

**本科毕业设计(论文)**

**题目: 用户信息可隐藏的匿名购物系统设计与实现**

学院名称　 **计算机科学与技术学院**

　　　　　专业班级　 **软件工程xxxx**

　　　　　学生姓名　 **郭XX**

　　　　　学 号　 **xxxxxxxxxxxx**

　　　　　导师姓名　 **赵盛荣**

年 月 日

**齐鲁工业大学本科毕业设计（论文）原创性声明**

本人郑重声明：所呈交的毕业设计（论文），是本人在指导教师的指导下独立研究、撰写的成果。设计（论文）中引用他人的文献、数据、图件、资料，均已在设计（论文）中加以说明，除此之外，本设计（论文）不含任何其他个人或集体已经发表或撰写的成果作品。对本文研究做出重要贡献的个人和集体，均已在文中作了明确说明并表示了谢意。本声明的法律结果由本人承担。

　　　　　　　　　　　　　　　毕业设计（论文）作者签名：

　　　　　　　　　　　　　　　　　　　　　　　　　　年　　月　　日

**齐鲁工业大学关于毕业设计（论文）使用授权的说明**

本毕业设计（论文）作者完全了解学校有关保留、使用毕业设计（论文）的规定，即：学校有权保留、送交设计（论文）的复印件，允许设计（论文）被查阅和借阅，学校可以公布设计（论文）的全部或部分内容，可以采用影印、扫描等复制手段保存本设计（论文）。

指导教师签名：　　　　　　　　　　　毕业设计（论文）作者签名：

　　　　　年　　月　　日　　　　　　　　　　　　　　年　　月　　日

用户信息可隐藏的匿名购物系统设计与实现

**作 者 姓 名 郭XX**

**专 业 软件工程**

**指导教师姓名 赵XX**

**专业技术职务 副教授**

目 录

[摘 要 I](#_Toc42172746)

[ABSTRACT II](#_Toc42172747)

[第一章 绪论 1](#_Toc42172748)

[1.1 课题简介及目的 1](#_Toc42172749)

[1.2 课题研究意义 1](#_Toc42172750)

[1.3 研究现状和发展趋势 1](#_Toc42172751)

[第二章 系统开发环境及平台 4](#_Toc42172752)

[2.1 开发环境简介 4](#_Toc42172753)

[2.2 Ubuntu 18 4](#_Toc42172754)

[2.3 OpenJDK 4](#_Toc42172755)

[2.4 Intellij IDEA 4](#_Toc42172756)

[2.5 MySQL 5](#_Toc42172757)

[2.6 DataGrip 5](#_Toc42172758)

[2.7 Bootstrap Studio 5](#_Toc42172759)

[2.8 Firefox 5](#_Toc42172760)

[2.9 Nginx 5](#_Toc42172761)

[2.10 Tomcat9 5](#_Toc42172762)

[2.11 FastDFS 5](#_Toc42172763)

[2.12 Dubbo 6](#_Toc42172764)

[2.13 SpringBoot 6](#_Toc42172765)

[2.14 ActiveMQ 6](#_Toc42172766)

[2.15 Redis 6](#_Toc42172767)

[第三章 系统分析 7](#_Toc42172768)

[3.1 系统目标分析 7](#_Toc42172769)

[3.2 可行性分析 7](#_Toc42172770)

[3.2.1 经济可行性 7](#_Toc42172771)

[3.2.2 技术可行性 8](#_Toc42172772)

[3.2.3 风险隐私及对策 8](#_Toc42172773)

[3.3 系统需求分析 8](#_Toc42172774)

[3.3.1 用户管理模块 8](#_Toc42172775)

[3.3.2 商品管理模块 9](#_Toc42172776)

[3.3.3 商品搜索模块 9](#_Toc42172777)

[3.3.4 商品详情模块 10](#_Toc42172778)

[3.3.5 购物车模块 10](#_Toc42172779)

[3.3.6 身份认证模块模块 10](#_Toc42172780)

[3.3.7 订单模块 10](#_Toc42172781)

[3.3.8 支付模块 10](#_Toc42172782)

[3.3.9 商品评分模块 10](#_Toc42172783)

[3.4 业务流程分析模块 11](#_Toc42172784)

[3.4.1 SPU管理 11](#_Toc42172785)

[3.4.2 商品详情 11](#_Toc42172786)

[3.4.3 单点登录与用户认证 13](#_Toc42172787)

[3.4.4 支付 16](#_Toc42172788)

[3.5 数据流图 17](#_Toc42172789)

[3.6 性能需求 17](#_Toc42172790)

[第四章 系统设计 19](#_Toc42172791)

[4.1 系统架构设计 19](#_Toc42172792)

[4.2 项目结构设计 19](#_Toc42172793)

[4.2.1 Parent模块 20](#_Toc42172794)

[4.2.2 api模块 21](#_Toc42172795)

[4.2.3 common-util模块 21](#_Toc42172796)

[4.2.4 Web-Util模块 22](#_Toc42172797)

[4.2.5 Service-Util模块 23](#_Toc42172798)

[4.2.6 业务Web模块 23](#_Toc42172799)

[4.2.7 业务Service模块 24](#_Toc42172800)

[4.3 主要功能设计 24](#_Toc42172801)

[4.4 数据库设计 25](#_Toc42172802)

[4.4.1 概念结构设计 25](#_Toc42172803)

[4.4.2 数据库表设计 26](#_Toc42172804)

[第五章 系统开发 30](#_Toc42172805)

[5.1 项目结构与描述 30](#_Toc42172806)

[5.2 后台管理模块系实现 31](#_Toc42172807)

[5.2.1 子项目搭建 31](#_Toc42172808)

[5.2.2 主要页面 34](#_Toc42172809)

[5.2.3 主要代码 34](#_Toc42172810)

[5.3 商品详情模块实现 35](#_Toc42172811)

[5.3.1 主要页面 35](#_Toc42172812)

[5.3.2 主要代码 36](#_Toc42172813)

[5.4 搜索模块实现 37](#_Toc42172814)

[5.4.1 主要页面 37](#_Toc42172815)

[5.4.2 主要代码 37](#_Toc42172816)

[5.5 购物车模块实现 38](#_Toc42172817)

[5.5.1 主要页面 38](#_Toc42172818)

[5.5.2 主要代码 39](#_Toc42172819)

[5.6 单点登录模块实现 42](#_Toc42172820)

[5.6.1 主要页面 42](#_Toc42172821)

[5.6.2 主要代码 42](#_Toc42172822)

[5.7 订单模块实现 44](#_Toc42172823)

[5.7.1 主要页面 44](#_Toc42172824)

[5.7.2 主要代码 44](#_Toc42172825)

[5.8 支付模块实现 45](#_Toc42172826)

[5.8.1 主要页面 45](#_Toc42172827)

[5.8.2 主要代码 47](#_Toc42172828)

[5.9 订单模块完善 48](#_Toc42172829)

[5.9.1 主要页面 48](#_Toc42172830)

[5.9.2 主要代码 49](#_Toc42172831)

[5.10 商品评分模块 50](#_Toc42172832)

[5.10.1 主要页面 50](#_Toc42172833)

[5.10.2 主要代码 51](#_Toc42172834)

[5.11 后台管理模块完善 52](#_Toc42172835)

[5.11.1 主要页面 52](#_Toc42172836)

[5.11.2 主要代码 52](#_Toc42172837)

[第六章 系统测试与调试 54](#_Toc42172838)

[6.1 测试目的 54](#_Toc42172839)

[6.2 测试方法 54](#_Toc42172840)

[6.3 测试环境与测试条件 54](#_Toc42172841)

[6.4 测试项目 54](#_Toc42172842)

[6.4.1 单点登录测试 55](#_Toc42172843)

[6.4.2 订单和支付模块模块测试 55](#_Toc42172844)

[6.5 测试结论 57](#_Toc42172845)

[6.6 遇到的问题及解决方案 57](#_Toc42172846)

[6.7 系统存在的不足 59](#_Toc42172847)

[第七章 总结 61](#_Toc42172848)

[参考文献 62](#_Toc42172849)

[致 谢 63](#_Toc42172850)

# ****摘 要****

随着互联网行业的发展，网络购物行业也逐步趋于成熟，不同的网络购物平台发展出了不同的服务，完善了市场的分工。随着IT技术的发展，以及公众对网络安全和隐私的关注度逐渐提高，网络安全变得越来越重要。因此安全可靠，并且不侵犯用户隐私的网络购物平台在未来会有更大的发展空间，并且这将成为主流。

目前的网络购物平台在用户使用的时候会搜集大量的用户信息，这些用户数据将会进行脱敏处理，然后通过计算得到用户画像，分析用户喜好和最近可能要购买的商品，最后将商家的广告或商品信息定向推销给用户。借助国家实名制的浪潮，这些平台逐步取得了获得用户信息的“合法权利”。从用户的浏览历史到购物记录，从付款方式到快递物流，还有智能手机的各种权限的获取，我们就好像在这些平台上“裸奔”。而匿名购物中，用户仅仅是购物，不会在平台上进行任何与购物无关的活动，平台也只是提供交易的场所。需要指出的是，与暗网的交易模式不同，在本文中匿名购物是指在网购时，仅在必要时候使用有限的用户信息，利用最少用户信息实现身份认证以及可靠购物。

本文就分析探讨匿名网络购物的基础上，逐步开发出能为用户提供隐私保障和安全保障的用户信息可隐藏的匿名购物系统。本系统主要包含商品管理模块，商品详情模块，搜索模块，购物车模块，身份认证模块，用户管理模块，订单模块，支付模块，评论模块。

关键字**：**电商平台 网络安全 匿名购物 身份认证

# ****ABSTRACT****

As the development of IT, online-shopping has been more perfect. Some different service which improve the division of market has been developed by various E-commerce platforms. As the cyber skill developing and the public’s continues attention of cyber-security and privacy, the cyber-security has been more important. As a result, there has more potential for E-commerce platform which is secure, dependable, and uninvasive of privacy, and it will be the trend of E-commerce platform in the future.

The current E-commerce platform collects a large amount of user information when users use it. These user data will be desensitized, and then the user portrait will be calculated, and finally the merchant’s advertisement or product information will be recommended to user according to the portrait. Under the help of the national real-name legislation, these platforms have gradually achieved "legal full rights" to obtain users’ information. From the users’ browsing history to shopping records, from payment methods to express delivery and logistics, and the acquisition of various permissions on smartphones, we seem to be "streaking" on these platforms. In anonymous shopping, users are only shopping, they will not conduct activities those are nothing to do with shopping on the platform, and the platform is only a place for trading. It should be pointed out that, unlike the dark web transaction mode, anonymous shopping in this article refers to only using limited user information when necessary, and using minimal user information to achieve identity authentication and reliable shopping.

Based on the analysis and discussion of anonymous online shopping, I will gradually develop an anonymous shopping system that can hide user information and provide users with privacy and security protection. This system mainly includes E-commerce product platform management module, product detail module, product search module, shopping cart module, identity authentication module, user management module, order management module, payment module and comment module.

**Keyword:** E-commerce platform; Cyber-Security; Anonymous Shopping; Identity Anthentication

# 绪论

## 课题简介及目的

随着信息技术的发展，数据逐渐成为很多公司的重要财富。目前大多数电商平台都会在用户使用过中搜集用户信息，浏览历史，购物记录，手机GPS位置信息，相机权限，录音权限，拨号、发短信权限，日历权限等等。他们的模式是，除非该信息法律规定不能搜集或用户可选择并且拒绝了，否则使用一切手段搜集一切可以搜集到的信息。这些信息会被脱敏[[[1]](#endnote-2)]，然后利用各种算法计算出用户画像，然后向用户定向推销商品或广告，从中谋取高额利润。而搜集这些信息就意味着这些信息有被泄露，被滥用，被买卖的风险。

避免这一切的方法之一就是仅搜集必要的用户信息，使用最小权限集。密码学的发展为这些安全技术的实现提供了可能。

因此，我们需要一个可靠的网络匿名购物平台，既方便用户购物，又能保护用户隐私，因为不搜集不必要的用户信息，所以理论上根本不存在用户数据泄露的风险。用户可以使用高效的分类检索系统，花最少的时间找到自己喜欢的产品，而不是在系统推荐中迷失自我。

## 课题研究意义

真实世界的交易通过使用现金给了消费者保留隐私的特权，因此不用提供类似姓名和家庭住址之类的个人信息[[[2]](#endnote-3)]，因此这就需要一种能提供线下等价体验的交易系统，即买卖双方就好像面对面一手交钱一手交货一样，卖家不需要知道买家详细的身份信息，而买家也不需要那些自己根本不需要的服务。简单直接的用户体验是中国所有的电子商务平台所欠缺的。尽管由于电子商务的特殊性，有些服务的使用可能会麻烦一些，但是整体还是缺乏简洁的感官。本课题将设计出足够简洁同时又有安全保障的可靠的匿名购物系统，对于规范互联网行业普遍存在的用户数据搜集行为具有重要意义。

## 研究现状和发展趋势

国内著名的购物平台，淘宝和京东，都没有提供匿名购物的服务，在淘宝下单时提供的匿名购买选项也仅仅是在评论商品时不会显示用户的会员名和姓名而已。国外的著名电商平台如亚马逊，亚马逊的匿名购物也是体现在评论商品时不会显示全部的用户名，卖家无法得知买家的真实身份和联系方式，但是亚马逊本身保存着用户的信息。网购的的隐私不在于卖家对于用户的信息掌握了多少，而是平台本身就掌握了大量的用户数据。近年来，一连串的电商企业客户数据泄露事件，比如阿里巴巴、京东、当当、Uber 等发生之后，不仅使得企业受到处罚或高管被迫辞职，而且这类企业未能保护消费者数据免受互联网黑客入侵的一系列备受瞩目的数据泄露事件，在《电子商务中个人信息泄露成因分析与防范研究》[[[3]](#endnote-4)]中对三起信息泄漏事件的分析中不难看出，内部人员利用职务之便可以很轻松的获取到用户的数据，包括用户名、密码、电子邮件地址、QQ帐号、电话号码和账户ID号等。而这是大多数理性的消费者不希望的，尽管消费者必须将自己的一些信息交给平台。

非对称密钥，也就是公钥密码系统，类似于一把钥匙一把锁，每个用户持有一对密钥，一个称为公钥，用于唯一标识一个用户，对所有人公开，另个一为私钥，仅持有者自己知道。可以用来实现端点鉴别，使用Hash函数对要传送的明文进行Hash运算生成消息鉴别码，使用发送者的私钥加密消息鉴别码，也就是生成签名，接收方使用发送方的公钥解密签名，从而得到原始的消息鉴别码，接收方再对收到的信息使用与发送方相同的密码散列函数运算得到消息鉴别码，与解密得到的消息鉴别码对比，从而实现端点鉴别。为了防止公钥被他人冒用，而引入了受信任的第三方，为了应对公钥密码机制面临的中间人替换公钥攻击的风险，目前有三类主要的解决方案[[[4]](#endnote-5)]，第一种是使用公钥基础设施（Public Key Infrastructure: PKI）技术，它使用了一整套完整的证书管理机制来实现身份认证服务，但是管理成本相对较高；第二种是使用基于标识的密码技术，这种标识在系统内是唯一的，只有用户本人通过认证，登录系统才能使用，从而确保了用户身份的唯一性。在无证书的公钥密码[[[5]](#endnote-6)]体系中，仍然需要一个可信的第三方密钥生成中心KGC（Key Generation Center）,它拥有系统的主密钥（Master Key），KGC的作用是根据用户的身份和系统主密钥计算用户的部分私钥,并安全地传送给用户，另一半则由用户选取，公钥由用户自己的秘密值、身份和系统参数计算得出,并以可靠的方式公布，攻击者无法伪造签名，因为伪造的签名将无法通过端点鉴别。使用这种技术，可以降低成本，同时提高安全性。

关于安全支付，为了避免重复造轮子，决定使用支付宝作为首选的支付工具。支付宝为开发者提供了完整的开发文档。但是为了避免支付宝通过对比用户的支付转账以及平台给商家的转账记录而的得出用户的购物详情，借鉴了洋葱路由的思想，Tor是一个三重代理, Tor客户端先与目录服务器通信获得全球活动中继节点信息, 然后再随机选择三个节点组成电路（circuit），用户流量跳跃这三个节点 （hop）之后最终到达目标网站服务器[[[6]](#endnote-7)]。而用户在确认收货后，平台将用户事先支付的钱分三次转给商家，这三次转账总的金额是商品的价格，每次转账的数目随机生成，并且在生成转账金额的同时生成一个随机的转账时间，在系统全局保存一个转账队列，通过时间先后顺序将转账插入到队列中，在设定的时间自动转账，单笔订单完成全部3笔转账后系统将通知商家，同时更新订单状态。

个人信息被称为“新石油”。如果某一商家把控有大量的个人信息，通过对个人信息的分析和整合研究，就可以识别利用并创造市场需求，成为商业竞争中的佼佼者，为自身带来巨额的经济财富[[[7]](#endnote-8)]。而对电商平台的处罚相比他们的收入而言是很不对称的，整个行业以及形成了对于用户信息利用的趋势，大的平台可以做一些脱敏处理，一定程度上保护了使用者的隐私，但是小的公司很难做到这一点。垃圾营销短信的出现就是建立在大量用户信息泄露的基础上。

# 系统开发环境及平台

## 开发环境简介

越来越多的复杂系统平台采用分布式的架构，并且前后端分离，不同的模块分开开发，既降低了开发难度，又降低了系统耦合度，同时提高了系统的可维护性，还提高了软件代码的复用。因此本系统将采用分布式的架构，使用OpenJDK11，服务器使用Ubuntu18，后端系统采使用集成开发环境Intellij IDEA进行开发，前端的页面设计使用Bootstrap Studio设计，同时大量的图片文件使用阿里巴巴的分布式文件存储系统FastDFS保存。还有很多用到的工具将在之后的小节详细说明。

## Ubuntu 18

Ubuntu是基于Debian的Linux发行版，其中Ubuntu Server的发行版不包括Gnome桌面环境和一些桌面版软件。服务器使用了Vultr和华为云的Ubuntu18.04，华为云的服务器提供后端业务的处理，Vultr的服务器负责处理前端请求。华为云的服务器处理能力更强，但是在国内只有网站备案之后基于http协议的网络应用才能使用域名解析服务，Vultr的服务器不需要备案就可以配置域名访问，通过将Vultr服务器的一些请求转回华为的服务器来处理，再返回到Vultr的服务器，最后返回到用户，一定程度上保护了数据服务器的安全，同时将访问频率高的数据放在Vultr的服务器上，可以降低数据服务器的访问压力。前后端分离是指将前端和后端从之前的相互融合中分离开来[[[8]](#endnote-9)], 两者不再共用一个Server, 前端作为一个独立Server存在，通过前后端的分离，方便实现负载均衡，提高了系统可靠性、稳定性以及高并发性。

## OpenJDK

OpenJDK11是最新的一个长期维护的OpenJDK版本，同时由于是开源的，避免了使用Oracle JDK所面临的版权问题。OpenJDK11包含了JDK8、JDK9、JDK10中的新特性，同时又删减了大量不再适用的代码，另外Ubuntu18默认的JDK源就是OpenJDK11，安装也更加方便。

## Intellij IDEA

IntelliJ IDEA是有JetBrains公司开发的、跨平台的Java集成开发环境工具软件。具有很多强大的功能，如代码自动补全、变量名智能建议，同时还有丰富的插件。

## MySQL

MySQL是一个关系型数据库管理系统，本系统使用MySQL8作为数据库服务器。

## DataGrip

DataGrip是有JetBrains公司开发的、跨平台的数据库管理工具，面向使用SQL数据库的开发人员。项目运行后数据库将运行部署在华为的云服务器上，对数据库的直接管理通过DataGrip来实现。

## Bootstrap Studio

Bootstrap Studio是专有的Web设计和开发应用程序，可以快速的构建HTML页面。 它提供了大量用于构建响应页面的组件，包括页眉，页脚，画廊和幻灯片以及基本元素（例如span和div）。

## Firefox

Firefox是一个由Mozilla基金会赞助的开源、支持多平台的网页浏览器，内置了很多易于开发调试的工具。

## Nginx

Nginx是一款免费的、开源的高性能异步框架的网页服务器，也可以用来作为反向代理，负载均衡和web缓存。Nginx是一卷面向性能设计的HTTP服务器，占用内存少，稳定性高，有丰富的第三方库和模块，配置灵活，可以同时处理大量并发连接。Tomcat无法处理大量的高并发连接，Nginx作为Tomcat前端负载均衡器的组合方式, 被许多企业和开发团队所采用[[[9]](#endnote-10)]，Nginx处理静态页面，同时在后端配置多个Tomcat实例，利用Nginx实现多个实例之间的负载均衡。

## Tomcat9

Tomcat是Apache 基金会赞助的开源项目，使用Java语言开发，最新版是2020年3月11日发布的9.0.33，出于安全以及技术支持的原因使用了9.0.30的版本。主要负责处理后台数据以及相应前端的服务。

## FastDFS

FastDFS是阿里巴巴开发者余庆开发的分布式文件系统。

## Dubbo

Dubbo是阿里巴巴公司开源的一个高性能、轻量级的开源Java RPC框架，同时可以和Spring框架实现无缝集成。它提供了三大核心能力：面向接口的远程方法调用、智能容错和负载均衡，以及服务自动注册和发现 [[[10]](#endnote-11)]。Dubbo是面向运输层的通信协议，Spring家族中的SpringCloud也是RPC框架，但是SpringCloud是面向应用层的框架，使用HTTP作为通信协议，因此同等条件下Dobbo表现更加优异。

## SpringBoot

SpringBoot是Spring家族的框架之一，特别适用于快速搭建分布式的应用。采用了特定的配置文件，而且Intellij IDEA内置了SpringBoot的插件，可以快速搭建应用，使用maven作为依赖管理，非常有利于模块化开发。

## ActiveMQ

ActiveMQ是又Apache基金会赞助开发的开源的消息中间件，主要是在高并发的情况下，很多请求来不及做同步处理，请求往往会发生堵塞，从而导致服务器压力过大直接宕机，导致应用无法正常工作。通过使用消息队列，可以将原本串行处理的业务流程改为并行异步处理。消息中间件主要是采用用高效且可靠的消息传递机制，实现不同系统应用之间的进程通信，这些应用可能运行在多台完全不同的服务器上。对于消息中间件，常见的角色大致也就有Producer（生产者，发送消息的进程）、Consumer（消费者，监听消息队列的进程）。

## Redis

Redis 是一个开源（BSD许可）的，基于内存的数据库系统，主要使用键值对的形式存储数据，通过将系统中经常被访问的数据存储到redis中，可以极大的降低数据库的访问压力，提高前台页面的响应速度。

# 系统分析

## 系统目标分析

本系统旨在建立一个匿名购物系统，即使用最少的用户隐私信息实现可靠的网络购物。因此主要有以下几个方面的目标：

1. 本系统首先是作为一个电子商务系统，应该有基本的商品管理；
2. 除了商品的管理外将商品展示出来也是电子商务系统最基本的功能之一，因此前台系统需要用不同的方法将商品展示出来，首页基于评分、订单量、评价等因素，分类检索——根据商品的分类以及商品的平台属性，关键词搜索——通过商品的名称中关键字快速根据用户输入的关键字检索到相关的商品；
3. 购物车也是电子商务系统必不可少的功能模块之一，应该在用户登录和未登录的情况下都可以使用，未登录的情况下购物车中的数据保存在用户浏览器cookie中，登陆后则保存在缓存中；
4. 订单系统是连接消费者和商家的重要桥梁，通过结算，下单，用户实现了最终的购物，卖家需要使用管理系统完成订单的处理和物流的管理；
5. 支付是电子商务系统的核心问题之一，选择支付宝作为主要支付渠道，同时保留微信支付和PayPal支付接口，方便以后扩展；
6. 另外由于系统采用分模块开发，前后端分离，不同的模块之间的调用涉及身份认证，要有一个认证中心解决用户身份认证的问题；
7. 用户对订单的评价对于系统中商品检索功能至关重要，因此需要一个评论模块用来处理用户对与商品的评论，更新商品的评分。

## 可行性分析

本系统的意义在于从根源上解决用户隐私泄露的问题——不搜集不必要的用户信息，因此可行性分级基于以下几个方面：

### 经济可行性

由于互联网行业的快速发展，各种服务层出不穷，传统的电子商务系统，比如京东、淘宝、天猫已经占据了中国市场的半壁江山，新兴的电子商务系统比如拼多多，也已经快速占领了低收入消费群体，美团和饿了么则占据了中国的外卖市场。对于本系统而言，已经很难在市场中占据一席之地，由于不搜集用户的不必要信息，所以不存在以牺牲用户隐私为代价而换取所谓方便的推荐系统，对于不在乎个人隐私的消费者而言，他们更愿意选择那些传统的电商；对于隐私敏感的群体更愿意选择本系统，因为以前他们是没得选择，而本系统将可以为他们提供新的选择。

### 技术可行性

电子商务系统的日趋成熟可以为本系统的开发和设计通过良好的借鉴，不论是商品管理、商品展示还是身份认证。本系统的数据库设计借鉴了尚硅谷“谷粒商城”教学项目所使用的数据库设计，同时又根据需求对数据库做了很多调整；网页使用Bootstrap Studio内置的模板快速开发，使用thymeleaf 和 JQuery 框架在前端进行渲染；微服务管理使用阿里巴巴的dubbo，这是阿里巴巴大多数产品都在使用的服务注册框架；考虑到了数据访问压力，本系统将常用的数据放在redis缓存中，因为用户操作过程中对数据库读的操作的频率远远高于写的操作，将常用的数据放在缓存中可大大降低数据库的访问压力；电商系统最重要的是数据的安全，由于本系统规模较小，使用一般的拦截器配置和简单的访问控制就可以实现安全访问。

### 风险隐私及对策

由于时间问题，网站没有备案，后期可能无法在国内上线运行，针对这个问题，可以在后期向公安局申请备案；由于用户真实身份，包括身份证号等数据不被搜集，仅仅凭很少的信息网络购物，所以交易可能不会受到法律的保护，针对这个问题，可以建立简单的信用机制以及商家担保交易机制，一定程度上保护消费者与商家的交易；由于进行的是商业活动，需要很完善的各类手续，短期内不肯准备齐全，因此本系统可能会面临非法交易的指控，这个问题需要在后期专门做工作，去申请各类手续。

## 系统需求分析

本系统主要的模块包含以下七个模块，以下对于从不同用户的角度出发分析每个模块的需求：

### 用户管理模块

本系统的主要用户包括消费者，卖家，平台管理员。用户模块主要包括用户身份以及个人信息的需求，具体需求如下：

**消费者：**

注册账户，修改密码，修改个人用户名，忘记密码后重置密码，重置账户（清空所有账号信息），设置默认收货地址；

**卖家：**

注册卖家账户，修改密码，修改个人用户名，忘记密码后重置密码，注销卖家账户（卖家账户注销必须在最后一笔订单完成的两个月后，确保用户可以找到售后）；

**平台管理员：**

由于平台管理员身份的特殊性，不提供注册功能，而是直接在数据库中写入管理员的身份，同时管理员账户具有修改密码，忘记密码后重置密码的功能。

### 商品管理模块

商品管理模块的使用者包括了本系统的两类用户，卖家和平台管理员，具体需求如下：

**平台管理员：**

1. 管理平台属性（本系统对商品的分类管理采用了三级分类的模式，一级分类包括手机，家用电器，数码，家居家装，电脑办公等21个分类，在手机的二级分类包括手机，通讯运营商，手机配件三个分类，手机的三级分类下包含手机，对讲机，通过三级分类就可以将所有商品的分类结构划到比较细致，手机的具体分类包含着平台属性，平台属性是指本平台所维护的商品的属性，手机的平台属性包含了机身尺寸，内存，屏幕尺寸，机身存储，cpu，外壳，待机时间等22个属性，每个属性又包含若干个属性值），平台属性管理包括对平台属性的增删改查；
2. SPU管理（Standard Product Unit，标准产品单元，一个系列的产品可以称为一个SPU），对SPU的管理包括对SPU的增删改查。

**卖家：**

1. 查找SPU；
2. 库存保有单位管理（Stock keeping Unit，SKU是对每一个产品和服务的唯一标示符，同一系列下的某一个具体型号的产品可以称为一个SKU），SKU的管理包括对SKU的增删改查。

### 商品搜索模块

商品搜索模块的主要使用者是消费者：

1. 分类检索商品，通过三级分类以及具体商品分类下对应的平台属性和商品的品牌查找商品；
2. 模糊搜索，通过输入关键字查找商品；
3. 查看商品详情，通过点击搜索结果查看商品的具体信息。

### 商品详情模块

商品详情模块的使用者面向所有用户，主要需求是展示商品详情。

### 购物车模块

购物车模块的使用者主要是消费者，具体需求如下：

1. 在商品详情页将商品添加到购物车；
2. 从购物车中删除已经添加的商品；
3. 修改购物车中商品的数量；
4. 改变购物车中商品的选中状态。

### 身份认证模块模块

身份认证模块的使用者涉及到消费者、卖家、平台管理员，主要需求如下：

1. 验证登录状态；
2. 验证用户登录信息是否正确；
3. 验证用户令牌。

### 订单模块

订单模块的使用者主要包括消费者和卖家，具体需求如下

**消费者：**

1. 选择购物车中的商品使用支付宝结算下单；
2. 在个人中心查看订单状态；
3. 确认收货并评价。

**卖家：**

1. 再管理页查看订单；
2. 发货；
3. 到货。

### 支付模块

支付模块的使用者主要是消费者，主要需求：确认订单信息，选择支付方式（支付宝），使用支付宝扫码付款。

### 商品评分模块

商品评分属于系统的功能，本系统的三类用户都不会用到。商品的评分主要是对商SPU的评分，同一SPU包含不同型号的产品，这也是目前大型电商平台主要使用的评分策略，商品评分展示在商品的搜索结果，详情页中。

## 业务流程分析模块

本小节主要包含了系统核心业务的业务流程。

### SPU管理

图片文件上传

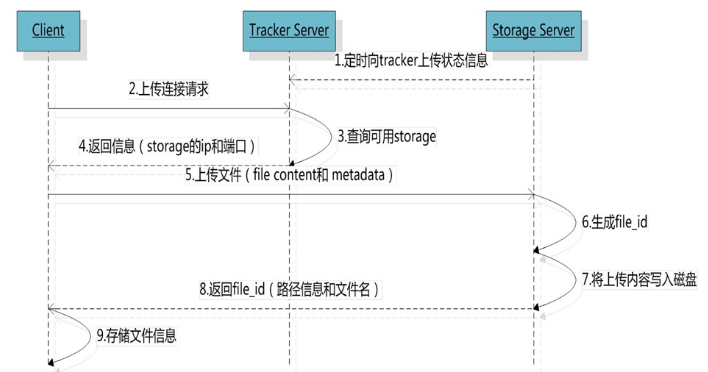


图3-1 FastDFS文件上传

图片文件使用分布式文件存储系统FastDFS来存储，整个上传流程分为如下几个步骤：

1. Web应用发送连接请求到跟踪器；
2. 跟踪器查询可用的存储节点，将改存储节点的信息返回给web应用；
3. Web应用直接向上一步返回的存储节点发送要存储的文件；
4. 存储节点对于上传的文件，生成一个唯一的文件id，并将文件写入服务器磁盘；
5. 存储节点返回文件id，web应用调用相应服务将文件id写入数据库。

### 商品详情

查看商品详情

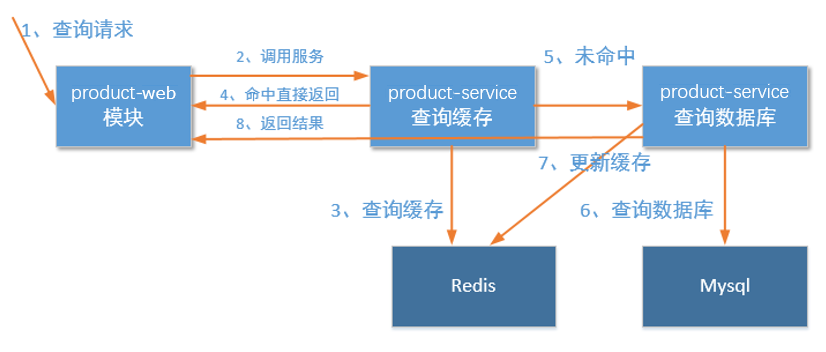


图3-2 商品详情

**详细步骤：**

1. 用户点击搜索结果中的商品或首页的商品，发送查询请求；
2. 控制器调用product-service相应方法；
3. product-service查询Redis缓存；
4. 如果缓存中有要查询的对象则返回；
5. 如果没有就查询MySQL数据库；
6. 得到查询结果，将查询结果写入缓存，并将查询结果返回前端。

### 单点登录与用户认证

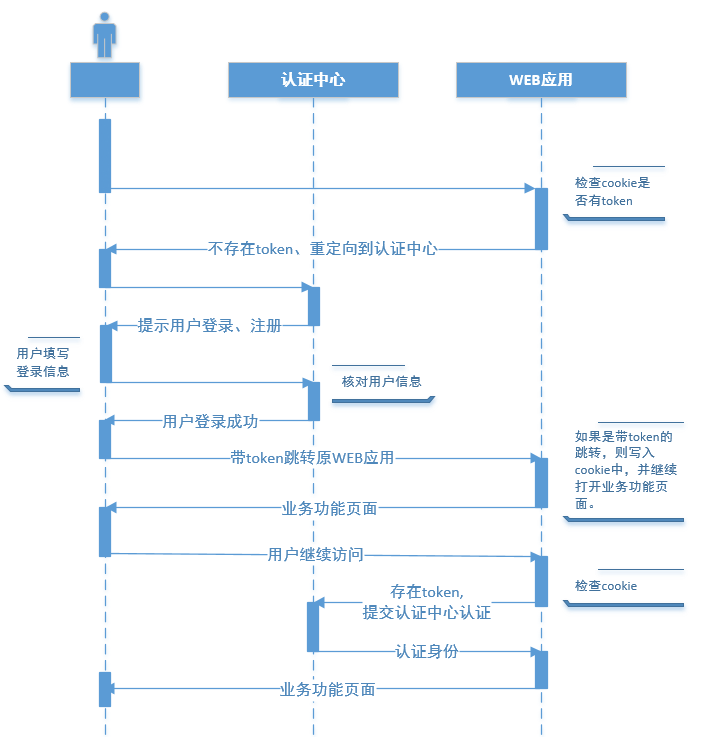


图3-3 单点登录流程

**用户登录流程：**

1. 用户在登录页填写用户名和密码或邮箱和密码；
2. 请求发送至认证中心，认证中心调用用户Service，核对用户信息；
3. 用户Service首先检查缓存中是否有用户信息，该用户可能不久之前登录过；
4. 如果缓存中有用户信息，则得到缓存中的用户信息；
5. 通过BcrypetPasswordEncoder检查用户填写的密码与数据库中加密后的密码是否匹配；
6. 如果匹配返回缓存中的用户信息，如果不匹配返回null，表示登录失败；
7. 如果缓存中没有用户信息，可能是之前用户没有登录过或缓存中的身份信息已经过期，这时通过用户名查询数据库，由于每个用户的用户名是唯一的，所以要么能查询到一条用户信息，要么查不到；
8. 如果查询到一条用户信息，重复步骤5和6；
9. 如果没有查询到，则返回空值null；
10. 认证中心得到的返回值不为空，表示登录成功，之后通过JWT（Json Web Token，是一个紧凑的采用URL安全表示方法的声明，用于在两方之间传输[[[11]](#endnote-12)]）生成令牌，改令牌包含三部分；
    1. 头（图3-4中HEADER部分）：包含了令牌类型和之用的签名算法，本系统使用HS256(HMAC-SHA256)算法作为签名算法；
    2. 载荷（图3-4中PAYLOAD部分）：包含了用户id和用户名；
    3. 签名（图3-4中VERIFU SIGNATURE部分）：使用HS256算法生成的签名，签名部分会使用一个密钥值和盐值，盐值是为了加入更多的随机性，盐值在本系统中使用用户的ip地址，好的密钥值的选取可以保证签名的安全

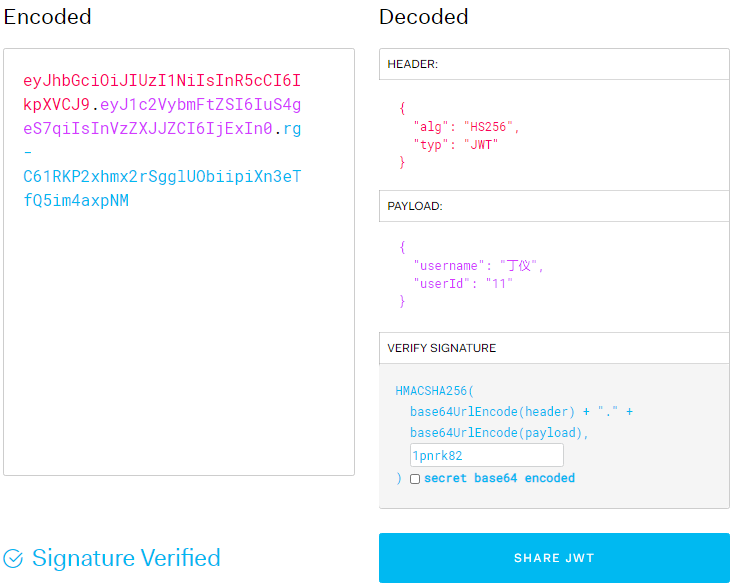


图3-4 JWT加密

1. 认证中心返回令牌，令牌将会存储在用户的浏览器cookie中。

**单点登录认证**

通过专门的认证中心，解决了多应用下的用户身份认证问题。在单应用系统的模式中，可以使用session存储关于用户身份的信息，但是在SOA的架构下，用户身份信无法使用session存储，因为session无法跨应用实现信息共享。单点登录认证结合自定义注解@LoginRequired，可以解决这个问题。LoginRequired注解有一个属性为LoginSuccess，默认值问true，标志改方法之后再登陆后才能使用，如果该方法不需要登录也可以使用，可以将该属性的值设置为false。

在所有页面跳转的控制器请求上加入LoginRequired注解，通过该注解，会进入到授权了拦截器，拦截器定义在了Web-Util中，系统中所有的Web应用都可以使用具体步骤如下：

1. 拦截器检查该方法的LoginRequired属性判断该方法的使用是否必须登录；
2. 拦截器检查用户的令牌，如果已经登录过，该令牌会存储cookie中，cookie中的命名为oldToken，在用户的request请求中也会携带改令牌，命名为newToken，根据oldToken和newToken是否为空可以有表3-1中的的笛卡尔积：

表3- 1 token与登录状态

|  |  |  |
| --- | --- | --- |
|  | NewToken为空 | NewToken不为空 |
| oldToken为空 | 从未登录 | 刚刚登录 |
| oldToken不为空 | 之前登录过 | oldToken过期 |

1. 如果newToken不为空，则向认证中心发送认证请求, 请求参数为newToken；
2. 认证中心收到认证请求，通过使用生成token时的密钥和用户的ip地址，解密令牌，将存储的用户名和用户id转换为明文放入返回结果中，返回结果还包括解密是否成功的标志；
3. 如果该方法的使用必须在登录状态下进行，但是用户没有登录，则会重定向到登录页；
4. 如果用户已经登录，也即token不为空，则判断(4)中的验证结果，如果返回结果中是否成功的标志为fail，则重定向到登录页，如果为success，则将结果中的username和userId放入request；
5. 更新cookie中的token，将oldToken的值设置为newToken，然后返回执行控制其中的方法，控制器将能在request中取得username和userId；
6. 如果该方法在用户没有登录的情况下也可以使用，这时只需要判断第(4)步的返回结果中成功标志是否为success，如果为success则执行和(6)中seccess情况下相同的动作和(7)中的动作，其他情况下都什么都不做。

### 支付

支付宝支付流程如图3-5所示：

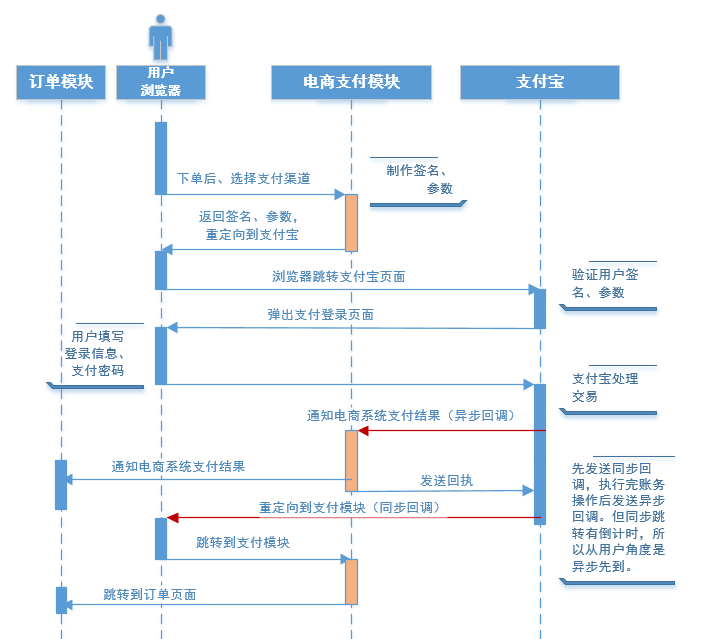


图3-5 支付流程

**支付流程**

1. 消费者确认订单并点击支付支付；
2. 重定向到支付系统；
3. 消费者选择支付宝作为支付方式并向支付系统提交支付请求；
4. 系统返回给客户端一个重定向命令；
5. 消费者向想支付宝发送请求；
6. 支付宝给客户端返回一个支付页面；
7. 消费者扫码付款；
8. 支付宝处理交易并返回结果；
9. 修改订单支付状态；
10. 返回给用户结果；
11. 支付宝通知系统的服务器支付结果；
12. 平台处理订单、物流。

## 数据流图

单点登录数据流图如图3.6所示

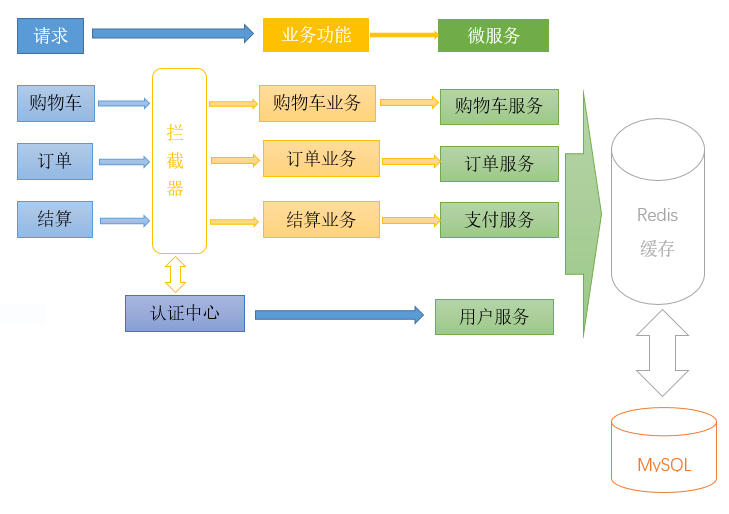


图3-6 拦截与身份认证

## 性能需求

本系统的设计充分考虑到了高并发情况下可能带来的性能消耗。本系统预计用户数为10000人，可以同时处理1000条并发的连接。SpringBoot内置了tomcat，每个tomcat服务器可以承载200条并发连接，而每个SpringBoot应用的运行平均需要500M的内存，一个4G内存双核心的云服务器可以运行3个SpringBoot应用。系统最终大约会有15个SpringBoot应用，每个应用只运行一个实例就需要最少5台服务器。而要搭建集群就需要更多服务器，考虑到有些应用不常用，比如后台管理，一个实例就足以承载所有的并发连接，而有些应用使用频率很高，比如商品首页，详情处理，搜索等，这些应用每个都需要至少3到5个实例，这就需要至少20台云服务器才能处理1000条并发的连接。

本系统使用nginx作为负载均衡服务器，单个nginx应用的处理能力是有限的，因此将需要用到层级架构，第一层只使用一个nginx服务器，将收到的请求按三级域名转发到不同的第二层nginx服务器，第二层nginx再按照指定的分流策略将请求分发给SpringBoot web应用。

本系统使用 FastDFS作为文件存储服务器，由于电商网站需要处理大量的图片数据，图片数据对服务器的磁盘空间又更大的需求，因此也至少需要两台4G内存双核心的服务器，而且磁盘空间为500G。

本系统使用了Dubbo作为注册中心来协调不同应用的访问，如果仅使用一个注册中心，那么该服务器一旦宕机将会使所有应用都无法正常访问。因此Dubbo注册中心至少需要三个实例。

本系统使用Redis作为缓存服务器，由于数据是直接存储在内存上的，可以很大程度地减小数据库的访问压力，同时意味着缓存服务器必须使用内存较大的服务器作为运行环境。MySQL的处理能力有限，如果缓存服务器宕机将使数据库服务器承受数倍的访问压力。因此缓存服务器也需要多个实例才能运行。

本系统使用MySQL作为数据库服务器，一个4G内存2核心10M带宽的服务器大概能承载1000条并发连接，而且考虑到数据库服务器可能回宕机，也将需要多个服务器运行。

本系统使用了ActiveMQ作为消息中间件，有些服务比如登录系统后cookie中购物车信息和数据库中购物车信息的合并，如果在登录验证通过之后就进行合并，那么就只能在合并完成后才能返回之前的页面，如果合并过程中出现数据处理异常，用户将无法返到登录前的页面。使用消息队列的模式，在登录之后发送合并购物车的消息，同时返回到前台页面，让登录和合并购物车作为并行的任务去执行而不是串行执行。因此ActiveMQ服务器对于系统的正常运行也至关重要，也需要运行在多个服务器来确保至少同时有个一服务器在正常工作。

# 系统设计

## 系统架构设计

本系统采用SOA（Service-Oriented Architecture, 面向服务）的分布式架构，如图4-1所示，因为考虑到未来功能的拓展以及对可能存在的高并发的适应，采用面向服务的架构，将应用的不同功能单元也即服务，进行拆分，通过远程过程调用（Remote Procedure Call, RPC）框架使得不同的服务可以交互，完成特定的功能。

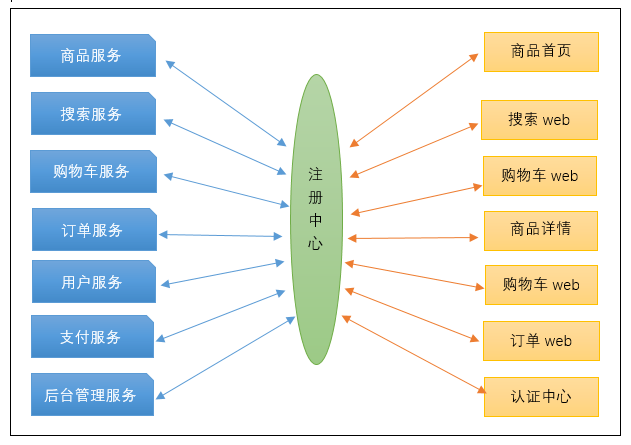


图4- 1 注册中心与SpringBoot应用

所有的服务向注册中心注册，注册中心为每个服务分配一个代理地址；所有的web应用通过远程注入的方式，从注册中心获得所需服务的代理访问地址，web应用和service应用使用JSON的数据格式和dubbo协议进行交互。

注册中心使用阿里巴巴的开源框架Dubbo，Dubbo主要由Provider（服务的提供者）、Consumer（服务的消费者）、Registry（服务注册中心）、Minitor（监控中心）四部分组成，所有的服务类应用都是服务的提供者，所有的web应用都是服务的消费者。

## 项目结构设计

使用maven的以来管理，通过父工程将项目中所有用到的第三方依赖包管理起来，简化了项目的以来版本管理。项目的结构设计如图4-2所示：

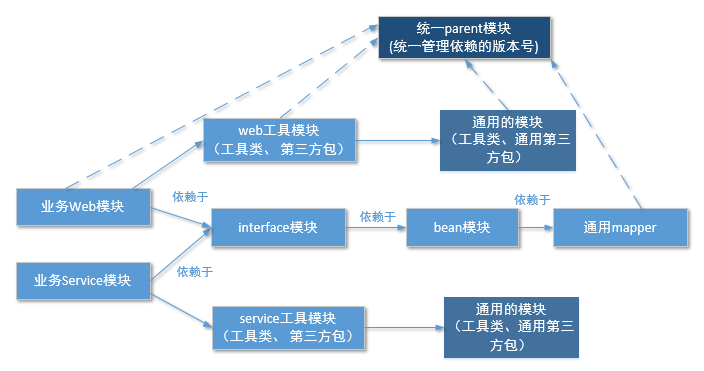


图4- 2 项目结构设计图

项目中的模块划分按照是否SpringBoot应用、应用的类型、以及模块在项目中的作用主要分为Parent模块、Web工具模块、Service模块、Interface模块、通用模块、业务Web模块、业务Service模块，接下来的小节将介绍这些模块。

### Parent模块

Parent模块中定义了项目的名称，版本，项目GroupID，第三方jar包的版本等，主要代码：

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version> <fastjson.version>1.2.68</fastjson.version>

<dubbo-starter.version>1.1.3</dubbo-starter.version>

<dubbo.version>2.6.8</dubbo.version> <zkclient.version>0.11</zkclient.version>

<mybatis.version>2.1.2</mybatis.version>

<nekohtml.version>1.9.22</nekohtml.version>

<xml-apis.version>1.4.01</xml-apis.version>

<batik-ext.version>1.12</batik-ext.version>

<jsoup.version>1.13.1</jsoup.version>

<httpclient.version>4.5.5</httpclient.version>

<commons-lang3.version>3.10</commons-lang3.version>

<mapper-starter.version>2.1.5</mapper-starter.version>

<jedis.version>2.9.0</jedis.version> <jest.version>6.3.1</jest.version>

<jna.version>5.5.0</jna.version>

<beanUtils.version>1.9.4</beanUtils.version>

<fastdfs-client-java.version>1.27-SNAPSHOT</fastdfs-client-java.version>

<pagehelper.version>4.1.1</pagehelper.version>

<redisson.version>3.10.5</redisson.version>

<jjwt.version>0.9.1</jjwt.version>

<activemq-pool.version>5.15.12</activemq-pool.version>

<mail.version>1.5.0-b01</mail.version>

</properties>

Parent模块的作用仅在于统一所有子模块的依赖包版本，不包含具体的java代码，最后以pom的形式打包。

### api模块

因为SOA架构的项目需要不同模块之间的相互调用，所以对于业务服务类的接口，应该放在公共接口模块中，方便所有其他模块的调用；同时所有的bean类也需要在所有的的模块中相互调用，所以bean类也应该放在通用接口模块中。

通用 Mapper 是对MyBatis的基础上进一步抽象的框架，通过定义接口继承Mapper类，并通过泛型动态生成对应数据表实体类的增删改查的SQL代码。通用mapper由于需要使用的时候会利用反射的机制得到bean类的实例，所以通用mapper也应该包含在api模块中。

### common-util模块

首先可以把所有的第三方依赖包分为如下四种：

1. 通用型的第三方jar包，比如fastjson、httpclient、apache工具包等；
2. Web业务模块用到的第三方jar包，比如文件上传客户端、页面渲染工具等；
3. Service业务模块用到的第三方jar包，比如jdbc、jedis、mybatis、activemq工具包等；
4. 只有本模块用到的依赖包。

通用模块是指包含第一类依赖包以及一些项目中通用的工具类的模块，在项目编译时本模块会被打包成jar包，主要代码：

<dependencies>

<dependency>

<groupId>com.alibaba</groupId><artifactId>dubbo</artifactId>

</dependency>

<dependency>

<groupId>com.101tec</groupId><artifactId>zkclient</artifactId>

<exclusions><exclusion>

<groupId>org.slf4j</groupId><artifactId>slf4j-log4j12</artifactId>

</exclusion></exclusions>

</dependency>

<dependency>

<groupId>com.gitee.reger</groupId>

<artifactId>spring-boot-starter-dubbo</artifactId>

</dependency>

<dependency>

<groupId>com.alibaba</groupId><artifactId>fastjson</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.apache.httpcomponents</groupId>

<artifactId>httpclient</artifactId>

</dependency>

<dependency>

<groupId>org.apache.commons</groupId><artifactId>commons-lang3</artifactId>

</dependency>

<dependency>

<groupId>commons-beanutils</groupId>

<artifactId>commons-beanutils</artifactId>

</dependency>

<dependency>

<groupId>commons-codec</groupId><artifactId>commons-codec</artifactId>

</dependency>

<dependency>

<groupId>com.github.pagehelper</groupId><artifactId>pagehelper</artifactId>

</dependency>

</dependencies>

### Web-Util模块

Web-Util模块是指包含了第二类依赖包和web应用工具类的模块，web应用的工具类包含cookie工具、邮件工具、jwt工具，主要代码：

<dependencies><dependency>

<groupId>indi.faniche.anonyshop</groupId>

<artifactId>anonyshop-util-common</artifactId>

<version>1.0-SNAPSHOT</version>

</dependency><dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-thymeleaf</artifactId>

</dependency><dependency>

<groupId>net.sourceforge.nekohtml</groupId>

<artifactId>nekohtml</artifactId><version>${nekohtml.version}</version>

</dependency><dependency>

<groupId>xml-apis</groupId>

<artifactId>xml-apis</artifactId><version>${xml-apis.version}</version>

</dependency><dependency>

<groupId>org.apache.xmlgraphics</groupId>

<artifactId>batik-ext</artifactId><version>${batik-ext.version}</version>

</dependency><dependency>

<groupId>org.csource</groupId><artifactId>fastdfs-client-java</artifactId>

<version>${fastdfs-client-java.version}</version>

</dependency><dependency>

<groupId>io.jsonwebtoken</groupId><artifactId>jjwt</artifactId>

<version>${jjwt.version}</version>

</dependency><dependency>

<groupId>javax.mail</groupId>

<artifactId>mail</artifactId><version>${mail.version}</version>

</dependency>

</dependencies>

### Service-Util模块

Service-Util模块是指包含了第三类依赖包和Service应用工具类的模块，Service应用的工具类包含Redis工具类，主要代码：

<dependencies>

<dependency>

<groupId>indi.faniche.anonyshop</groupId>

<artifactId>anonyshop-util-common</artifactId><version>1.0-SNAPSHOT</version>

</dependency><dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-jdbc</artifactId>

</dependency><dependency>

<groupId>org.mybatis.spring.boot</groupId>

<artifactId>mybatis-spring-boot-starter</artifactId>

</dependency><dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId><scope>runtime</scope>

</dependency><dependency>

<groupId>redis.clients</groupId><artifactId>jedis</artifactId>

</dependency><dependency>

<groupId>org.redisson</groupId><artifactId>redisson</artifactId>

<version>${redisson.version}</version>

</dependency><dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-activemq</artifactId>

<exclusions><exclusion>

<groupId>org.slf4j</groupId><artifactId>slf4j-log4j12</artifactId>

</exclusion></exclusions>

</dependency><dependency>

<groupId>org.apache.activemq</groupId><artifactId>activemq-pool</artifactId>

<version>${activemq-pool.version}</version>

<exclusions><exclusion>

<groupId>org.slf4j</groupId><artifactId>slf4j-log4j12</artifactId>

</exclusion></exclusions>

</dependency>

</dependencies>

### 业务Web模块

具体的Srpingboot应用，主要负责与用户交互，调用后端业务Service模块处理不同的业务。由于已经构建过Interface模块和Web-Util模块，在具体的业务模块只需要引入这两个模块，pom文件中的主要代码：

<dependencies>

<dependency>

<groupId>indi.faniche.anonyshop</groupId>

<artifactId>anonyshop-api</artifactId><version>1.0-SNAPSHOT</version>

</dependency><dependency>

<groupId>indi.faniche.anonyshop</groupId>

<artifactId>anonyshop-util-web</artifactId> <version>1.0-SNAPSHOT</version>

</dependency>

</dependencies>

### 业务Service模块

具体的SpringBoot应用，主要负责与数据库服务器和缓存服务器交互，根据Web模块控制器的调用完对数据库和缓存中的数据进行增删改查。由于已经构建过Interface模块和Service-Util模块，在具体的业务模块只需要引入这两个模块，pom文件中的主要代码：

<dependencies>

<dependency>

<groupId>indi.faniche.anonyshop</groupId>

<artifactId>anonyshop-api</artifactId><version>1.0-SNAPSHOT</version>

</dependency>

<dependency>

<groupId>indi.faniche.anonyshop</groupId>

<artifactId>anonyshop-util-service</artifactId><version>1.0-SNAPSHOT</version>

</dependency>

</dependencies>

## 主要功能设计

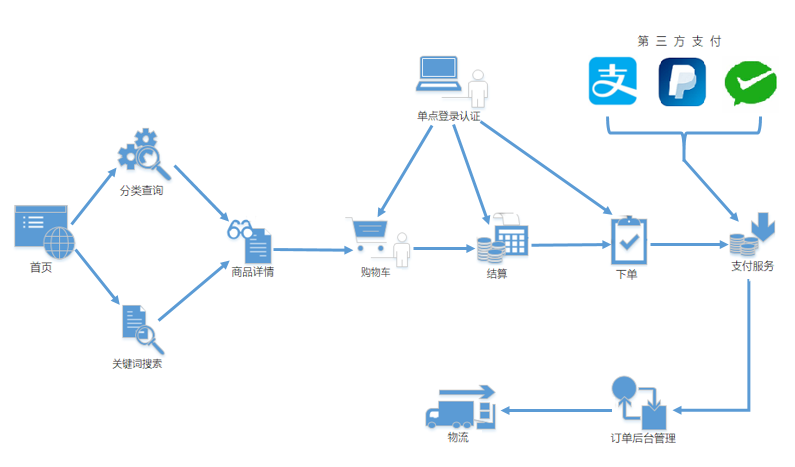


图4- 3 功能结构图

用户在首页可以查看商品，也可以分类查询，可通过关键词搜索商品，点击具体的商品可以进入详情页查看商品的详情，时候点击添加购物车的按钮可以将商品添加到购物车，购物在在不登录的情况下也可以使用，购物车中的商品信息保存在用户浏览器的cookie中，从购物车开始，之后的功能都需要用户在登录的状态下才能使用。

在通过认证的情况下用户可以对购物车中的商品结算，下单，使用第三方支付平台付款，之后就是后台处理订单，物流配送。由于物流系统又涉及到其他知识，本系统中将通过卖家手动修改方式直接模拟物流的派送。

## 数据库设计

数据库的设计对一个系统的正常运行起着至关重要的作用，最初的数据库设计也只是在设计数据库的时候认为时最佳的设计，在实际编写代码中也会陆续发现很多设计不合理的地方。好的设计是不断打磨出来的。本系统中数据库一共使用了37张数据表，由于篇幅限制，不能一一叙述，接下来的两小节将着重介绍与商品管理有关的数据表。

### 概念结构设计

1. 平台属性相关数据表设计：

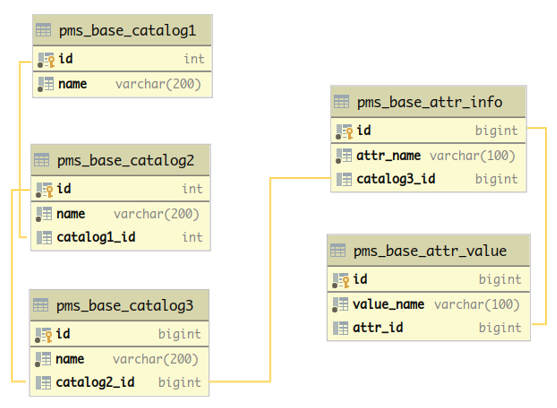


图4- 4 平台属性数据表结构图

1. SPU相关数据表设计：

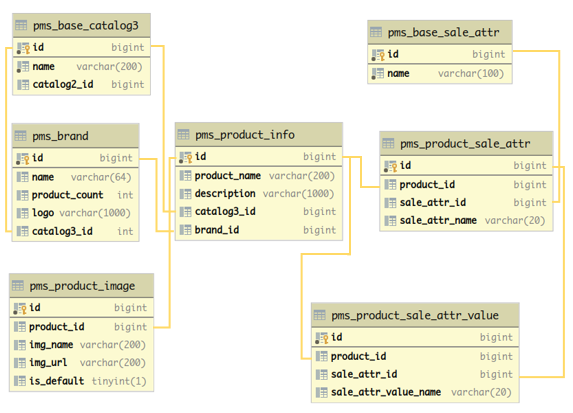


图4- 5 SPU结构图

1. SKU相关数据表设计：

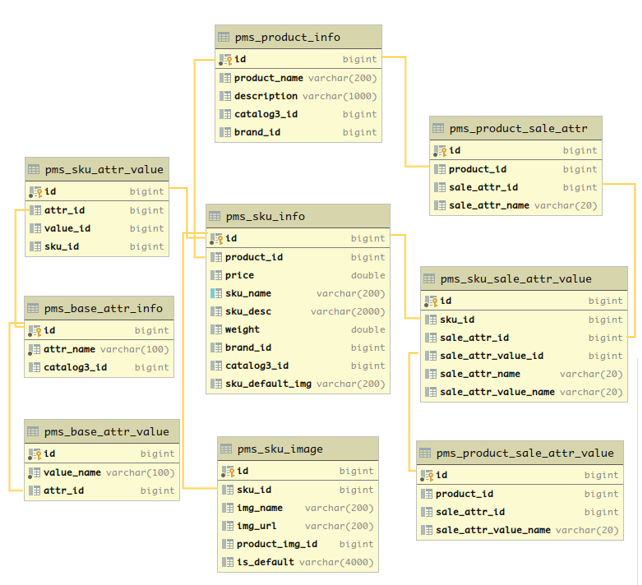


图4- 6 SKU结构图

1. 商品评论相关数据表设计：

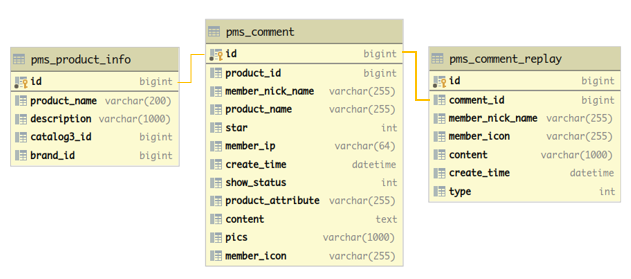


图4- 7 商品评论数据表结构

### 数据库表设计

本小节仅展示商品管理模块的17张表的设计。

表4- 2 pms\_base\_attr\_info

| Field | Type |  | Null | Key | Default | Extra |
| --- | --- | --- | --- | --- | --- | --- |
| id | bigint |  | NO | PRI | N | auto\_increment |
| attr\_name | varchar(100) |  | NO |  | N |  |
| catalog3\_id | bigint |  | YES |  | N |  |

表4- 3 pms\_base\_attr\_value

| Field | Type | Null | Key | Default | Extra |
| --- | --- | --- | --- | --- | --- |
| id | bigint | NO | PRI | N | auto\_increment |
| value\_name | varchar(100) | NO |  | N |  |
| attr\_id | bigint | YES |  | N |  |

表4- 4 pms\_base\_catalog1

| Field | Type | Null | Key | Default | Extra |
| --- | --- | --- | --- | --- | --- |
| id | int | NO | PRI | N | auto\_increment |
| name | varchar(200) | NO |  | N |  |

表4- 5 pms\_base\_catalog2

| Field | Type | Null | Key | Default | Extra |
| --- | --- | --- | --- | --- | --- |
| id | int | NO | PRI | N | auto\_increment |
| name | varchar(200) | NO |  | N |  |
| catalog1\_id | int | YES |  | N |  |

表4- 6 pms\_base\_catalog3

| Field | Type | Null | Key | Default | Extra |
| --- | --- | --- | --- | --- | --- |
| id | bigint | NO | PRI | N | auto\_increment |
| name | varchar(200) | NO |  | N |  |
| catalog2\_id | bigint | YES |  | N |  |

表4- 7 pms\_base\_sale\_attr

| Field | Type | Null | Key | Default | Extra |
| --- | --- | --- | --- | --- | --- |
| id | bigint | NO | PRI | N | auto\_increment |
| name | varchar(100) | NO |  | N |  |

表4- 8 pms\_brand

| Field | Type | Null | Key | Default | Extra |
| --- | --- | --- | --- | --- | --- |
| id | bigint | NO | PRI | N | auto\_increment |
| name | varchar(64) | YES |  | N |  |
| product\_count | int | YES |  | N |  |
| logo | varchar(1000) | YES |  | N |  |
| catalog3\_id | int | YES |  | N |  |

表4- 9 pms\_comment

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| id | bigint | NO | PRI | N | auto\_increment |
| product\_id | bigint | YES |  | N |  |
| member\_nick\_name | varchar(255) | YES |  | N |  |
| product\_name | varchar(255) | YES |  | N |  |
| star | int | YES |  | N |  |
| member\_ip | varchar(64) | YES |  | N |  |
| create\_time | datetime | YES |  | N |  |
| show\_status | int | YES |  | N |  |
| product\_attribute | varchar(255) | YES |  | N |  |
| content | text | YES |  | N |  |
| pics | varchar(1000) | YES |  | N |  |
| member\_icon | varchar(255) | YES |  | N |  |
| store\_id | bigint | YES |  | N |  |

表4- 10 pms\_comment\_replay

| Field | Type | Null | Key | Default | Extra |
| --- | --- | --- | --- | --- | --- |
| id | bigint | NO | PRI | N | auto\_increment |
| comment\_id | bigint | YES |  | N |  |
| member\_nick\_name | varchar(255) | YES |  | N |  |
| member\_icon | varchar(255) | YES |  | N |  |
| content | varchar(1000) | YES |  | N |  |
| create\_time | datetime | YES |  | N |  |
| type | int | YES |  | N |  |

表4- 11 pms\_product\_image

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| id | bigint | NO | PRI | N | auto\_increment |
| product\_id | bigint | YES |  | N |  |
| img\_name | varchar(200) | YES |  | N |  |
| img\_url | varchar(200) | YES |  | N |  |
| is\_default | tinyint(1) | YES |  | 0 |  |

表4- 12 pms\_product\_info

| Field | Type | Null | Key | Default | Extra |
| --- | --- | --- | --- | --- | --- |
| id | bigint | NO | PRI | N | auto\_increment |
| product\_name | varchar(200) | YES |  | N |  |
| description | varchar(1000) | YES |  | N |  |
| catalog3\_id | bigint | YES |  | N |  |
| brand\_id | bigint | YES |  | N |  |

表4- 13 pms\_product\_sale\_attr

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| id | bigint | NO | PRI | N | auto\_increment |
| product\_id | bigint | YES |  | N |  |
| sale\_attr\_id | bigint | YES |  | N |  |
| sale\_attr\_name | varchar(20) | YES |  | N |  |

表4- 14 pms\_product\_sale\_attr\_value

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| id | bigint | NO | PRI | N | auto\_increment |
| product\_id | bigint | YES |  | N |  |
| sale\_attr\_id | bigint | YES |  | N |  |
| sale\_attr\_value\_name | varchar(20) | YES |  | N |  |

表4- 15 pms\_sku\_attr\_value

| Field | Type | Null | Key | Default | Extra |
| --- | --- | --- | --- | --- | --- |
| id | bigint | NO | PRI | N | auto\_increment |
| attr\_id | bigint | YES |  | N |  |
| value\_id | bigint | YES |  | N |  |
| sku\_id | bigint | YES |  | N |  |

表4- 16 pms\_sku\_image

| Field | Type | Null | Key | Default | Extra |
| --- | --- | --- | --- | --- | --- |
| id | bigint | NO | PRI | N | auto\_increment |
| sku\_id | bigint | YES |  | N |  |
| img\_name | varchar(200) | YES |  | N |  |
| img\_url | varchar(200) | YES |  | N |  |
| product\_img\_id | bigint | YES |  | N |  |
| is\_default | varchar(4000) | YES |  | 0 |  |

表4- 17 pms\_sku\_info

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| id | bigint | NO | PRI | N | auto\_increment |
| product\_id | bigint | YES |  | N |  |
| price | double | YES |  | N |  |
| sku\_name | varchar(200) | YES | MUL | N |  |
| sku\_desc | varchar(2000) | YES |  | N |  |
| weight | double | YES |  | N |  |
| brand\_id | bigint | YES |  | N |  |
| catalog3\_id | bigint | YES |  | N |  |
| sku\_default\_img | varchar(200) | YES |  | N |  |
| store\_id | bigint | YES |  | N |  |
| quantity | int | YES |  | N |  |

表4- 18 pms\_sku\_sale\_attr\_value

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| id | bigint | NO | PRI | N | auto\_increment |
| sku\_id | bigint | YES |  | N |  |
| sale\_attr\_id | bigint | YES |  | N |  |
| sale\_attr\_value\_id | bigint | YES |  | N |  |
| sale\_attr\_name | varchar(20) | YES |  | N |  |
| sale\_attr\_value\_name | varchar(20) | YES |  | N |  |

# 系统开发

## 项目结构与描述

项目工程结构如图5-1所示，项目名称为anonyshop，其中anony为anonymous的前五个字母，意为匿名。

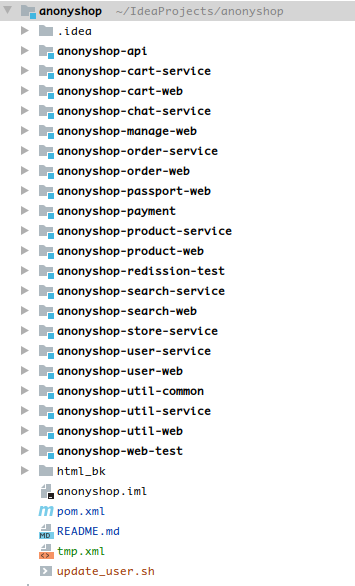


图5- 1 项目文件结构

父项目为anonyshop，不包含java代码，仅包含html\_bk，pom.xml，README.md，tmp.xml，update\_user.sh，其中html\_bk为静态资源的备份文件夹，pom.xml为项目的maven配置文件，README.md为项目的介绍文件，包含了表5-1中定义的内容，为不同子项目的运行端口，域名，和描述，update\_user.sh为更新html文件的shell脚本，由于静态页面是使用Bootstrap Studio制作的，要和项目整合就要将生成的文件导入到不同的web应用下。

表5- 1 子模块名称及运行端口

|  |  |  |  |
| --- | --- | --- | --- |
| 项目子模块 | 端口 | 三级域名 | 描述 |
| dubbo-admin | 8000 | dubbo.chiry.loc/dubbo/ | dubbo注册管理中心 |
| anonyshop-user-service | 8011 |  | 用户Service |
| anonyshop-user-web | 8012 | user.anonyshop.tech | 用户web |
| anonyshop-manage-web | 8020 | manage.anonyshop.tech | 后台管理 |
| anonyshop-product-service | 8030 |  | 商品Service |
| anonyshop-search-web | 8040 | search.anonyshop.tech | 搜索web |
| anonyshop-search-service | 8050 |  | 搜索service |
| anonyshop-item-web | 8060 | item.anonyshop.tech | 商品详情,首页,分类检索 |
| anonyshop-redission-test | 8070 | redisson.chiry.test | redisson测试 |
| anonyshop-web-test | 8090 | anonyshopo.tech | 首页 |
| anonyshop-cart-web | 8100 | cart.anonyshopo.tech | 购物车 |
| anonyshop-cart-service | 8101 |  | 购物车Service |
| anonyshop-passport-web | 8110 | passport.anonyshop.tech | 认证中心 |
| anonyshop-order-web | 8120 | order.anonyshop.tech | 订单web |
| anonyshop-order-service | 8121 |  | 订单service |
| anonyshop-payment | 8130 | payment.anonyshop.tech | 支付模块 |
| anonyshop-store-service | 8140 |  | 店铺Service |
| anonyshop-ware-service | 8150 |  | 仓库Service |

每个子模块预留了从该模块端口起的10个端口作为该模块的运行端口，如果要扩展集群，这10个端口就是集群的运行端口。

关于系统首页的nginx的配置：

upstream anonyshop.tech {

server 127.0.0.1:8090;

}

server {

listen 80;

server\_name anonyshop.tech;

location / {

proxy\_pass http://anonyshop.tech;

proxy\_set\_header X-forwarded-for $proxy\_add\_x\_forwarded\_for;

}

}

其他web应用的配置和首页类似，只是server\_name和端口不同，不在赘述。

## 后台管理模块系实现

### 子项目搭建

1. 选择父项目，新建模块，如图5-2所示，选择Spring Initailizer，新建spring boot应用，Module SDK选择OpenJDK1.8，点击Next；

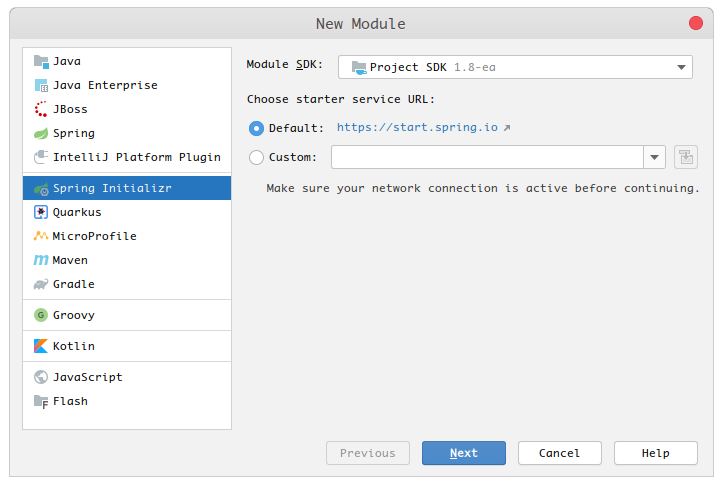


图5- 2 新建SpringBoot模块

1. 配置子项目信息，如图5-3所示，Group为indi.faniche.anonyshop，表示这是Faniche的个人项目，Artifact为anonyshop-manage-web，表示这是后台管理模块的web应用；Type标识应用类型，选择Maven Project，表示使用maven的项目结构；Packaging选择Jar，表示项目编译后打包成jar包；Java Version选择8，因为在验证项目中出现了问题，有些依赖包长期没有更新，导致在OpenJDK11下无法正常使用；Version默认为0.0.1-Snapshot；name为项目名字，和Artifact保持一致；Description为项目描述，在这里填写“Web application of manage”；Package为indi.faniche.anonyshop.manage，表示源代码的统一包名。

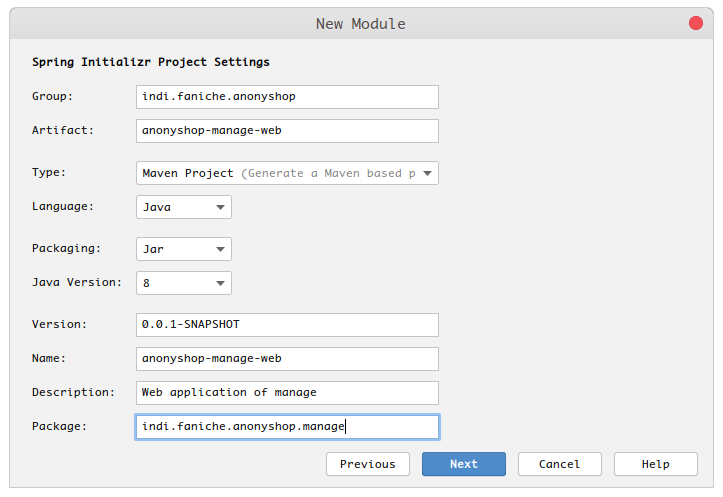


图5- 3 填写模块配置

1. 选择子项目中要使用的库，如图5-4所示，在这里什么都不选，因为已经在anonyshop-parent中定义了所有要使用的依赖包，只需要在新建完子项目后手动添加对anonyshop-util-web的依赖；

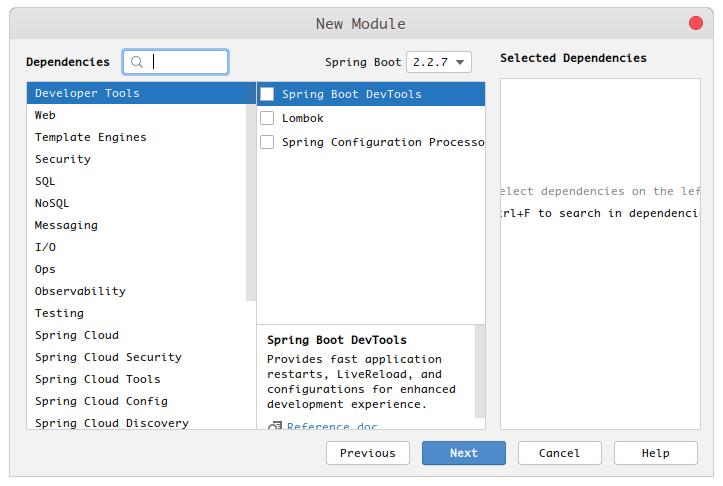


图5- 4 选择子模块依赖

1. 完成子项目的创建如图5-5所示

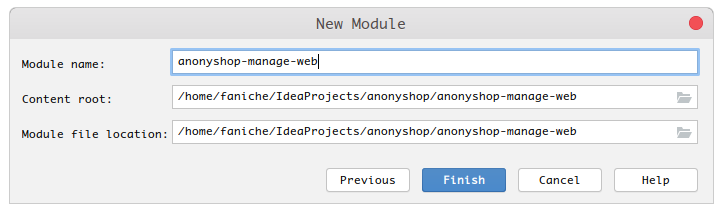


图5- 5 完成子模块创建

1. 修改pom.xml文件，添加anonyshop-api和anonyshop-util-web的依赖：

<dependency>

<groupId>indi.faniche.anonyshop</groupId>

<artifactId>anonyshop-api</artifactId>

<version>1.0-SNAPSHOT</version>

</dependency>

<dependency>

<groupId>indi.faniche.anonyshop</groupId>

<artifactId>anonyshop-util-web</artifactId>

<version>1.0-SNAPSHOT</version>

</dependency>

### 主要页面

1. 平台属性管理页面如图5-6所示

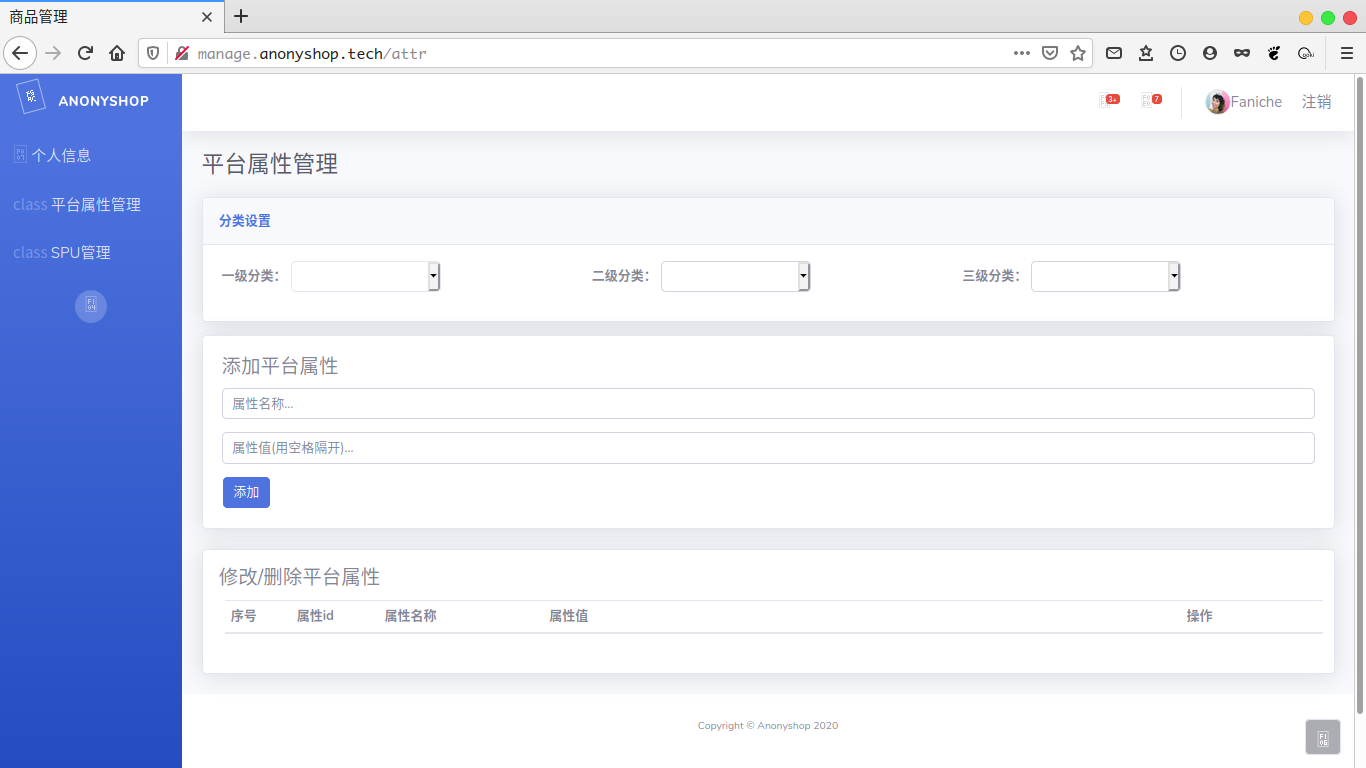


图5- 6 平台属性管理页面

1. SPU管理页面如图5-7所示

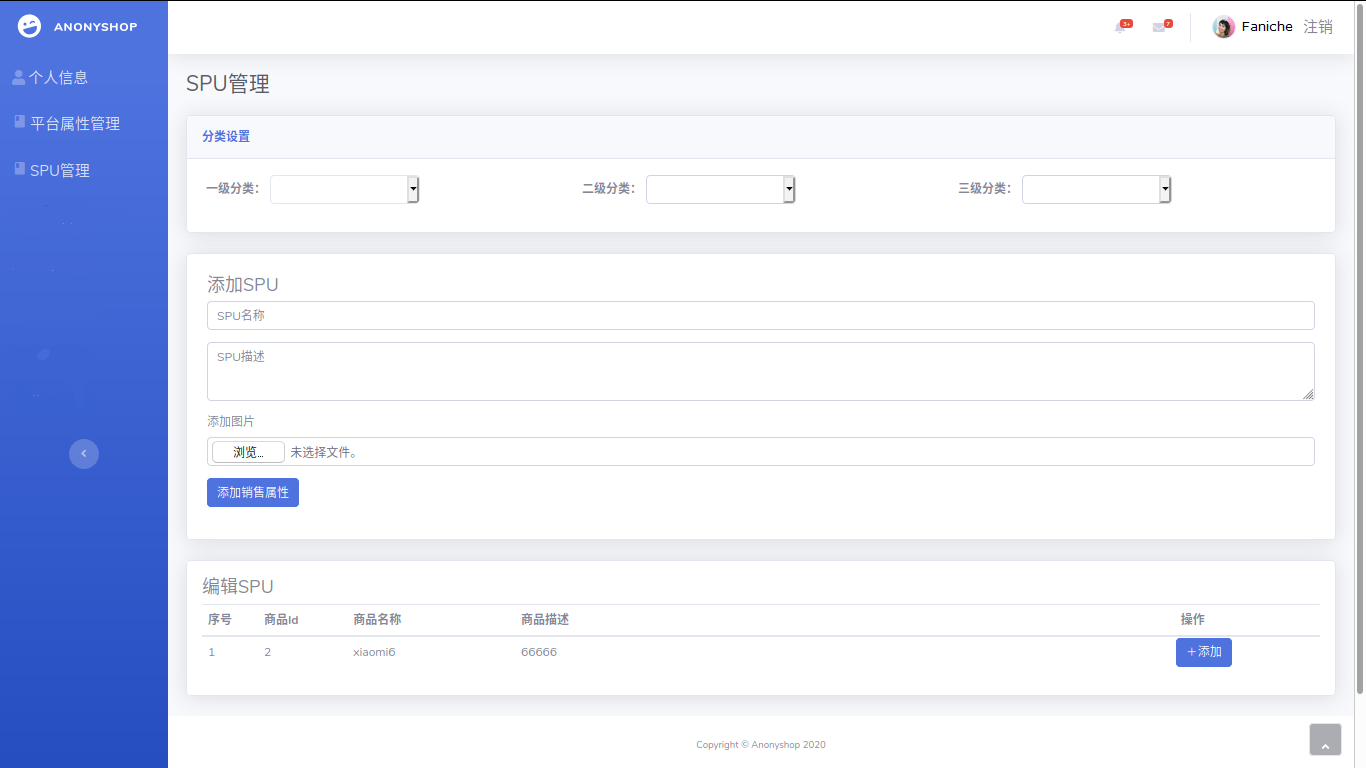


图5- 7 SPU管理页面

### 主要代码

@RequestMapping("getAttrInfoList") @ResponseBody

public List<PmsBaseAttrInfo> getAttrInfoList(String catalog3Id){

return attrService.getAttrInfoList(catalog3Id);

}

/\* 添加平台属性 \*/

@RequestMapping("addAttrInfo")

@ResponseBody

public List<PmsBaseAttrInfo> addAttrInfo(PmsBaseAttrInfo pmsBaseAttrInfo){

attrService.addAttrInfo(pmsBaseAttrInfo);

return attrService.getAttrInfoList(pmsBaseAttrInfo.getCatalog3Id());

}

/\* 删除平台属性 \*/

@RequestMapping("deleteAttrInfo")

@ResponseBody

public List<PmsBaseAttrInfo> deleteAttrInfo(PmsBaseAttrInfo pmsBaseAttrInfo){

attrService.deleteAttrInfo(pmsBaseAttrInfo);

return attrService.getAttrInfoList(pmsBaseAttrInfo.getCatalog3Id());

}

/\* 修改平台属性 \*/

@RequestMapping("editAttrInfo") @ResponseBody

public List<PmsBaseAttrInfo> editAttrInfo(PmsBaseAttrInfo pmsBaseAttrInfo){

attrService.editAttrInfo(pmsBaseAttrInfo);

return attrService.getAttrInfoList(pmsBaseAttrInfo.getCatalog3Id());

}

## 商品详情模块实现

子项目的搭建和上一节中大致相同，不再赘述。

### 主要页面

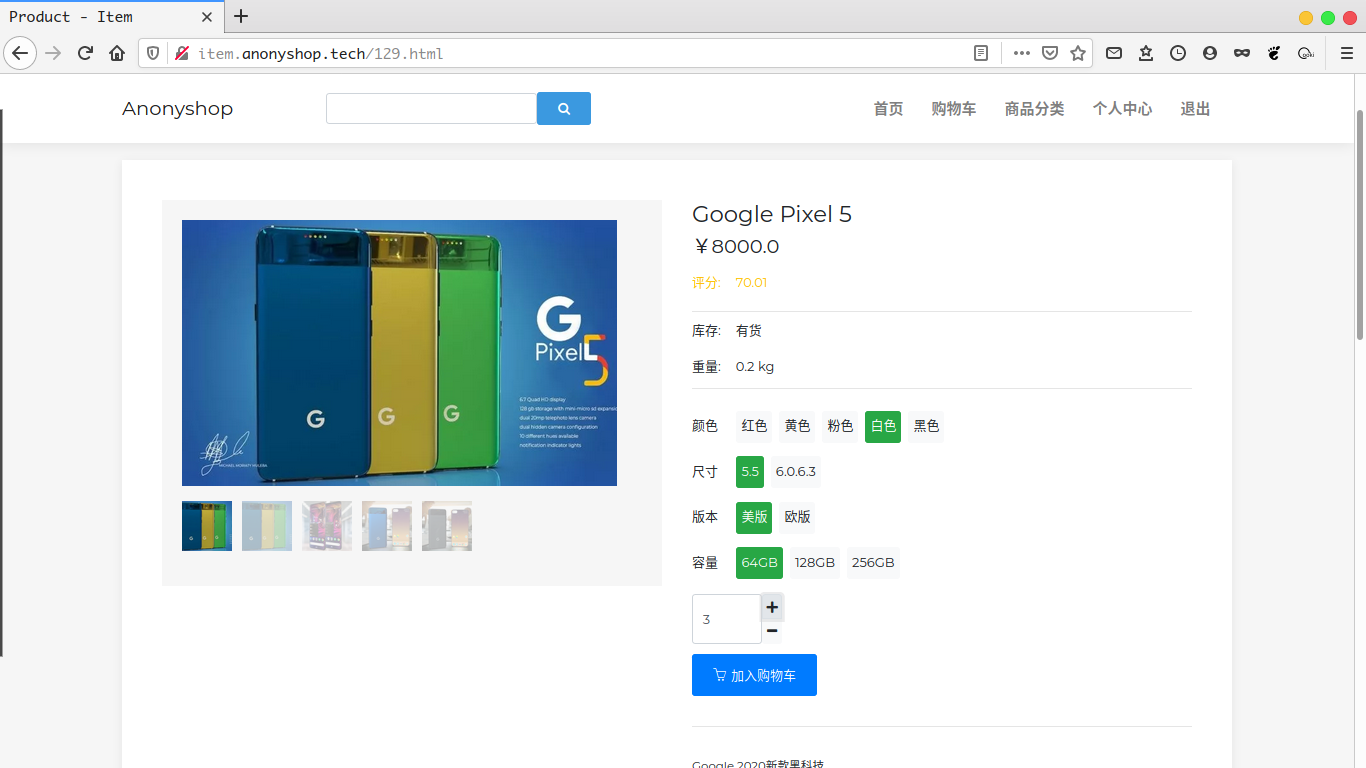


图5- 8 商品详情页面

### 主要代码

@RequestMapping("{skuId}.html")

@LoginRequired(loginSuccess = false)

public String item(@PathVariable String skuId, Model model, HttpServletRequest request){

String username = (String) request.getAttribute("username");

String userId = (String) request.getAttribute("userId");

PmsSkuInfo pmsSkuInfo = skuService.getSkuById(skuId, request.getRemoteAddr());

SmsStore smsStore = storeService.getStoreByStoreId(pmsSkuInfo.getStoreId());

BigDecimal productScore = storeProductService.getSProcuctScore(pmsSkuInfo.getProductId(), pmsSkuInfo.getStoreId());

model.addAttribute("productScore", productScore);

List<PmsComment> comments =

commentService.getComments(pmsSkuInfo.getProductId(), pmsSkuInfo.getStoreId());

model.addAttribute("comments", comments);

pmsSkuInfo.setSmsStore(smsStore);

model.addAttribute("skuInfo", pmsSkuInfo);

List<PmsProductSaleAttr> productSaleAttrs =

spuService.spuSaleAttrLis (pmsSkuInfo.getProductId(), pmsSkuInfo.getId());

model.addAttribute("spuSaleAttrListCheckBySku",pmsProductSaleAttrs);

Map<String, String> skuSaleAttrHash = new HashMap<>();

List<PmsSkuInfo> pmsSkuInfos =

skuService.getSkuSaleAttrValueList (pmsSkuInfo.getProductId());

for (PmsSkuInfo skuInfo : pmsSkuInfos) {

String k = "";

String v = skuInfo.getId();

List<PmsSkuSaleAttrValue> skuSaleAttrValueList = skuInfo.getSkuSaleAttrValueList();

for (PmsSkuSaleAttrValue pmsSkuSaleAttrValue : skuSaleAttrValueList) {

k += pmsSkuSaleAttrValue.getSaleAttrValueId() + "|";

}

skuSaleAttrHash.put(k,v);

}

String skuSaleAttrHashJsonStr = JSON.toJSONString(skuSaleAttrHash);

model.addAttribute("skuSaleAttrHashJsonStr",skuSaleAttrHashJsonStr);

addUserInfoToModel(request, model);

return "product/product-page";

}

## 搜索模块实现

### 主要页面

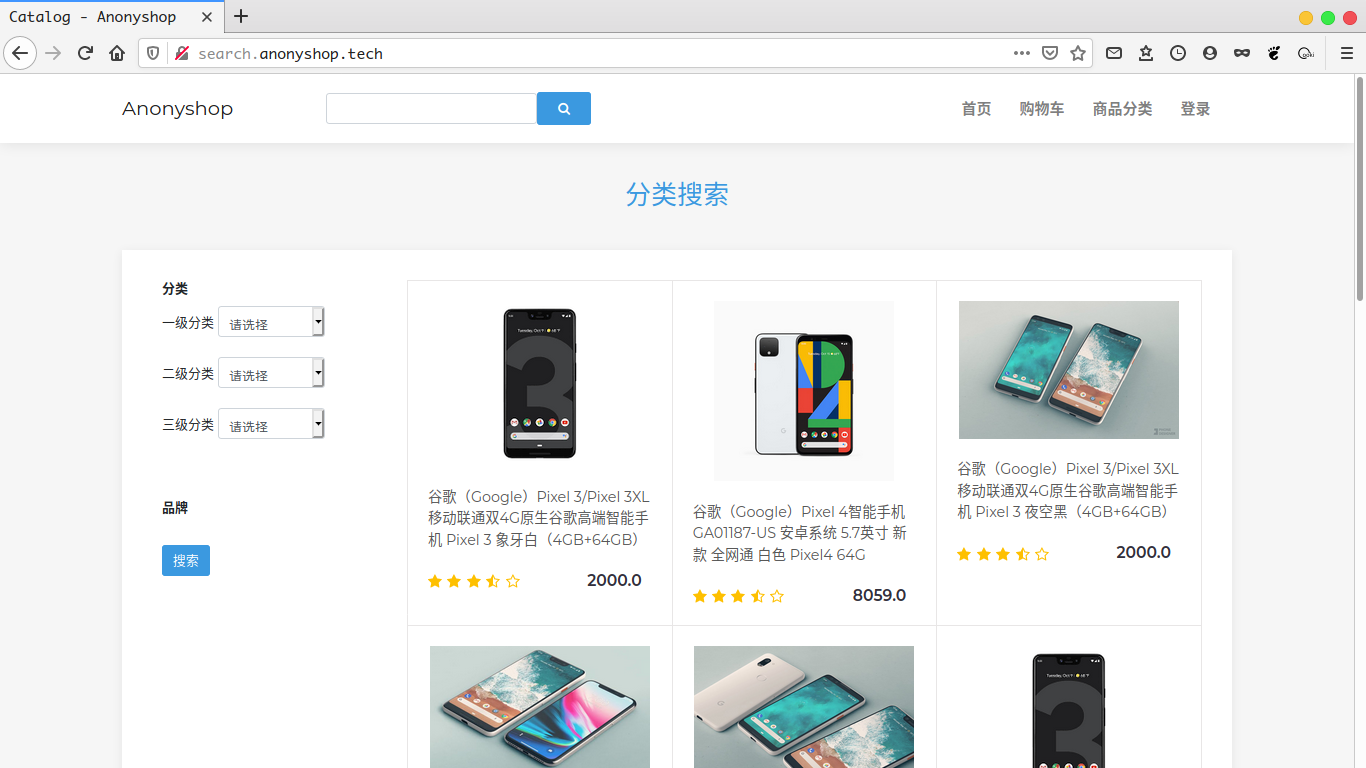


图5- 9 分类搜索页面

### 主要代码

@RequestMapping(value = {"", "classSearch", "/"})

@LoginRequired(loginSuccess = false)

public String toClassSearch(Model model, HttpServletRequest request) {

String username = (String) request.getAttribute("username");

String userId = (String) request.getAttribute("userId");

model.addAttribute("username", username);

model.addAttribute("userId", userId);

PageInfo<PmsSkuInfo> skuInfoList = skuService.getClassSearchPageRandomList(1);

model.addAttribute("skuInfoList", skuInfoList);

return "search/catalog-page";

}

@RequestMapping("randomRet")

public String randomRet(Integer pageNum, Model model) {

PageInfo<PmsSkuInfo> skuInfoList = skuService.getClassSearchPageRandomList(pageNum);

model.addAttribute("skuInfoList", skuInfoList);

return "search/catalog-page::class-search-result";

}

@RequestMapping("class\_search")

public String classSearch(ClassSearch classSearch, HttpSession session, Model model) {

List<PmsSkuInfo> skuInfos = searchService.classSearch(classSearch);

sortSearchRetById(skuInfos)

PageInfo<PmsSkuInfo> skuInfoList = getPage(skuInfos, 1);

session.setAttribute("skuInfoList", skuInfoList);

model.addAttribute("skuInfoList", skuInfoList);

return "search/catalog-page::class-search-result";

}

@RequestMapping("classSearchList")

public String classSearchList(HttpSession session, Integer pageNum, Model model) {

List<PmsSkuInfo> skuInfos = (List<PmsSkuInfo>) session.getAttribute("skuInfoList");

PageInfo<PmsSkuInfo> skuInfoList = getPage(skuInfos, pageNum);

model.addAttribute("skuInfoList", skuInfoList);

return "search/catalog-page::class-search-result";

}

@RequestMapping("list")

@LoginRequired(loginSuccess = false)

public String ambSearch(String keyword, Model model) {

List<PmsSkuInfo> skuInfoList = searchService.ambSearch(keyword);

model.addAttribute("skuInfoList", skuInfoList);

return "search/search-result";

}

## 购物车模块实现

### 主要页面

购物车页面如图5-10所示

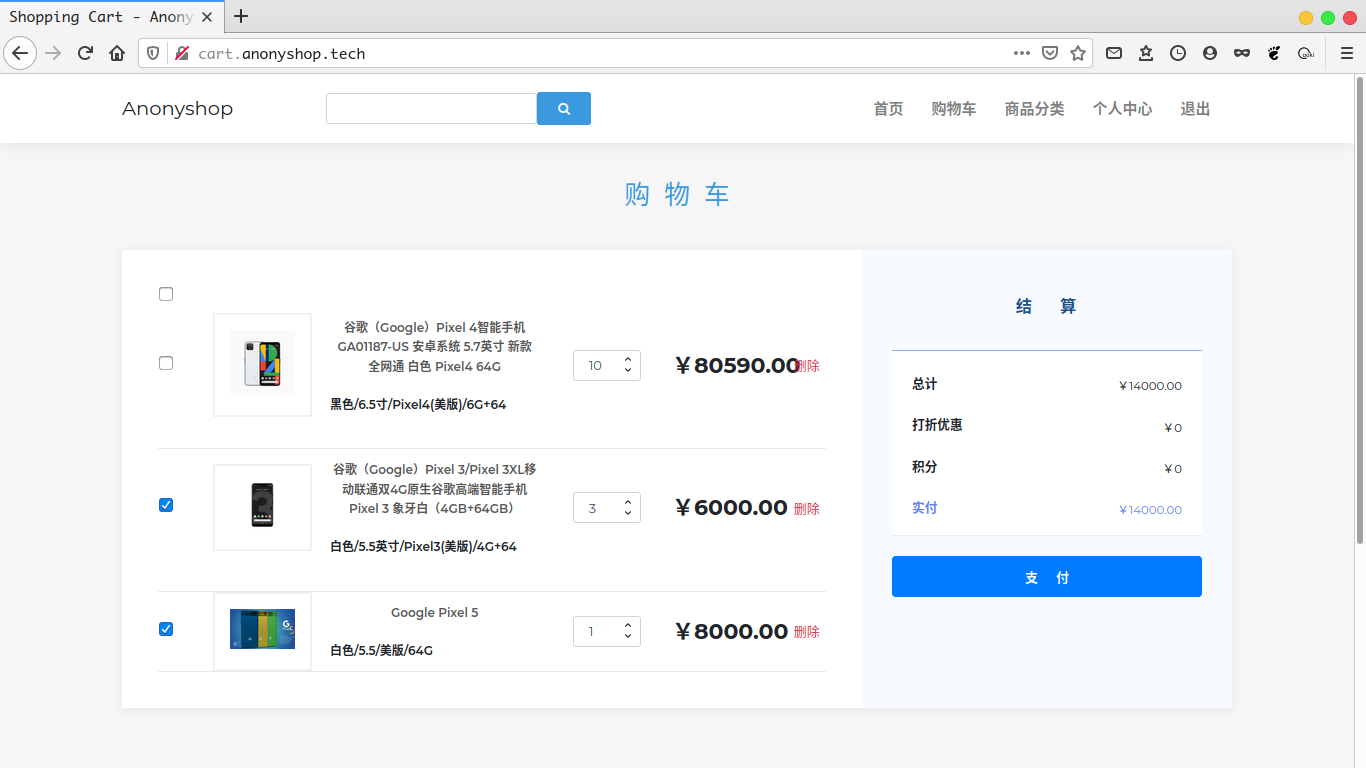


图5- 10 购物车页面

### 主要代码

@RequestMapping("delete")

@LoginRequired(loginSuccess = false)

public String deleteCartItem(HttpServletRequest request, HttpServletResponse response, String skuId, Model model) {

List<OmsCartItem> omsCartItems = new ArrayList<>();

String userId = (String) request.getAttribute("userId");

if (StringUtils.isBlank(userId)) {

String cartListCookie = CookieUtil.getCookieValue(request, "cartListCookie", true);

omsCartItems = JSON.parseArray(cartListCookie, OmsCartItem.class);

for (int i = 0; i < omsCartItems.size(); i++){

if (omsCartItems.get(i).getProductSkuId().equals(skuId)) {

omsCartItems.remove(i);

break;

}

}

setCookie("cartListCookie", JSON.toJSONString(omsCartItems), 60 \* 60 \* 72);

} else {

cartService.deleteCartItem(userId, skuId);

cartService.syncCartCache(userId);

omsCartItems = cartService.cartList(userId);

}

model = sortAndAddToModel(model, omsCartItems);

return "cart/cart-list-inner";

}

@RequestMapping("checkCart")

@LoginRequired(loginSuccess = false)

public String checkCart(HttpServletRequest request, HttpServletResponse response, String isChecked, String skuId, Model model) {

List<OmsCartItem> omsCartItems = new ArrayList<>();

String userId = (String) request.getAttribute("userId");

if (StringUtils.isBlank(userId)) {

String cartListCookie = CookieUtil.getCookieValue(request, "cartListCookie", true);

omsCartItems = JSON.parseArray(cartListCookie, OmsCartItem.class);

for (OmsCartItem cartItem : omsCartItems) {

if (cartItem.getProductSkuId().equals(skuId)) {

cartItem.setIsChecked(isChecked);

break;

}

}

setCookie ("cartListCookie", JSON.toJSONString(omsCartItems), 60 \* 60 \* 72);

} else {

OmsCartItem omsCartItemFromDb = cartService.ifCartExistByUser(userId, skuId);

omsCartItemFromDb.setIsChecked(isChecked);

cartService.updateCart(omsCartItemFromDb);

cartService.syncCartCache(userId);

omsCartItems = cartService.cartList(userId);

}

model = sortAndAddToModel(model, omsCartItems);

return "cart/cart-list-inner";

}

@RequestMapping("checkAll")

@LoginRequired(loginSuccess = false)

public String checkAllCartItem(HttpServletRequest request, HttpServletResponse response, String isChecked, Model model) {

List<OmsCartItem> omsCartItems = new ArrayList<>();

String userId = (String) request.getAttribute("userId");

if (StringUtils.isBlank(userId)) {

String cartListCookie = CookieUtil.getCookieValue(request, "cartListCookie", true);

omsCartItems = JSON.parseArray(cartListCookie, OmsCartItem.class);

for (OmsCartItem cartItem : omsCartItems) {

cartItem.setIsChecked(isChecked);

}

setCookie("cartListCookie", JSON.toJSONString(omsCartItems), 60 \* 60 \* 72);

} else {

cartService.checkAll(userId, isChecked);

cartService.syncCartCache(userId);

omsCartItems = cartService.cartList(userId);

}

model = sortAndAddToModel(model, omsCartItems);

model.addAttribute("checkAll", isChecked);

return "cart/cart-list-inner";

}

@RequestMapping("chgItemQuantity")

@LoginRequired(loginSuccess = false)

public String chgItemQuantity(HttpServletRequest request, HttpServletResponse response, Integer quantity, String skuId, Model model) {

List<OmsCartItem> omsCartItems = new ArrayList<>();

String userId = (String) request.getAttribute("userId");

if (StringUtils.isBlank(userId)) {

String cartListCookie = CookieUtil.getCookieValue(request, "cartListCookie", true);

omsCartItems = JSON.parseArray(cartListCookie, OmsCartItem.class);

for (OmsCartItem cartItem : omsCartItems) {

if (cartItem.getProductSkuId().equals(skuId)) {

cartItem.setQuantity(quantity);

break;

}

}

setCookie("cartListCookie", JSON.toJSONString(omsCartItems), 60 \* 60 \* 72);

} else {

OmsCartItem omsCartItemFromDb = cartService.ifCartExistByUser(userId, skuId);

omsCartItemFromDb.setQuantity(quantity);

cartService.updateCart(omsCartItemFromDb);

cartService.syncCartCache(userId);

omsCartItems = cartService.cartList(userId);

}

model = sortAndAddToModel(model, omsCartItems);

return "cart/cart-list-inner";

}

@RequestMapping("addToCart")

@LoginRequired

public String addToCart(String skuId, Integer quantity, HttpServletRequest request, HttpServletResponse response) {

List<OmsCartItem> omsCartItems = new ArrayList<>();

OmsCartItem omsCartItem = getOmsCartItemFromSkuId(skuId, quantity);

String userId = (String) request.getAttribute("userId");

String username = (String) request.getAttribute("username");

if (StringUtils.isBlank(userId)) {

String cartListCookie = CookieUtil.getCookieValue(request, "cartListCookie", true);

if (StringUtils.isBlank(cartListCookie)) {

omsCartItems.add(omsCartItem);

} else {

omsCartItems = JSON.parseArray(cartListCookie, OmsCartItem.class);

if (if\_cart\_exist(omsCartItems, omsCartItem)) {

for (OmsCartItem cartItem : omsCartItems) {

if (cartItem.getProductSkuId().equals(omsCartItem.getProductSkuId())) {

cartItem.setQuantity(cartItem.getQuantity() + omsCartItem.getQuantity());

break;

}

}

} else {

omsCartItems.add(omsCartItem);

}

}

setCookie("cartListCookie", JSON.toJSONString(omsCartItems), 60 \* 60 \* 72,);

} else {

OmsCartItem omsCartItemFromDb = cartService.ifCartExistByUser(userId, skuId);

if (omsCartItemFromDb == null) {

omsCartItem.setUserId(userId);

omsCartItem.setUsername(username);

omsCartItem.setQuantity(quantity);

cartService.addCart(omsCartItem);

} else {

omsCartItemFromDb.setQuantity(omsCartItemFromDb.getQuantity() + (omsCartItem.getQuantity()));

cartService.updateCart(omsCartItemFromDb);

}

cartService.syncCartCache(userId);

}

return "redirect:/cartList";

}

@RequestMapping("cartList")

@LoginRequired(loginSuccess = false)

public String cartList(HttpServletRequest request, Model model) {

List<OmsCartItem> omsCartItems = new ArrayList<>();

String userId = (String) request.getAttribute("userId");

String username = (String) request.getAttribute("username");

if (StringUtils.isNotBlank(userId)) {

omsCartItems = cartService.cartList(userId);

} else {

String cartListCookie = CookieUtil.getCookieValue(request, "cartListCookie", true);

if (StringUtils.isNotBlank(cartListCookie)) {

omsCartItems = JSON.parseArray(cartListCookie, OmsCartItem.class);

}

}

for (OmsCartItem omsCartItem : omsCartItems) {

omsCartItem.setTotalPrice(omsCartItem.getPrice().multiply(

new BigDecimal(omsCartItem.getQuantity())));

}

model = sortAndAddToModel(model, omsCartItems);

addUserInfoToModel(request, model);

return "cart/shopping-cart";

}

## 单点登录模块实现

### 主要页面

登录页面如图5-11所示



图5- 11 登陆页面

### 主要代码

@RequestMapping(value = {"", "/"})

public String toLogin(HttpServletRequest request, Model model) {

addUserInfoToModel(request, model);

model.addAttribute("returnUrl", request.getHeader("Referer"));

return "passport/login";

}

@RequestMapping("login")

@ResponseBody

public String authenticate(UmsLogin umsLogin, HttpServletRequest request) {

String token = "";

String username = umsLogin.getUsername();

if (username.contains("@")) {

umsLogin.setEmail(username);

umsLogin.setUsername("");

}

UmsLogin login = userService.login(umsLogin);

if (login != null) {

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

String userId = login.getId();

userMap.put("userId", login.getId());

userMap.put("username", login.getUsername());

userMap.put("roleId", login.getRoleId());

String remoteAddr = request.getHeader("s-forward-for");

token = JwtUtil.encode("2020anonyshop", userMap, remoteAddr);

userService.addToken(token, userId);

return JSON.toJSONString(token);

} else {

return JSON.toJSONString("fail");

}

}

@RequestMapping("verify")

@ResponseBody

public String verify(String token, String currentIp) {

Map<String, String> map = new HashMap<>();

Map<String, Object> decode = JwtUtil.decode(token, "2020anonyshop", currentIp);

if (decode != null) {

map.put("status", "success");

map.put("userId", (String) decode.get("userId"));

map.put("username", (String) decode.get("username"));

map.put("roleId", (String) decode.get("roleId"));

} else {

map.put("status", "fail");

}

return JSON.toJSONString(map);

}

@RequestMapping("logout")

@LoginRequired(loginSuccess = false)

public String logout(HttpServletRequest request, HttpServletResponse response) {

String userId = (String) request.getAttribute("userId");

CookieUtil.deleteCookie(request, response, "oldToken");

userService.deleteToken(userId);

return "redirect:http://anonyshop.tech";

}

## 订单模块实现

### 主要页面

支付确认页面如图5-12所示

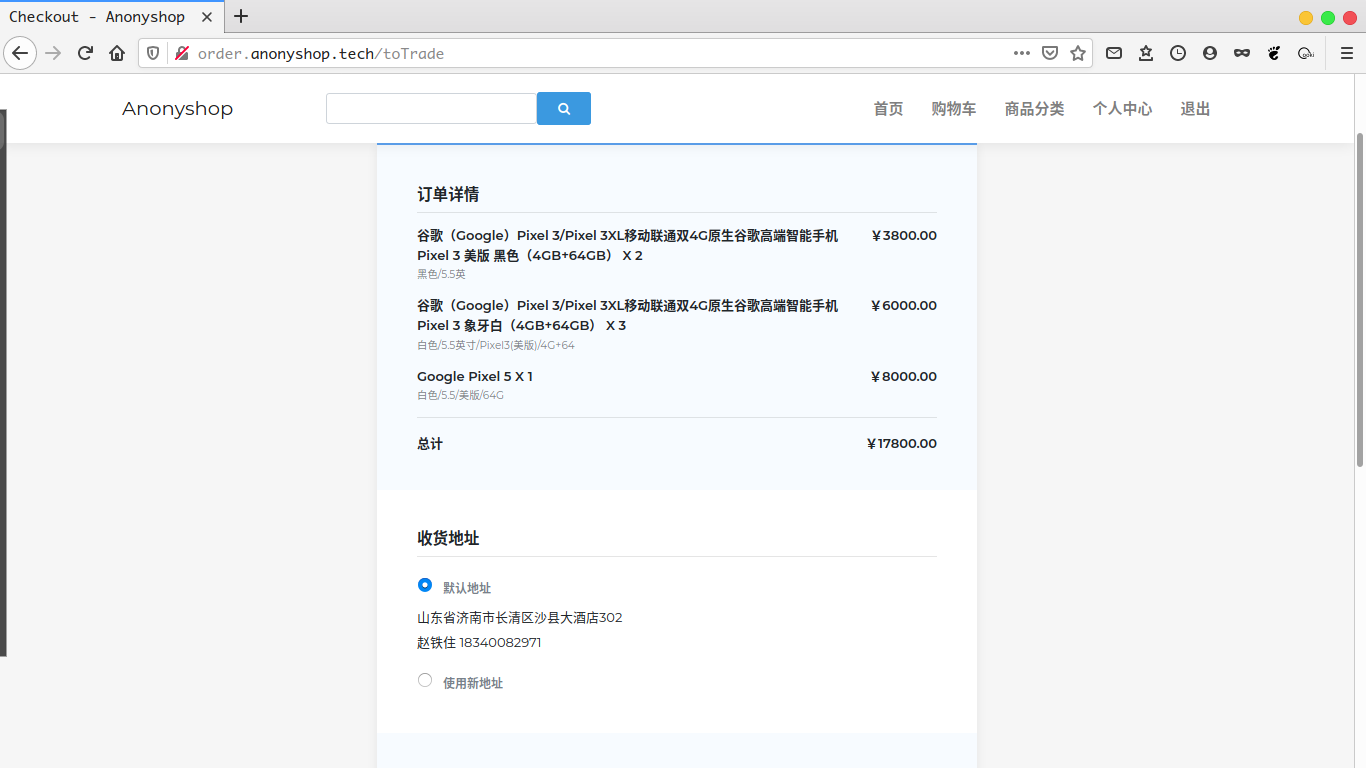


图5- 12 支付确认页面

### 主要代码

@RequestMapping(value = {"toTrade", "", "/"})

@LoginRequired(loginSuccess = true)

public String toTrade(HttpServletRequest request, HttpServletResponse response, HttpSession session, Model model) {

addUserInfoToModel(request, model);

String userId = (String) request.getAttribete("userId");

UmsDefaultAddress defaultAddress = addressService.getDefaultAddress (userId);

List<OmsCartItem> omsCartItems = cartService.cartList(userId);

List<OmsOrderItem> omsOrderItems = new ArrayList<>();

for (OmsCartItem omsCartItem : omsCartItems) {

if (omsCartItem.getIsChecked().equals("1"))

omsOrderItems.add(getOrderItemFromCartItem(omsCartItem, true));

}

}

List<AddrProvince> provinceList = cascadeAddressService.getAllProvince();

model.addAttribute("defaultAddress", defaultAddress);

model.addAttribute("orderList", omsOrderItems);

model.addAttribute("totalPrice", getTotalAmount(omsCartItems));

model.addAttribute("tradeCode", orderService.genTradeCode(userId));

model.addAttribute("provinceList", provinceList);

return "order/order";

}

@RequestMapping("submit")

@LoginRequired(loginSuccess = true)

public ModelAndView submitOrder(OmsOrder subOmsOrder, HttpServletRequest request) {

String userId = (String) request.getAttribute("userId");

String username = (String) request.getAttribute("username");

String success = orderService.checkTradeCode(userId, subOmsOrder.getTradeCode());

ModelAndView modelAndView = new ModelAndView();

if (success.equals("success")) {

subOmsOrder = genOmsOrder(subOmsOrder, userId, username);

List<OmsOrderItem> omsOrderItems = new ArrayList<>();

List<OmsCartItem> omsCartItems = cartService.cartList(userId);

for (OmsCartItem omsCartItem : omsCartItems) {

if (omsCartItem.getIsChecked().equals("1")) {

boolean ret = skuService.checkPrice(omsCartItem.getProductSkuId(), omsCartItem.getPrice());

if (ret == false) {

modelAndView.setViewName("order/fail");

return modelAndView;

}

OmsOrderItem omsOrderItem = chgCartItemToOrderItem(omsCartItem, false);

omsOrderItem.setOrderSn(subOmsOrder.getOrderSn());

omsOrderItem.setProductId(omsCartItem.getProductId());

omsOrderItem.setIntegrationAmount(new BigDecimal(0));

omsOrderItems.add(omsOrderItem);

subOmsOrder.setBillContent(subOmsOrder.getBillContent() != null? subOmsOrder.getBillContent() + omsCartItem.getProductName() + "\n" : omsCartItem.getProductName());

}

}

subOmsOrder.setStoreId(omsCartItems.get(0).getStoreId());

subOmsOrder.setOmsOrderItemList(omsOrderItems);

orderService.saveOrder(subOmsOrder);

for (OmsCartItem omsCartItem : omsCartItems) {

if (omsCartItem.getIsChecked().equals("1"))

cartService.deleteCartItem(userId, omsCartItem.getProductSkuId());

cartService.syncCartCache(userId);

modelAndView.setViewName("redirect:http://payment.anonyshop.tech");

modelAndView.addObject("outTradeNo", subOmsOrder.getOrderSn());

modelAndView.addObject("totalAmount", subOmsOrder.getTotalAmount());

return modelAndView;

} else {

modelAndView.setViewName("order/fail");

return modelAndView;

}

}

## 支付模块实现

### 主要页面

1. 选择支付方式如图5-13：

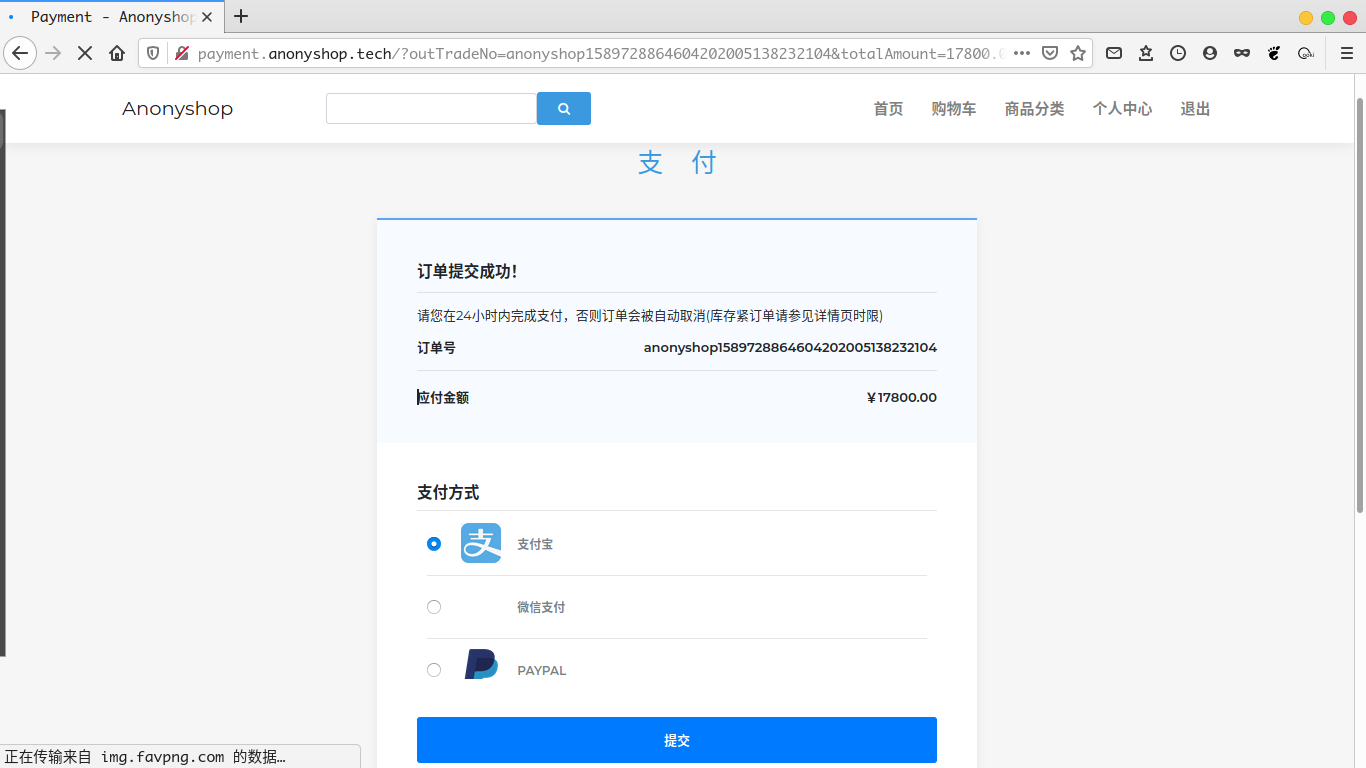


图5- 13 选择支付方式

1. 扫码付款如图5-14所示：

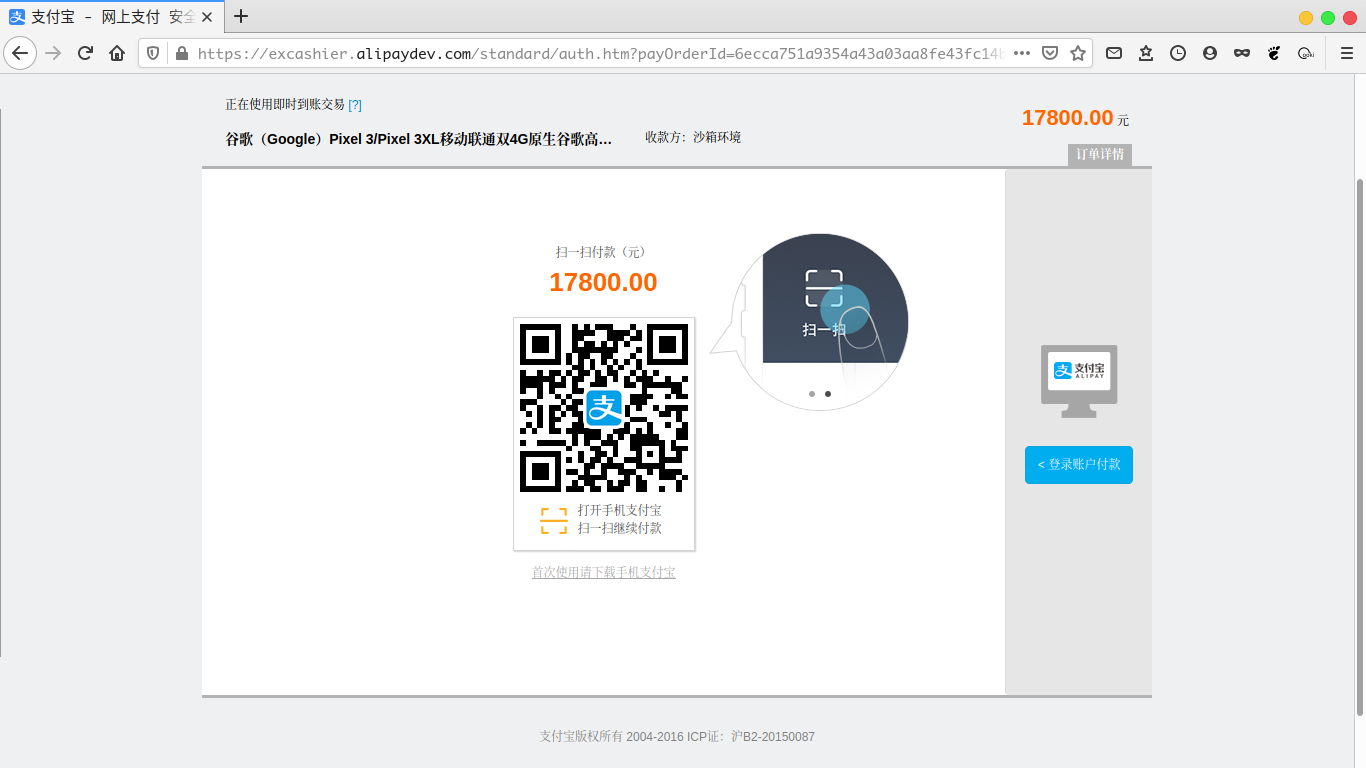


图5- 14 扫码付款

1. 付款成功如图5-15：

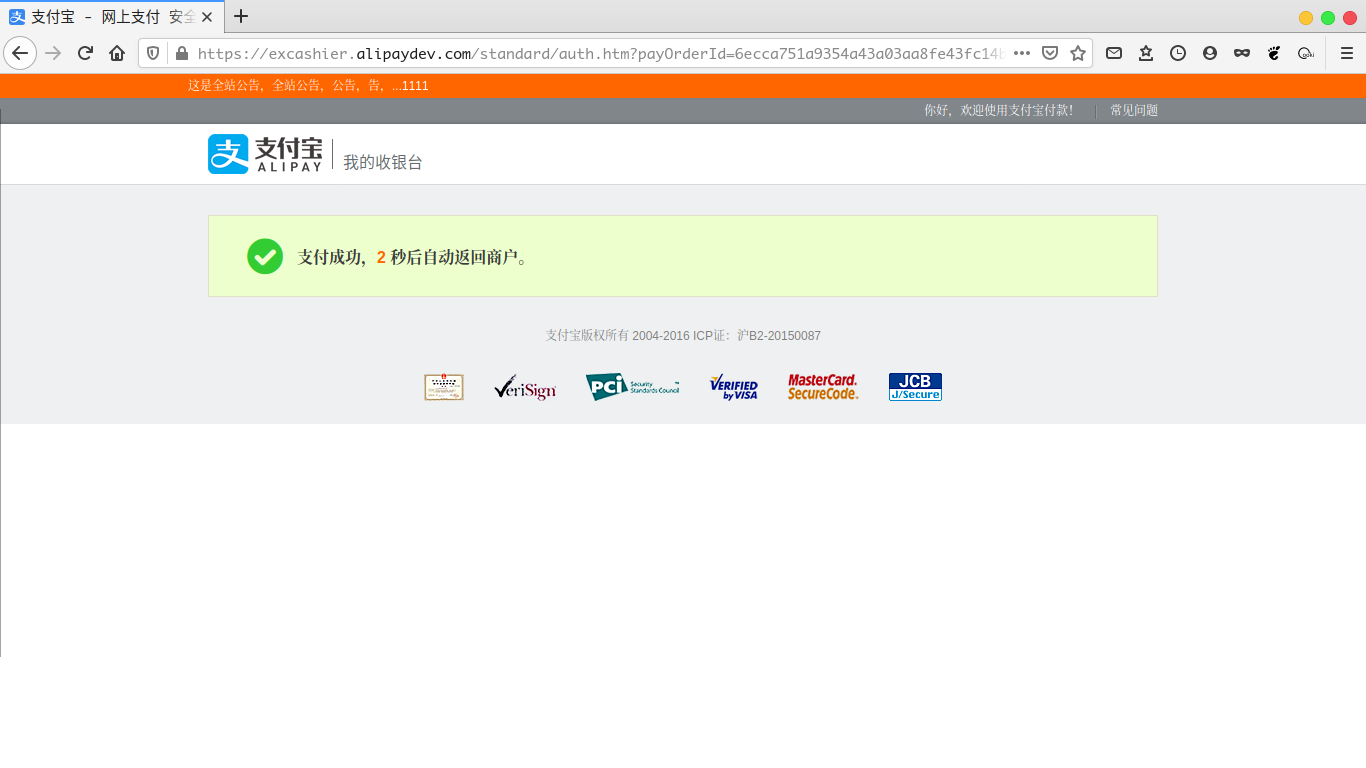


图5- 15 付款成功

### 主要代码

@RequestMapping("alipay/success") @LoginRequired(loginSuccess = true)

public String alipaySuccess(HttpServletRequest request, Model model) {

String userId = (String) request.getAttribute("userId");

String username = (String) request.getAttribute("username");

String sign = request.getParameter("sign");

String trade\_no = request.getParameter("trade\_no");

String out\_trade\_no = request.getParameter("out\_trade\_no");

String trade\_status = request.getParameter("trade\_status");

String total\_amount = request.getParameter("total\_amount");

String subject = request.getParameter("subject");

String call\_back\_content = request.getQueryString();

Date paymentTime = new Date();

if (StringUtils.isNotBlank(sign)) {

PaymentInfo paymentInfo = new PaymentInfo();

paymentInfo.setOrderSn(out\_trade\_no); paymentInfo.setPaymentStatus("已支付");

paymentInfo.setAlipayTradeNo(trade\_no);

paymentInfo.setCallbackContent(call\_back\_content);

paymentInfo.setCallbackTime(new Date());

paymentService.updatePayment(paymentInfo, total\_amount, paymentTime);

} model.addAttribute("result", "付款成功"); addUserInfoToModel(request, model);

return "payment/success";

}

@RequestMapping("alipay") @LoginRequired(loginSuccess = true) @ResponseBody

public String alipay(String outTradeNo, BigDecimal totalAmount, Model model, HttpServletRequest request) {

String userId = (String) request.getAttribute("userId");

String username = (String) request.getAttribute("username");

String form = null;

AlipayTradePagePayRequest alipayRequest = new AlipayTradePagePayRequest();

alipayRequest.setReturnUrl(AlipayConfig.return\_payment\_url);

alipayRequest.setNotifyUrl(AlipayConfig.notify\_payment\_url);

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

map.put("out\_trade\_no", outTradeNo); map.put("total\_amount", totalAmount);

map.put("product\_code", "FAST\_INSTANT\_TRADE\_PAY");

OmsOrder omsOrder = orderService.getOrderByOutTradeNo(outTradeNo);

map.put("subject", omsOrder.getBillContent());

alipayRequest.setBizContent(JSON.toJSONString(map));

try {

form = alipayClient.pageExecute(alipayRequest).getBody();

} catch (AlipayApiException e) {

e.printStackTrace(); }

PaymentInfo paymentInfo = new PaymentInfo();

paymentInfo.setCreateTime(new Date());

paymentInfo.setOrderId(omsOrder.getId());

paymentInfo.setOrderSn(outTradeNo);

paymentInfo.setSubject(omsOrder.getBillContent());

paymentInfo.setTotalAmount(totalAmount);

paymentInfo.setPaymentStatus("未付款");

paymentService.savePaymentInfo(paymentInfo);

paymentService.sendDelayPaymentResultCheckQueue(outTradeNo, 5);

return form;

}

## 订单模块完善

### 主要页面

1. 卖家订单页面：

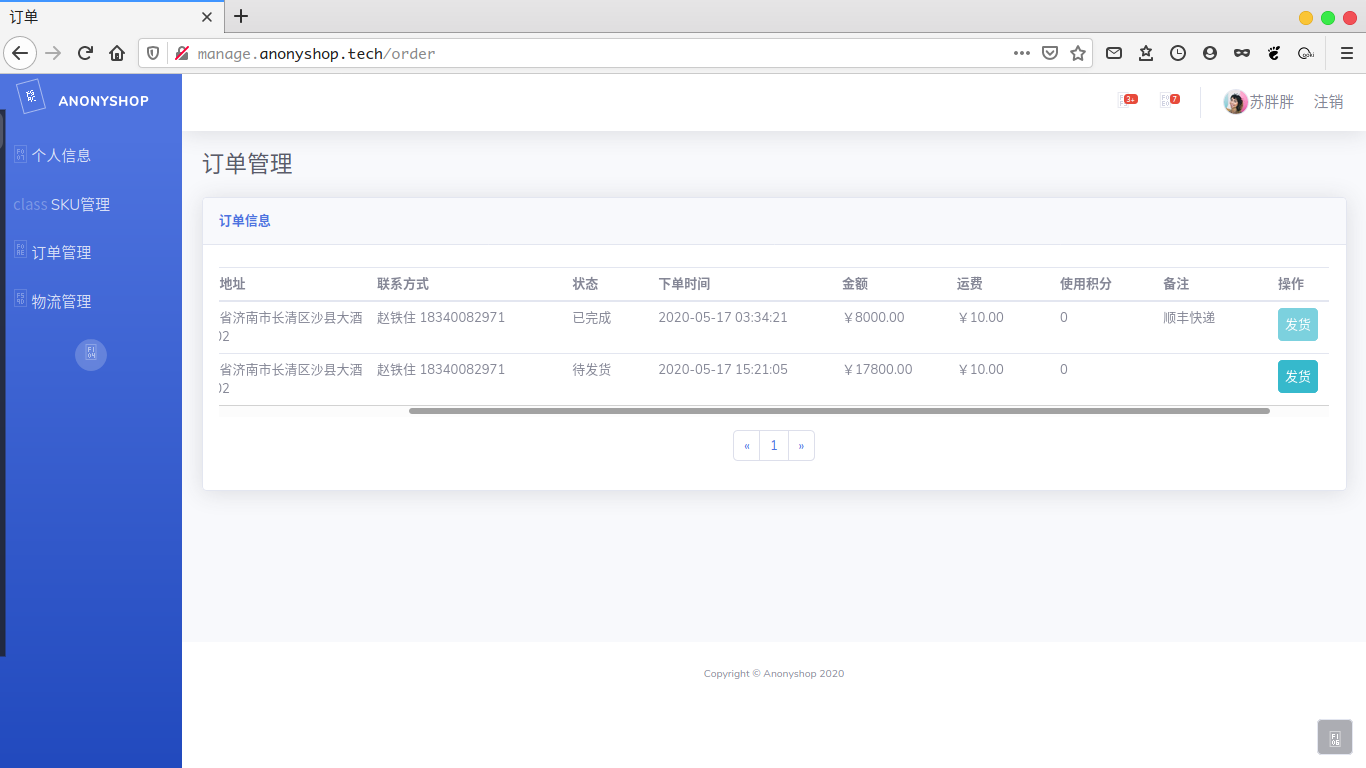


图5- 16 卖家个人中心订单页面

1. 用户订单页面：



图5- 17 用户个人中心订单页面

### 主要代码

@RequestMapping @LoginRequired

public String toOrder(HttpServletRequest request, Model model){

addUserInfoToModel(request, model);

PageInfo<OmsOrder> orderList = null;

if (roleId.equals("1")) {

orderList = orderService.getUserOrderList(1, userId);

for (OmsOrder omsOrder : orderList.getList()){

String storeName = storeService.getStore (omsOrder.getStoreId()).getName();

omsOrder.setStoreName(storeName); }

} else { orderList = orderService.getStoreOrderList(1, getStoreId(request));

String storeStr = CookieUtil.getCookieValue(request, "store", true);

SmsStore store = JSON.parseObject(storeStr, SmsStore.class);

String storeName = store.getName();

for (OmsOrder omsOrder : orderList.getList()){

omsOrder.setStoreName(storeName); } }

model.addAttribute("orderList", orderList);

return "user\_seller/order";

}

@RequestMapping("delete") @LoginRequired

public String delOrder(Integer pageNum, String orderId, HttpServletRequest request, Model model){

addUserInfoToModel(request, model);

orderService.delOrderById(orderId);

OmsDelivery delivery = new OmsDelivery();

delivery.setOrderId(orderId);

deliveryService.delDeliveryByOrderId(delivery);

PageInfo<OmsOrder> orderList = orderService.getUserOrderList(pageNum, getStoreId(request));

for (OmsOrder omsOrder : orderList.getList()){

String storeName = storeService.getStoreByStoreId(omsOrder.getStoreId()).getName();

omsOrder.setStoreName(storeName); }

model.addAttribute("orderList", orderList);

return "user\_seller/order";

}

@RequestMapping("list")

@LoginRequired

public String orderList(Integer pageNum, HttpServletRequest request, Model model){

addUserInfoToModel(request, model);

PageInfo<OmsOrder> orderList = null;

if (roleId.equals("1")) {

orderList = orderService.getUserOrderList(pageNum, userId);

for (OmsOrder omsOrder : orderList.getList()){

String storeName = storeService.getStore (omsOrder.getStoreId()).getName();

omsOrder.setStoreName(storeName);

}

} else {

orderList = orderService.getStoreOrderList(pageNum, getStoreId(request));

String storeStr = CookieUtil.getCookieValue(request, "store", true);

SmsStore store = JSON.parseObject(storeStr, SmsStore.class);

String storeName = store.getName();

for (OmsOrder omsOrder : orderList.getList()){

omsOrder.setStoreName(storeName);

}

}

model.addAttribute("orderList", orderList);

return "user\_seller/order";

}

## 商品评分模块

### 主要页面

1. 订单评论页面如图5-19所示：

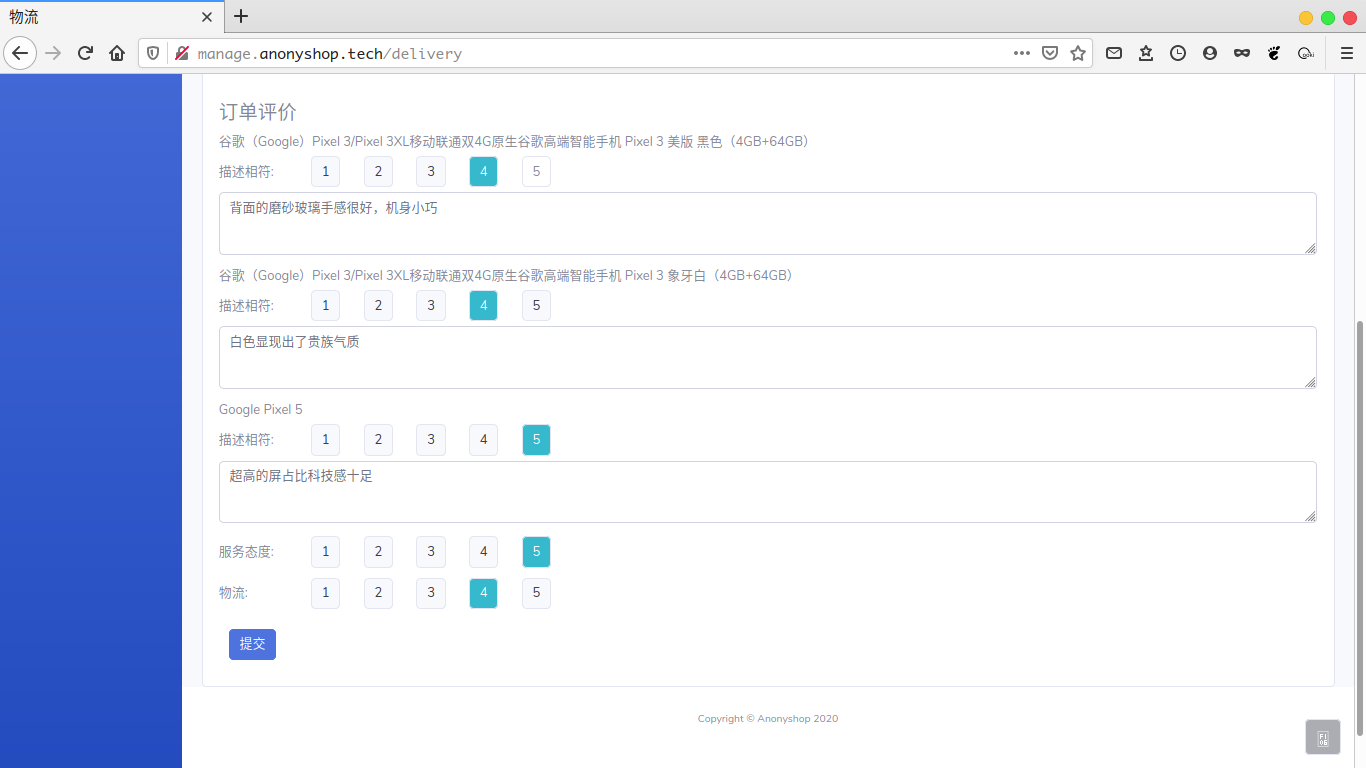


图5- 19 订单评论页面

1. 商品详情评论展示如图5-20所示：

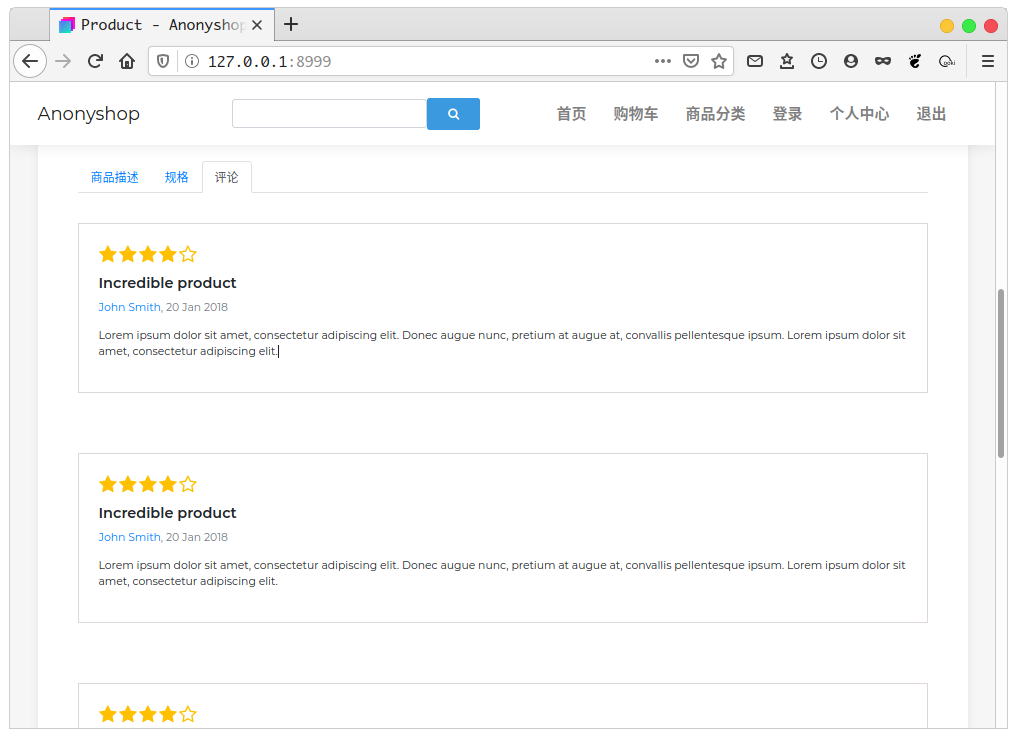


图5- 20 商品详情评论展示

### 主要代码

public void addCommment(Comments comments) {

List<PmsComment> pmsComments = comments.getComments();

for (PmsComment comment : pmsComments) {

pmsCommentMapper.insertSelective(comment);

}

Connection connection = null;

Session session = null;

try {

connection = activeMQUtil.getConnectionFactory().createConnection();

session = connection.createSession(true, Session.SESSION\_TRANSACTED);

Queue product\_review\_queue = session.createQueue("PRODUCT\_REVIEW\_QUEUE");

MessageProducer producer = session.createProducer(product\_review\_queue);

TextMessage textMessage = new ActiveMQTextMessage();

textMessage.setText(JSON.toJSONString(pmsComments));

producer.send(textMessage);

session.commit();

} catch (Exception ex) {

try { session.rollback(); }

catch (JMSException e1) { ex.printStackTrace(); }

} finally {

try { connection.close(); }

catch (JMSException e1) { ex.printStackTrace(); }

}

}

}

@JmsListener(destination = "PRODUCT\_REVIEW\_QUEUE", containerFactory = "jmsQueueListener")

public void newComment(TextMessage textMessage) throws JMSException {

String pmsCommentsStr = textMessage.getText();

List<PmsComment> pmsComments = JSON.parseArray(pmsCommentsStr, PmsComment.class);

smsStoreProductService.updateProductScore(pmsComments);

}

public void updateProductScore(List<PmsComment> pmsComments) {

for (PmsComment pmsComment : pmsComments) {

SmsStoreProduct smsStoreProduct = new SmsStoreProduct();

smsStoreProduct.setPorductId(pmsComment.getProductId());

smsStoreProduct.setStoreId(pmsComment.getStoreId());

smsStoreProduct = smsStoreProductMapper.selectOne(smsStoreProduct);

BigDecimal commentScore = getCommentScore(pmsComment);

smsStoreProduct.setScore(smsStoreProduct.getScore().add(commentScore));

smsStoreProductMapper.updateByPrimaryKeySelective(smsStoreProduct);

}

}

private BigDecimal getCommentScore(PmsComment pmsComment) {

double commentScore = pmsComment.getProductScore() \* 0.5

+ pmsComment.getServiceScore() \* 0.3

+ pmsComment.getDeliveryScore() \* 0.2;

if (commentScore >= 3) commentScore = commentScore / 5 \* 0.01;

else commentScore = commentScore / 3 \* 0.01;

return new BigDecimal(commentScore);

}

## 后台管理模块完善

### 主要页面

1. 卖家物流管理列表页面：

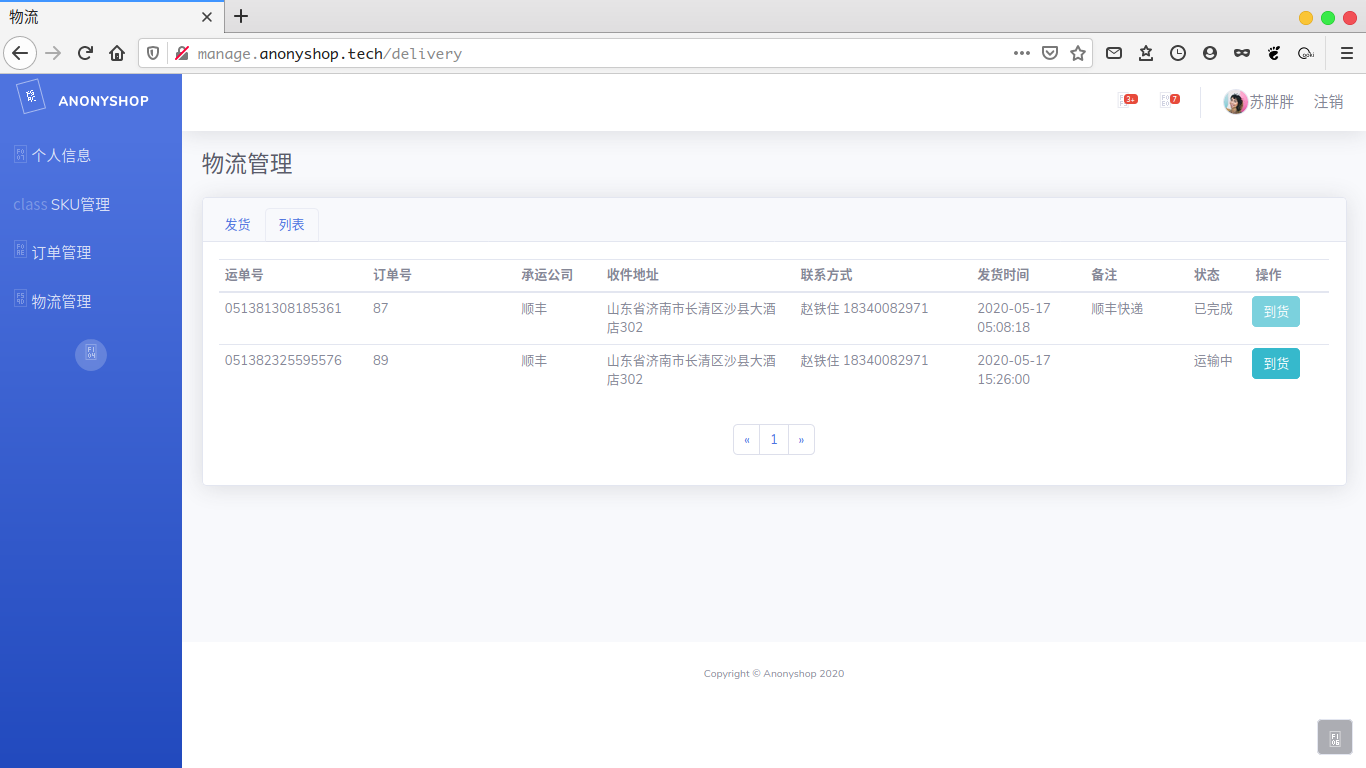


图5- 21 卖家个人中心物流管理页面

1. 卖家物流管理发货页面：

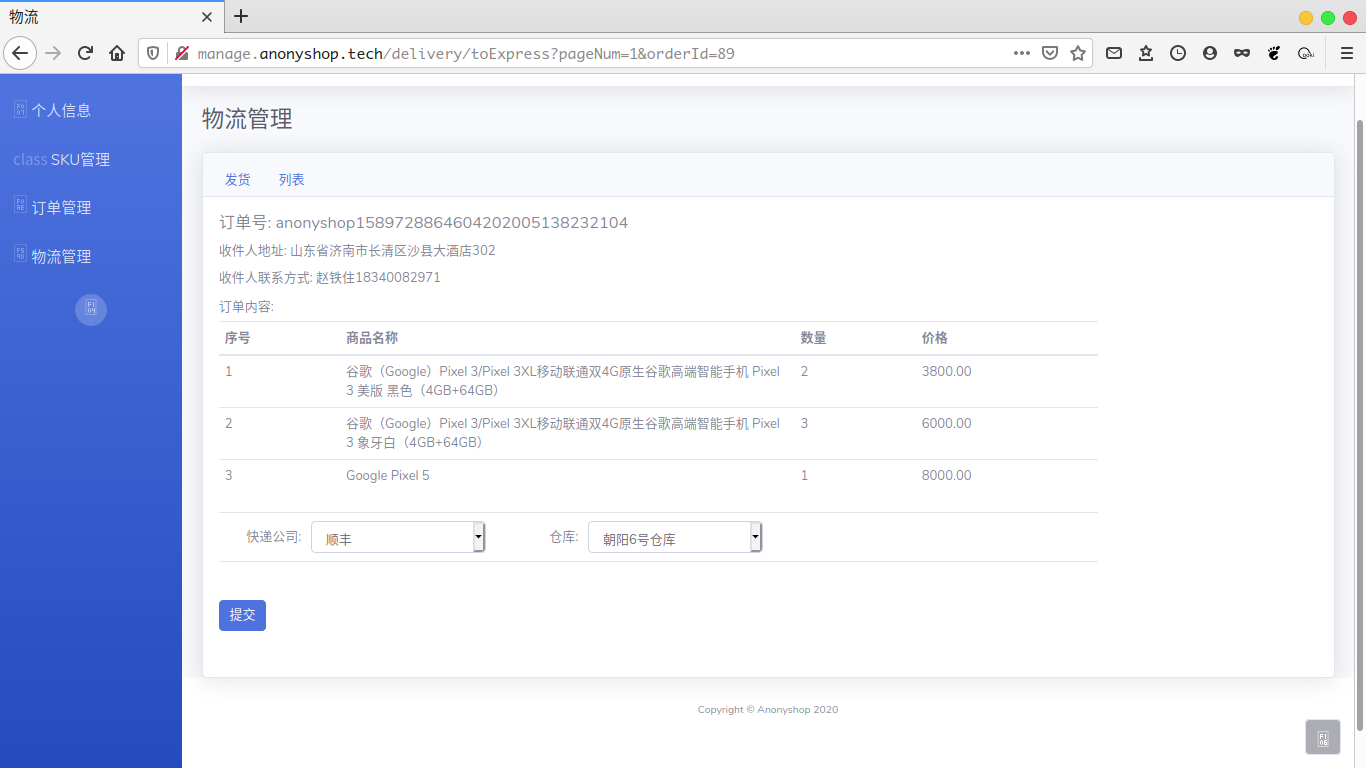


图5- 22 卖家个人中心物流管理发货页面

### 主要代码

@RequestMapping("send")

@ResponseBody

public String sendExpress(OmsDelivery delivery, Model model) {

deliveryService.sendDelivery(delivery);

OmsOrder omsOrder = new OmsOrder();

omsOrder.setId(delivery.getOrderId());

omsOrder.setStatus("2");

orderService.updateOrderDeliveryStatus(omsOrder);

return JSON.toJSONString("已发货");

}

@RequestMapping(value = {"toExpress", ""})

@LoginRequired

public String toExpress(HttpServletRequest request, Integer pageNum, String orderId, Model model) { addUserInfoToModel(request, model);

PageInfo<OmsDelivery> deliveryList = null;

if (roleId.equals("1")){

deliveryList = deliveryService.getUserProgressDelivery(1, userId);

} else {

deliveryList = deliveryService.getStoreProgressDelivery(1, getStoreId(request));

}

model.addAttribute("deliveryList", deliveryList);

if (StringUtils.isNotBlank(orderId)) {

OmsDelivery delivery = deliveryService.getDeliveryByOrderId(orderId);

String returnUrl = "http://manage.anonyshop.tech/order/list?pageNum=" + pageNum;

OmsOrder order = orderService.getOrderById(orderId);

List<WmsWareInfo> wareInfos = wareService.getAllWares(getStoreId(request));

model.addAttribute("delivery", delivery);

model.addAttribute("returnUrl", returnUrl);

model.addAttribute("order", order);

model.addAttribute("wareList", wareInfos);

model.addAttribute("storeId", getStoreId(request));

}

return "user\_seller/delivery";

}

@RequestMapping("list")

public String deliveryList(HttpServletRequest request, Integer pageNum, String userId, Model model) {

PageInfo<OmsDelivery> deliveryList = null;

if (StringUtils.isNotBlank(userId)) {

deliveryList = deliveryService.getUserProgressDelivery(pageNum, userId);

} else {

deliveryList = deliveryService.getStoreProgressDelivery(pageNum, getStoreId(request));

}

model.addAttribute("deliveryList", deliveryList);

if (StringUtils.isNotBlank(userId)) {

return "user\_seller/delivery::tab-delivery-user";

}

return "user\_seller/delivery::tab-delivery-seller";

}

@RequestMapping("arrive")

public String deliveryArrive(HttpServletRequest request, String orderId, Integer pageNum, Model model) {

OmsDelivery delivery = new OmsDelivery();

delivery.setOrderId(orderId);

delivery.setStatus("2");

delivery = deliveryService.updateDeliveryByOrderId(delivery);

OmsOrder order = new OmsOrder();

order.setStatus("3");

order.setId(orderId);

orderService.updateOrderDeliveryStatus(order);

PageInfo<OmsDelivery> deliveryList = deliveryService.getStoreProgressDelivery(pageNum, getStoreId(request));

model.addAttribute("deliveryList", deliveryList);

return "user\_seller/delivery::tab-delivery-seller";

}

# 系统测试与调试

软件测试的四个阶段分别为单元测试、集成测试、系统测试、验收测试，因为本项目为毕业设计项目，没有验收测试，单元测试也已经在代码编写的过程中做过，所以本章将主要讲述集成测试和系统测试。

## 测试目的

测试的目的在于系统地找出软件中潜在的各种错误和缺陷，证明软件的功能和性能和需求文档中的定义相符合，测试通过不能证明软件中不存在错误或缺陷，它只能说明软件中存在尚未发现的错误或缺陷。

## 测试方法

单元测试：对代码中的最小可测试单元进行测试，在项目中引入Junit之后可以通过@Test注解对类中的单个方法进行测试，目的是检查一个可执行单元是否按照预期进行执行。

集成测试：在单元测试的基础上，将子模块所属的SpringBoot应用以调试模式启动，在关键位置添加断点，跟踪所有方法联动执行的过程以及传递的参数是否与预期一致。

系统测试：在集成测试的基础上，启动所有模块，按照一定的顺序进行操作。系统测试的目的在于通过与系统的需求做对比，发现系统与需求分析相矛盾或不一致的地方。

## 测试环境与测试条件

电脑型号：Dell Inspiron 14 7447

系统：Ubuntu 20.04 LTS

内核：5.4.0-29-generic

内存：12 GiB

硬盘容量：480.1 GB

处理器：Intel® Core™ i5-4210H CPU @ 2.90GHz × 4

显卡：NV117 / Intel® HD Graphics 4600 (HSW GT2)

## 测试项目

由于篇幅原因，本章只对项目的核心模块进行测试，包括单点登录模块、订单和支付模块。

### 单点登录测试

1. 登录验证

表6- 1 登录测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 功能特性 | 用户登录验证 | | | | |
| 测试目的 | 验证用户输入的身份信息是否有效 | | | | |
| 测试数据 | 用户名：程心 邮箱：18340082971@163.com 密码：123456 | | | | |
| 测试内容 | 操作描述 | 数据 | 期望结果 | 实际结果 | 测试状态 |
| 1 | 输入正确的用户名和密码，点击“登录”按钮。 | 用户名：程心  密码：123456 | 正确登录，带token返回首页 | 正确登录，带token返回首页 | 与期望结果相同 |
| 2 | 输入正确的邮箱地址和密码，点击“登录”按钮。 | 邮箱：[18340082971@163.com](mailto:18340082971@163.com)  密码：123456 | 正确登录，带token返回首页 | 正确登录，带token返回首页 | 与期望结果相同 |
| 3 | 输入正确的用户姓和错误的密码，点击“登录”按钮。 | 用户名：程心  密码：1111111 | 返回首页，token为fail | 返回首页，token为fail | 与期望结果相同 |
| 4 | 输入正确的邮箱地址和错误的密码，点击“登录”按钮。 | 邮箱：[18340082971@163.com](mailto:18340082971@163.com)  密码：11111111 | 返回首页，token为fail | 返回首页，token为fail | 与期望结果相同 |

1. 用户令牌验证

表6- 2 令牌验证

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 功能特性 | 用户令牌验证 | | | | |
| 测试目的 | 验证用户输入的身份信息是否有效 | | | | |
| 测试数据 | 用户名称：程心 密码：123456 | | | | |
| 测试内容 | 操作描述 | 数据 | 期望结果 | 实际结果 | 测试状态 |
| 1 | 输入正确的用户名和密码，点击“登录”按钮，返回首页后点击导航栏的购物车 | 用户名：程心  密码：123456 | 拦截器拦截到请求，验证token，将token中加密的用户信息解密并放入请求中，正确到达购物车页面，导航栏的登录按钮变成个人中心 | 拦截器拦截到了请求，请求中携带的token通过了认证，导航栏的登录按钮变成个人中心 | 与期望结果相同 |
| 2 | 输入正确的用户名和错误的密码，点击“登录”按钮，返回首页后点击导航栏的购物车 | 用户名：程心  密码：111111 | 拦截器拦截到请求，验证token，token中没有数据，没有通过验证，返回的请求头没有数据，购物车页面导航栏没有变化 | 拦截器拦截到了请求，验证token，token中没有数据，没有通过验证，返回的请求头没有数据，购物车页面导航栏没有变化 | 与期望结果相同 |

### 订单和支付模块模块测试

订单和支付模块的测试如表6-3中所示：

表6- 3 下单、支付、结算

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 功能特性 | 订单支付验证 | | | | |
| 测试目的 | 验证支付流程 | | | | |
| 测试数据 | 用户名、密码：程心(123456)，苏胖胖(123456) | | | | |
| 测试内容 | 操作描述 | 数据 | 期望结果 | 实际结果 | 测试状态 |
| 1 | 输入正确的用户名和密码, 点击“登录”按钮,返回首页后点击导航栏的购物车 | 用户名：程心  密码：123456 | 通过验证，显示出购物车中的商品 | 通过验证，显示出购物车中的商品 | 与期望结果相同 |
| 2 | 选择两样商品，点击结算 |  | 跳转到订单确认页面，显示出订单内容，订单价格，选择收货地址 | 跳转到订单确认页面，显示出订单内容，订单价格，选择收货地址 | 与期望结果相同 |
| 3 | 选择收货地址，点击确认按钮 |  | 生成订单号，跳转到选择付款方式页面 | 生成订单号，跳转到选择付款方式页面 | 与期望结果相同 |
| 4 | 选择支付宝付款，点击确认按钮 |  | 跳转到支付宝结算页面 | 跳转到支付宝结算页面 | 与期望结果相同 |
| 5 | 扫码付款 |  | 跳转到支付成功页面 | 跳转到支付成功页面 | 与期望结果相同 |
| 6 | 进入个人中心，查看订单 |  | 订单显示已付款 | 订单显示也付款 | 与期望结果相同 |
| 7 | 登录卖家账号，查看订单 | 用户名：苏胖胖  密码：123456 | 有新的订单数据，状态为待发货 | 有新的订单数据，状态为待发货 | 与期望结果相同 |
| 8 | 点击发货按钮 |  | 跳转到物流管理页面，并有新的待发货订单 | 跳转到物流管理页面，并有新的待发货订单 | 与期望结果相同 |
| 9 | 选择物流公司和仓库，点击发货按钮 |  | 发货成功，生成订单号，物流状态变为运输中，订单状态便问已发货， | 发货成功，生成订单号，物流状态变为运输中，订单状态便问已发货， | 与期望结果相同 |
| 10 | 退出卖家账号，登录用户账号，点击个人中心，查看订单和物流状态 | 用户名：程心  密码：123456 | 订单状态显示为已发货，物流状态显示为运输中 | 订单状态显示为已发货，物流状态显示为运输中 | 与期望结果相同 |
| 11 | 退出用户账号，登录卖家账号，进入物流管理，点击到货 | 用户名：苏胖胖  密码：123456 | 订单状态更新为待签收，物流状态更新为已送达 | 订单状态更新为待签收，物流状态更新为已送达 | 与期望结果相同 |
| 12 | 退出卖家账号，登录用户账号，点击个人中心，查看订单和物流状态，在物流管理页面点击签收 | 用户名：程心  密码：123456 | 订单状态更新为待签收，物流状态更新为已送达，点击签收后物流状态变为已完成，返回订单页面后订单状态变为以签收，出现评价按钮 | 订单状态更新为待签收，物流状态更新为已送达，点击签收后物流状态变为已完成，返回订单页面后订单状态变为以签收，出现评价按钮 | 与期望结果相同 |

## 测试结论

关键的流程运行都没有问题，本系统的目标是能用，而不是足够好，所以已经达到了目标。在下单和支付模块使用了三个消息队列，分不同的时间去修改订单状态、生成支付信息并生成物流信息，而不是在同一时间段串行地去执行这些任务，明显缩短了支付成功后返回前台页面地延时。最重要的是用户地身份认证，传统的session无法在分布式系统中存在，单独的认证中心很好的解决了用户身份认证地问题。

## 遇到的问题及解决方案

1. Dubbo注册中心没法运行，2019年阿里巴巴对dubbo进行了升级，在Github的仓库中不再提供打包编译好的jar包。因此需要手动编译，在编译源码的时候一直出现各种各样的错误，调试了很久，结合网上的解决方案，主要问题在于jdk版本，dubbo版本和mysql驱动版本。

解决方案：将jdk版本换为OpenJDK8，不使用bubbo2.8的版本，换为2.6.8的版本，替换掉阿里巴巴源码中的mysql驱动，使用mysql8的驱动，修改pom文件中关于mysql依赖版本的内容，重新使用maven编译修改后的阿里巴巴dubbo源码，将生成的war包放到tomcat目录下，重新启动tomcat。

1. FastDFS配置极其困难，服务器为Ubuntu18的系统，网上的一些解决方案基本都是基于CentOS系统，而且那些解决方案都是在windows开发环境下，在虚拟机中配置。但是本系统的开发环境主要是Ubuntu，而且由于内存都分给了Spring Boot应用，没有多余的内存再开虚拟机，打算部署FastDFS的服务器也是实实在在的云服务器，按照FastDFS官网的教程尝试了很久都没有成功。

解决方案：使用Docker，在Docker HUB上有很多一键安装的docker镜像，关于FasfDFS的也有，镜像中包含了运行FastDFS的所有第三方库以及已经编译好的FastDFS的可执行文件，只需要使用docker命令对FastDFS的连接配置就可以使用。

1. 本系统使用了Spring Boot 框架，每个模块在运行时需要大量内存，所有模块同时启动的时候会造成本机内存溢出。

解决方案：通过Intellij IDEA，手动调整为每个应用分配的内存，对于简单的web应用，分配128m内存，功能稍多的web应用分配256m内存，简单的service应用分配512m内存，复杂的service 应用分配1024m内存。

1. 管理系统（包括用户、卖家、平台管理员）中导航栏部分的BootStrap图标无法正常显示，Firefox的开发工具提示说字体在整合时出现了异常，暂无解决方案。
2. 对于要使用邮件的功能，如注册、修改密码等，系统发送出的邮件有很大的延时，本系统使用的是本文作者的私人雅虎邮箱，雅虎今年更新了安全协议，可能是这个原因，也可能是网络环境的问题。

解决方案：更换一家邮件服务厂商，比如微软的outlook 或者Google 的Gmail，考虑到这个问题并非是紧急的系统缺陷或Bug，暂时不进行更换。

1. 订单评论功能中点击评分按钮之后一直没有反应，设计中应该是点击之后按钮的样式发生改变，变成浅绿色，未选中的按钮变成白色。

解决方案：将JavaScript 脚本中点击评分按钮的函数修改成监听器模式，因为点击评分按钮显示评分区域，评分区域是使用异步加载后Thymeleaf重新渲染出来的，普通的JavaScript 函数无法更改渲染出来的页面的样式。

1. 商品详情页面的图片不知道什么时候开始无法正常显示，在最开始做商品详情模块的时候是可以正常显示的。因为HTML页面是使用Bootstrap Studio做出来的，导出的页面使用shell脚本在将不同的页面和JavaScript复制到不同的web应用中，所有页面使用了相同点css文件，可能是后来某个模块修改了css文件和之前发生了冲突，但是调试了很久都没有找出冲突的地方。

解决方案：在Bootstrap Studio中重做了详情页，将该页面单独整合product-web到项目中。

1. 分类检索模块中，由于要将大量的筛选条件放入表单中提交，导致返回的内容一直出现请求头过大引起的溢出，溢出的部分会冲掉HTTP响应报文的载荷部分，使得响应数据在渲染时一直出现异常。

解决方案：在search-web模块的application.properties文件中手动添加请求头允许大小取代默认值。

1. 评价商品之后发送一个队列消息提醒商品服务模块重新计算卖家店铺中商品评分，但是队列在获取connection 连接和session 会话时一直无法得到一个连接导致空指针异常。

解决方案：在manage-web应用的应用配置文件中添加ActiveMQ的配置，因为消息队列是在做订单模块时才添加的，也不是所有模块都会用到，所以忘了配置。

1. 单点登录模块，如果登录成功会返回一个为用户临时颁发的令牌，令牌是一串特殊的字符串，携带在请求头中，重新返回到登录前的页面，如果登录失败令牌内容是“fail”，登录失败后如果重新输入正确的用户信息登录成功，请求头中一直有“token=fail”而导致无法在跳转方法的拦截器中通过认证。

解决方案：尝试使用重定向而不是跳转来忽略之前登录失败时请求头中的信息，但是新的令牌无法放到请求头中。之后在通过在JavaScript 脚本中判断是否请求头中是否有“token=fail”，如果有就删除，问题得到了解决。

## 系统存在的不足

由于前期做了大量的准备工作，导致开发时间短，本系统着重做了商品管理的模块，其他模块的设计以及开发都有或多或少的缺陷，存在很多不足。

在订单模块，有很多对于卖家数据库相关表的操作是串行的，因为考虑到卖家相对于用户来说人数少很多，所以没有使用消息队列将这些操作分离，导致有些复杂操作在浏览器中获得反馈的时间会有些长。如果这些串行操作中某一步出现了异常就会使之后的操作无法顺利执行，应该将所有的串行操作分离，设计更多的队列。

系统中有很多数据是经常被访问的，比如首页的畅销商品数据，分类搜索页面中在没有开始搜索之前显示的商品数据，平台属性列表，这些数据访问量大，而且不是经常改变的数据，应该放到缓存中。但是因为时间有限，在开发过程中都没有做，缓存中只存放了更重要并且访问更加频繁的数据，比如用户在登录期间的身份信息，购物车列表，订单信息等。

代码中很多地方没有做安全操作的判断，比如所有对于数据库的写操作应该全部在try catch语句块中执行，并且对于修改和删除操作，应该判断传入的参数是否合理。在开发中，删除库存单元的操作涉及到四张表，在做删除库存单元的功能的时候，由于没有一步一步跟踪测试，导致传入的参数不合理，数据库误以为是模糊匹配删除，导致删除了库存单元平台属性和库存单元销售属性表中的所有数据。

由于涉及到金融交易，个人用户只能使用第三方支付服务提供的沙箱环境，之前设计的支付流程，用户付款后钱由平台代收，等确认收货后再分三笔在不同的时间转给卖家，这里无法实现，因为无法使用转账功能，个人用户申请到的一个卖家账户和一个卖家账户，而设计的流程需要三个支付宝账户，转账功能需要有资质的企业申请企业账号才可以使用。Payapl也只给个人用户提供两个账户，和支付宝差不多。

由于使用了第三方支付平台，第三方支付平台会完成付款人身份的认证，他们使用的技术也更加成熟。因为使用类似于SSH安全登录的技术，在web应用中会增加很多不必要的麻烦，而却对于普通用户来说并不是很友好，所以使用公开密钥的设计在本系统中没有实现。本系统中用户的密码是使用Bcrypt算法加密的，Bcrypt 算法是基于Blowfish加密设计的一个慢哈希算法，BCrypt是单向的, 而且经过salt和cost的处理, 与MD5加密算法相比, 安全性得到了极大地提高[[[12]](#endnote-13)]，即使数据库服务器被攻破，攻击者也无法获取用户真实的口令和用户个人的数据，因为他们无法得知用户名对应于用户的真实身份。系统中存储的用户邮箱也应该使用Bcrypt加密，这将更好的保护用户的身份信息，但是这就又给使用邮箱登录带来的新的挑战。在用户注册和用户重置密码时没有做用户密码强度的判断，对于强度较弱的密码很可能会被攻击者猜到。

用户删除订单的行为也没有做判断，应该是对于未付款订单和已完成订单可以删除，已经付款并且正在进行中的订单无法被删除。而且由于本系统旨在验整使用很少用户信息可以实现可靠网络购物，服务器资源有限，应该定期删除已完成的订单以空出更多的服务器资源，可以再创建一个SpringBoot应用，用来清理这些已经完成的订单数据，然后生成类似于单点登录中的token的加密订单记录，这种信息将更加紧凑，更加节省空间，将在处理买卖双方矛盾的时候使用，同时用户和卖家也可以使用这个记录来查看历史数据。

# 总结

本文主要记录了用户信息可隐藏的匿名购物系统的设计以及实现，最后得出结论，安全的网络购物主要有三个必要的要素，第一是一个有效的联系方式，第二是有效的收货地址，第三是支付系统。

对于第一个要素，一个有效的联系方式可以在物流抵达之后联系到买家，这里推荐使用电子邮箱。短信可以被窃听，截获，被有关部门随意查看，而且将用户电话写在快递信息上本身就很不安全。很多公益性网站提供了临时邮箱服务，时效长短不一，使用一次性的邮箱作为联系方式可以在很大程度上保护用户的隐私，不造成用户信息泄露。对于金额较小的交易，使用这种临时邮箱就可以了，对于数目较大的交易，推荐使用个人常用的邮箱之一，因为对于售后还存在潜在的需求，方便联系。

第二个要素，一个有效的收货地址，因为物流必须要有确切的收货地址才能送达。由于人类的群居属性，一般一个收货地址会有很多人，所以收货地址的泄露，在不知道用户身份的情况下对用户个人造成的影响也是微乎其微。

第三个要素，支付系统，买卖交易就需要金融系统的支持，由于金融系统的特殊性，尤其是网络金融系统，因此必须能够保证用户的财产安全。使用成熟的第三方支付是最明智的选择，尤其是在全世界范围内广泛使用的支付方式，比如Paypal。但是目前Paypal不支持人民币交易，汇率的换算为商品价格的展示带来了新的挑战，所以使用了在中国广泛使用的支付宝。

因此可以确信，目前大多数电商平台对于用户信息的搜集是不合理的，他们穷尽手段搜集所有能搜集到的信息，然后使用相应的算法给用户定向推销，从而谋取暴利，却没有给用户支付任何的信息使用费用，这本身就是不公平的。而且对于规模较小的平台，他们的安全性也值得考虑，他们没有更多的人力财力物力去保护搜集到的用户信息，使得用户的信息处于危险的境地。

当然，要符合法律规范，可能还需要很多信息才能使得网站真正的合法上线运行，但是上面提到的三点是最重要的。

本系统的开发时间很短，要为用户提供可靠的网络购物还需要很多工作要做，由于系统采用了分布式的架构，使得维护和后续开发也变得简单。后续工作将主要围绕第六章第7小节中提到的系统存在的不足进行。

# 参考文献

# 致 谢

这次毕业设计是本科期间的最后一次作业，融合了本科期间学到的很多知识，感谢本科期间教我的那些老师们，是你们的辛苦付出才有了我今天的收获。

另外需要感谢的是我的毕业设计指导老师赵盛荣以及我的班主任梁虎老师，在做毕业设计期间他们给予了我很多帮助。

毕业设计期间遇到了很多大大小小的问题，有几个技术好的朋友也帮助了我很多。

感谢所有帮助了我的老师和朋友。

最后感谢各位老师百忙之中抽出时间来对我的论文进行评审并给出了宝贵的意见。

1. ]刘明辉,张玮,陈湉等. 数据安全与隐私保护技术研究[J]. 邮电设计技术, 2019(04): 25-29. [↑](#endnote-ref-2)
2. ]Altawy Riham, Elsheikh Muhammad, Youssef Amr M, Gong Guang. Lelantos: A Blockchain-based Anonymous Physical Delivery System[C]. 2017 15th Annual Conference on Privacy, Security and Trust: 15-24. [↑](#endnote-ref-3)
3. ]王涛. 电子商务中个人信息泄露成因分析与防范研究[D]. 黑龙江大学, 2019. [↑](#endnote-ref-4)
4. ]程朝辉. 基于标准算法的高效无证书密码系统[J]. 中国信息安全, 2019(10): 91-95. [↑](#endnote-ref-5)
5. ]张福泰,孙银霞,张磊等. 无证书公钥密码体制研究[J]. 软件学报,2011,22(06): 1316-1332. [↑](#endnote-ref-6)
6. ]暗网的养成: Tor(洋葱路由)的故事[J]. 中国信息安全, 2017(11): 65-68. [↑](#endnote-ref-7)
7. ]师艳琴. 快递实名制下的个人信息民法保护研究[D]. 山西大学,2018. [↑](#endnote-ref-8)
8. ]李宇,刘彬. 前后端分离框架在软件设计中的应用[J]. 无线互联科技, 2018,15(17): 41-42. [↑](#endnote-ref-9)
9. ]徐小龙,王纯. 统一管理Tomcat和Nginx软件栈[J]. 电信工程技术与标准化, 2014,27(08): 86-90. [↑](#endnote-ref-10)
10. []唐国磊,宋向群,王文渊,郭子坚.基于Java开源框架的港口信息服务系统设计[J].水运工程,2011(05):23-26. [↑](#endnote-ref-11)
11. []范展源,罗福强.JWT认证技术及其在WEB中的应用[J].数字技术与应用,2016(02):114. [↑](#endnote-ref-12)
12. ]扈玮,王立华.基于加盐BCrypt算法的电商安全模块设计及实现[J].计算机系统应用,2019,28(10):80-85. [↑](#endnote-ref-13)