈希(Hash)、列表(list)、集合(sets)和有序集合(sorted sets)等类型。

鉴于Redis优秀的特性，项目中登录后记录账户状态用session的传统做法直接摒弃，利用Redis服务器，进行登录状态缓存，日志记录功能。这不但减轻了数据库MySQL的压力，还减少了登录过程、登录成功时的繁琐存取操作，提高用户体验。

* **JWT登录：**

JWT,即JSON Web Token，是为了在网络应用环境间传递状态、声明而执行的一种基于JSON的开放式标准。项目中使用JWT做登录状态的传输，能保证用户状态在会话过程中的一致性，同时也解决了传统Cookie易篡改不安全的缺点。再者，传统服务端使用session保存用户状态，服务器压力大。使用JWT保存状态，服务端只需要对其加密解密操作即可实现，非常轻便。

* **阿里云对象存储OSS：**

对象存储OSS（Object Storage Service）是一款海量、安全、低成本、高可靠的云存储服务，可提供99.9999999999%（12个9）的数据持久性，99.995%的数据可用性。多种存储类型供选择，全面优化存储成本。与平台无关的RESTFUL API接口，可以在任何应用、任何时间、任何地点存储和访问任意类型的数据。[[[4]](#endnote-3)]

在此次项目中，主要使用OSS的WEB端文件直传技术，前端利用OSS提供的Post Object接口，通过表单上传的方式将文件上传到OSS。该方案兼容大部分浏览器，但在网络状况不好的时候，如果单个文件上传失败，只能重试上传。过程中后端只需要对云服务器进行授权签名。

* **Spring Security ：**

它是基于Spring的一个安全框架，有着很好的扩展性。项目中，结合Redis，JWT，Spring Security，主要实现了用户的登录，认证和授权。利用JWT的Token传输用户登录状态和信息，在Spring Security框架上进行授权认证，利用缓存数据库减少查询MySQL的次数。

## 前端技术运用

* **VUE：**

是一套构建用户界面的渐进式框架。开发过程中只关心视图层，采用自底向上增量开发的设计。VUE的目标是通过尽可能简单的 API 实现响应的数据绑定和组合的视图组件。其中心思想是组件化，与Java的对象化编程十分契合。

* **Element UI：**

一套为开发者、设计师和产品经理准备的基于 VUE 2.0 的桌面端组件库，配合VUE使用。

* **V-charst:**

绘图插件，方便绘制折线图、饼图、环形图等统计图，且有多个配置项。

## 系统架构设计模型

在整个系统中，项目采用前后端分离技术，用户通过浏览器直接访问前端项目，其发出请求到后端服务器中，后端服务器接收到请求参数，对数据库进行查询，处理的结果响应给前端服务器，再交由用户展示。如图2-3：

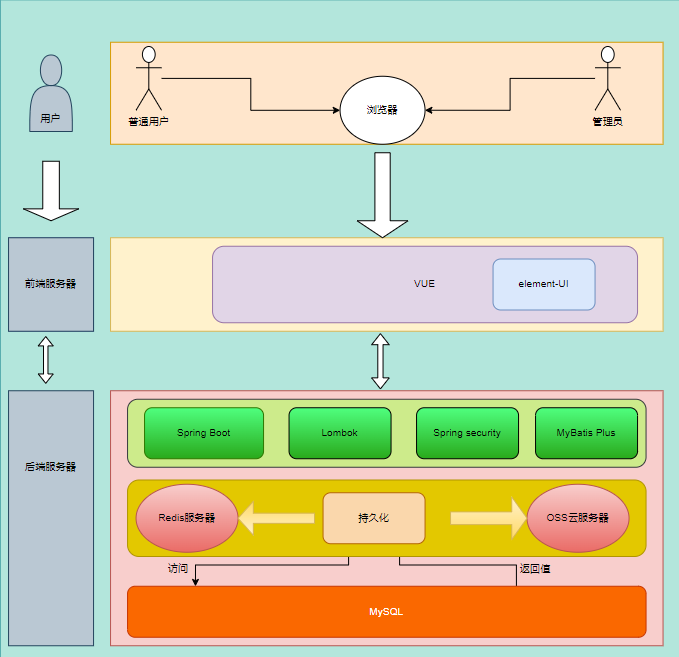


图2-3

# 系统分析

## 系统可行性分析

spring boot作为一个web网站简易开发的服务框架，拥有不错的性能以及开发可操作性、可扩展性，兼容当下市场的绝大部分服务框架、中间件。对于商城开发的后续功能改动和产品维护提供了各种监控工具，在当下开发任务下十分契合。

开发所使用的绝大多数工具以及中间件都是开源且免费的，几乎可以实现零成本开发。

系统使用的是传统商城样式模板UI，对于用户来说没有对购物界面做出任何改动，其功能使用方式也十分简洁，无需过多操作。对于管理员端界面，采用的是数据可视化处理，更直观的统计与管理。

## 需求分析

系统主要服务于售卖商品的个体商户，以及购买该商户产品的普通群众。系统主要分为前端商城页面和后端可视化商品管理页面。普通用户通过访问商城页面，可正常选购相关商品。还支持退货系统等售后服务。可视化管理页面供商户及管理员使用，有资源分配，文件上传，商品管理与审核，商品销售指标总览，产品运维等功能。综上所述：

* 普通用户：登录、注册、购物车、支付、搜索。
* 销售商：商品管理、销量统计。
* 商品销售运维人员：订单管理、商品状态跟踪、产品分析
* 系统运维人员：资源管理、产品审核、用户管理

需求建模：

使用UML例图来分析用户和管理员角色，可以得到如下图所示的用例图：如图3-2-1，图3-2-2：

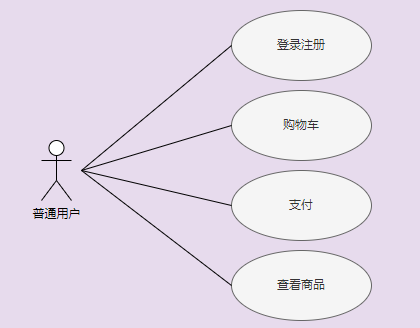


图3-2-1

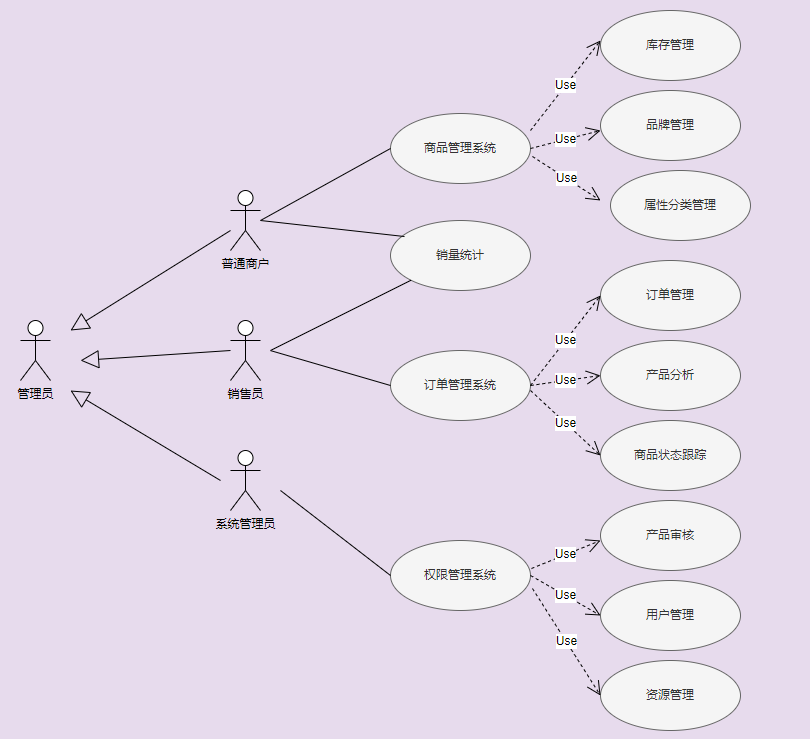


图3-2-2

## 非功能性需求分析

1. 系统界面需比较美观，界面布局需紧凑合理，方便用户的使用.
2. 系统可维护性、可靠性：系统中各功能在维护时，应保证其易维护，且不易崩溃，增加其可靠程度。
3. 系统应该具有一定的扩展能力。系统需要具备一定的技术性。
4. 通过主页和登录建立良好的印象，让界面看起来更专业简洁，实现可公布信息的信息透明，符合用户预期,以此消除用户的不确定感。
5. 传达安全策略，建立安全范围。在一些操作上，增强用户的控制感。系统整体架构需要以业务模块进行分组，方便后期的服务化与扩展。

## 数据库分析

数据库表对前端后端统一管理：

* cms\_\*：内容管理模块相关表
* oms\_\*：订单管理模块相关表
* pms\_\*：商品管理模块相关表
* sms\_\*：营销销售模块相关表
* ums\_\*：用户会员模块相关表

# 系统总体设计

## 系统架构设计

在开发框架上，系统结合了实际情况的需求，以及自身技术的掌握情况对于本系统的后台采用了Spring Boot + MyBatis + MySQL作为后台的基础开发框架。Spring Boot，它简化了后台开发的流程，能快速的为项目的构建搭好基础的框架。

在数据交互上，MyBatis作为持久层框架，为后台与数据库的交互提供了强有力的帮助，使用 MyBatis Plus能快速的将其需要的连接文件生成。

数据库方面，则是使用的MySQL关系型数据库和Redis非关系型数据库。Redis在项目中主要作为登录日志功能和登录状态检测功能实现提供帮助。而MySQL只负责存储长期有效的数据。

前端系统使用VUE.JS作为基础的开发框架，运用了Element前端UI库作为后台管理PC端的开发。Element UI能帮助网站快速的成型，使用框架开发使用已经预制好的UI组件能够极大的减小工作量，同时该框架的配色也较友好，开发时不用思索太多设计美学相关的内容。前期前端的框架搭建，则是使用的GitHub上的开源脚手架快速搭建项目结构。

综上所述，用于编写本系统的详细技术架构如图4-1所示：

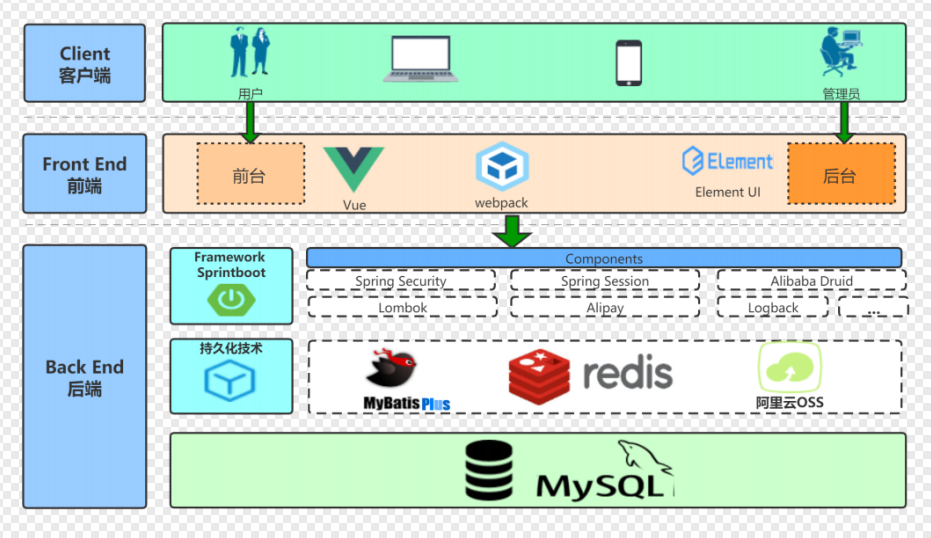


图4-1

## 系统模块总体设计

普通用户：

* 登录注册：用户可以以游客、会员、普通顾客身份对其使用系统。
* 个人中心：属于用户的个性化设置
* 查看商品：包括商品详细内容，搜索商品，商品分类内容等等。
* 购买与支付：购买商品，生成订单，在线支付。
* 发布评论：用户可以在商品页发布一些相关的信息，如文字等。
* 客服服务：其中包括有关的售后服务，如退货，修改收货地点等。
* 购物车：实现商品提前选购，加入购物车。

系统运维：

* 资源管理：对数据的请求，响应路径进行统一管理、统一认证。
* 用户、权限管理：对登录的角色身份进行判断处理，角色权限统一管理认证。显示不同菜单。
* 产品审核：展示系统使用情况，查看系统的用户，账号禁用管理。

商品运维人员：

* 商品统计：展示商品时间段内的销售情况。
* 产品分析：针对产品，展示物流情况，出货进货记录，仓库内数据。
* 销量预测：评估商品销量，通过系统算分，提出建议（目前建议内容只给出三个状态）
* 评论管理：显示用户评论，管理评论。

个体商户：

* 商品管理：拥有商品上架，商品添加功能，修改商品信息，删除商品等。
* 售后服务：针对用户购物需求，处理客户需求的功能，包括退货申请，订单处理等功能。

**系统功能架构如下，图4-2：**

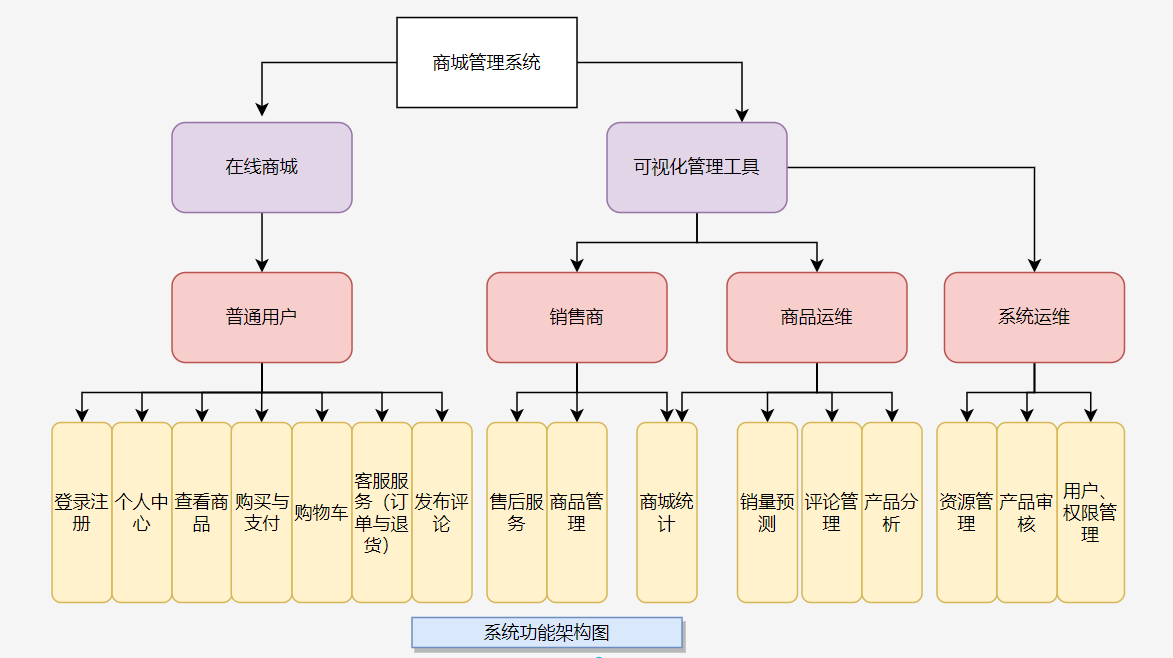


图4-2

## 系统功能使用流程图

普通用户：

普通用户使用在线商城页面，未登录可以直接访问首页，首页可以实现商品的搜索，查看。但不能进行购物车、评论、购买等私有操作。用户通过点击登录按钮，或通过私有操作跳转，可访问登录注册页面，开启商城全部功能。在支付界面，用户支付成功，则跳转至首页，未成功支付，则订单开始倒计时过期，过期订单用户不可以再次打开。过程如图：图4-3-1；

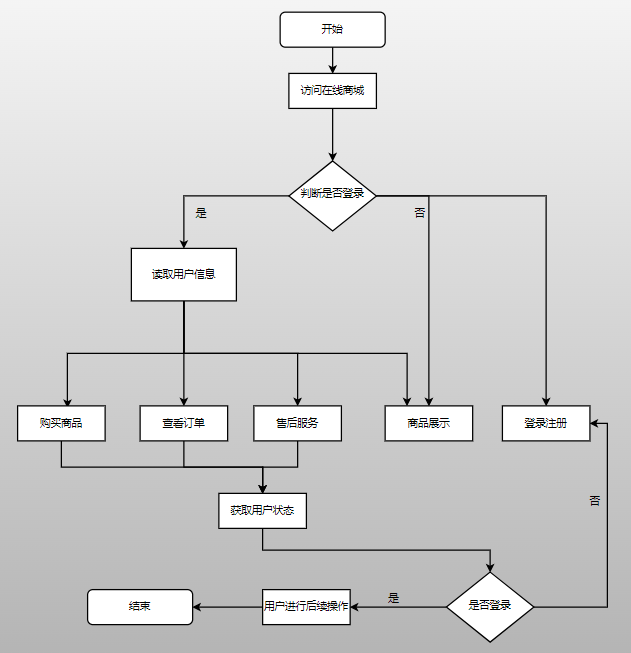


图4-3-1

管理员：

管理员使用后台管理工具，进入管理页面必须强制登录，登录成功后，不同的角色，拥有不同的管理权限，展示的管理页面菜单项也不尽相同，当用户执行某种管理操作时，会进行身份、权限、登录状态的验证，无权限则更新失败。过程如下图：图4-3-1；

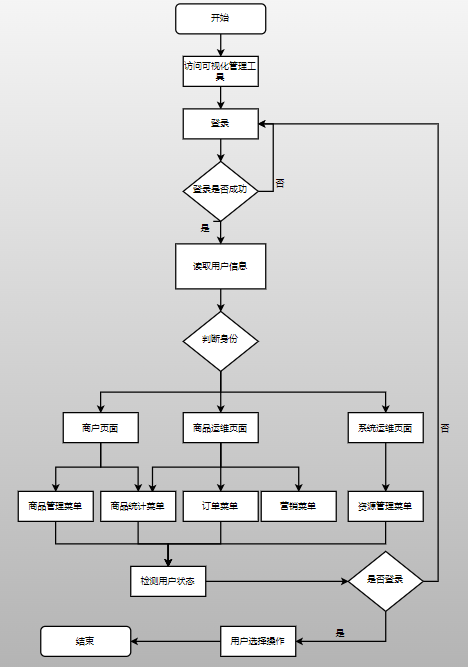


图4-3-2

## 数据库设计

**概念设计：**

这个系统主要分为两个页面，前端商城展示页面，和后台商品管理页面，而后台管理员也存在维护账户的管理员，因此，存在三个系统的用户，及普通用户、管理员和系统管理员，而商品是商城的主体，因此存在商品实体相关，即商品评论，商品订单和商品售卖详细信息。图4-4-1：

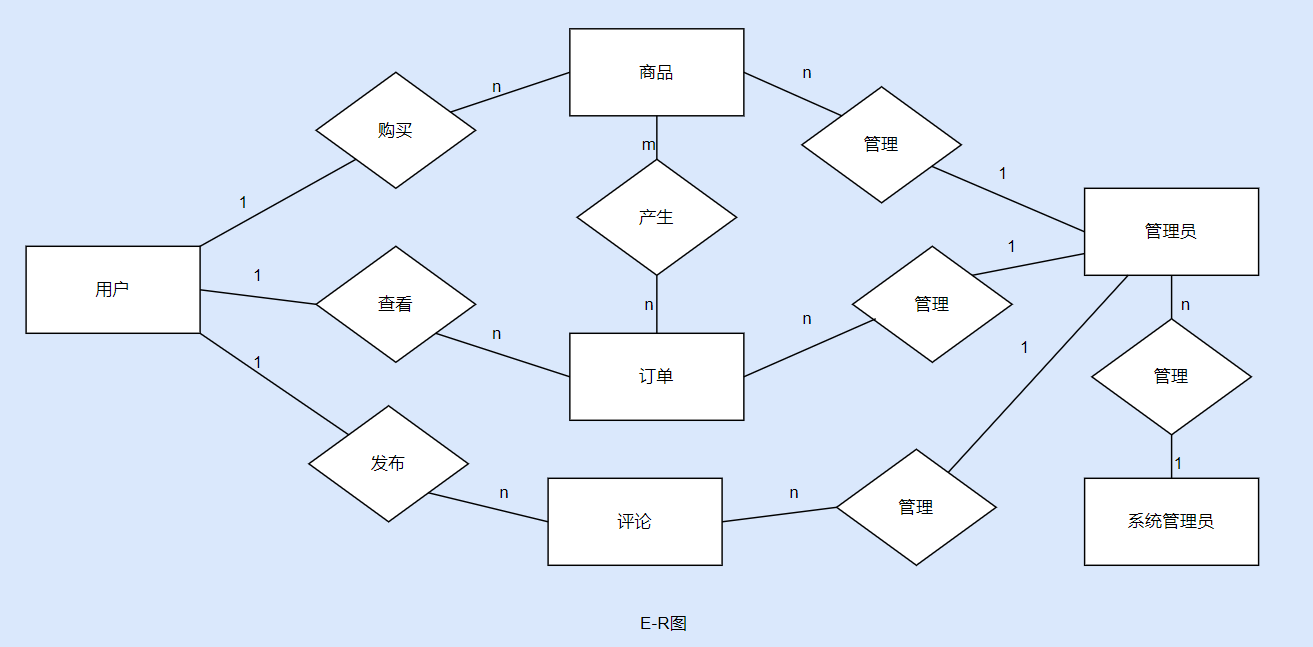


图4-4-1

实体属性关系如下图。图4-4-2：

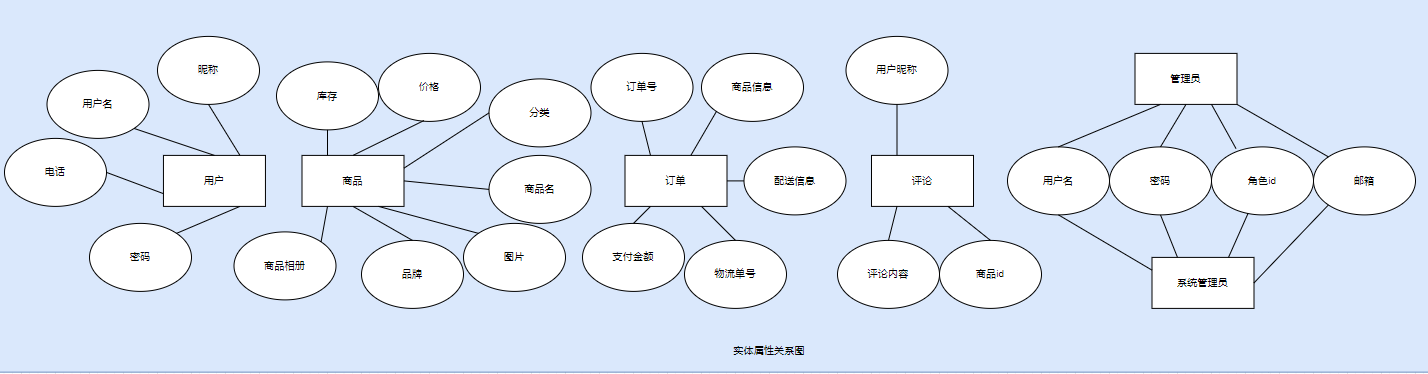


图4-4-2

**逻辑设计：**

根据概念模型设计，系统管理员应当可管理普通管理员，普通用户则可购买商品，进而生成订单，而订单实体需要根据商品库存查询进行商品发货，管理员又维护商品相关信息。综上所述：在实体类中下图根据概念设计模型的逻辑结构的E-R图。图4-4-3：

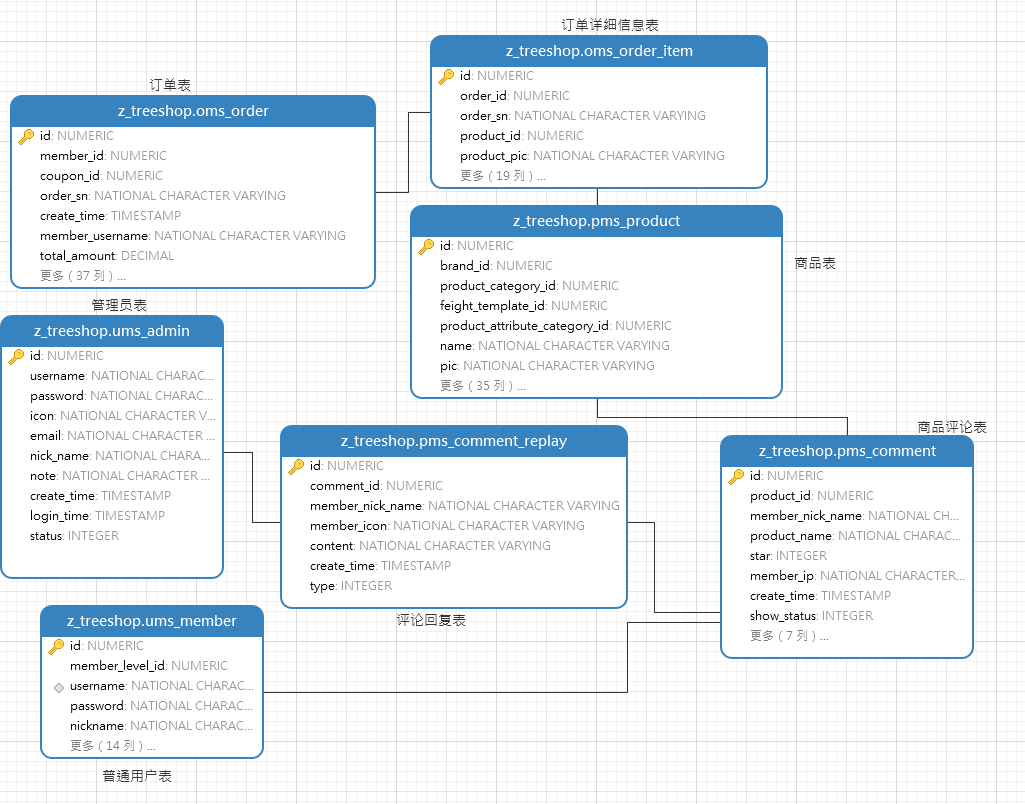


图4-4-3

**数据库物理结构设计：**

数据库的设计初衷是按照功能模块划分数据库表的，因此，数据库表大致分为四部分，分别对功能模块进行管理。

用户表：

主要功能实现表有：用管理员用户表、普通用户表。用于存用户基本信息，用户所属角色，拥有的权限等用户的管理。

管理员表，如图4-4-4：

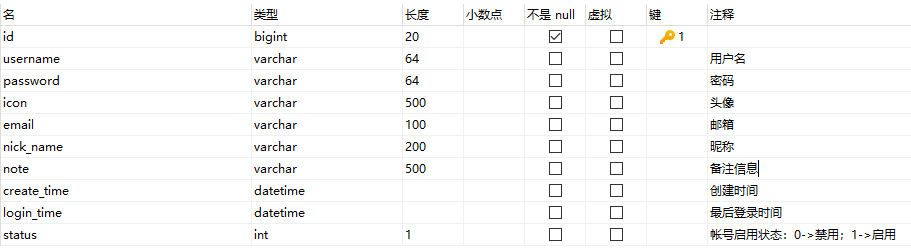


图4-4-4

普通用户表，如图4-4-5：



图4-4-5

商品表：

主要功能实现表有：商品表，并且需要对商品实体类做明确划分，进而需要使用有关分类的表。其用于存商品基础信息，如品牌信息，分类id，商品属性id等。具体如下。图4-4-6：



图4-4-6

订单表：

主要功能实现表有：订单表、订单详细商品表，订单表对订单交易内容做记录，订单商品信息表对每个订单的商品做关联（与商品属性、库存表关联）。如下图为订单表。图4-4-7：



图4-4-7

订单商品信息表，用来关联商品的购买详细信息，如商品属性，商品价格等。如图4-4-8：



图4-4-8

评论表：

主要实现表有：评论表和回复表。主要存储用户在商品下的评论。评论主要分两种，一种是商品下母评论，一种是评论下的子评论。评论表如图。图4-4-9：



图4-4-9

回复表，记录回复评论下的评论。如图4-4-10：

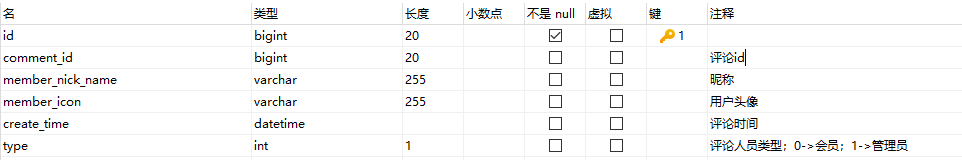


图4-4-10

# 系统详细设计

## 系统开发环境

开发工具：IDEA、WebStorme

JDK: 1.8

MySQL：版本为5.7+

Redis：版本3.2.100

操作系统：win10环境开发

## 主要功能模块实现

**登录功能实现：**

因为设计这个项目的目的就是微商城，也就是后端商品管理工具页面只对一个店铺服务，多个销售员管理，多个商品运维人员维护，所以系统运维必须是数据库事先注册好的用户，登录界面不对外开放有注册功能。在线商城则是正常的登录注册过程。

而登录采用Redis数据库进行缓存用户信息的方式实现，而Mysql表持久化特殊用户，且只开启一级缓存。登录功能操作流程图如下。图5-2-1：

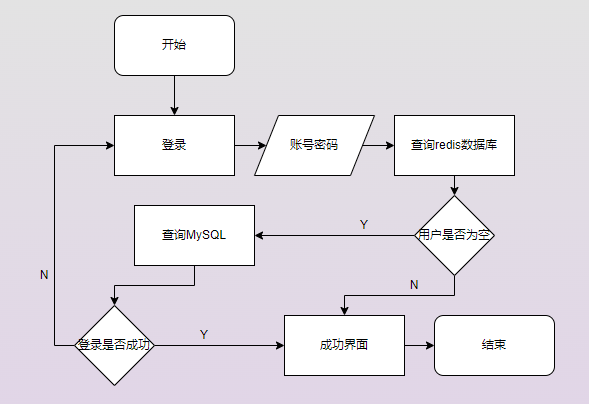


图5-2-1

*登录过程联调MySQL、Redis：*

为提高用户使用体验，登录过程摒弃了传统的Session记录账户登录状态的方式改而使用Token验证，账户可以实现一次登录持久使用，且账户生命周期依据策略调整。

**算法分两个过程：**

*登录过程：*

* 用户输入正确账号密码，发起登录请求。
* 后端接收数据，查询数据库（Redis或MySQL）
* 得到用户详细信息，根据用户账号密码生成基于JWT的Token字符串。（加密过程）
* 登录成功，响应给前端固有格式的Token。
* 前端将Token字符串存入Coking中。

*任意请求过程：*

* 用户发起某个请求。请求头携带Token。
* 后端Spring Security过滤器执行JWT的Token验证（解密过程）
* 从Token中获取用户名。
* 根据用户名，从Spring Security容器中拿到用户信息。
* 将用户相关信息存入Spring Security容器认证通过。标记为登录状态，后续过滤器直接放行。

下图为登录过程的功能实现时序图：图5-2-2

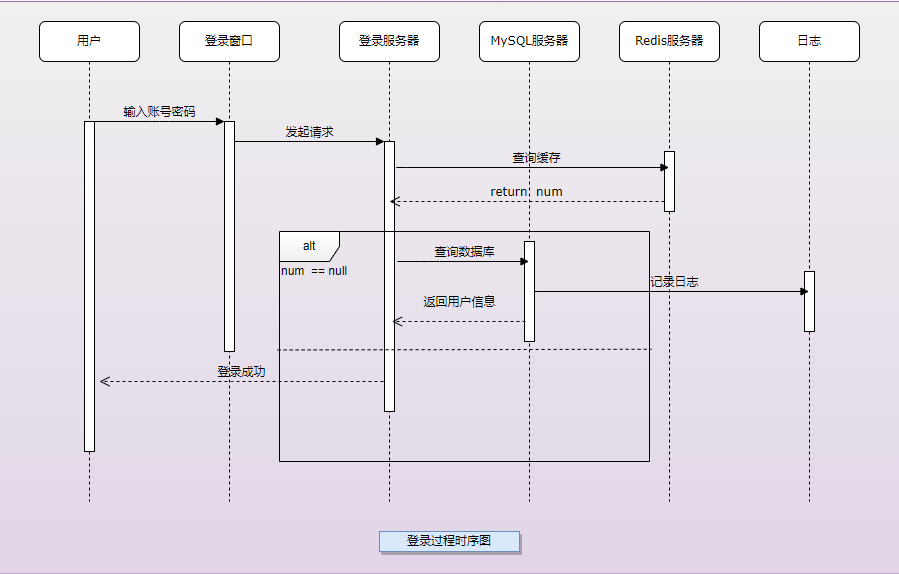


图5-2-2

任意请求过程的功能实现时序图。如图5-2-3：

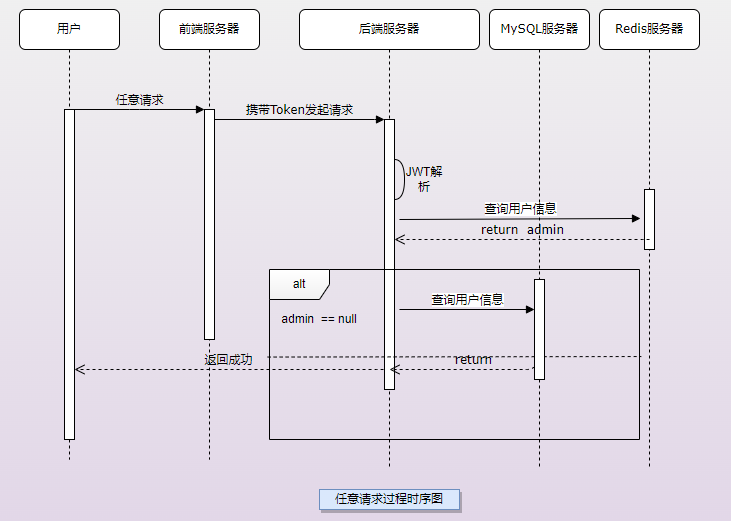


图5-2-3

***核心代码实现：***

* 查询数据库过程，包括Redis和MySQL：

//根据用户名查询用户信息

@Override

public UserDetails loadMemberByUsername(String name) {

UmsAdmin admin = getAdminByUsername(name);

List<UmsRole> roleList = getRoleList(admin.getId());

if (admin != null) {

return new UserDetailsProperties(admin, roleList);

}

throw new ApiException("用户名密码错误!");

}

@Override

public UmsAdmin getAdminByUsername(String username) {

//查询Redis数据库

UmsAdmin admin = adminCacheService.getAdmin(username);

if(admin!=null) return admin;

QueryWrapper<UmsAdmin> wrapper = new QueryWrapper<>();

wrapper.lambda().eq(UmsAdmin::getUsername,username);

List<UmsAdmin> adminList = list(wrapper);

if (adminList != null && adminList.size() > 0) {

admin = adminList.get(0);

adminCacheService.setAdmin(admin);

return admin;

}

return null;

}

* 正确登录后返回用户信息，根据用户信息生成Token如下：

***//加密： 根据用户名生成token***

***public String generateUserNameStr(@NonNull String username, String password, String nickname) {***

***Map<String, Object> claims = new HashMap<>();***

***claims.put(CLAIM\_KEY\_USERNAME,username);***

***claims.put(PASS\_WORD,password);***

***if (nickname == null) {***

***claims.put(NICK\_NAME,username);***

***}else {***

***claims.put(NICK\_NAME,nickname);***

***}***

***claims.put(CLAIM\_KEY\_CREATED, new Date());***

***return generateToken(claims);***

***}***

* 前端记录Token，并在每次请求头上加入Token：

// request拦截器

service.interceptors.request.use(config => {

//在全局变量中获取Token，并放入请求头中。

if (store.getters.token) {

config.headers['Authorization'] = getToken()

}

return config；

}, error => {

// Do something with request error

console.log(error) // for debug

Promise.reject(error)

})

* 后端抓取请求头中的Token，对其解密，获得用户信息的过滤器实现：

public class JwtAuthenticationFilter extends OncePerRequestFilter {

@Autowired

private UserDetailsService userDetailsService;

@Autowired

JwtTokenUtil tokenUtil;

@Value("${jwt.tokenHead}")

private String tokenHead;

@Value("${jwt.tokenHeader}")

private String tokenHeader;

@Override

protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain) throws ServletException, IOException {

//获取请求头参数，不为空则为登录状态，注册状态

String token = request.getHeader(tokenHeader);

if (!StringUtils.isEmpty(token) && token.startsWith(tokenHead)) {

token = token.substring(tokenHead.length());

String memberName = tokenUtil.getUserNameFromToken(token);

if (!StrUtil.isBlank(memberName)) {

UserDetails member = userDetailsService.loadUserByUsername(memberName);

if (member != null) {

//生成security框架专属token

UsernamePasswordAuthenticationToken authenticationToken = new UsernamePasswordAuthenticationToken(member,null,member.getAuthorities());

authenticationToken.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));

SecurityContextHolder.getContext().setAuthentication(authenticationToken);

}

}

}

filterChain.doFilter(request,response);

}

}

**关于用户资源访问与权限认证的实现：**

本次项目后台用户身份较前台多，因此，并没有针对账户身份拦截请求，而是直接在页面显示上去除用户不存在的功能选项，从而从根本上解决不同身份用户权限不同功能不同的问题。一切非法从导航栏传入的资源路径都将被Spring Security过滤出去，防止一部分资源被非法访问。对于种类繁多的后台管理员，需要对其功能菜单做处理，无权则不显示。

如下流程图为通过Spring Security 安全框架实现的授权与认证。图5-2-4

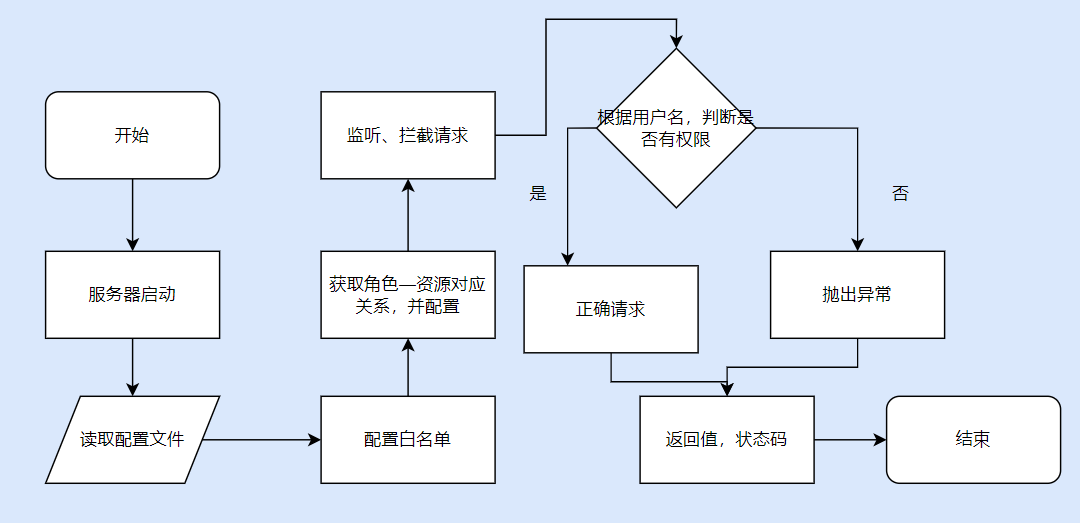


图5-2-4

***核心代码实现：***

* 配置Config代码实现：

public class SecurityConfig extends WebSecurityConfigurerAdapter {

//非必须自动注入类

@Autowired(required = false)

SecurityResourceRoleInterface resourceRoleInterface;

@Autowired(required = false)

private DynamicSecurityMetadataSource dynamicSecurityService;

@Override

protected void configure(HttpSecurity http) throws Exception {

//放行白名单

ExpressionUrlAuthorizationConfigurer<HttpSecurity>.ExpressionInterceptUrlRegistry registry = http.authorizeRequests();

for (String url : getUrlSecurityList().getUrls()) {

registry.antMatchers(url).permitAll();

}

//静态角色资源分配

if (resourceRoleInterface != null) {

Map<String, List<String>> resourceMap = resourceRoleInterface.getResource();

for (String url : resourceMap.keySet()) {

// todo Object数组转String数组

List<String> res = resourceMap.get(url);

registry.antMatchers(url).hasAnyAuthority(res.toArray(new String[res.size()]));

}

}

//允许跨域请求 OPTION请求

registry.antMatchers(HttpMethod.OPTIONS).permitAll();

//其他请求需要身份验证

registry

.anyRequest().authenticated()

.and().cors()

.and()

//关闭CSRF请求伪造

.csrf() .disable()

//禁止session

.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS)

//自定义处理类

.and().exceptionHandling()

//没权限

.accessDeniedHandler(getAccessHandler())

//没登录

.authenticationEntryPoint(getAuthHandler())

.and()

//在security的配置中，添加过滤器，且在最前面，默认第一个是UsernamePasswordAuthenticationFilter.class，所以插入在它之前

.addFilterBefore(jwtAuthenticationFilter(), UsernamePasswordAuthenticationFilter.class);

//有动态权限配置时添加动态权限校验过滤器

if(dynamicSecurityService!=null){

registry.and().addFilterBefore(dynamicSecurityFilter(), FilterSecurityInterceptor.class);

}

}

}

* 重写Spring Security中UserDetailsService接口的loadUserByUsername方法：

@Override

public UserDetails loadMemberByUsername(String name) {

UmsAdmin admin = getAdminByUsername(name);

List<UmsRole> roleList = getRoleList(admin.getId());

if (admin != null) {

return new UserDetailsProperties(admin, roleList);

}

throw new ApiException("用户名密码错误!");

}

**商品管理关于OSS前端直传图片的技术实现：**

利用前端与云服务器直接交互，后端只实现对前端数据的签名认证，可以更高效的实现文件上传下载，但前提是需要网络支持。住要用在商品上架时的商品图片上传，以及加载页面图片时。其时序图如下图所示：图5-2-5

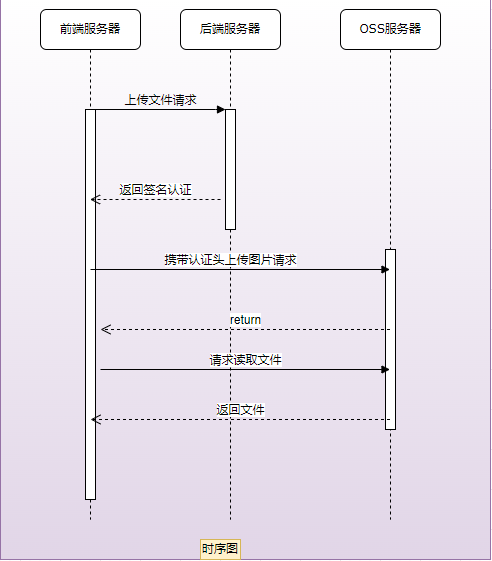


图5-2-5

***核心代码实现：***

* 前端直传代码：

<template>

<div>

<el-upload

:action="useOss?dataObj.host:minioUploadUrl"

:data="useOss?dataObj:null"

list-type="picture"

:multiple="false" :show-file-list="showFileList"

:file-list="fileList"

:before-upload="beforeUpload"

:on-remove="handleRemove"

:on-success="handleUploadSuccess"

:on-preview="handlePreview">

<el-button size="small" type="primary">点击上传</el-button>

<div slot="tip" class="el-upload\_\_tip">只能上传jpg/png文件，且不超过10MB</div>

</el-upload>

<el-dialog :visible.sync="dialogVisible">

<img width="100%" :src="fileList[0].url" alt="">

</el-dialog>

</div>

</template>

js实现访问后端：

beforeUpload(file) {

let \_self = this;

if(!this.useOss){

//不使用oss不需要获取策略

return true;

}

return new Promise((resolve, reject) => {

policy().then(response => {

\_self.dataObj.policy = response.data.policy;

\_self.dataObj.signature = response.data.signature;

\_self.dataObj.OSSAccessKeyId = response.data.accessId;

\_self.dataObj.key = response.data.dir + '/${filename}';

\_self.dataObj.dir = response.data.dir;

\_self.dataObj.host = response.data.host;

// \_self.dataObj.callback = response.data.callback;

resolve(true)

}).catch(err => {

console.log(err)

reject(false)

})

})

}

* 后端签名认证：

@Override

public OssResultDTO policy() {

String host = "https://"+bucketName+"."+endpoint;

SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMddHH");

String dir\_date = dir + sdf.format(new Date());

OSS ossClient = ossConfig.buildOssClient();

try {

long expireTime = 30;

long expireEndTime = System.currentTimeMillis() + expireTime \* 1000;

Date expiration = new Date(expireEndTime);

PolicyConditions policyConds = new PolicyConditions();

policyConds.addConditionItem(PolicyConditions.COND\_CONTENT\_LENGTH\_RANGE, 0, 1048576000);

policyConds.addConditionItem(MatchMode.StartWith, PolicyConditions.COND\_KEY, dir\_date);

String postPolicy = ossClient.generatePostPolicy(expiration, policyConds);

byte[] binaryData = postPolicy.getBytes("utf-8");

String encodedPolicy = BinaryUtil.toBase64String(binaryData);

String postSignature = ossClient.calculatePostSignature(postPolicy);

OssResultDTO dto = new OssResultDTO();

dto.setAccessId(accessId);

dto.setDir(dir\_date);

dto.setExpire(String.valueOf(expireEndTime / 1000));

dto.setHost(host);

dto.setPolicy(encodedPolicy);

dto.setSignature(postSignature);

return dto;

} catch (Exception e) {

System.out.println(e.getMessage());

} finally {

ossClient.shutdown();

}

return null;

}

**评论商品与管理评论功能实现：**

**在线商城商品评价功能：**

普通会员用户，可以通过登录后进入商品详细页面，对商品进行评价打星，用户对已经发布的评价可以做回复功能。其具体流程为。图5-2-6：

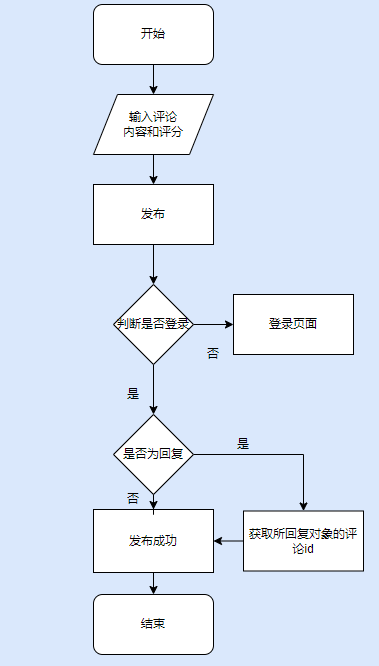


图5-2-6

***核心代码：***

* 前端js核心方法：

sendComment() {

if (!this.replyComment) {

this.$message({

showClose: true,

type: 'warning',

message: '评论不能为空'

})

} else {

let a = {}

let input = document.getElementById('replyInput')

a.commentUser = this.myName

a.headIcon = this.myHeader

a.commentContent = this.replyComment

a.ProductIdParam = this.ProductIdParam

a.star = this.star

this.comments.push(a)

this.replyComment = ''

input.innerHTML = ''

this.saveComment(a)

}

},

saveComment(comment){

let formData = new FormData();

formData.append("memberNickName", comment.commentUser);

formData.append("content", comment.commentContent);

formData.append("productId", comment.ProductIdParam);

formData.append("memberIcon", comment.headIcon);

formData.append("star", comment.star);

this.axios.post('/comment/opeMain',formData).then(()=>{

this.getCommentData()

})

}

* 后端实现接口：

@Override

public Boolean saveComment(PmsComment comment) {

comment.setCreateTime(new Date());

comment.setShowStatus(0);

String productName = productService.getProductName(comment.getProductId());

comment.setProductName(productName);

UmsMember member = memberService.getMemberId();

comment.setMemberIp(member.getId().toString());

return save(comment);

}

**后台商城管理系统对评论的管理功能：**

因为后台主要目的是更方便的管理在线商城。所以管理员应当对评论进行多种管理，包括商品评论删除、回复功能。并且获得商品评分的平均值。

***核心代码：***

@Override

//查询评论信息，并实现条件查询

public Page<PmsComment> fetchList(PmsComment comment) {

Page<PmsComment> iPage = new Page<>(comment.getPageNum(), comment.getPageSize());

QueryWrapper<PmsComment> queryWrapper = new QueryWrapper<>();

queryWrapper.lambda().eq(PmsComment::getProductId, comment.getProductId())

.like(StrUtil.isNotBlank(comment.getMemberNickName()), PmsComment::getMemberNickName, comment.getMemberNickName())

.eq(comment.getShowStatus() != null,PmsComment::getShowStatus, comment.getShowStatus())

.orderByDesc(PmsComment::getSort,PmsComment::getCreateTime);

return page(iPage, queryWrapper);

}

//回复评论，并对评论表的回复数加一

@Override

@Transactional

public Boolean sendCommentReply(PmsCommentReplay comment) {

comment.setCreateTime(new Date());

UmsAdmin admin = adminService.getAdmin();

comment.setMemberIcon(admin.getIcon());

comment.setMemberNickName(admin.getNickName());

comment.setType(0);

UpdateWrapper<PmsComment> updateWrapper = new UpdateWrapper<>();

updateWrapper.setSql("`replay\_count` = `replay\_count` + 1").lambda().eq(PmsComment::getId, comment.getCommentId());

update(updateWrapper);

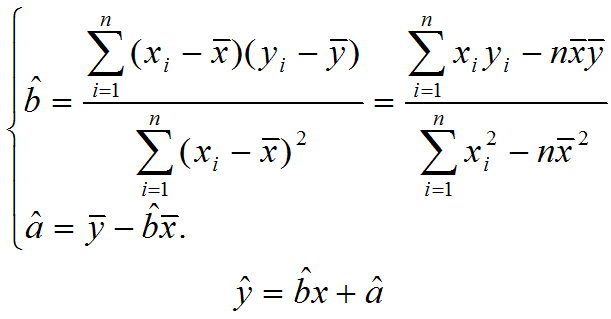
return commentReplayService.save(comment);

}

**关于商品销量的预测：**

本项目由于数据量有限，对预测模型的选用，采用比较简单基础的线性回归算法，利用线性回归方程，简单计算出对应时间下的销量。

利用基础公式：



算法流程：

1. 获取评论表下的对应商品评分。
2. 获取订单详细信息表商品对应时间和销量。
3. 计算因变量为：销量\*评分。
4. 取样本时间的时间戳为自变量。
5. 获取线性回归方程：y = bx+a.
6. 计算时间偏移量在未来七天的数据。

***核心代码：***

***@Override***

***public List<DataStatisticsDTO> forecastShopSale(DataQueryDTO queryDTO) {***

***Long productId = queryDTO.getId();***

***//todo 1.查询商品前30天销量、商品相应的评分***

***Double score = commentService.getScore(productId);//商品评分***

***List<DataStatisticsDTO> order = orderItemService.getOrderProduct(productId);***

***order = order.stream().map(o->{***

***o.setY(score\*o.getSale());***

***o.setX(DateUtil.parse(o.getTime()).getTime());***

***return o;***

***}).collect(Collectors.toList());***

***//todo 2.获取直线回归方程***

***Long xSum = 0l;***

***Double ySum = 0d;***

***for (DataStatisticsDTO dto : order) {***

***xSum = xSum + dto.getX();***

***ySum = ySum + dto.getY();***

***}***

***Long \_x = xSum/order.size();***

***Double \_y = ySum/order.size();***

***Double b1 = 0d;***

***Long b2 = 0l;***

***for (DataStatisticsDTO dto : order) {***

***b1 =b1 + (dto.getY()-\_y)\*((dto.getX()-\_x));***

***b2 = b2 + (dto.getX()-\_x)\*(dto.getX()-\_x);***

***}***

***Double b = b1/b2;***

***Double a = \_y-(b\*\_x);***

***DateTime date = DateUtil.date();***

***SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");***

***String time = format.format(date);***

***DateTime forecastTime = DateUtil.parse(time, "yyyy-MM-dd");***

***for(int i = 1; i<= 7;i++){***

***DataStatisticsDTO dto = new DataStatisticsDTO();***

***Double forecastSale = getXforY(a, b, DateUtil.offsetDay(forecastTime, i), score);***

***dto.setTime(DateUtil.format(DateUtil.offsetDay(forecastTime, i),"yyyy-MM-dd"));***

***dto.setSale(forecastSale.intValue());***

***order.add(dto);***

***}***

***//todo 3.根据方程获得未来7天销量***

***return order;***

***}***

***private Double getXforY(Double a,Double b, DateTime time, Double score){***

***Double num = b\*time.getTime()+a;***

***if (num==0){***

***return num;***

***}***

***return num/score;***

***}***

**关于商品管理的产品分析功能：**

产品分析，主要是对某个商品做销量、销售额占比，退货次数占比，统计并给予评价。

主要是通过查询订单表，对其商品订单的销量和营业额做求和，再在前端对数据求比例。评分的过程主要是对销量占比、营业额占比、退货占比进行计算，令销量的权重为2，营业额权重为1，退货权重为3。

***核心代码：***

@Override

public AnalysisDTO analysisShop(DataQueryDTO queryDTO) {

Long id = queryDTO.getId();

//销量

Map<String, Object> sum = queryOrderSum(null);//总销量

Map<String, Object> shopSum = queryOrderSum(id);//商品销量

//销售额

Map<String, Object> price = queryOrderSale(null);//销售额

Map<String, Object> shopPrice = queryOrderSale(id);//商品销售额

//订单退货量

Map<String, Object> returnSumCount = queryOrderReturn(null);//退货数

Map<String, Object> shopReturnSumCount = queryOrderReturn(id);//商品退货数

Object sumCount = sum.get("sumCount");

Object shopSumCount = shopSum.get("sumCount");

Object sumPrice = price.get("sumPrice");

Object shopSumPrice = shopPrice.get("sumPrice");

Object returnSum = returnSumCount.get("returnSum");

Object shopReturnSum = shopReturnSumCount.get("returnSum");

int i = Integer.parseInt(sumCount.toString());

int i1 = Integer.parseInt(shopSumCount.toString());

BigDecimal bigDecimal1 = new BigDecimal(sumPrice.toString());

BigDecimal bigDecimal2 = new BigDecimal(shopSumPrice.toString());

int j = Integer.parseInt(returnSum.toString());

int j2 = Integer.parseInt(shopReturnSum.toString());

AnalysisDTO analysisDTO = new AnalysisDTO();

analysisDTO.setSumCount(i);

analysisDTO.setShopSumCount(i1);

analysisDTO.setSumPrice(bigDecimal1);

analysisDTO.setShopSumPrice(bigDecimal2);

analysisDTO.setReturnSum(j);

analysisDTO.setShopReturnSum(j2);

return analysisDTO;

}

前端核心代码：

<div class="app-container">

<!-- 添加-->

<el-card class="operate-container" shadow="never">

<i class="el-icon-tickets"></i>

<span>产品分析详细信息：</span>

<p>

<span style="float: left;margin: 30px;padding-left: 200px">分析评估分数：<span style="color: red">{{this.score}}</span>(满分100)</span>

<span style="float: left;margin: 30px;padding-right: 200px">商品存货：<span style="color: red">{{this.stock}}</span>件</span>

<span style="float: left;margin: 30px;padding-left: 100px">销售额：￥<span style="color: red">{{this.shopSumPrice}}</span></span>

<span style="float: left;margin: 30px;padding-right: 200px">销量：<span style="color: red">{{this.shopSumCount}}</span>份</span>

<span style="float: left;margin: 30px;padding-left: 200px">评估建议：<span style="size: 20px;color: #3c763d">{{this.score | formatScore}}</span></span>

<span style="float: left;margin: 30px;padding-left: 100px">退货量：<span style="color: red">{{this.shopReturnSum}}</span>单</span>

</p>

</el-card>

<el-card class="pie">

销量占比

</el-card>

<ve-pie :data="chartData1" ></ve-pie>

<el-card class="pie">

销售额占比

</el-card>

<ve-pie :data="chartData2" ></ve-pie>

<el-card class="pie">

退货占比

</el-card>

<ve-ring :data="chartData3" ></ve-ring>

</div>

Js：

getDataLine() {

getAnalysis({id:this.$route.query.id}).then(result => {

this.returnSum=result.data.returnSum;

this.shopReturnSum= result.data.shopReturnSum;

this.shopSumCount= result.data.shopSumCount;

this.shopSumPrice= result.data.shopSumPrice;

this.sumCount= result.data.sumCount;

this.sumPrice= result.data.sumPrice;

this.handleData();

});

# 系统测试

## 测试目的和分类

软件测试是我们日常开发不可或缺的一个重要流程，能够防止产品上线后产生一些不可避免的错误，更重要的是能够使软件功能实现上趋近于完美。

此篇主要研究在线商城及后台管理系统的功能实现方法，因此，测试着重点在功能是否能完整展示，实现功能的算法是否可取。故为功能性测试为主。

## 主要核心功能的测试

* 登录测试：

后台初始用户/密码为admin/123456，该账户事先存于数据库中，后续账号由系统账号代为注册。密码必须大于3位，不能为空，当不满足其中一个条件，则会弹窗警告错误登录信息。如图6-2-1



图6-2-1

菜单显示及首页展示：

由于系统管理员只负责系统内部用户账号及资源使用情况，菜单仅展示权限栏和首页。系统用户可以对账户进行分配角色，分配资源，也可以对账户进行管理，对不同角色进行菜单显示设置。同时，可以对资源、菜单、用户、角色进行增删改查等基础操作。其展示页面如下：图6-2-2

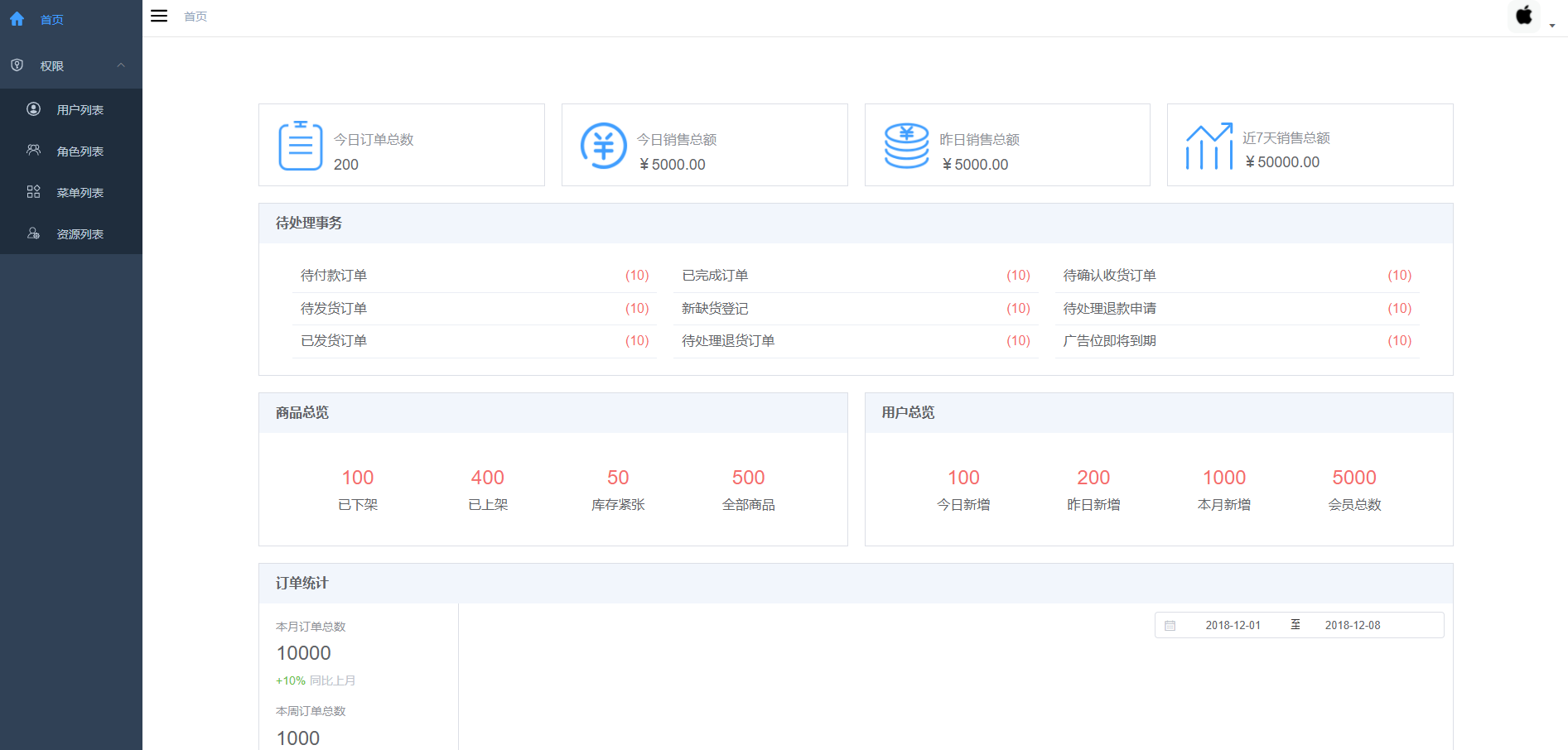


图6-2-2

* 权限管理：

在用户管理模块中，页面加载调用后端接口，对用户信息表做查询，同时关联角色表，可作增删改查操作，还可分配多角色、单角色。其功能完整实现：图6-2-3

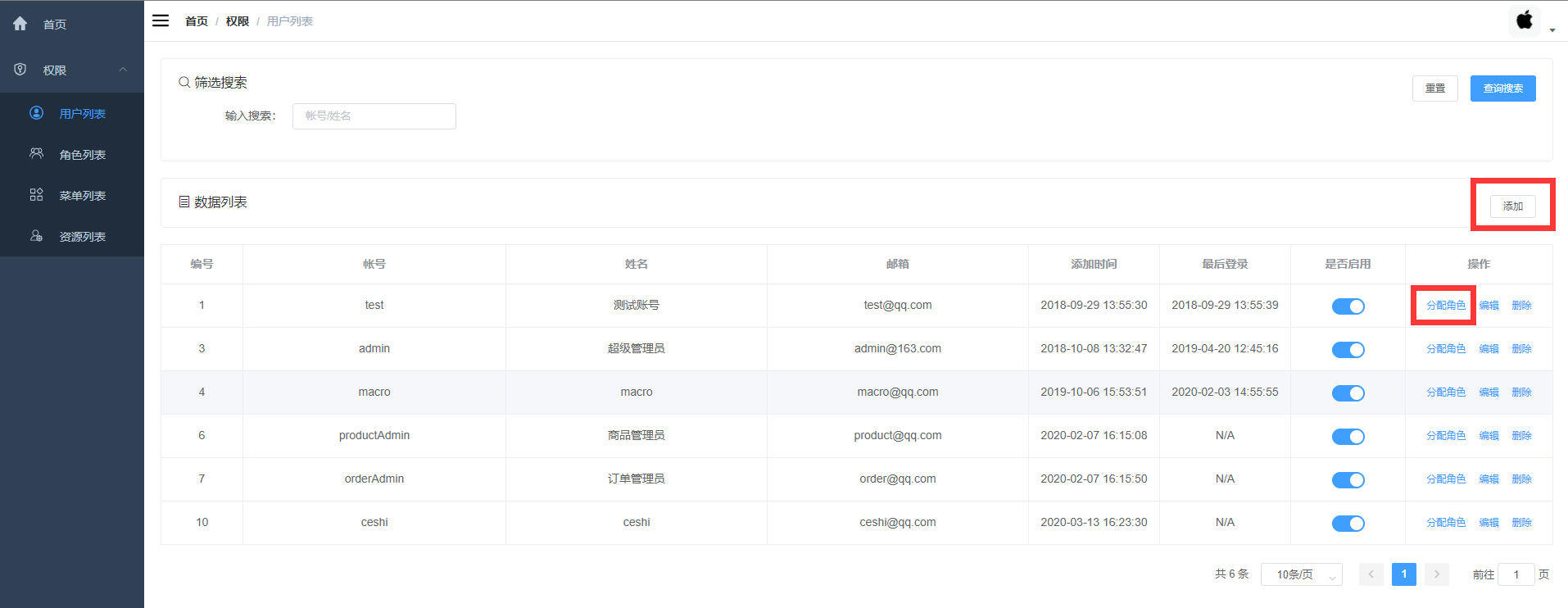


图6-2-3

在角色管理模块中，页面加载角色表中的数据，同时关联菜单表和资源表，可作增删改查操作，还可分配菜单和访问资源，其功能可完整实现：图6-2-4

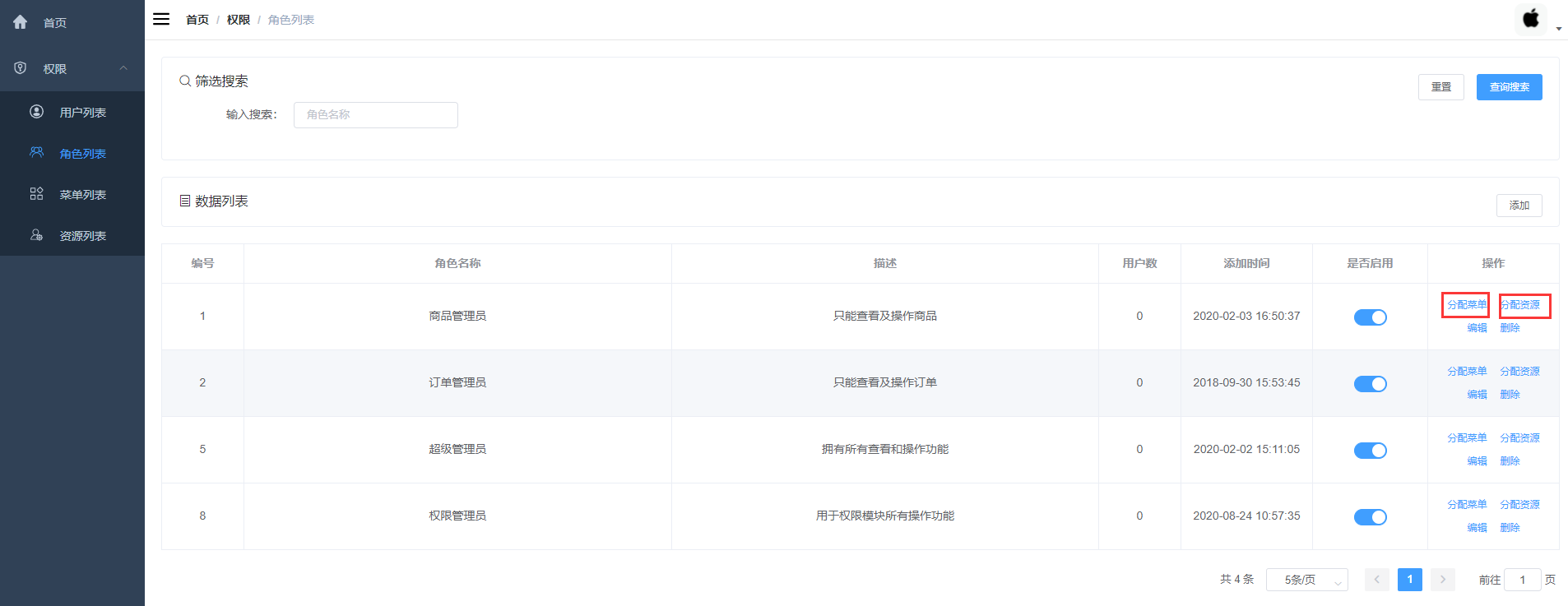


图6-2-4

* 商品管理：

当系统管理员添加账户并分配角色后，普通商户即可登录后台管理，对商品列表进行分类关联和库存、属性设置。商品搜索做模糊查询，名称可为左关联右关联，网络条件充沛时，商品图片上传功能正常，网络环境差时，则会在传输许久后弹出上传失败字样。图6-2-5、图6-2-6

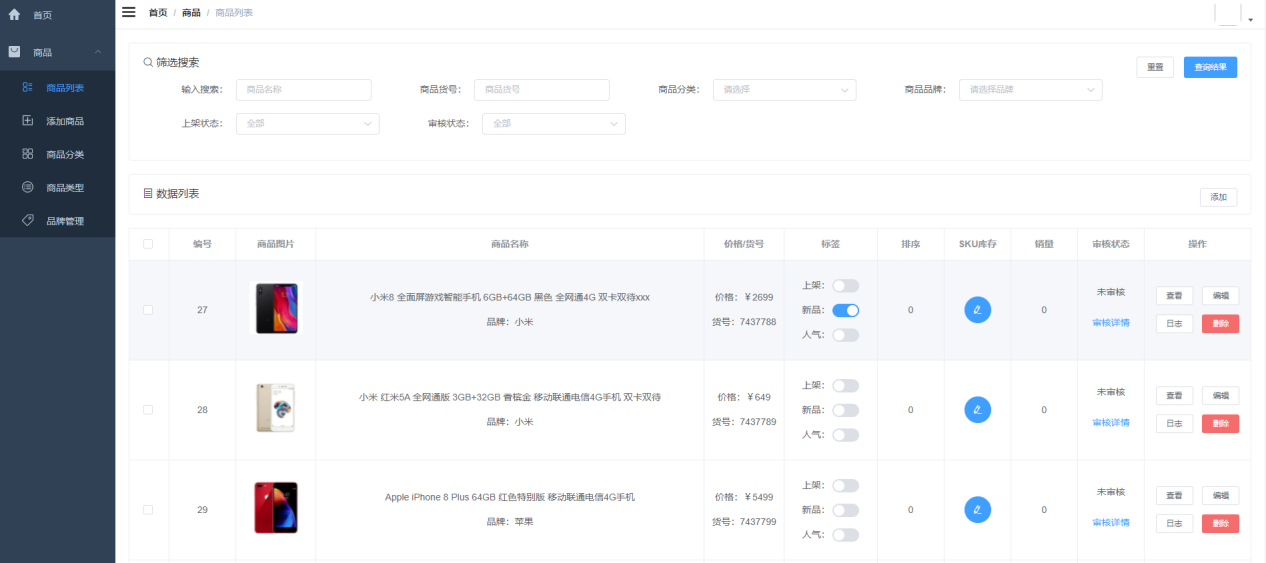


图6-2-5

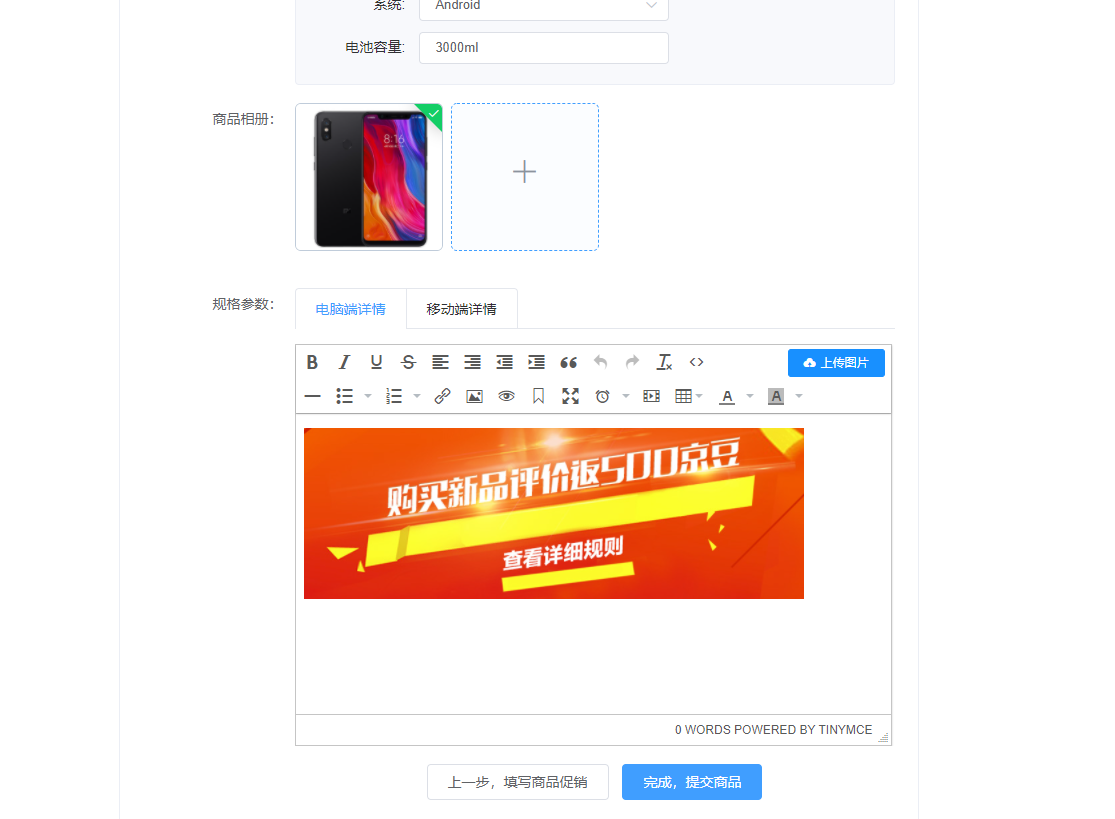


图6-2-6

商家对品牌的管理，品牌页面实现了基础的增删改功能，且与商品添加页面的品牌选项做关联，增加商品时选择维护品牌列表中的品牌，成功将品牌表与商品表关联。图6-2-7



图6-2-7

* 订单管理：

系统管理员分配角色账户为订单管理员成功，登录后只显示订单系统，对订单进行处理，进入页面，查询订单表，展示的订单存在四种种状态，即待处理、待发货、已完成、已关闭，订单列表展示如下：图6-2-8

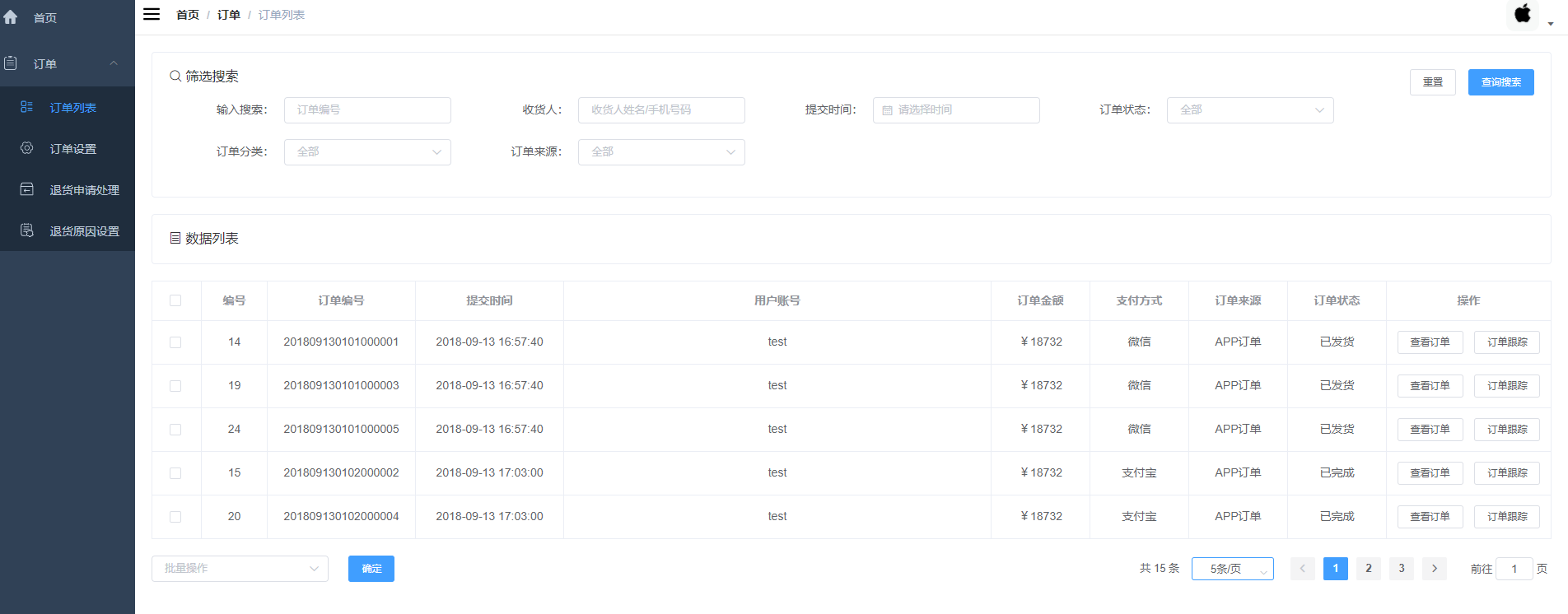


图6-2-8

订单的设置是根据当前用户的个人设置实现的增与改，存入数据库中存在默认值。图6-2-9



图6-2-9

退货信息管理，退货信息有四种状态，退货中、已拒绝、待处理、已关闭，管理员通过改变状态，可操作页面内容修改部分不同，且每次操作都会记录日志，操作记录的操作人。页面显示如下：图6-2-10

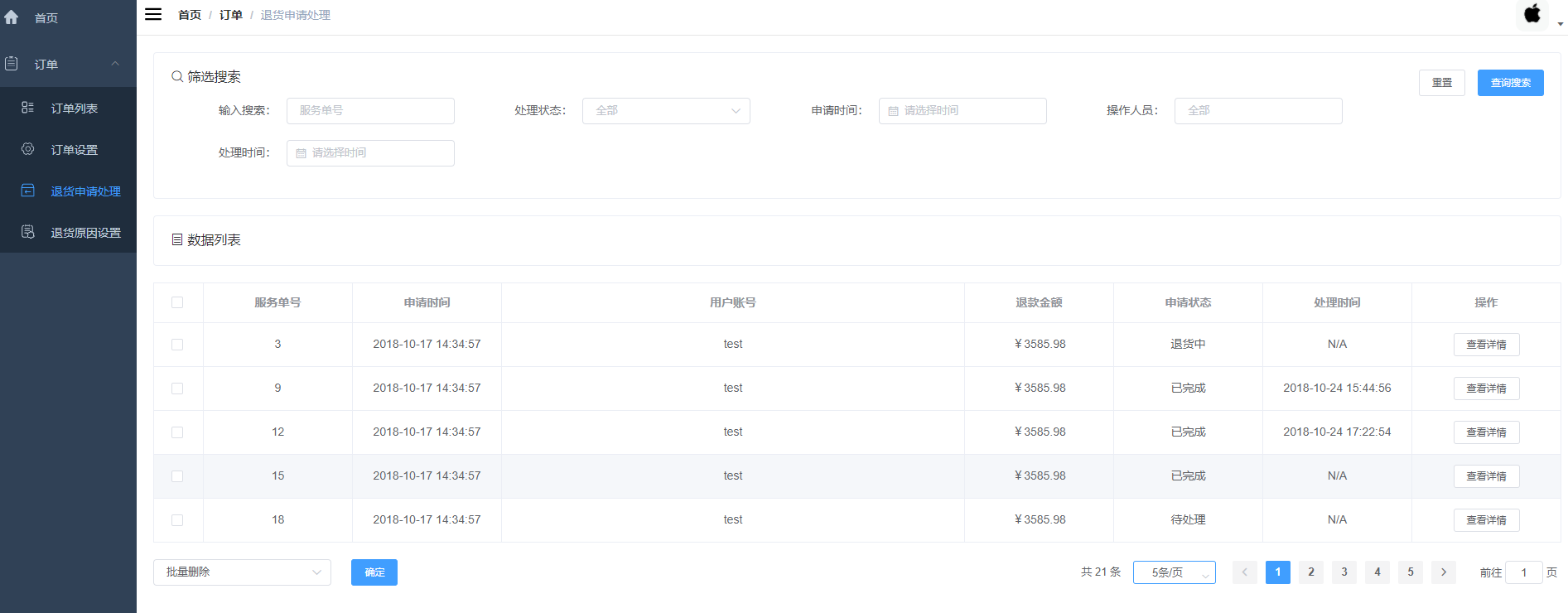


图6-2-10

退货原因模块，主要功能类似记录，用来统一管理用户退货原因，方便下次改正。其页面主要是增删改查的功能，页面显示如下：图6-2-11

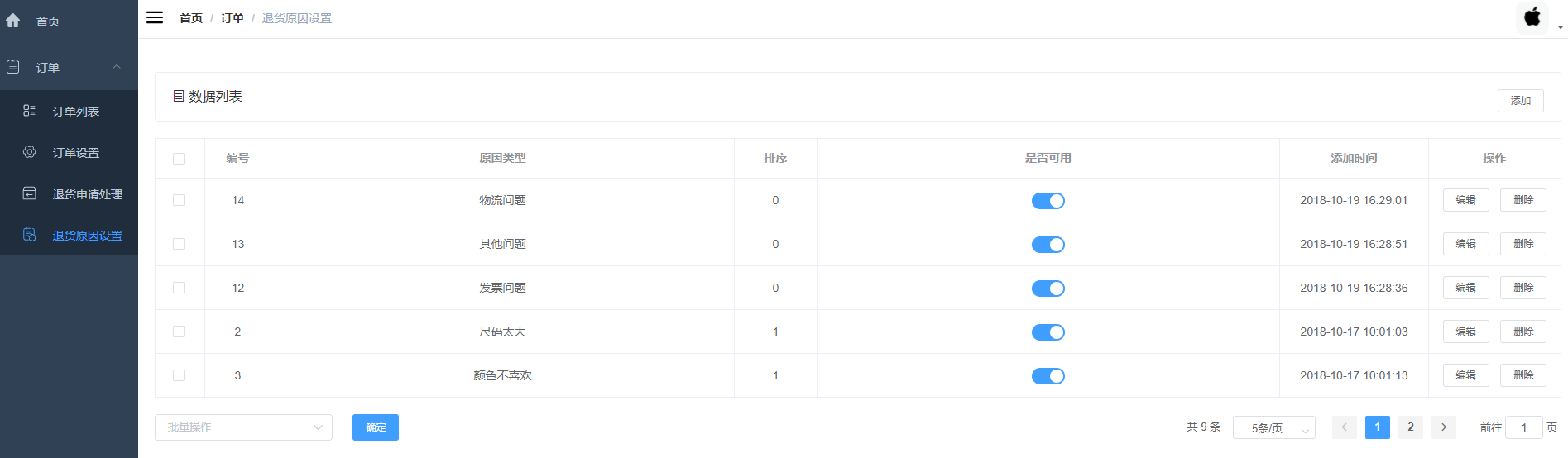


图6-2-11

* 促销管理：

促销模块主要是管理在线商城推荐部分，广告展示部分，活动秒杀专题部分。其功能主要是关联各种表结构实现增删改查，定义时间节点，方便在线商城实现定时任务。页面功能可正常设置商品加入促销活动，也可以正确设置时间段对活动开启关闭。图6-2-12

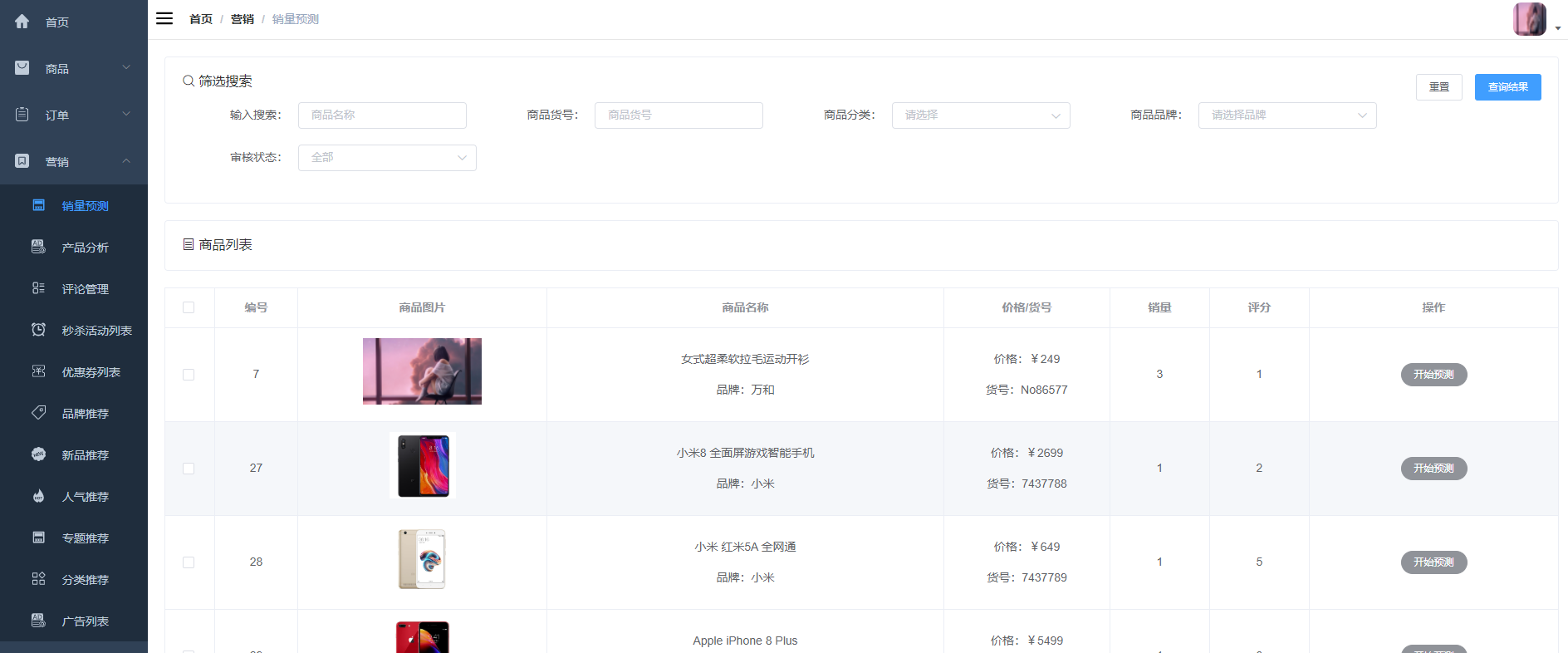


图6-2-12

对于商品预测功能和产品分析功能，点击相应菜单项，进行开始分析或预测，则会展示商品的相关统计图。图6-2-13、图6-2-14

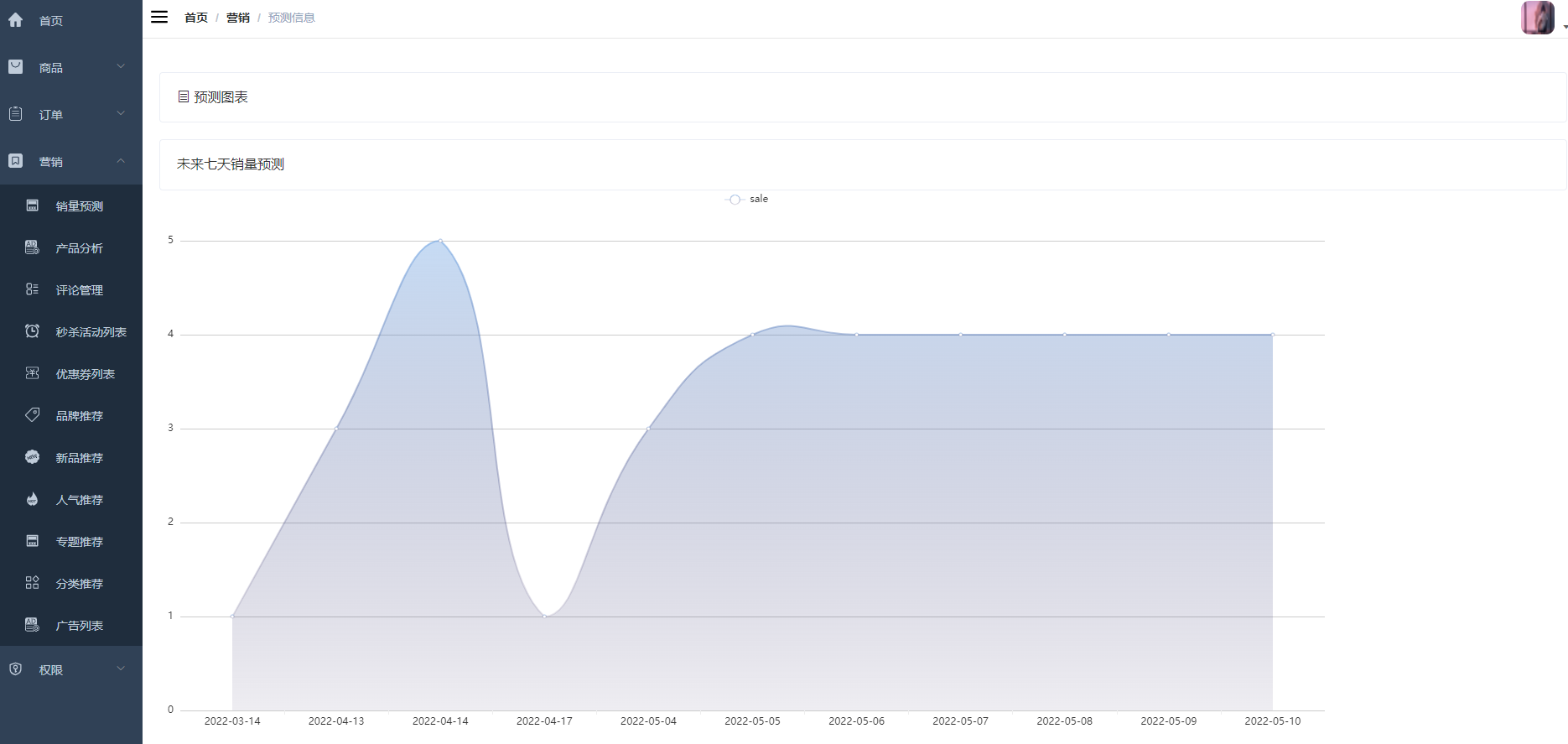


图6-2-13



图6-2-14

* 在线商城的商品评论打分功能：

在用户登录状态下，用户可以在商品页面下发布相关评论，同时可以对商品打分。点击查看评论，展示效果如下：图6-2-15

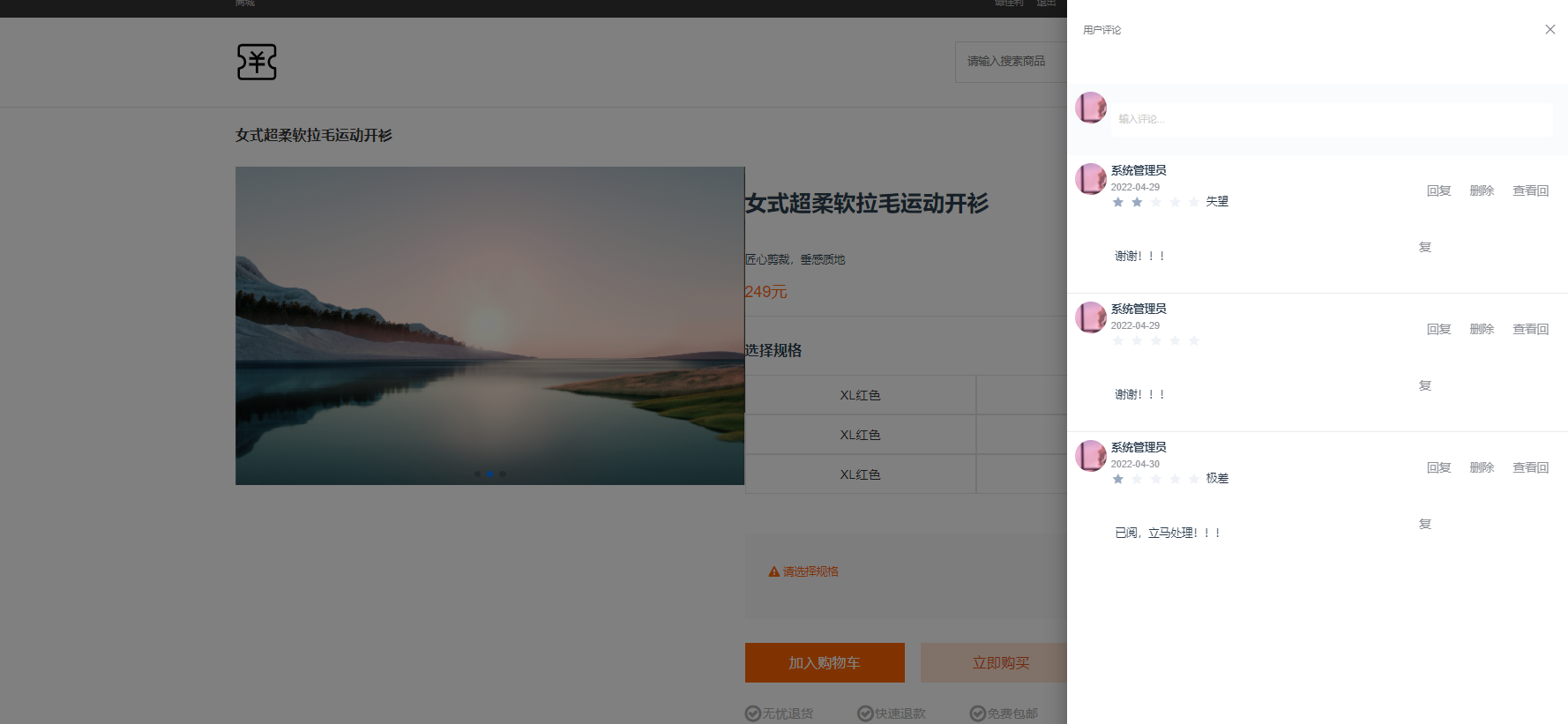


图6-2-15

后端则对所有商品进行统一管理和发布信息，其展示状态如下：图6-2-16

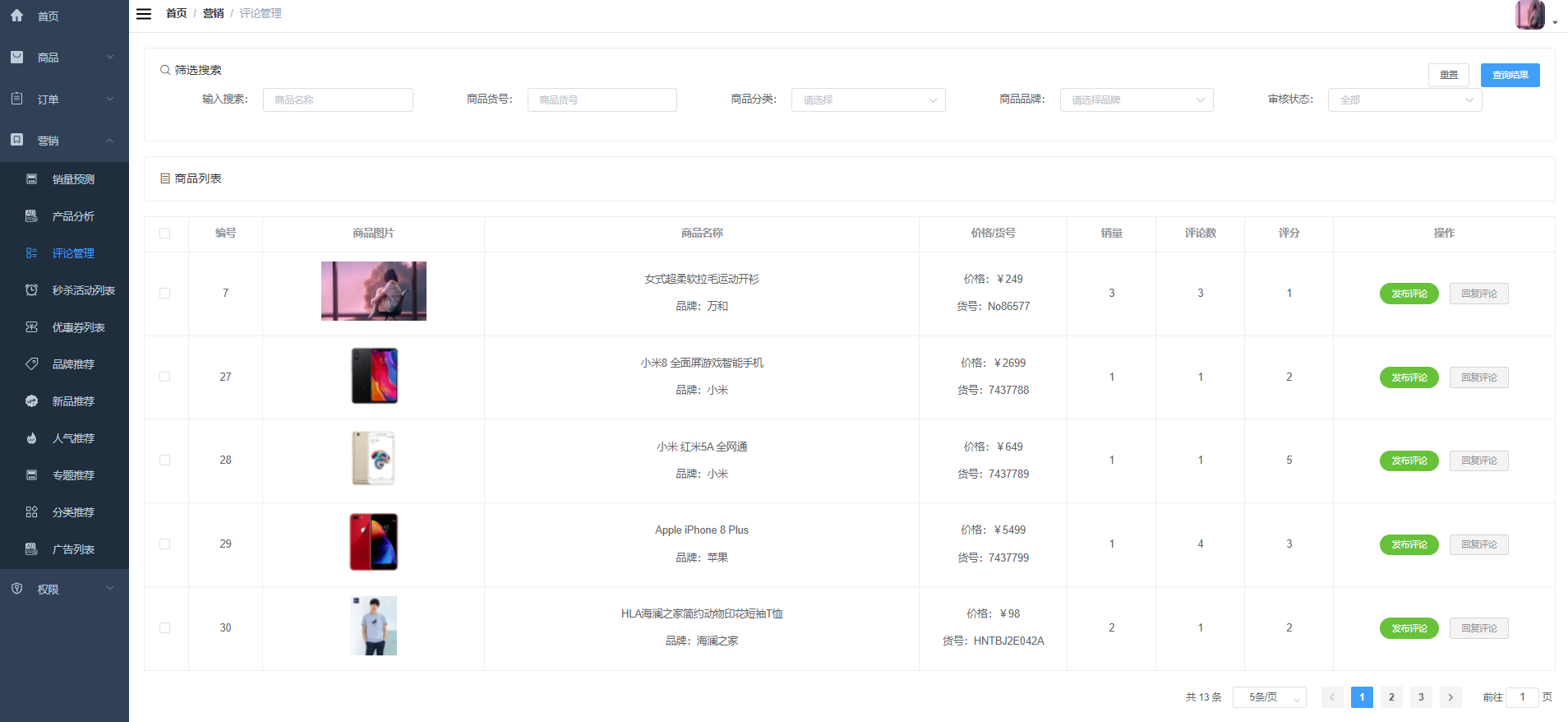


图6-2-16

# 结论

通过本次毕业设计，主要任务是熟悉并掌握框架的、企业级的开发流程。在本次毕业设计的过程中，一开始的茫然到后来的各种学习、参考、查阅资料，每一步都走的极其艰难。最终在各方面的请教、指导，最终完成了论文的编写。

通过这个商城管理系统的开发过程，我不仅仅将毕生所学的知识应用到了本次系统中，还通过查阅资料获取了许多更深层次的知识。同时也是在这个过程中，我见识到了当下一些比较流行的第三方开发框架，在简易开发的过程中，将所学的知识融会贯通形成了自己的知识链。我不仅仅学了很多专业知识，还学会了如何查阅资料，学习新知识。经过这段时间的努力，由于能力的提升也让我的自信心得到了提升，让我能更加自信的面临以后工作中遇到的问题。在本次设计过程中汲取的东西，也是一笔宝贵的财富，让我受益匪浅。

# 致谢

忙忙碌碌了两个多月，我的毕业设计终于是迎来了尾声。纵观这整个过程，从最初的懵懵懂懂，到后面对项目流程的了如执掌，实在是经历了太多太多。因为自身能力的原因，项目的工时远远超过了自己的预期，整个项目也有很多不尽人意的地方，比如页面的布局和设计美感并没有达到论文初期设计的标准，许多功能并没有更贴合用户体验，即不人性化，更倾向于开发者的使用习惯。但是，我深知一个道理：一件事，不要在意结果如何，更可贵的是解决这件事的整个过程，所有的收货。

仔细想想，毕业设计是大学最后一次交作业了。很快，我就要步入社会，去到更广阔的天地。在此，我想感谢四年来给了我帮助的所有老师和同学，是你们给了我一个丰富的大学生活，给了我人生宝贵的财富。特别是感谢我的毕业设计指导老师张霞老师。张霞老师对学术的严谨，使我认识到，做事一定要做好自己。起初，在开题报告中，老师给我提供了许多建议和意见，这保证了我后面论文不会走太多弯路。感谢老师能耐心的督促我完成毕业设计，提供给了我很多毕设上难题的解决方法，比如：在对功能的筛选，研宄对象的确定以及研宄设计方面遇到困难的时候，张老师总是会很认真的给我指导。没有张霞老师的耐心教导，也许我完成毕业设计会更加困难。

# 参考文献

1. [] 刘威. 图书商城系统的Web服务器性能优化研究与实现[D].昆明理工大学,2017. [↑](#endnote-ref-0)
2. []耿庆阳. 基于Spring Boot与Vue的电子商城设计与实现[D].西安石油大学,2020.DOI:10.27400/d.cnki.gxasc.2020.000569. [↑](#endnote-ref-1)
3. [] 752018766@qq.com.网上购物国内外研究现状[N/OL],[2018-6-13].[网上购物国内外研究现状\_毕业论文 (youerw.com)](http://www.youerw.com/yanjiu/lunwen_17637.html). [↑](#endnote-ref-2)
4. [] 阿里巴巴集团.阿里云服务平台. 关于云对象存储技术OSS企划白皮书.产品简介[Z]，2021. [↑](#endnote-ref-3)