



摘要

随着计算机技术与电子商务的快速发展,带有推荐功能的导向性购物网站已成为时代主流,一方面可以根据地理位置、浏览行为、购买行为等多个维度进行商品推荐,以更加个性化、差异化的服务用户。另一方面,随着个人用户量的增多,调整网站的技术架构、提升高并发处理能力也成为技术难点。二者正是困扰许多中小型电商企业的问题。

本设计针对电商服装购物网站,通过采用 Spring Cloud 微服务框架整合 Spring Boot 后台开发框架、Mybatis 数据库访问组件、Redis 缓存、Maven 依赖管理、前后端分离等技术手段来实现一个分布式、微服务的架构,从而提升网站的高并发处理能力,并且通过记录用户购物行为、IP 定位等手段来实现多维度的推荐系统,达到一定的购物导向性。其主要功能包括用户管理、服装商品与店铺管理、服装推荐、短信验证、服装店定位、公交线路规划等。

本论文详细阐述了本网站开发的分析、设计、实现等各阶段工作过程,可为同类型电商购物网站的设计开发提供一定的说明,对中小型电商企业实现导向性推荐系统、高并发技术架构两方面都具有较好的参考价值。

【关键词】 导向性 微服务 服装网站



Abstract

With the rapid development of computer technology and e-commerce, guided shopping websites with recommendation function have become the mainstream of the times. On the one hand, commodity recommendation can be carried out according to geographical location, browsing behavior, purchase behavior and other dimensions in order to provide more personalized and differentiated service users. On the other hand, with the increase of the number of individual users, it has become a technical difficulty to adjust the technical framework of the website and enhance the high concurrent processing capacity. Both of them are the problems that perplex many small and medium-sized e-commerce enterprises.

This design aims at e-commerce clothing shopping website. It integrates Spring Boot background development framework, Mybatis database access component, Redis cache, Maven dependency management, front-end and back-end separation and other technical means to achieve a distributed and micro-service architecture by using Spring Cloud micro-service framework, so as to enhance the high concurrent processing ability of the website, and record users' shopping behavior, IP and so on. Positioning and other means to achieve multi-dimensional recommendation system, to achieve a certain degree of shopping orientation. Its main functions include user management, clothing merchandise and shop management, clothing recommendation, short message verification, clothing store positioning, bus route planning and so on.

This paper elaborates the analysis, design, implementation and other stages of the development of this website, which can provide a certain explanation for the design and development of similar e-commerce shopping websites, and has a good reference value for small and medium-sized e-commerce enterprises to achieve the guidance recommendation system and high concurrent technology architecture.

【Keywords】 : Orientation Micro Service Clothing Websit



目录

1 绪论	1
1.1 论文研究背景介绍	1
1.2 电商服装购物网站现状与发展趋势	1
1.3 导向性服装店购物网站开发过程中的特点分析	2
1.3.1 缓存组件的使用	2
1.3.2 前后端分离	2
1.3.3 分布式部署架构	2
1.3.4 推荐导向系统提升用户体验	3
1.4 论文主要工作	3
1.5 开发环境说明	5
1.6 开发的目的是意义	5
2 使用技术栈分析	5
2.1 技术栈概览	5
2.2 微服务架构分析	6
2.2.1 原始单一架构分析	6
2.2.2 传统分布式架构分析	7
2.2.3 微服务架构	7
2.2.4 微服务实现之 Spring Cloud 框架	9
2.3 其他技术栈介绍	11
2.3.1 开发框架 Spring Boot 分析	11
2.3.2 数据库访问组件 Mybatis、C3P0 分析	12
2.3.3 缓存组件 Redis 分析	12
2.3.4 依赖包管理 Maven 说明	12
2.3.5 秒嘀科技短信验证平台	12
2.3.6 高德地图 IP 定位	13
2.3.7 技术 Zxing、Gson、HttpURLConnection 介绍	13



3 软件可行性研究	13
4 需求分析	14
4.1 需求规范	14
4.1.1 产品背景	14
4.1.2 产品概述	15
4.2 功能需求	15
4.3 性能需求	16
5 概要设计	16
5.1 用例分析	16
5.2 概要逻辑设计	17
5.3 业务流程分析	20
5.4 软件架构设计	21
6 软件详细设计	22
6.1 数据库设计	22
6.1.1 数据表设计	23
6.1.2 数据库物理设计	29
6.2 命名规范	30
6.2.1 数据库命名规范	30
6.2.2 项目文件命名规范	30
6.3 模块详细设计	30
6.3.1 用户管理模块	30
6.3.2 商品管理模块	32
6.3.3 推荐系统模块	35
6.3.4 店铺管理模块	37
6.3.5 购物管理模块	38
6.3.6 定位与地图模块	39
7 软件编码	39



7.1 编码环境与技术.....	39
7.1.1 数据库编码环境.....	40
7.1.2 项目编码环境.....	40
7.1.3 编码技术.....	40
7.2 网站架构.....	40
7.3 设计模式.....	41
7.4 代码结构.....	41
8 软件测试.....	43
8.1 单元测试.....	44
8.2 集成测试.....	49
8.3 功能测试.....	52
8.4 测试结论.....	53
总结	54
致谢	55
参考文献	56



1 绪论

1.1 论文研究背景介绍

根据国内电商年度报告显示,以阿里巴巴、京东等为代表的电子商务企业,旗下都拥有自己的 B2C、C2C 个人端购物网站,如阿里巴巴的淘宝网、天猫商城以及京东的京东商城等。这些大型电商企业凭借其先进的互联网管理理念与运作方式,引进了高精尖的互联网技术人才,使其电商产品做的十分完善,能承受大量的并发访问,从而逐渐改变了人们的购物方式与生活方式。

而对于中小型企业来说,目前已经开通了网上商店的企业尝到了电子商务带来的甜头,在网上出售可以降低自己的销售成本与运营成本,许多知名品牌都已经开始打造自己的线上品牌专营店,对于一些知名度稍弱的品牌,也在酝酿为自己的产品在网络上打开另一条销售渠道。这样,就可以面向来自世界各地的消费群体,不仅是企业,越来越多的个人也将加入到网络销售中,变成电子商务大军中的一员。

对于中小型企业来说,因曾经开发 OA 系统、管理系统的经验,更偏向于使用传统技术与架构,如传统后端技术框架 SSH、SSM,同程序多服务器部署等,这并不利于其开展 B2C、B2C 等针对个人用户电商业务,由于个人用户的用户量大,并发访问高,对于服务器性能、数据库性能都是一个考验,再者电子商务的业务场景较为灵活,需求变更快,传统的技术架构不适用于电商购物网站的业务场景。

1.2 电商服装购物网站现状与发展趋势

根据《2018-2022 年中国服装零售市场深度评估及未来发展趋势报告》2018 年,我国服装网购市场交易规模达到 7232 亿元,年复合增长率将达到 14%左右。2015 年以来,服装行业线上销售额迅猛增长,线上成交额已占到总体行业销售额一半以上,渗透率在 2017 年以来左右突破 50%,时间点晚于食品、家电数码



及美妆行业，但早于鞋包行业。服装行业目前的线上渗透率在所有行业里处领先地位。

为了更好的体现其产品特色，满足用户差异化，定制化，个人化购物的需求，中小型电子商务企业、知名服装店在依附大型电商网站的同时，也在开发自己的专属购物网站。

1.3 导向性服装店购物网站开发过程中的特点分析

由于存在大量的个人用户注册、同时刻并发访问、需求变化较快等情况，电商服装购物网站的开发都围绕着如何更好的处理并发请求、更快的响应速度，更好的做版本迭代来进行。其开发往往存在以下特点，目的是为了改善用户体验、利用版本更新，从而提高用户满意度，从而减少用户流失。

1.3.1 缓存组件的使用

由于并发量大，而关系型数据库实际存放位置都为硬盘，如果每次请求数据都从数据库去查，则会带来性能瓶颈。对于某些特定数据，更新频率低、查找频率高，往往使用内存数据库 NO-SQL 来进行缓存，如 Redis、MongoDB，数据直接从内存中获取，从而提高后端接口的响应速度。

1.3.2 前后端分离

传统的网站前端，如 PHP、JSP 等，采用脚本式的将 PHP、Java 语言嵌入前端网页当中，部署时也需要专门的服务器来运行。优点是开发方便，简化前端人员的开发工作，缺点是需要依赖特定的服务器、部署不够灵活，同样也存在性能瓶颈。而使用前后端分离的架构方式，前端由纯 HTML、CSS、JS 构成，则数据解析处理只需要在客户端浏览器进行，省去了在服务器的数据解析过程，从而数据解析压力由服务器转到客户端浏览器，则可以很大程度减轻服务器的压力，从而提高并发访问的性能瓶颈。前后端的数据交互一般使用 XML、Json 格式数据，现目前大部分企业都使用 JSON 数据。

1.3.3 分布式部署架构

传统的应用程序部署一台服务器，当并发访问量较高时，解决方式是通过加



大带宽与服务器配置。这种方法会带来资源过剩与程序崩溃两个极端问题，一方面，加大服务器内存、CPU 配置等硬件条件，在一定程度上确实可以提高响应速度，但是还受应用程序本身，如 Tomcat 线程限制，数据库连接数量限制等，导致过大的硬件配置资源过剩；另一方面，采用这种部署方式，一旦程序崩溃，则软件直接无使用。目前 B/S 架构中、往往会将应用程序部署多个，分布式部署在多台服务器上，再由 Nginx 来实现轮询调度，负载均衡，共同承担用户的并发请求，从而提高并发量，不仅解决了单台服务器资源过剩问题、而且当一个程序崩溃时，其他程序依旧可以提供服务，软件依然可以使用。

微服务架构则是在普通分布式架构上再进一步优化，将大型应用程序拆分成若干个微服务，更细微的区分了存在性能瓶颈的功能点，通过增大带性能瓶颈服务的部署数量来提高并发处理能力，为程序服务器资源分配提供了有效依据，使得分布式部署更加合理。

1.3.4 推荐导向系统提升用户体验

由市场上的淘宝网、天猫商城、京东购物等大型电商购物网站的引领作用，中小型企业的购物网站，也需要推荐系统来更加个性化的发现用户喜好，进行商品推荐、店铺推荐，带来更加人性化的购物体验。

1.4 论文主要工作

本论文，以服装购物网站为例，对电商购物网站中的推荐系统作了研究与开发、对微服务分布式架构的开发技术进行了研究与运用。从需求分析开始，到测试完成，详细的记录与解释了导向性购物网站的技术栈、开发过程、测试与分析等。主要分为以下几个方面：

（1）研究 Redis 缓存、微服务分布式架构、IP 定位、推荐导向及开发框架、Git 版本控制等相应技术点与实现方式。

（2）确立需求分析和设计方案，利用主流技术框 Spring Boot、Mybatis 数据库访问组件与 C3P0 连接池进行各个功能模块进行后台开发。

（3）前端开发，使用 HTML5、CSS3、LayerUI、JavaScript、Bootstrap 进



行开发，利用 Ajax 实现前后端分离。

(4) 单元测试、集成测试、功能测试。对实现的功能和网页展现形式进行测试，使用 Spring Boot 结合 Junit 对数据库相关模块进行单元测试，使用 SwaggerUI 自动化测试框架对后端接口进行集成测试，保留测试代码与测试数据。

(5) 使用 Spring Cloud 实现微服务架构与部署，使用 Eureka 注册中心实现服务集中管理。

(6) 记录开发过程中遇到的困难以及难点，分析出网站的难点以及重点、实用价值及参考价值。

本文的研究内容包括 8 章：

第 1 章为绪论，主要对本论文的研究背景进行了介绍，对电商服装购物网站的发展现状以及发展趋势作了简单的分析，对导向性服装店购物网站开发过程中的特点进行了简单的描述，以及对论文的主要工作进行了罗列，对软件开发环境作了简单的介绍，以及介绍了本次开发导向性服装店购物网站的目的和意义。

第 2 章对该网站的使用的技术栈作了较为详细的分析，包括技术本身的优势、适用的场景以及与传统技术的对比说明。

第 3 章为可行性研究，主要从技术可行性、经济可行性、操作可行性、法律可行性等方面对网站进行了可行性分析。

第 4 章为需求分析，其中包括网站实现的功能与目标，功能的需求，性能的需求，以及设计思路和初步方案的选定。

第 5 章为网站的概要设计，主要用了用例分析、概要逻辑设计、软件架构设计。

第 6 章为软件详细设计，包括数据库的设计、项目命名规范、各个模块功能的详细设计。

第 7 章为软件编码，包括编码环境、软件架构、设计模式、代码结构。

第 8 章为软件的测试，从单元测试、接口集成测试、到功能测试进行了测试，以保证网站功能的完整性以及系统的稳定性。



1.5 开发环境说明

- (1) 操作系统: Windows 10 企业版、Ubuntu 16.04.4
- (2) 程序语言: Java、Html5、JavaScript、SQL
- (3) 后端开发工具: IntelliJ IDEA, 目前业界最流行的后端开发工具, 相对于 Eclipse 在开发与调试上有不少提升, 代码排列查找也更加方便。
- (4) 前端开发工具: Visual Studio Code, 一款界面美观, 插件丰富的前端开发 IDE。
- (5) 版本控制: Git, 代码仓库使用 Github 托管
- (6) 其他辅助工具: Xshell 远程连接工具、Navicat 数据库视图工具、Photoshop 图片处理、Chrome 浏览器调试等。

1.6 开发的目的是和意义

本设计针对电商服装购物网站, 进行推荐导向系统、微服务实现分布式架构两个关键点进行探索与实现。

在该开发过程中, 对技术进行研究与学习, 对代码风格进行规范, 锻炼自己的学习能力, 同时也为自己技术栈新增元素, 提升了自己技术实力。另外, 该开发可为同类网站的设计开发提供一定的说明, 对于中小型电商企业实现导向性推荐系统、高并发技术架构两方面都具有较好的参考价值。

2 使用技术栈分析

2.1 技术栈概览

(1) 后端: Spring Cloud 微服务框架、Spring Boot 开发框架、Mybatis 数据库访问组件、C3P0 数据库连接池、Redis 缓存、Maven 依赖包管理、Gson 格式化组件、秒嘀科技短信验证平台、IP 定位高德地图 API、Zxing 二维码生成、HttpURLConnection 网络请求与爬虫

(2) 前端: LayerUI、Bootstrap 前端框架、Html5、CSS3、Ajax、JavaScript

2.2 微服务架构分析

对于企业来说，随着公司业务量的飞速发展，平台面临的挑战已经远远大于业务量的增加。需求不断变化、用户数量增加，面临的复杂度也大大增加。在这个背景下，平台的技术架构也完成了从传统的单体应用到微服务化的演进。下面将对传统原始单一架构，到传统分布式架构，到微服分布式架构，分别进行分析与说明。

2.2.1 原始单一架构分析

原始的单一的架构，将前端页面与后端代码放在同一个项目中，然后将其部署在一个 WEB 容器（如 tomcat）中。如图 2.1 所示：

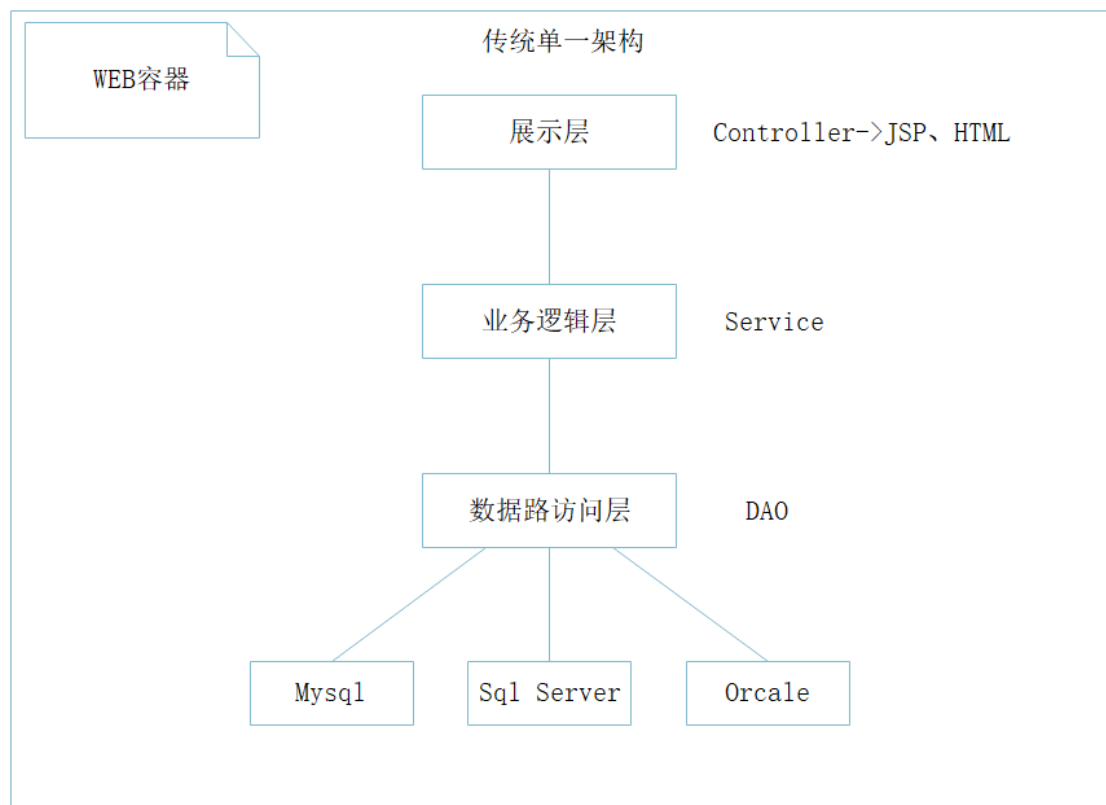


图 2.1 传统单一应用程序架构图

通过上图可以看出，展示层使用 JSP（PHP）脚本页面、HTML，使用 MVC 控制层控制器实现视图跳转。这样的架构是在用户流量较小时，为了节约开发成本，将所有应用都打包放到一个应用里面。缺点是并发承受能力较低，没有容灾措施，一旦服务器宕机，整个程序将无法运行。

2.2.2 传统分布式架构分析

传统的分布式架构，为前后端分离，使用 Json 或 XML 格式数据进行交互。同时将后端程序部署多个，如图 2.2 所示：

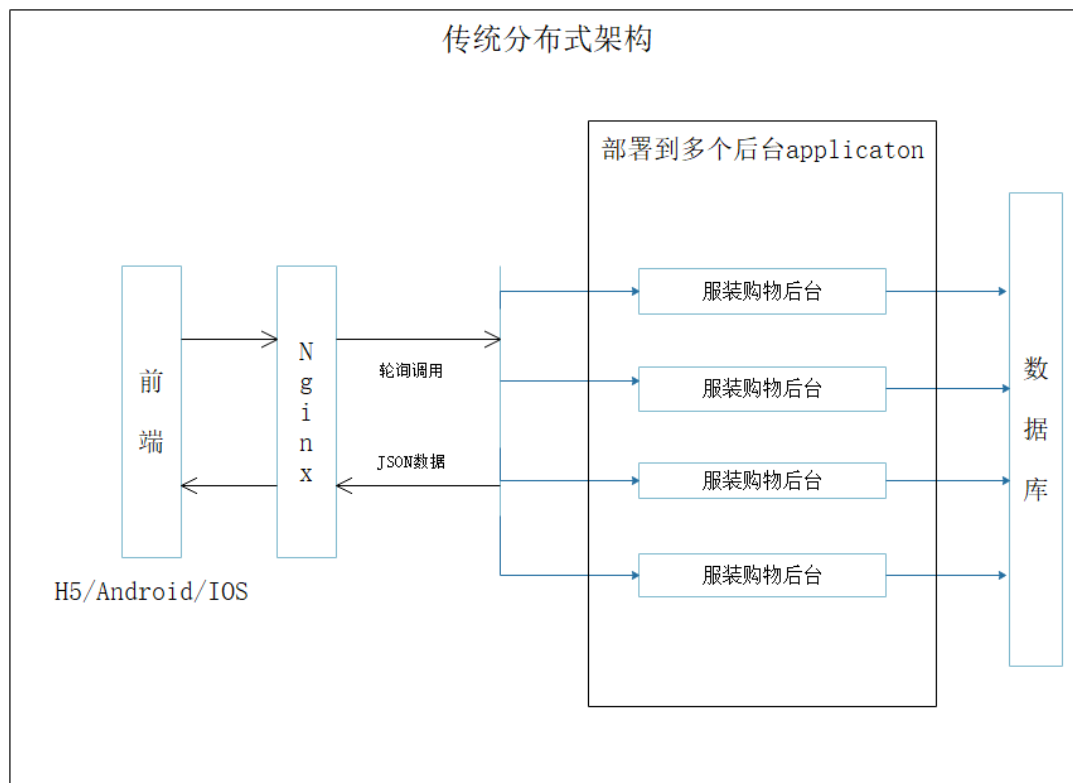


图 2.2 传统分布式架构

根据上图可以看出，传统分布式架构将后台程序部署了多个分别在不同的服务器上，而且前后端分离，如果其中一台服务器宕机，其后台程序仍然可以正常使用。相对于原始的单一应用程序架构，该方式可以提高并发处理能力，且带有一定的容灾能力，不会因为某一台服务器宕机导致整个后台处理能力崩溃，从而软件无法使用。

2.2.3 微服务架构

相对传统的分布式架构，微服务架构在应用程序级别将各个后台拆分为了若干个微服务，一个微服务架构示例，如图 2.3 所示：

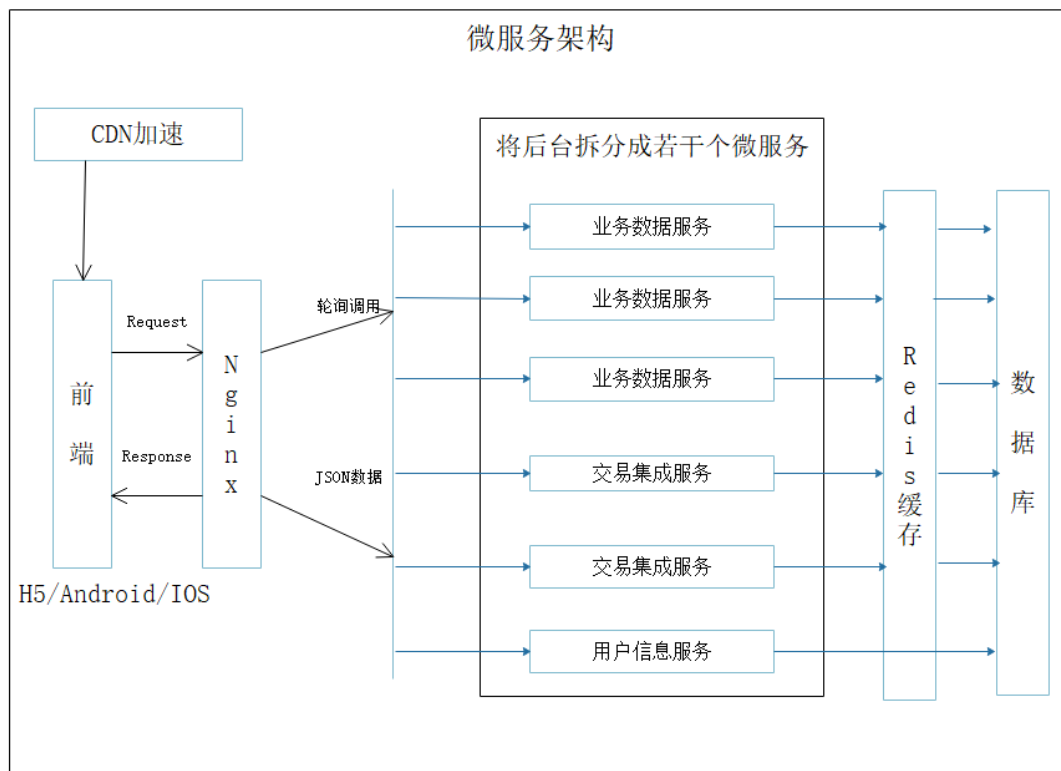


图 2.3 微服务架构示例

由上图可见，后台程序拆分成了业务数据服务、交易集成服务、用户信息服务三种不同的微服务，并且业务数据服务部署的最多，因为其与数据库交互最为频繁，其次是交易集成服务，用来统一处理前端请求与响应数据封装。用户信息服务则只针对用户登录、注册等简单的用户信息管理进行处理，相对功能单一，部署数量则较少。采用上面的微服务架构，有针对性的进行部署，增加接口耗时长或服务部署数量，从而提高服务器资源利用率，更好的服务于高并发业务场景。服务负责处理的业务示例如表 2.1 所示：

表 2.1 服务对应业务场景表

服务名称	业务数据服务	交易集成服务	用户信息服务
性能瓶颈	易产生性	易产生	不易产生
业务类型	数据库查询、更新	数据封装、接口处理	用户信息管理
业务示例	商品搜索	商品搜索结果过滤	用户登录
服务部署数量	多	较多	较少

通过这种方式，则有多个微服务共同来构成整个后台程序，其服务之间通过 Http 的方式进行 RPC 调用，实现服务之间的数据通信和信息交互，在该微服务架构下

的一个后台程序的服务拆分实例如图 2.4 所示：

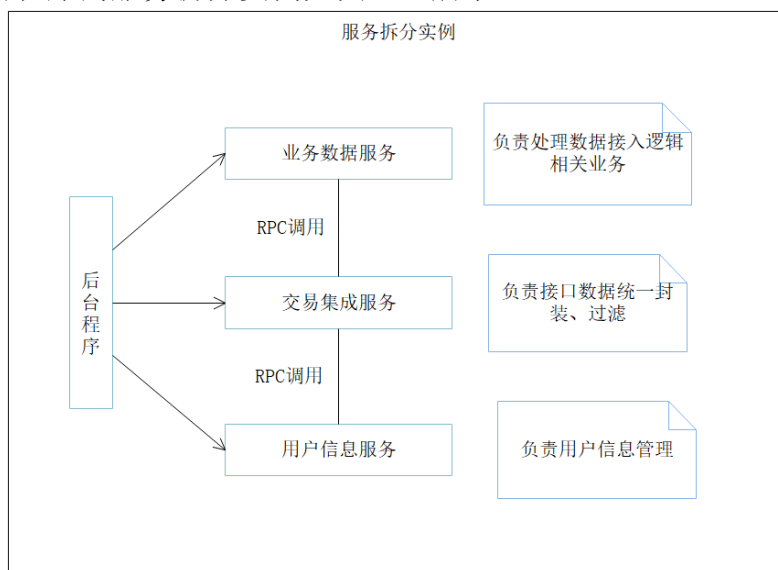


图 2.4 服务拆分实例图

2.2.4 微服务实现之 Spring Cloud 框架

Spring Cloud 是目前市场上主流的微服务实现技术栈，它是一个全家桶式的技术栈，包含了很多组件：Eureka、Ribbon、Feign、Hystrix、Zuul。

(1) Eureka 是 Spring Cloud 的注册中心，它自身也是一个服务，所有的服务运行以后，都需要在注册中心上去注册，Eureka 将维护一个注册表，包含所有注册服务的 IP 地址与端口信息，并且与各服务之间定时发送心跳包检测其活跃，一旦发现心跳检测不通，则将该服务从注册表中去除，服务之间调用之前会通过 Eureka 的注册表，去查找目标服务的 IP 地址与端口。示例如图 2.5 所示：

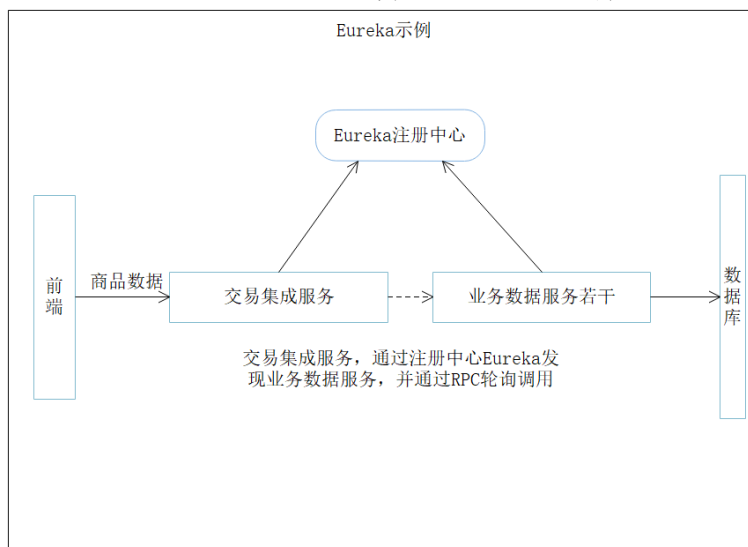


图 2.5 Eureka 示例图

交易集成服务和业务数据服务，都在 Eureka 上面注册，Eureka 保存了各服务所在的机器和端口号，并且会定时通过发送心跳检测服务是否正常运行。当服务需要互相调用时，比如交易集成服务，在接收到前端 POST 的商品发布数据，需要调用业务数据服务将商品数据存入数据库，则就需要通过 Eureka 去查找业务数据服务所在的机器 IP 地址和端口号，进行调用，再由业务数据服务执行对数据库的操作，而服务之间的 RPC 调用，则是通过 Ribbon 来进行。

(2) Ribbon 的作用是实现负载均衡，它首先会去 Eureka 去查找所有服务的注册表信息，再使用轮询算法去请求调用的目标服务。示例如图 2.6 所示：

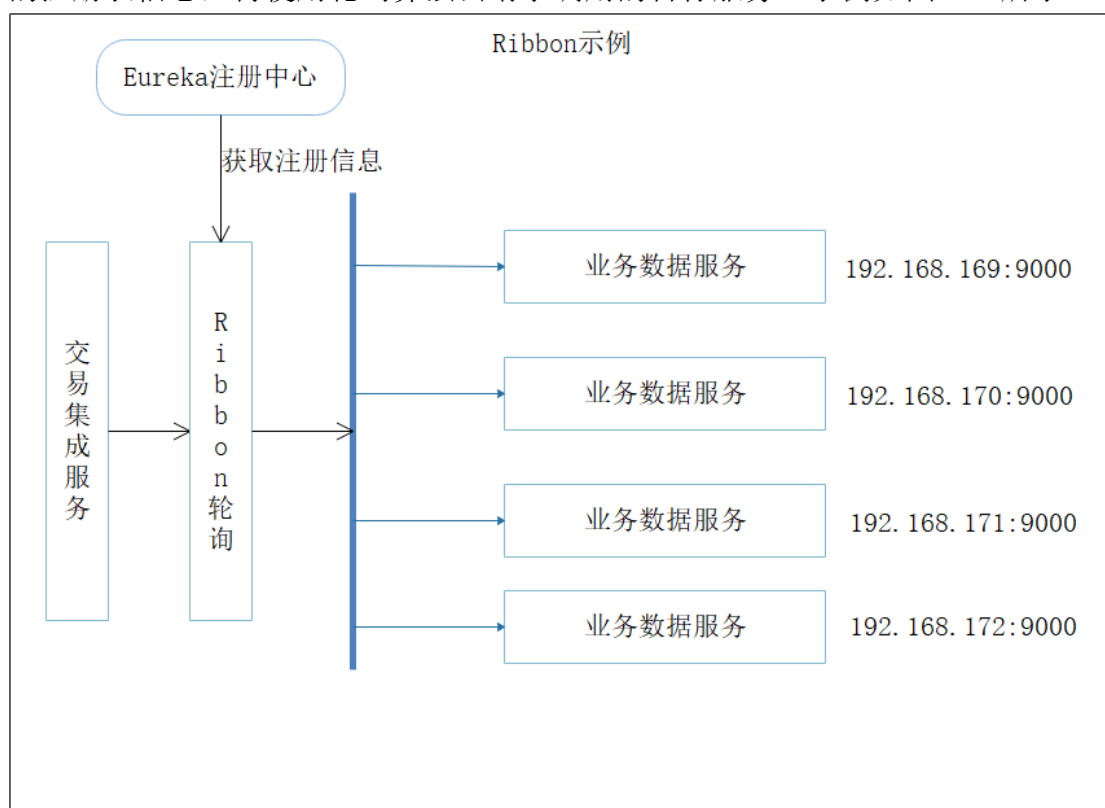


图 2.6 Ribbon 示例图

通过上图可见，有 5 台服务器都部署了业务数据服务，其 IP 地址都不同，当交易集成需要调用业务数据服务时，就会通过 Ribbon 去轮询调用，轮询的依据是来自 Eureka 提供的注册表信息。

(3) Hystrix 是一个服务熔断组件，作用是当发现服务调用超时，主动熔断调用，防止服务卡死等待，起到一定的并发容灾作用。假设在一个购物商城业务场景中，每产生一个订单就增加 5 个积分，订单服务需要调用积分服务，现在



假设积分服务自己最多只有 100 个线程可以处理请求，而此时积分服务不幸的崩溃了，每次订单服务调用积分服务的时候，都会卡住几秒钟，然后抛出一个超时异常，当多次调用将 100 个线程占完，而都卡主等待，系统响应就会十分的缓慢，这就是微服务架构中恐怖的服务雪崩问题。这么多服务互相调用，要是不做任何保护的话，某一个服务挂了，就会引起连锁反应，导致别的服务也挂。但是我们思考一下，就算积分服务挂了，订单服务也可以不用挂，结合业务来看：支付订单的时候，只要把库存扣减了，然后通知仓库发货就可以了，如果积分服务挂了，等它恢复之后，再手工恢复数据，不能因为一个积分服务挂了，就直接导致订单服务完全失去作用。Hystrix 就可以解决这个问题，Hystrix 会创建很多个小的线程池，比如订单服务请求库存服务是一个线程池，请求仓储服务是一个线程池，请求积分服务是一个线程池。每个线程池里的线程就仅仅用于请求那个服务，当积分服务挂了，前端请求订单服务，订单服务还是可以正常调用库存服务扣减库存，调用仓储服务通知发货，而对于积分服务，只要 Hystrix 检测调用超时，就直接在线程级别进行熔断，从而防止服务卡死等待，提高并发量。另外在熔断时，还可以作降级业务处理，当并发量较高时，积分服务被熔断，可以在熔断后添加积分记录，待恢复后根据记录来进行恢复。

(4) Zuul 组件，也就是微服务网关，负责服务路由。如果前端、移动端要调用后端系统，统一从 Zuul 网关进入，再由 Zuul 网关转发请求给对应的服务。

在实际使用过程中，并不是所有的组件都需要使用，Ribbon 组件与 RPC 调用通常由 Feign 来进行，Feign 是将 Ribbon 与 HTTP 网络调用结合后的 Spring Cloud 组件。在一般中小型企业中，很少使用熔断，因为熔断的维护成本很高，而是直接使用请求超时的用户提示。

2.3 其他技术栈介绍

2.3.1 开发框架 Spring Boot 分析

Spring Boot 简化了基于 Spring 的应用开发，通过少量的代码就能创建一个独立的、产品级别的 Spring 应用。Spring Boot 为 Spring 平台及第三方库



提供开箱即用的设置，多数 Spring Boot 应用只需要很少的 Spring 配置。因为其自身很好的融入了 MVC 模型、更清晰进行 Controller、Service、Dao 代码分层，采用简化的 XML 配置文件与注解，管理更为方便。使用 Spring Boot 可以实现很好的代替传统的 SSH、SSM 等企业级开发框架。

2.3.2 数据库访问组件 Mybatis、C3P0 分析

Mybatis 是一个较为灵活的数据库访问组件，其内部原理仍然是采用的 JDBC 操作，其主要功能是将 SQL 语句从代码中抽离出来，而写在 xml 配置文件中，程序员手写 SQL 的方式更加灵活，适用于电商行业。不同于传统的信息管理系统，电商行业的版本迭代很快，需求也经常改变，传统的将 SQL 语句写在代码中，每次修改 SQL 语句都需要重新编译整个代码，则维护成本较高，而将 SQL 语句放在配置文件中，只需要替换配置文件，而不用重新编译整个程序，降低了维护成本。

C3P0 是一个数据连接池，它是对 JDBC 中的数据库连接 Connection 进行复用，当连接使用完毕后，不再是调用 Close 关闭掉，而是放入连接池缓存起来，下次使用直接从连接池获取，从而提高数据库访问效率。

2.3.3 缓存组件 Redis 分析

Redis 是一个 NO-SQL 数据库，用于缓存场景，支持存入 string、hash、set、zset、list 的排序，相对传统的 Memcache 缓存数据结构更加丰富。对于一些查找频率高，更新频率低的信息，使用直接 Redis 存入内存，而不用每次去数据库查找，提高数据查询速度，从而则提高了接口的响应速度。

2.3.4 依赖包管理 Maven 说明

Maven 的作用是实现外部 jar 包的自动化管理，Maven 会建立自己的 jar 包仓库，在程序中，只需要集成 Maven 的配置文件，配置 jar 包的 groupId 与 ArtifactId 的依赖即可。如数据库连接时，不用再去人为的将依赖 jar 包拷贝到项目中，使得程序结构管理更加方便。

2.3.5 秒嘀科技短信验证平台

秒嘀科技是一家提供短信验证码、语音验证码等验证手段的三方平台，其接



入方便，使用 POST 请求，并且带有 64 位 Hash 加密签名，安全可靠。由于本次设计开发的服装网站，注册账号采用了手机号。为了避免用户填写虚假的手机号，使用他人的手机号，提高注册用户的身份真实性，同时为了兼顾成本，采用了爬虫与 HttpURLConnection 接入了第三方的短信验证。

2.3.6 高德地图 IP 定位

高德地图 IP 定位，网站不同于手机 APP，在 PC 端不能调用 GPS 进行定位，也无法调用手机号运营商基站的网络三角定位，故只能使用 IP 地址来进行定位。高德地图的 IP 精确地位，是基于其长年的数据服务累计的 IP 地址数据与地理位置数据，本次设计为了实现 PC 端定位功能，使用其 IP 定位。

2.3.7 技术 Zxing、Gson、HttpURLConnection 介绍

Zxing 是一个二维码生成工具、Gson 是一个 Json 数据格式化工具、HttpURLConnection 是一个 Java 语言原生的网络请求类，用于 Java 实现网络爬虫。

3 软件可行性研究

(1) 从技术可行性的角度，我在大学课程中学习了 Java、JavaWeb、前端相关技术，对该设计使用到的技术都有一定的了解与使用经验；另外曾在实验室学习过 Java 后端技术，自学了 Spring Boot 开发框架、Mybatis 数据库访问组件、C3P0 数据库连接池、Redis 缓存、Maven 依赖包管理、Gson 格式化组件、HttpURLConnection 网络请求与爬虫、Html5、CSS3、JavaScrip 等，并且在企业实习的过程中，对本设计的核心技术 Spring Cloud 以及上述技术都进行过系统的再次学习，运用这些技术参与过企业级项目开发，有过较完备的实战开发经验。对于前端 LayerUI 前端框架、秒嘀科技短信验证平台、IP 定位高德地图 API 平台，其官网上都有较为完备的开发文档供参考。综上，技术上是完全可行的。

(2) 从法律可行性角度，服装购物网站适应时代潮流，是电商的企业、服装店的刚需，其网站内容健康流行、网站使用技术均为开源技术与自行封装工具，



不会涉及相应的版权纠纷，且整个网站的研发、部署、运营都不会违反相关法律法规，故在法律上是完全可行的。

(3) 从操作可行性角度，该网站采用 LayerUI 前端框架进行开发，所实现的网站界面美观，并且带有对用户的友情界面提示，操作简单可行。

(4) 从经济可行性角度，该网站开发所涉及的技术资料均来自互联网和相关书籍，使用的短信验证、地图定位均为免费接口与开源接口，并没有过多的经济开销，另外在开发的过程中，也不需要额外的付费软件、硬件支撑，使用自己的笔记本电脑，无需任何其他费用，故经济上是完全可行的。

4 需求分析

4.1 需求规范

4.1.1 产品背景

根据《2018-2022 年中国服装零售市场深度评估及未来发展趋势报告》2018 年，我国服装网购市场交易规模达到 7232 亿元，年复合增长率将达到 14% 左右。2015 年以来，服装行业线上销售额迅猛增长，线上成交额已占到总体行业销售额一半以上，渗透率在 2017 年以来左右突破 50%，时间点晚于食品、家电数码及美妆行业，但早于鞋包行业。服装行业目前的线上渗透率在所有行业里处领先地位。

随着计算机技术与电子商务的快速发展，带有推荐功能的导向性的购物网站已成为时代主流，一方面可以根据地理位置、浏览行为、购买行为等多个维度进行商品推荐，以更加个性化、差异化的服务用户。另一方面，随着个人用户量的增多，如何调整网站的技术架构、提升高并发处理能力也成为技术难点。而二者也是困扰许多中小型电商企业的问题。为了更好的体现其产品特色，满足用户差异化，定制化，个人化购物的需求，购物导向性服装网站已经成为中小型电子商务企业、知名服装店的首要之选。

4.1.2 产品概述

购物导向性服装店网站设计，适用企业为：中小型电子商务企业、知名服装店。市场定位是一个 B2C、C2C 定位的网站，网站上既可以入驻普通商家、也支持实体店入驻，实体店自动定位。除了基本的用户信息管理以外，用户可以申请开店、发布服装、购买服装、使用购物车等；系统会记录用户上次登录的地理位置信息以及上次购买行为，并且主动为用户推荐同类型、当下季节、浏览偏好的服装，同时可以查看附近的服装实体店，进行公交线路导航。本网站采用微服务分布式架构，是目前市场上主流的电商网站架构之一，相对于普通网站，该网站的架构与一些主流技术栈的使用也为网站的需求变更、版本迭代、运维部署等提供了灵活的处理方式与入口。

4.2 功能需求

该设计包含用户端和管理员端。其用户端功能模块如图 4.1 所示：

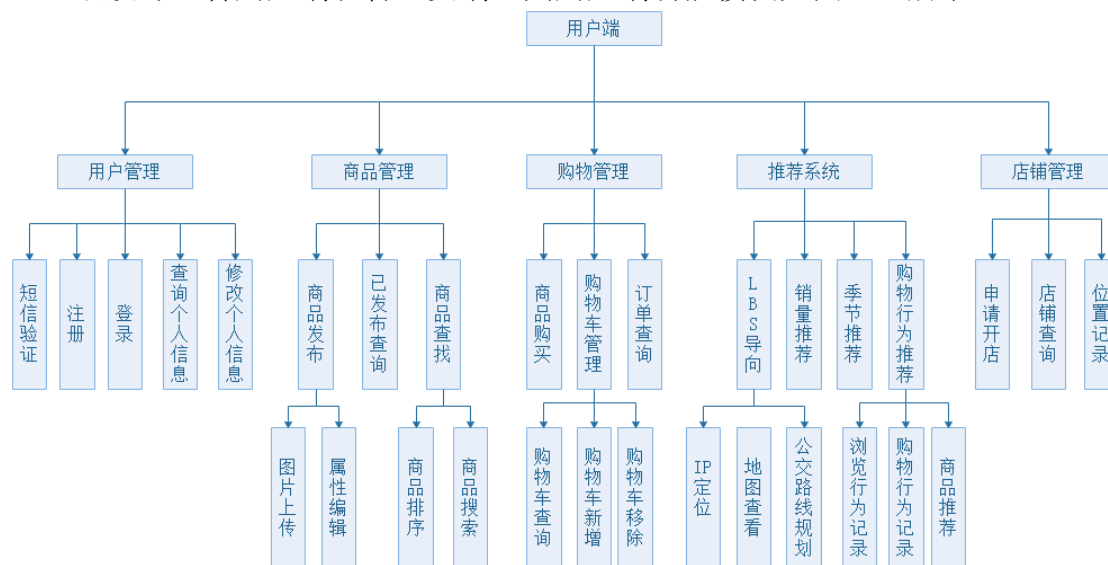


图 4.1 用户端功能模块图

如上图所示，用户端主要分为 4 个模块，分别为：用户管理、商品管理、购物管理、推荐系统、店铺管理。对功能模块进行展开，用户管理可以分为：短信验证、注册、登录、查询个人信息、修改个人信息 5 个子模块；商品管理可以分为：商品发布、已发布查询、商品查找 3 个子模块；购物管理可以分为：商品购买、购物车管理、订单查询 3 个子模块；推荐系统可以分为 LBS 导向、销量推荐、季节

推荐、购物行为推荐 4 个子模块；店铺管理可以分为：申请开店、店铺查询、位置记录 3 个子模块。

管理员端的功能主要为店铺管理，其功能模块结构如图 4.2 所示：

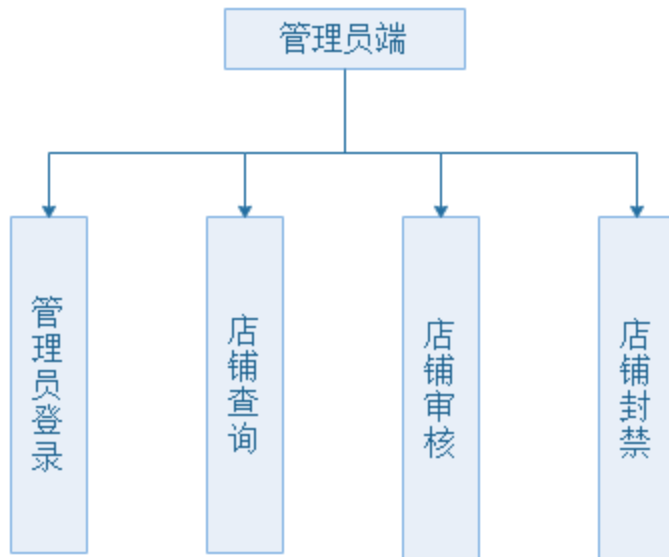


图 4.2 管理员端功能模块结构图

如上图所示，管理员分为 4 个功能模块，分别为：登录、店铺查询、店铺审核、店铺封禁。

4.3 性能需求

一方面，要求界面清新美观，符合响应式布局理念，同时具备电商购物网站的商业风格与流行服饰的独特的时尚感。另一方面，要求后台接口响应时间均为毫秒级，除文件上传以外，前端页面的点击要求带给用户响应速度快、操作流畅的使用体验感，架构要求采用 Spring Cloud 微服务分布式架构，将后台程序拆分为：业务数据服务、交易集成服务。

5 概要设计

5.1 用例分析

结合需求分析，分析用户与管理员关于用户管理、商品管理、推荐系统、店铺管理 4 个功能模块。分析这些模块所包含的用例角色，找出其中每个角色的功

能点，画出用例图如 5.1 所示：

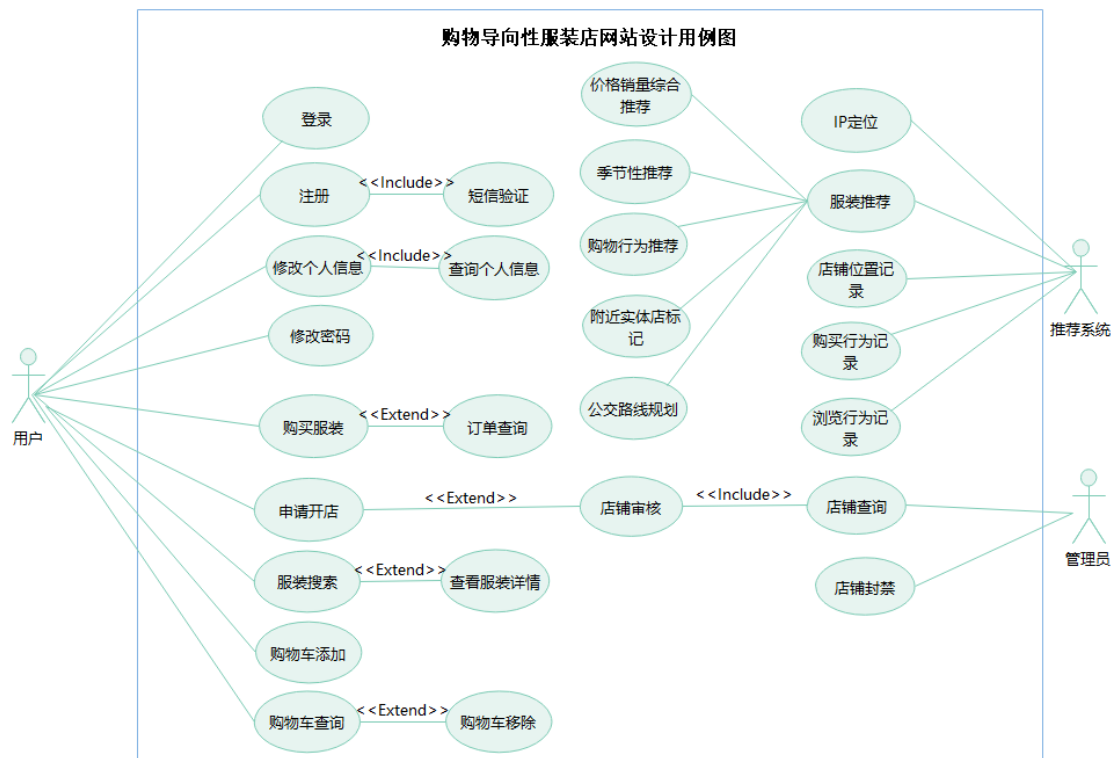


图 5.1 购物导向性服装店网站设计用例图

如上图，根据系统边界将用例中分为三个角色：用户、管理员、推荐系统。其中用户用例的为：短信验证、登录、注册、查询个人信息、修改个人信息、修改密码、购买服装、订单查询、申请开店、服装搜索、查看服装详情、购物车添加、购物车查询、购物车移除；推荐系统用例为：IP 定位、服装价格销量综合推荐、服装季节性推荐、购物行为推荐、附近实体店标记、店铺位置记录、购物行为记录、浏览行为记录；管理员用例为：店铺审核、店铺查询、店铺封禁。

5.2 概要逻辑设计

结合需求分析与用例设计，参考实体关系模型，将用例中设计的对象进行抽象成实体，分析实体之间的关系，同时找出每个实体包含的属性。共有：用户、商品、购物车、发布记录、购物行为记录、订单、店铺、属性共计 8 个实体。共有用户与商品之间的发布商品、用户与商品之间的加入购物车、用户与店铺之间的入驻、用户与商品之间的浏览、用户与商品之间购买、商品与商品属性之间的包含共计 4 个关系。采用实体关系模型，将实体与关系相联系起来，同时确定实

体之间的一与多对应关系，可以画图 E-R 图，如图 5.2 所示：

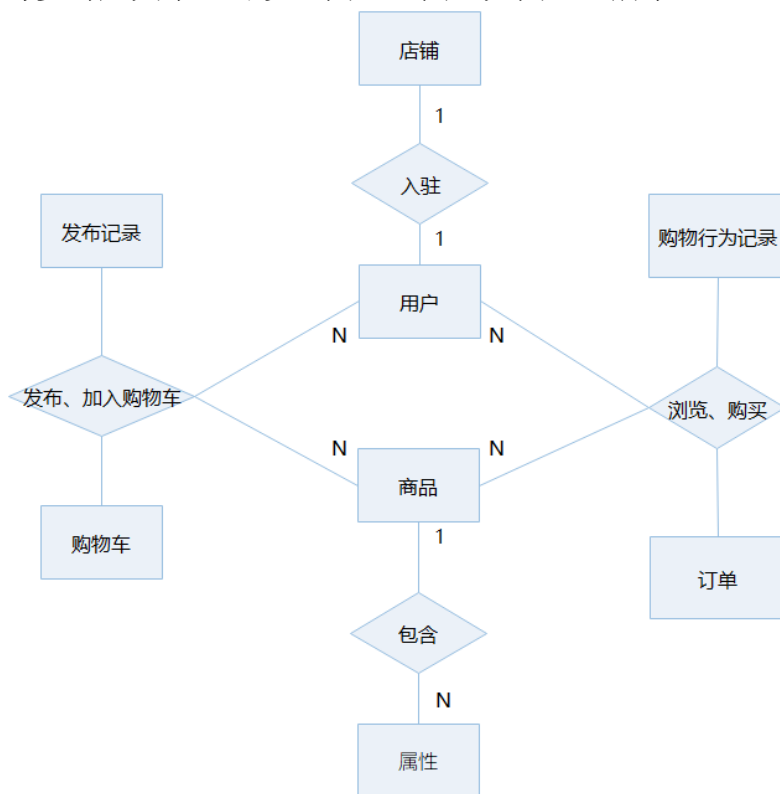


图 5.2 E-R 图

将 E-R 进行展开，其中用户、商品、管理员 E-R 如图 5.3 所示：

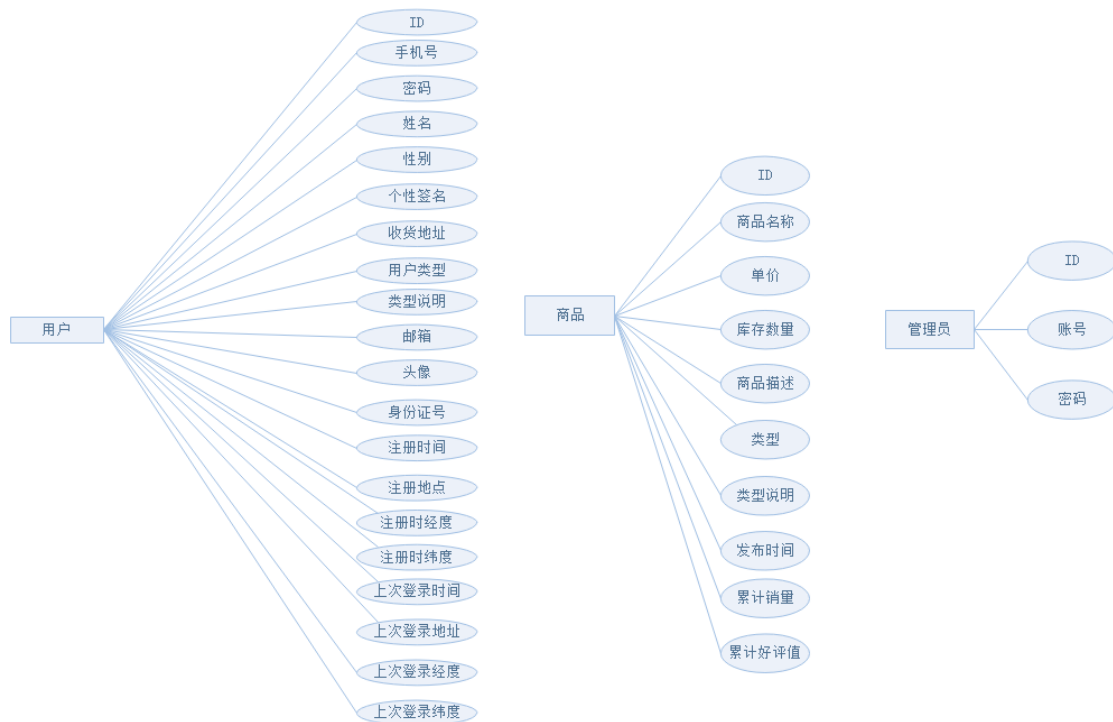


图 5.3 用户、商品、管理员相关 E-R 图

店铺、商品图片、商品属性相关 E-R 图如图 5.4 所示：

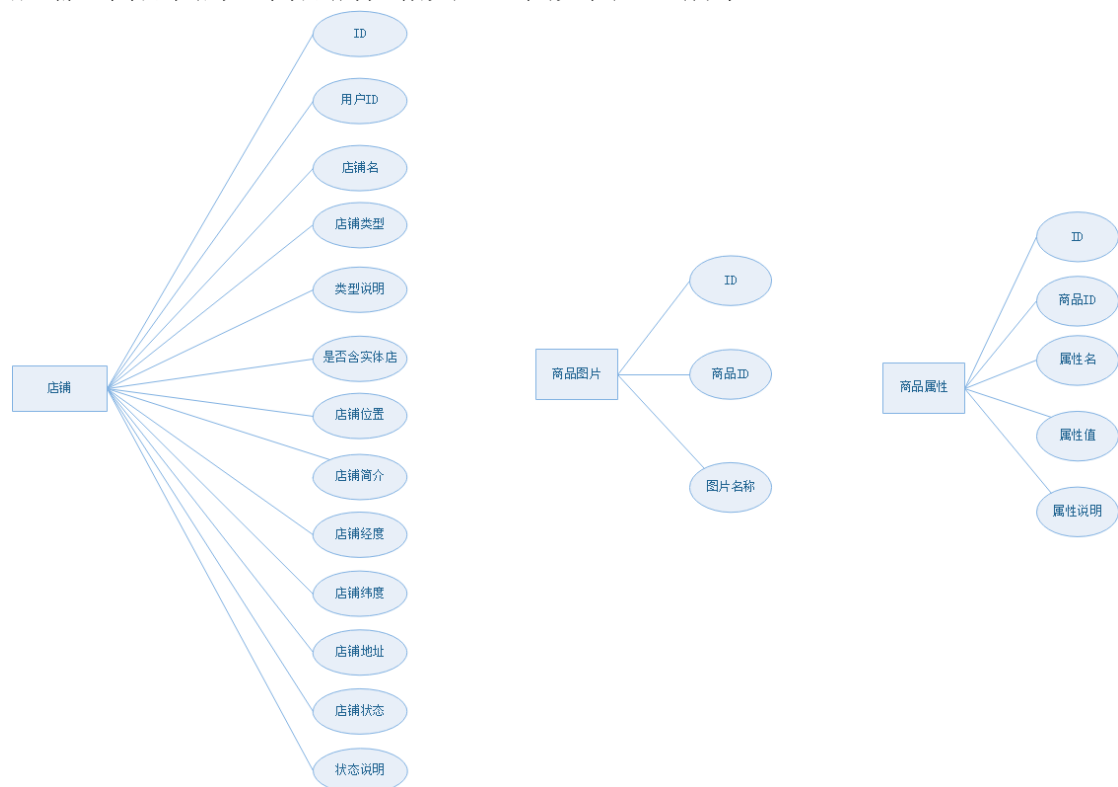


图 5.4 店铺、商品图片、商品属性相关 E-R 图

发布记录、购物行为记录、购物车相关 E-R 图如图 5.5 所示：

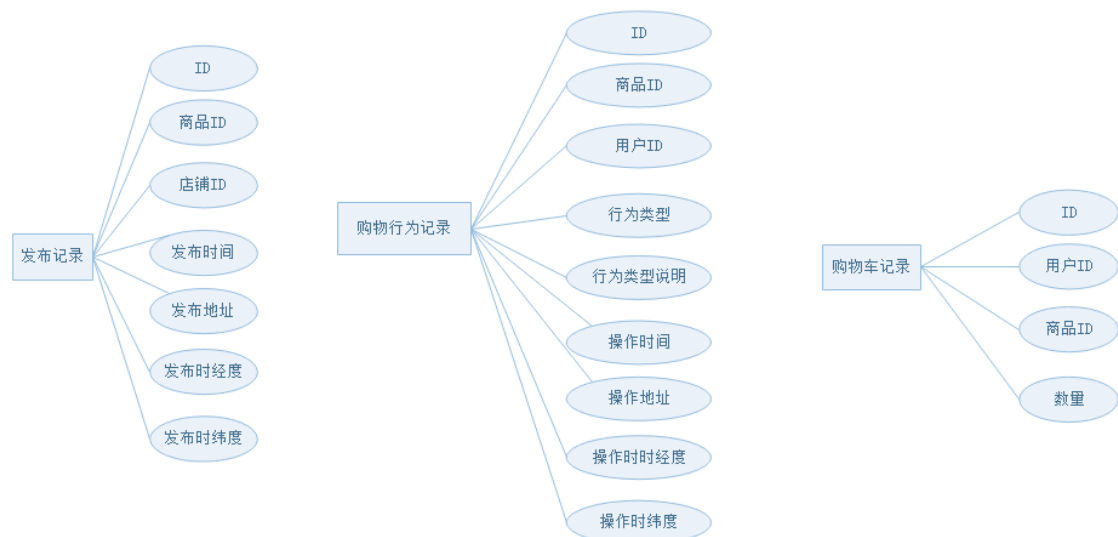


图 5.5 发布记录、购物行为记录、购物车相关 E-R 图

订单、下单记录相关 E-R 图如图 5.6 所示:

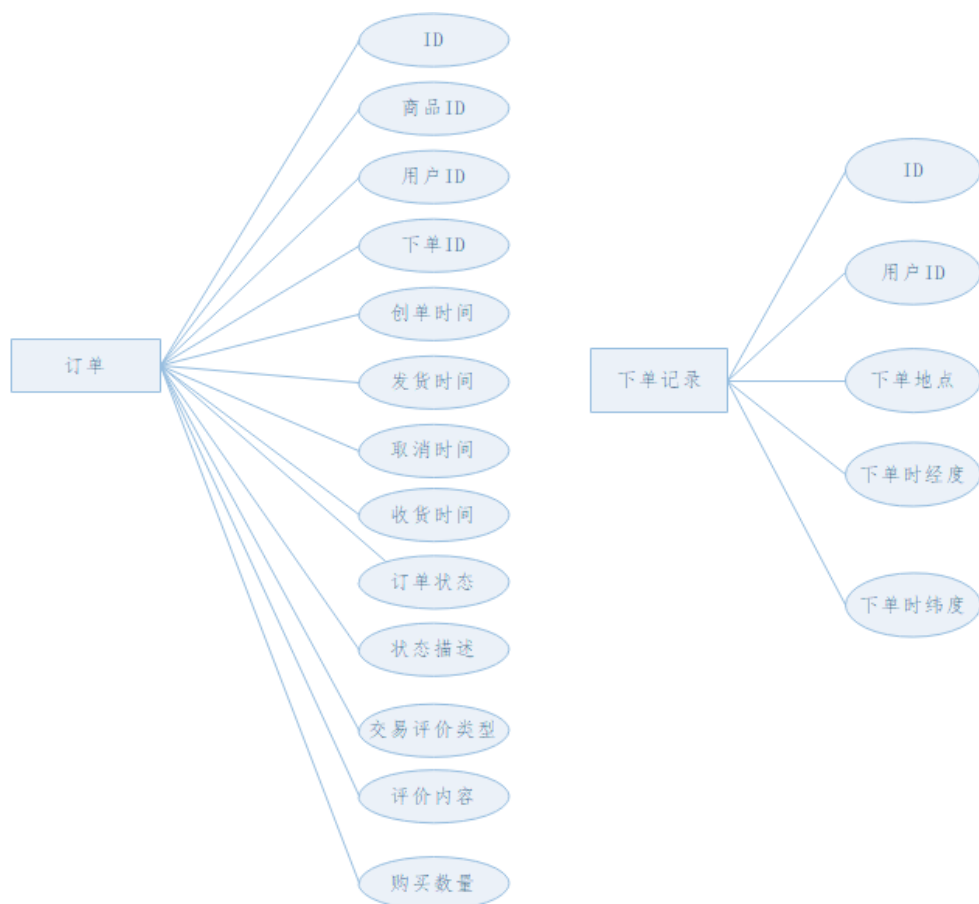


图 5.6 订单、下单记录相关 E-R 图

5.3 业务流程分析

本购物网站的一次的购物流程：用户首先进入系统，然后浏览商品，浏览时可以进行商品分类查找、按名称进行商品搜索、查看商品详情。当发现自己想要购买的商品后，可以选择直接购买与加入购物车，在进行相应操作时，系统会检测用户是否登陆，若未登陆则会跳转到登录页面。当用户身份确定后，会进行对应商品的库存检测，为了避免并发造成的库存量脏读，在真正购买和加入购物车之前会从数据库查询以事物级别的去查询库存量，然后后台程序判断库存量是否满足当前用户的购买数量，若库存充足，会执行写订单、减库存等一系列购买操作，最后将订单信息保存到数据库。若库存不足时，则会直接提示库存量不足，结束购买操作。由于查询商品库存量与减库存、写订单操作是放在同一个事物中的，利用数据库 Mysql 的行级锁性质，可以很好的规避并发时的脏读问题，从而

保证了购买时的一致性。业务的流程图如图 5.7 所示：

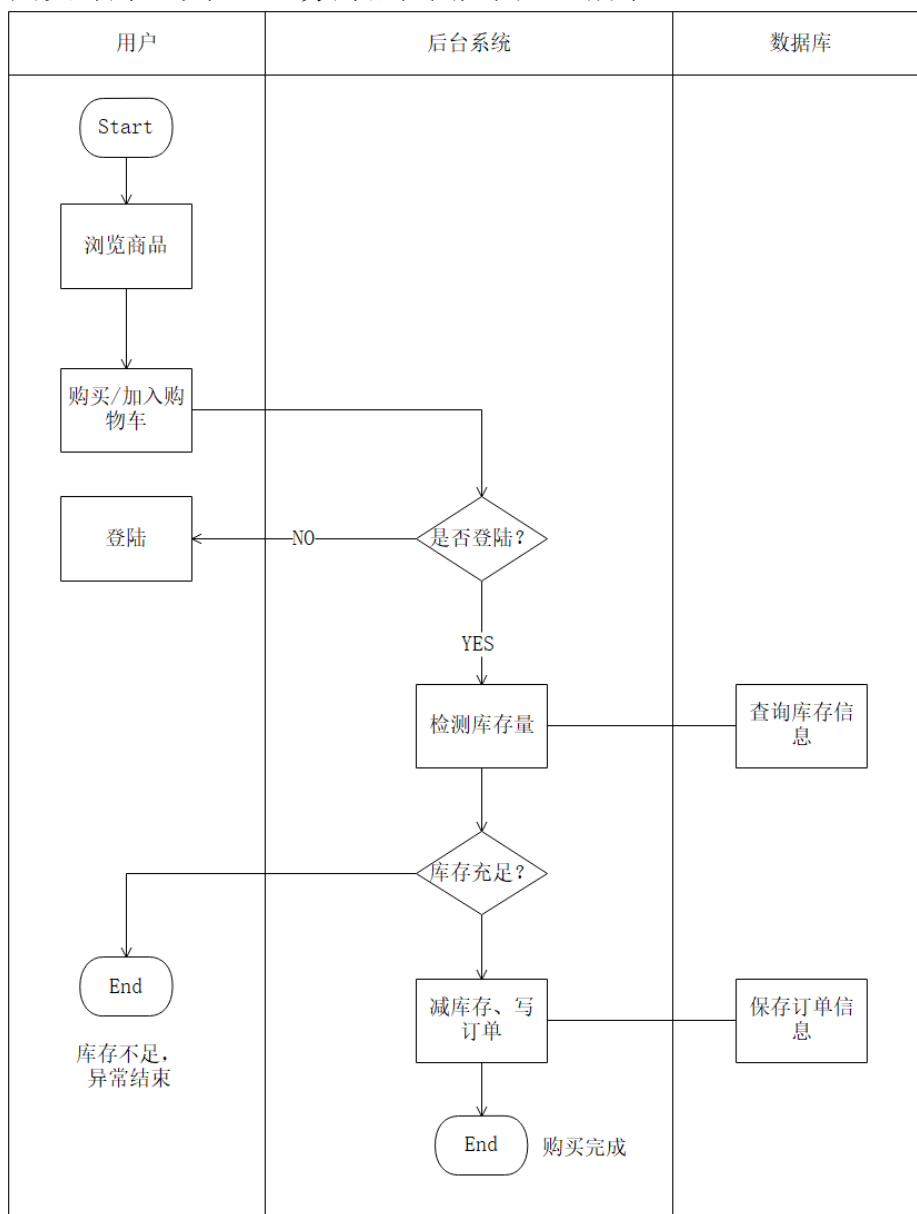


图 5.7 购物流程图

如上图可见，一次成功的购买流程为：用户登陆系统，浏览商品，点击购买，系统检测库存充足、执行减库存、写订单操作，购买完成。

5.4 软件架构设计

本次设计使用了 Spring Cloud 框架，用到了其注册中心组件 Eureka、负载均衡组件 Ribbon 与 RPC 的结合组件 Feign，前端使用 Html 页面，利用 Ajax 实现前后端完全分离，界面利用 LayerUI 与 Bootstrap 进行编写，不适用任何需要额

外服务器解析的脚本语言，前后端使用 Json 传递数据。整个网站后台分为三个部分：注册中心服务，交易集成服务，业务数据服务。后二者属于业务处理级别的服务，不属于 Spring Cloud 的组件，其中业务数据服务负责处理所有数据库相关逻辑，交易集成服务负责统一处理请求数据的解析与返回数据封装，注册中心服务负责维护交易集成服务与业务数据服务的注册表信息，定时检测心跳，为二者之间调用提供 IP 与端口信息。整体架构如图 5.8 所示：

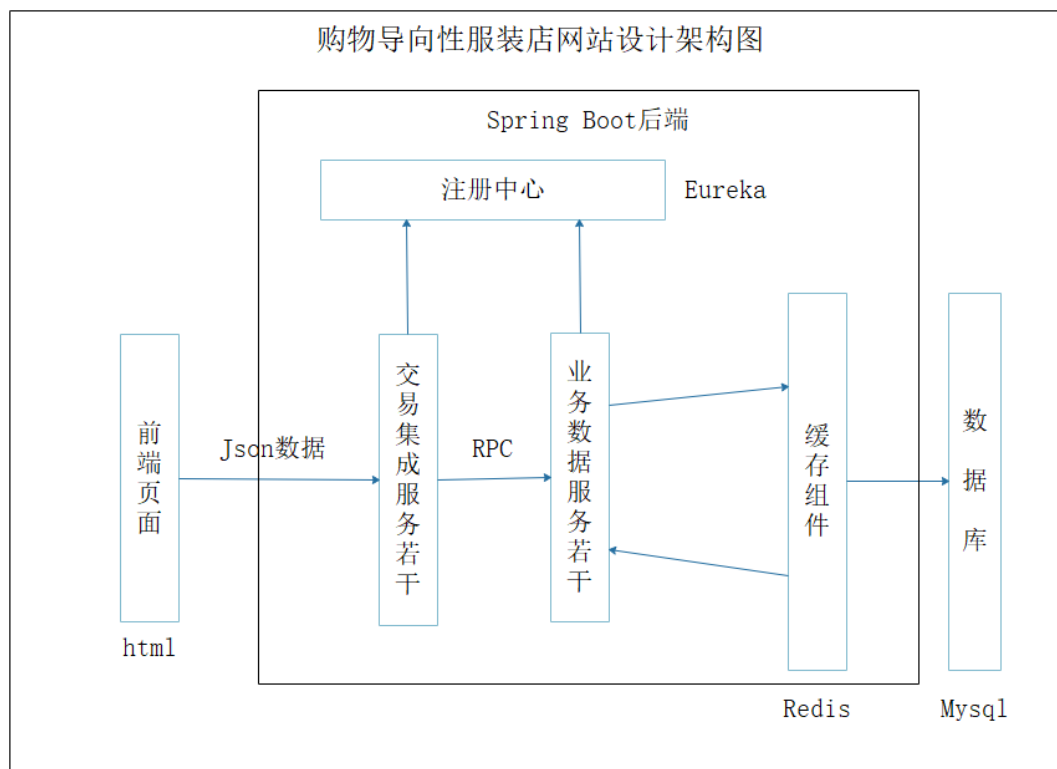


图 5.8 系统架构图

如图上图所示，由 Eureka 作为注册中心，统一调度交易集成服务和业务数据服务若干个，将高查询、低更新频率的数据放入 Redis 缓存，前后端完全分离，使用 Json 数据进行交互。

6 软件详细设计

6.1 数据库设计



6.1.1 数据表设计

通过对 E-R 图所表示的关系模式进行分析，确定以下数据表如下：

表 6.1 用户表

字段名称	字段类型	长度	允许为空	备注
id	bigint	20	N	用户 ID
phone	char	12	N	手机号
password	varchar	40	N	密码
name	char	20	N	姓名
sex	char	2	Y	性别
signature	varchar	100	Y	个性签名
receive_address	varchar	120	Y	收货地址
type	tinyint	4	N	用户类型
type_explain	char	20	N	类型说明
email	char	20	Y	邮箱
head_image	char	36	Y	头像
id_card	char	19	Y	身份证号
register_time	datetime		N	注册时间
register_location	varchar	120	Y	注册地址
register_longitude	decimal	12,6	Y	注册纬度
register_latitude	decimal	12,6	Y	注册经度
last_login_time	datetime		Y	登录时间
last_login_location	varchar	120	Y	登录地址
last_login_longitude	decimal	12,6	Y	登录纬度
last_login_latitude	decimal	12,6	Y	登录经度
create_time	timestamp		N	创建时间
update_time	timestamp		N	修改时间
deleted	tinyint	1	N	是否删除

用户表如表 6.1 所示，其中 id 为用户表的主键。



表 6.2 店铺表

字段名称	字段类型	长度	允许为空	备注
id	bigint	20	N	店铺 ID
user_id	bigint	20	N	用户 ID
name	char	30	N	店铺名
type	tinyint	4	N	店铺类型
type_explain	char	20	N	类型说明
entity_store	tinyint	1	N	实体店
address	varchar	100	Y	所在城市
introduce	varchar	120	Y	店铺简介
longitude	decimal	12,6	Y	所在纬度
latitude	decimal	12,6	Y	所在经度
location	varchar	120	N	所在地址
status	tinyint	4	N	店铺状态
status_explain	char	15	N	状态说明
create_time	timestamp		N	创建时间
update_time	timestamp		N	修改时间
deleted	tinyint	1	N	是否删除

店铺表如表 6.2 所示，其中 id 为店铺表的主键。

表 6.3 管理员表

字段名称	字段类型	长度	允许为空	备注
id	bigint	20	N	ID
account	char	20	N	账号
password	varchar	40	N	密码
create_time	timestamp		N	创建时间
update_time	timestamp		N	修改时间
deleted	tinyint	1	N	是否删除



管理员表如表 6.3 所示，其中 id 为管理员表的主键。

表 6.4 商品表

字段名称	字段类型	长度	允许为空	备注
id	bigint	20	N	商品 ID
name	char	50	N	商品名称
price	decimal	18,2	N	单价
amount	int	11	N	库存数量
introduce	varchar	120	Y	商品描述
type	int	11	N	商品类型
type_explain	char	20	N	类型说明
introduce	varchar	120	N	店铺简介
add_time	datetime		N	发布时间
sale_volume	int	11	N	累计销量
comment_sum	int	11	N	好评价
create_time	timestamp		N	创建时间
update_time	timestamp		N	修改时间
deleted	tinyint	1	N	是否删除

商品表如表 6.4 所示，其中 id 为主键。

表 6.5 商品图片表

字段名称	字段类型	长度	允许为空	备注
id	bigint	20	N	图片 ID
goods_id	bigint	20	N	商品 ID
image_name	char	36	N	图片名称
create_time	timestamp		N	创建时间
update_time	timestamp		N	修改时间
deleted	tinyint	1	N	是否删除

商品图片表记录了商品与图片的关系如表 6.5 所示，其中 id 为图片表主键。



表 6.6 发布记录表

字段名称	字段类型	长度	允许为空	备注
id	bigint	20	N	记录 ID
goods_id	bigint	20	N	商品 ID
store_id	bigint	20	N	店铺 ID
release_time	datetime		N	发布时间
release_location	varchar	120	N	发布地址
release_longitude	decimal	12,6	N	发布纬度
release_latitude	decimal	12,6	N	发布经度
create_time	timestamp		N	创建时间
update_time	timestamp		N	修改时间
deleted	tinyint	1	N	是否删除

发布记录表记录了用户与商品发布信息，如表 6.6 所示，其中 id 为主键

表 6.7 商品属性表

字段名称	字段类型	长度	允许为空	备注
id	bigint	20	N	属性 ID
goods_id	bigint	20	N	商品 ID
name	char	15	N	属性名
value	char	36	N	属性值
property_explain	varchar	100	Y	属性说明
create_time	timestamp		N	创建时间
update_time	timestamp		N	修改时间
deleted	tinyint	1	N	是否删除

商品属性表记录了商品与其属性，如表 6.7 所示，其中 id 为主键。



表 6.8 订单记录表

字段名称	字段类型	长度	允许为空	备注
id	bigint	20	N	记录 ID
user_id	bigint	20	N	用户 ID
location	varchar	120	N	下单地点
longitude	decimal	12, 6	Y	下单纬度
latitude	decimal	12, 6	Y	下单经度
sum_money	decimal	18, 2	N	总金额
create_time	timestamp		N	创建时间
update_time	timestamp		N	修改时间
deleted	tinyint	1	N	是否删除

订单记录表记录了购买记录，如表 6.8 所示，其中 id 为主键。

表 6.9 购物行为表

字段名称	字段类型	长度	允许为空	备注
id	bigint	20	N	行为 ID
user_id	bigint	20	N	用户 ID
goods_id	bigint	20	N	商品 ID
Type	tinyint	4	N	行为类型
type_explain	char	20	N	行为说明
operation_time	datetime		N	操作时间
operation_location	varchar	120	N	操作地点
operation_longitude	decimal	12, 6	N	操作纬度
operation_latitude	decimal	12, 6	N	操作经度
create_time	timestamp		N	创建时间
update_time	timestamp		N	修改时间
Deleted	tinyint	1	N	是否删除

购物行为表记录了用户的浏览、购买商品行为，如表 6.9 所示，其中 id 为主键。



表 6.10 订单明细表

字段名称	字段类型	长度	允许为空	备注
Id	bigint	20	N	订单 ID
goods_id	bigint	20	N	商品 ID
user_id	bigint	20	N	用户 ID
orders_record_id	bigint	20	N	购买 ID
add_time	datetime		N	创单时间
deliver_time	datetime		Y	发货时间
cancel_time	datetime		Y	取消时间
receive_time	datetime		Y	收货时间
Status	tinyint	4	N	订单状态
status_explain	char	15	N	状态描述
comment_type	tinyint	4	N	评价类型
comment_content	varchar	120	Y	评价内容
Amount	int	11	Y	购买数量
sum_money	decimal	18,2	N	订单金额
create_time	timestamp		N	创建时间
update_time	timestamp		N	修改时间
Deleted	tinyint	1	N	是否删除

订单明细表记录了用户购买商品时产生的订单明细，多条订单明细记录共同构成一次购买记录。订单明细表如表 6.10 所示，其中 id 为其主键。



表 6.11 购物车表

字段名称	字段类型	长度	允许为空	备注
Id	bigint	20	N	购物车 ID
user_id	bigint	20	N	用户 ID
goods_id	bigint	20	N	商品 ID
Amount	int	11	N	购物数量
create_time	timestamp		N	创建时间
update_time	timestamp		N	修改时间
Deleted	tinyint	1	N	是否删除

购物车表记录了用户添加商品到购物车的记录，如表 6.11 所示，其中 id 为主键。

6.1.2 数据库物理设计

根据数据表的设计，在 mysql 中生成数据库，如图 6.1 所示：

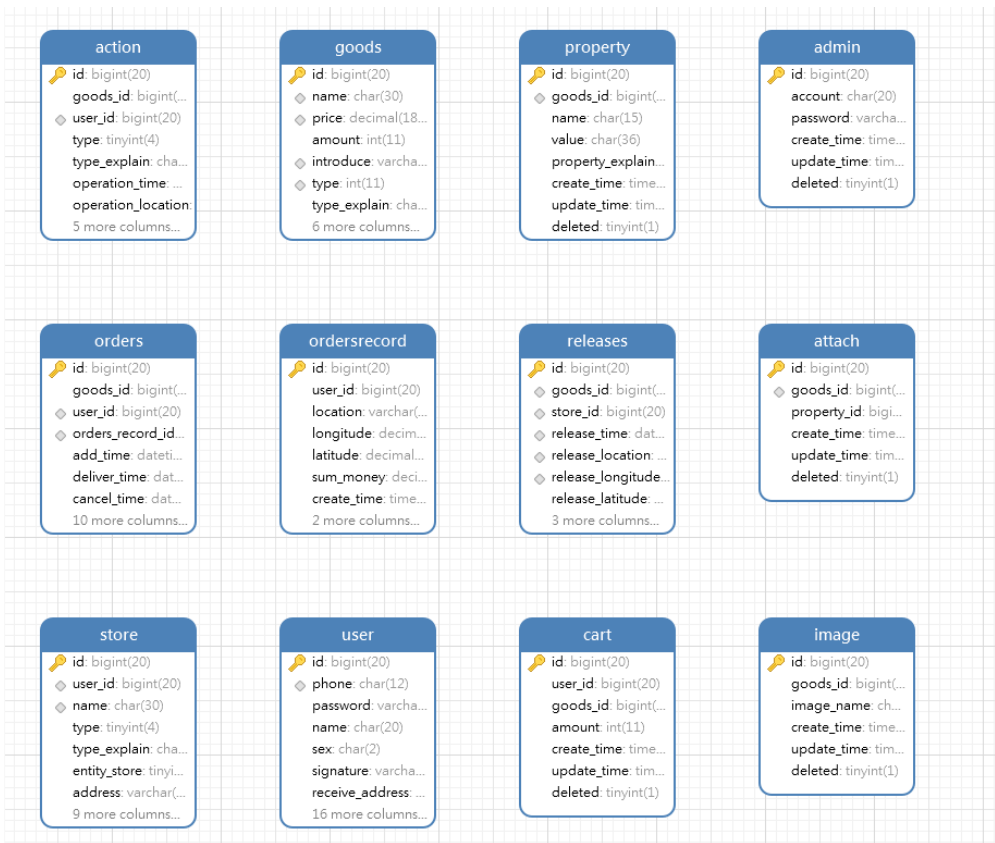


图 6.1 数据库详细设计



6.2 命名规范

6.2.1 数据库命名规范

- (1) 数据库名：英文单词全拼，首字母大写，驼峰命名。
- (2) 数据表名：英文单词全拼，所有字母均小写。
- (3) 字段名称：英文单词缩写，所有字母均小写，多词语下划线分隔。

6.2.2 项目文件命名规范

对项目的文件进行命名规范，方便对项目文件进行管理，有利于查阅，节省时间并且利用修改维护。

(1) 在后端设计中，包名均为 `com.clothing.xxx`，严格按照 `dao`、`service`、`controller`、`util`、`domain`、`enums`、`consts` 等子文件夹存储对应的类，且类文件名首字母大写、驼峰命名。资源文件放在 `resources` 下，使用 `yaml` 取代 `properties` 属性文件，均按照符合企业级专业项目命名规范。

(2) 在前端设计中，`html`、`css`、`js` 严格分离，存放于自己的文件夹中，方便资源的统一管理与维护。

(3) 项目名：英文单词全拼，所有单词均首字母大写，驼峰命名，如：`ClothingDataBussinessService` 作为业务数据服务命名。

6.3 模块详细设计

6.3.1 用户管理模块

(1) 模块描述：用户登录、注册、个人信息管理的相关模块。

(2) 后台设计：对前端暴露获取验证码、登录、注册、查询个人信息、修改密码、修改个人信息的接口，以 `Json` 格式数据与前端交互。接口命名及请求方式定义在用户控制层 `UserController`，具体接口地址与接口设计，如图 6.2 所示：

user-controller : User Controller			Show/Hide	List Operations	Expand Operations
POST	/api/user/addStore	申请开店			
POST	/api/user/getVerifyCode	获取验证码			
POST	/api/user/login	登录			
POST	/api/user/queryOneAction	查询一次购物行为			
POST	/api/user/queryStore	查询个人店铺信息			
POST	/api/user/queryStoreById	根据ID查询店铺信息			
POST	/api/user/queryUser	查询个人信息			
POST	/api/user/register	注册			
POST	/api/user/updatePassword	修改密码			
POST	/api/user/updateUser	修改个人信息			

图 6.2 用户控制层接口说明

(3) 界面设计

登录界面如图 6.3 所示：

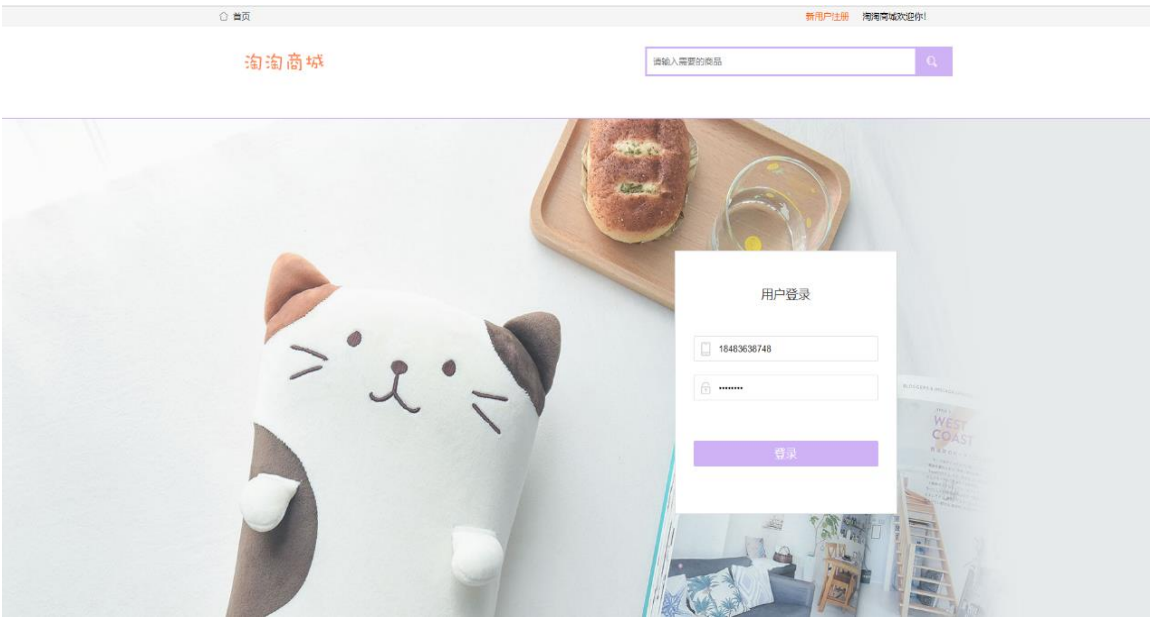


图 6.3 登录界面

登录时前端会验证用户的手机号格式是否正确，为 1 开头的 11 位数字，密码格式是否正确为字母与数字的组合，长度为 6 至 12 位。注册界面如图 6.4 所示：

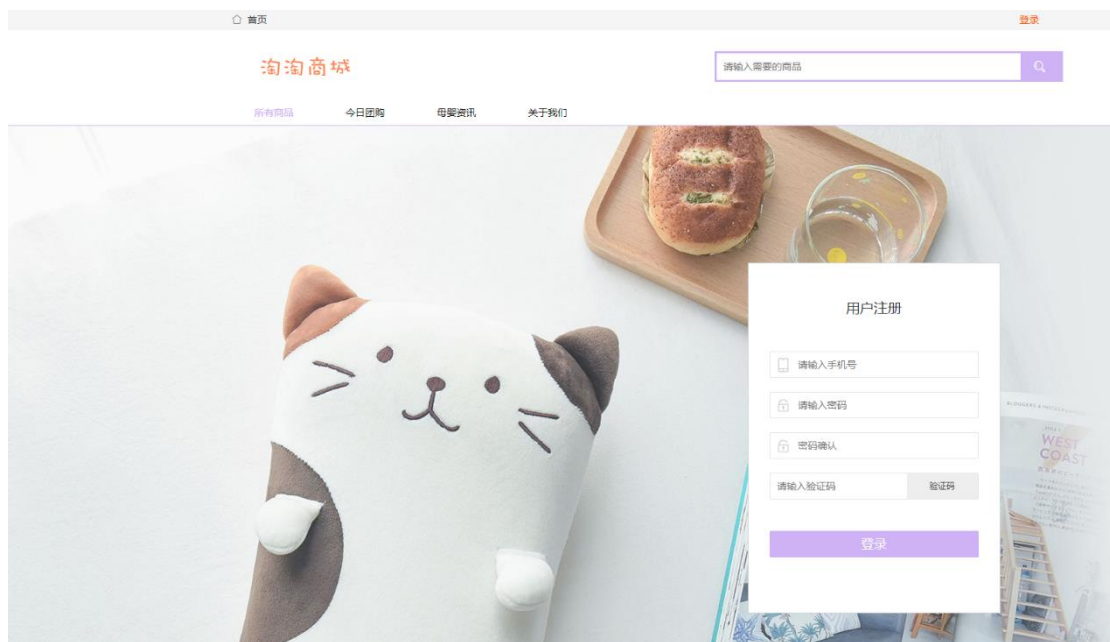


图 6.4 注册界面

个人信息界面如图 6.5 所示：

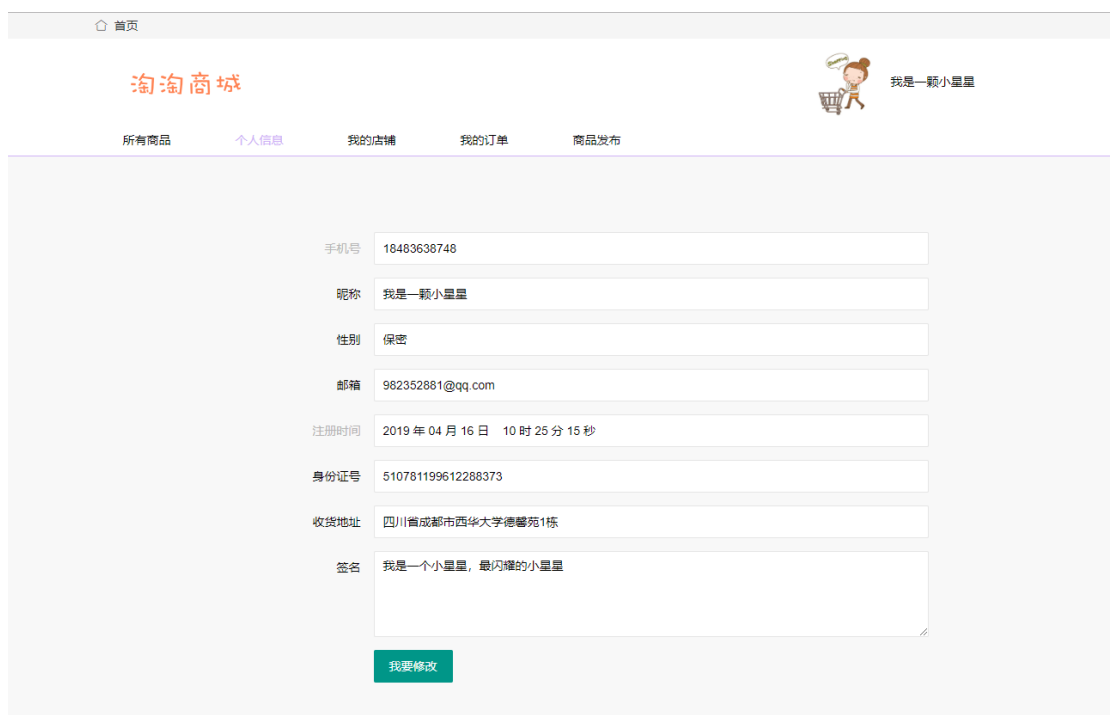


图 6.5 个人信息界面

6.3.2 商品管理模块

(1) 模块描述：商品发布、商品搜索、商品详情查看相关模块。

(2) 后台设计：暴露商品分页条件查询，根据 ID 查询商品信息，商品详情查询 Restful 格式的接口，以 Json 数据与前端交互。查询条件分为：销量、价格、发布时间以及商品名称模糊查询。接口命名及请求方式参见商品控制层 GoodsController，如图 6.6 所示：

goods-controller : Goods Controller			Show/Hide	List Operations	Expand Operations
POST	/api/goods/addToCart				加入到购物车
POST	/api/goods/buyGoods				购买商品
POST	/api/goods/queryCart				查询购物车
POST	/api/goods/queryGoodsByConditionByPage				条件查询商品 (带分页)
POST	/api/goods/queryGoodsById				根据ID查询商品信息
POST	/api/goods/queryGoodsDetail				查询商品详情
POST	/api/goods/queryOrders				查询个人订单
POST	/api/goods/queryStoresByLocation				根据定位查询店铺
POST	/api/goods/releaseGoods				发布商品
POST	/api/goods/removeOfCart				移除购物车

图 6.6 商品控制层接口说明

(3) 界面设计

商品发布界面如图 6.7 所示：

淘淘商城

时尚小糖果

所有商品个人信息我的店铺我的订单商品发布

商品名称

男士卫衣

品类

请选择

商品类型

请选择种类

单价

99.9

库存量

100

商品属性

品牌

adadis

材质

adadis

尺码

多个尺码用逗号隔开, 若含字母为大写

颜色

填写衣服颜色, 多个用逗号隔开

季节

填写适用季节

原价

参考价格, 不代表实际价值

性别

年龄

特点

商品描述

在此输入商品描述

图 6.7 商品发布界面

商品图片上传界面如图 6.8 所示：



图 6.8 商品图片上传界面

商品查找界面，包括了按商品类型查找，按销量排序，按价格排序，按发布时间排序，同时支持按商品名称进行搜索，点击商品可以查看商品的详细信息，界面如图 6.9 所示：

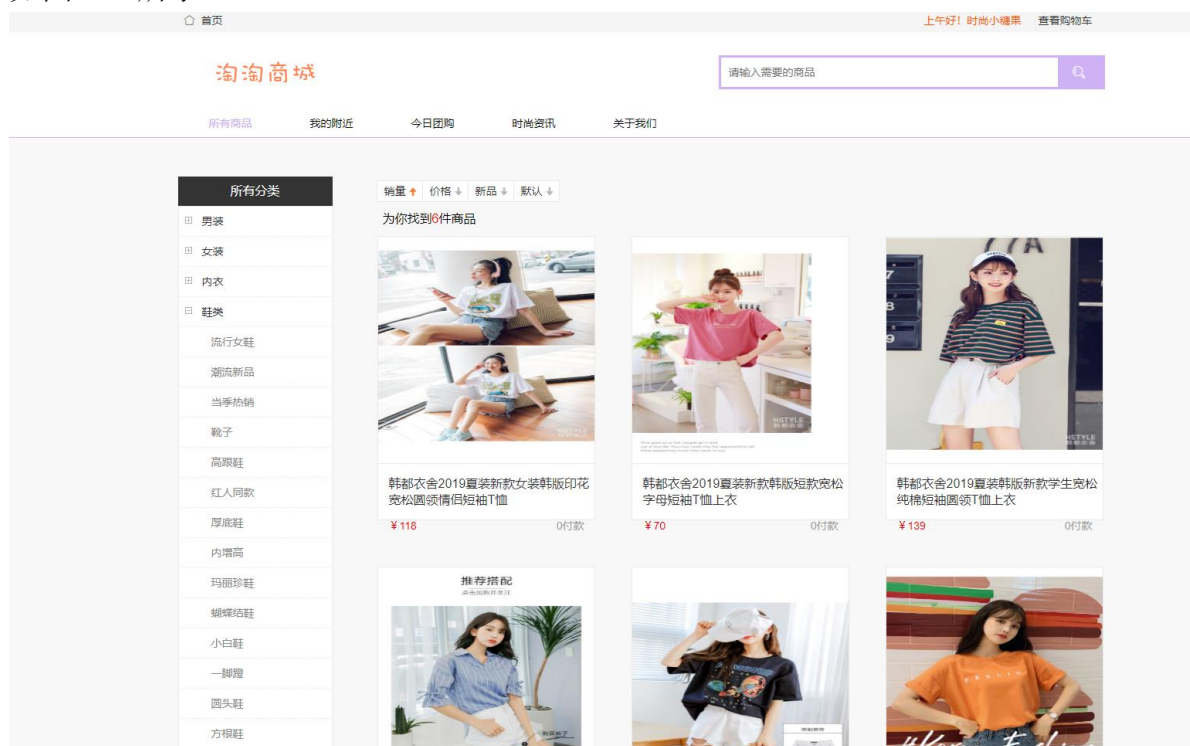


图 6.9 商品查找界面

在商品详情界面中，前端渲染后端传过来的数据，会显示出商品的属性相关内容，包括商品名称、商品价格、原价、库存量以及一些商品的特征信息，同时会在购买按钮左边展示商品缩略图，在购买按钮下部展示分辨率较高的较所有商品图片。详情页如图 6.10 所示

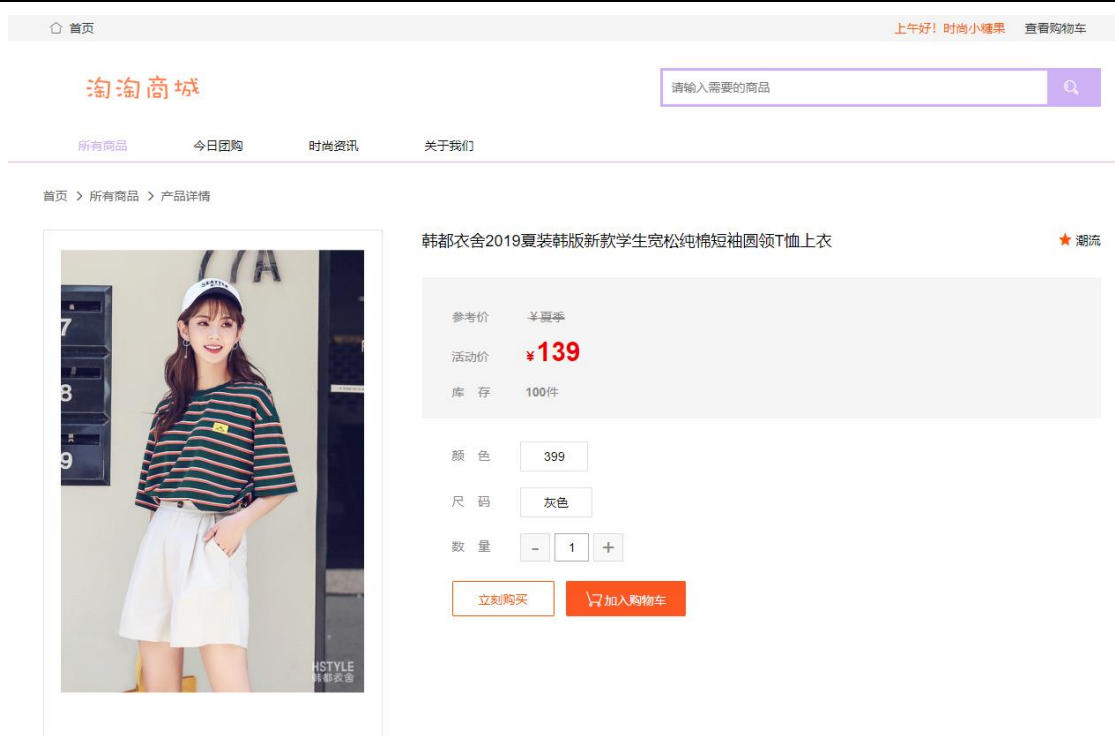


图 6.10 商品详情界面

6.3.3 推荐系统模块

(1) 模块描述：记录店铺地理位置，用户购物行为，并根据记录的信息进行服装推荐的模块。

(2) 后台设计：购物行为的业务逻辑融入到购买接口中，浏览行为的业务逻辑融入到查看商品详情业务逻辑中，这二者不再单独提供接口，仅对外暴露一个购物行为查询接口，查询上一次浏览、购买过的商品，根据其类型，关联类似商品，实现推荐功能。

(3) 界面设计

查看商品详情时，在立即购买按钮下部展示商品所有图片，在立即购买按钮左边展示缩略图，在缩略图下部则展示与正在浏览商品相同类型的推荐商品，推荐商品信息有限，只展示一张缩略图片、商品名称以及商品价格，最多推荐 5 条，最少推荐 0 条，点击所关联的推荐商品，可以进入推荐商品的详情页，同类型商品推荐展示界面如图 6.11 所示：

热销推荐



韩都衣舍2019夏装新款女装韩版印花宽

松圆领情侣短袖T恤 ¥118



韩都衣舍2019夏装韩版新款学生宽松纯

棉短袖圆领T恤上衣 ¥139

详情

品牌: adadis

材质: M



搭配推荐

图 6.11 同类型商品推荐展示

进入首页时，根据购买行为，或季节进行推荐展示，界面如图 6.12 所示：

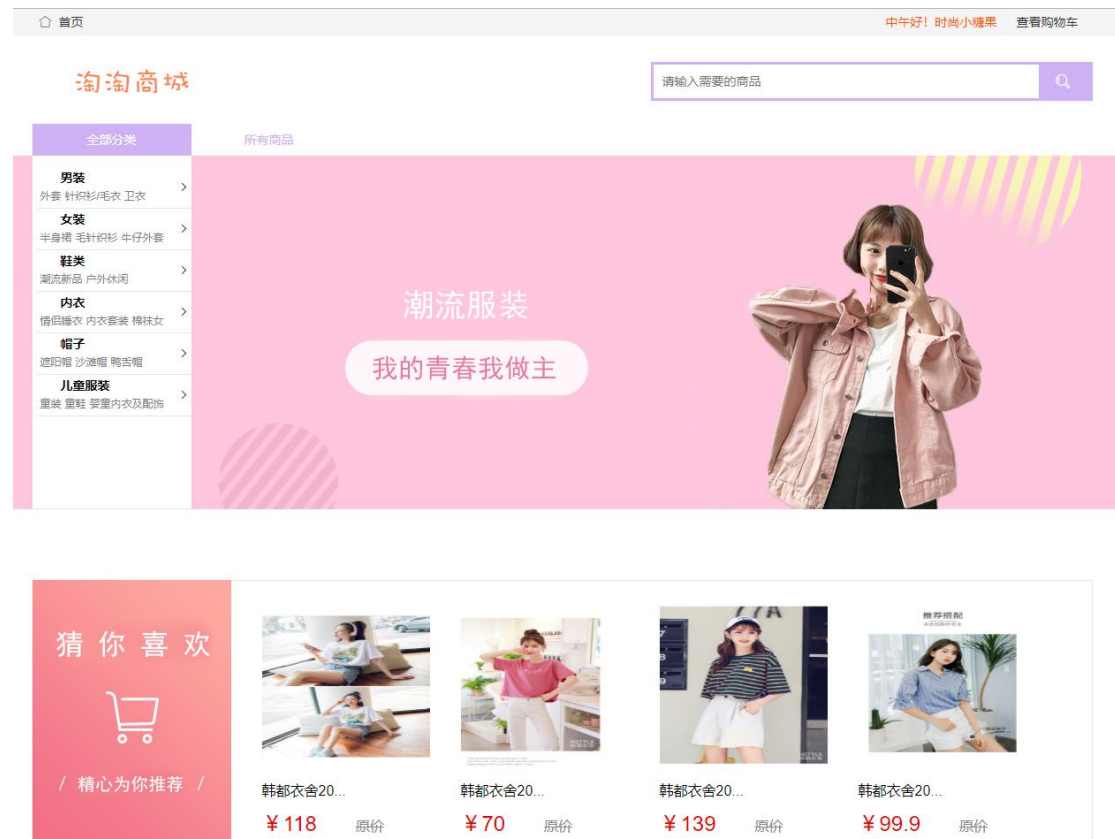


图 6.12 首页商品推荐页

6.3.4 店铺管理模块

(1) 模块描述：用户申请开店，管理员审核、封禁店铺相关模块

(2) 后台设计：暴露申请开店、分页查询店铺、审核店铺、下架店铺 Restful 格式接口，以 Json 格式数据与前端进行交互。接口命名及请求方式参见管理员控制层 AdminController，如图 6.13 所示：

admin-controller : Admin Controller			Show/Hide	List Operations	Expand Operations
POST	/api/admin/deleteStore				下架店铺
POST	/api/admin/login				登录
POST	/api/admin/queryStoreByPage				分页查询店铺
POST	/api/admin/verifyStore				审核店铺

图 6.13 管理员接口控制层

(3) 界面设计

用户申请开店界面如图 6.14 所示：

首页

淘淘商城

我是一颗小星星

所有商品个人信息我的店铺我的订单商品发布

店铺名

请输入店铺名

所在城市

请输入城市

如：四川成都

店铺类型

请选择

是否实体店

含实体店

店铺地址

店铺地址自动检测中...

店铺简介

请输入内容

立即提交

重置

图 6.14 用户申请开店界面

管理员审核界面使用 Bootstrap 开发，界面主要展示店铺信息，包括用户新申请入驻的店铺，可以对其进行审核，对已经正常运营的店铺，可以点击封禁按钮对其进行封禁。管理界面如图 6.15 所示：



淘淘商城

用户店铺管理

店铺审核

店铺列表

Show 10 entries

Search:

店铺编号	用户编号	店铺名	店铺类型	是否实体店	所在城市	店铺简介	经度	纬度	地址	店铺状态	操作
1000	1001	韩都衣舍	女装店	是	四川成都	时尚女装，连衣裙店铺多年销量第一，引领潮流。	30.78302	103.95459	四川省成都市金牛区西华街道西华大学锦馨苑6西华大学	审核通过，正常运营	封禁
1001	1002	MG小象欧美街拍时尚女装店	女装店	是	四川成都	时尚女装	30.78302	103.95459	四川省成都市金牛区西华街道西华大学锦馨苑6西华大学	审核通过，正常运营	封禁
1002	1011	Nike官方旗舰店	服装店	是	四川成都	Nike专卖店	null	null	四川省成都市郫都区龙城国际	未审核	审核
1003	1011	Adidas官方旗舰店	服装店	是	四川成都	Nike专卖店	null	null	四川省成都市郫都区龙城国际	未审核	审核

图 6.15 管理员审核店铺界面

6.3.5 购物管理模块

- (1) 模块描述：购物车添加、移除，商品购买相关模块
- (2) 后台设计：暴露购物车查询、商品购买、订单查询、购物车添加、购物移除的 Restful 格式接口，以 Json 格式数据与前端进行交互。
- (3) 界面设计：

购物车界面如图 6.16 所示：

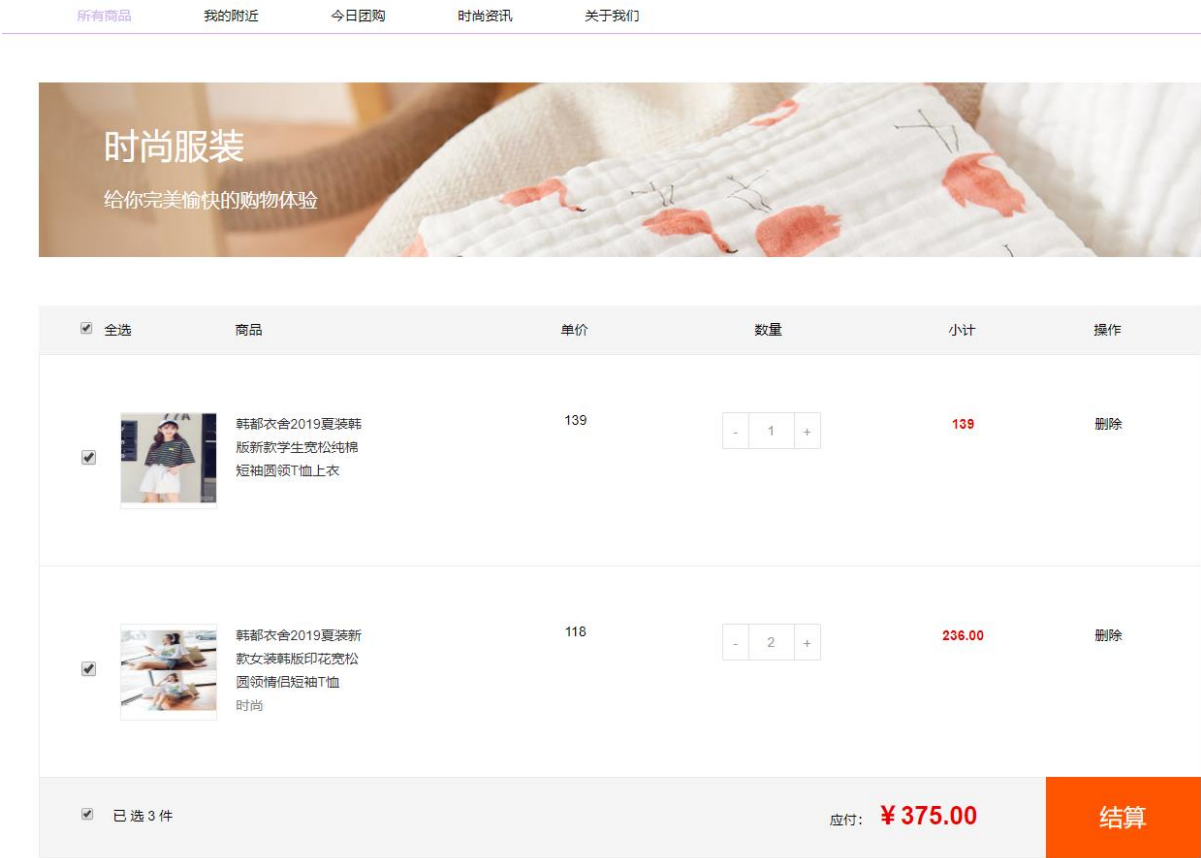


图 6.16 购物车界面

支付界面如图 6.17 所示：



图 6.17 支付界面

6.3.6 定位与地图模块

地图与定位为前端调用高德地图 API，界面设计如图 6.18 所示：



图 6.18 地图界面

7 软件编码

7.1 编码环境与技术



7.1.1 数据库编码环境

- (1) 数据库: Mysql5.5
- (2) 图形化工具: Navicat for Mysql
- (3) 字符编码: UTF-8

7.1.2 项目编码环境

- (1) 开发工具: IntelliJ IDEA、Visual Studio Code、Photoshop
- (2) UML 建模工具: 亿图
- (3) 浏览器: Chrome、Edge
- (4) 服务器: Tomcat 8.5
- (5) 开发语言: Java、JavaScript、Html、SQL
- (6) 版本管理: Git
- (7) 代码仓库: Github

7.1.3 编码技术

(1) 前端页面: 使用技术涉及 HTML5, CSS3, JavaScript, AJAX, JQuery、LayerUI、Bootstrap; 整个网站的页面搭建采用 Bootstra 与 LayerUI 前端框架, 结合 JQuery 与 Ajax, 实现了前后端完全分离。

(2) 后台实现: 使用技术涉及 Java、Spring Cloud、Spring Boot、Mybatis 数据库访问组件, C3P0 数据库连接池、Redis 缓存、Maven 依赖包管理、Gson 格式化组件、秒嘀科技短信验证平台、IP 定位高德地图 API、Zxing 二维码生成、URLConnection 网络请求与爬虫。

7.2 网站架构

整个程序前后端分离、使用 Json 数据进行交互。后端使用 Spring Cloud 微服务分布式架构。将整个后端分为交易集成服务、业务数据服务两个服务。所有前端请求与数据响应统一由交易集成服务进行, 所有与数据库交互逻辑统一由业务数据服务处理, 交易集成服务端口使用 13000, 业务数据服务端口号使用 12000。网站架构图如图 7.1 所示:

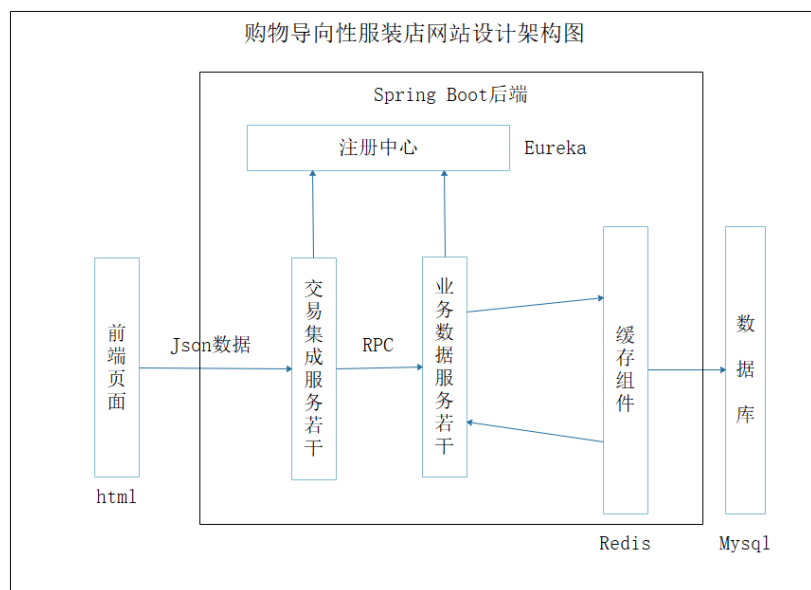


图 7.1 网站架构图

7.3 设计模式

本设计编码中使用的设计模式为：工厂模式、单例模式

（1）工厂模式运用在 Spring 的 IOC 中，使用静态工厂模式创建对象，并且 Spring 容器统一管理，使用工厂模式使得对象创建更加灵活。

（2）单例模式运用在 Spring 的依赖注入 DI 中与自行封装工具类中，在 Spring 的 DI 中使用@Autowired 注解，以单例的形式注入对象，在 Controller 单例注入 Service 对象，在 Service 中单例注入 Dao 对象，从而增加对象复用。在自行封装的工具类 GsonUtil 也使用到，使用单例模式节省程序内存。

7.4 代码结构

项目的编码对象主要为：注册中心服务、业务数据服务、交易集成服务、前端。其中业务数据服务于交易集成服务为业务逻辑类服务，注册中心服务为 Spring Cloud 必要组件，维护后二者服务的注册信息，前端为单独项目。

业务数据服务代码结构见图 7.2:

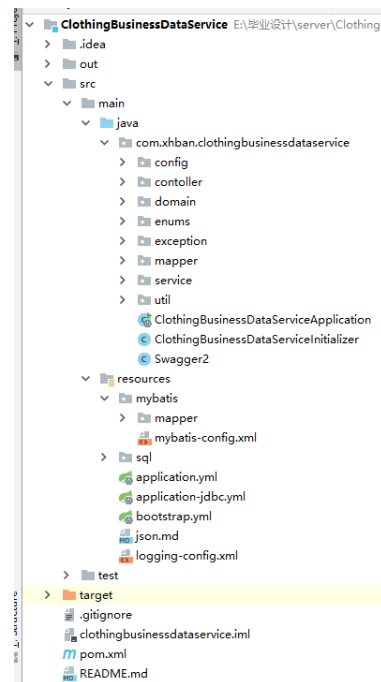


图 7.2 业务数据服务代码结构

交易集成服务代码结构如图 7.3 所示:

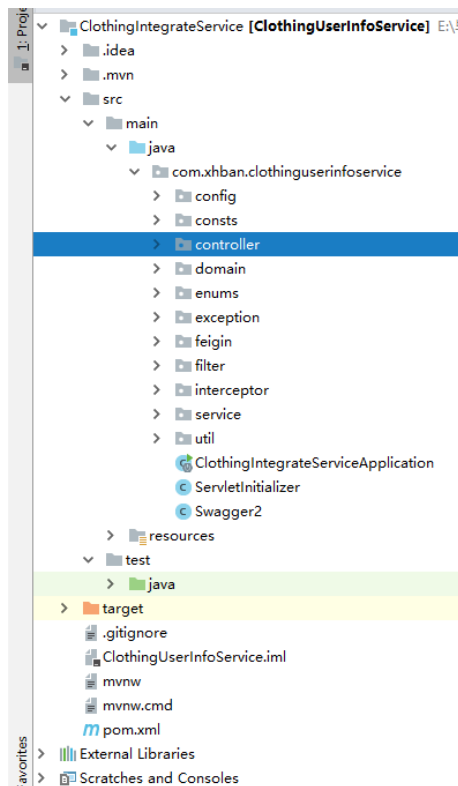


图 7.3 交易集成服务代码结构

前端代码结构见图 7.4:

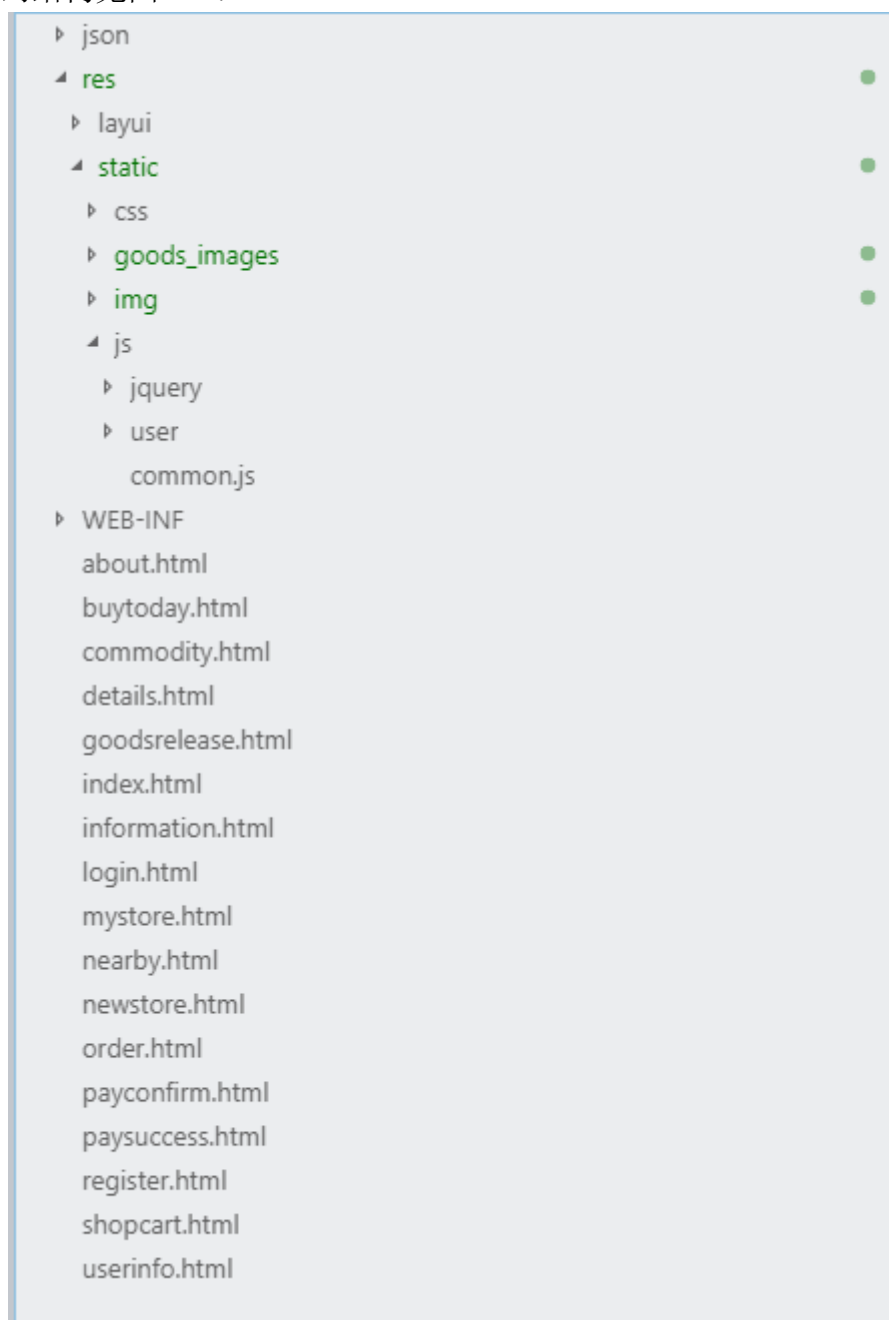


图 7.4 前端代码结构

8 软件测试

软件测试的主要目的是验证软件开发的正确性、可靠性、响应速度等，确保系统的开发实现了需求所定义的功能。在本设计中对业务数据服务进行了单元测试，利用 Swagger 对接口进行了集成测试，对用户的用例进行功能了测试。

8.1 单元测试

用户相关单元测试如图 8.1:



```
1 package com.xhban.clothingbusinessdataservice;
2
3 import ...
4
15 @RunWith(SpringRunner.class)
16 @SpringBootTest
17 public class UserMapperTest {
18     @Autowired
19     private UserMapper userMapper;
20
21
22     @Test
23     public void testAdd() {
24         UserBO userBO = new UserBO();
25         userBO.setPhone("18781691047");
26         userBO.setPassword("123456");
27         int result = userMapper.addUser(userBO);
28         System.out.println(result);
29     }
30
31     @Test
32     public void testUpdateUser() {
33         UserBO userBO = new UserBO();
34         userBO.setId(1001L);
35         userBO.setName("大涛涛");
36         userBO.setEmail("982352881@qq.com");
37         userBO.setSex("男");
38         userBO.setIdCard("510781199612288373");
39         userBO.setLastLoginLocation("四川成都");
40         userBO.setLastLoginLatitude(new BigDecimal( val: "111.12"));
41         userBO.setLastLoginLongitude(new BigDecimal( val: "110.55"));
42         userBO.setType(2);
43         userBO.setTypeExplain("商家");
44         userBO.setSignature("hhehehe");
45         userBO.setLastLoginTime(new Date());
46         userBO.setReceiveAddress("西华大学");
47         userBO.setRegisterLatitude(new BigDecimal( val: "113.22"));
48         userBO.setRegisterLocation("四川成都");
49         userBO.setRegisterLongitude(new BigDecimal( val: "115.12"));
50         userBO.setRegisterTime(new Date());
51         int result = userMapper.updateUser(userBO);
52         System.out.println(result);
53     }
54
55     @Test
56     public void testDeleteUser() {
57         UserBO userBO = new UserBO();
58         userBO.setId(1002L);
59         int result = userMapper.deleteUser(userBO);
60         System.out.println(result);
61     }
62 }
```

图 8.1 用户相关单元测试截图

用户用相关测试补充截图，如图 8.2 所示：

```
@Test
public void testCheckPhone() {
    Long count = userMapper.checkPhone("string");
    System.out.println(count);
}

@Test
public void testCheckUser() {
    UserBO userBO = new UserBO();
    userBO.setPhone("18483638748");
    userBO.setPassword("0");
    Long count = userMapper.checkUser(userBO);
    System.out.println(count);
}

@Test
public void queryUserByPhoneAndPassword() {

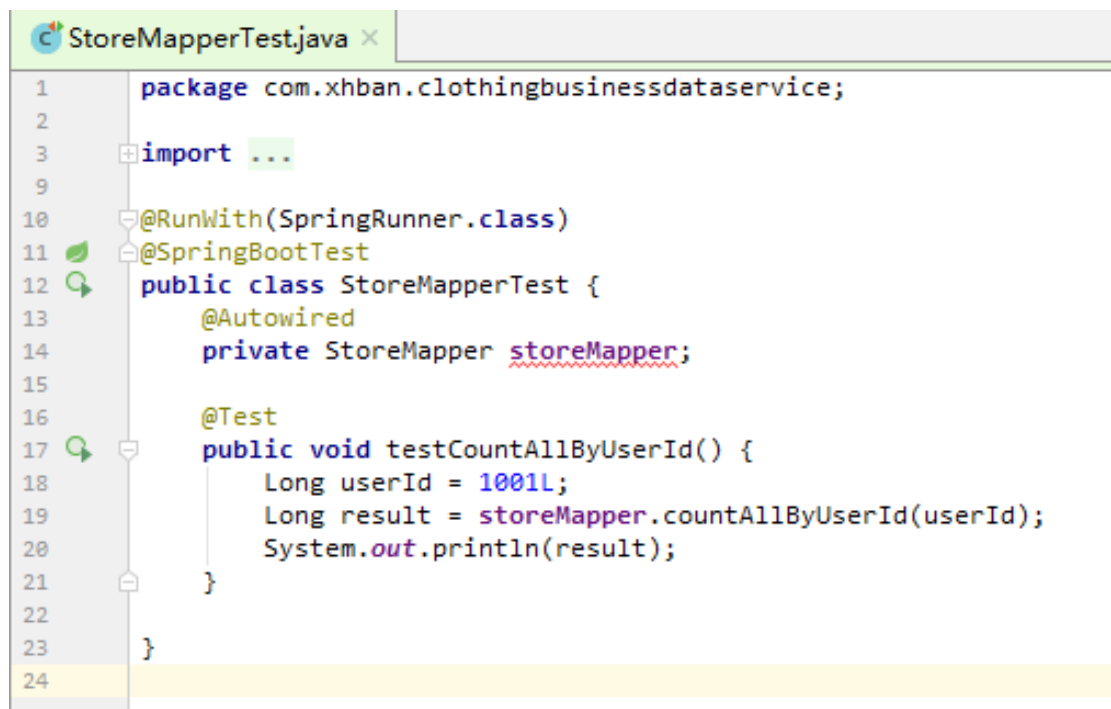
    UserBO userBO = userMapper.queryUserByPhoneAndPassword( phone: "18483638748", password: "123456");
    System.out.println(userBO);
}

@Test
public void testUpdatePassword() {
    Long id = 1001L;
    String originPassword = "1245";
    String newPassword = "123";
    Integer result = userMapper.updatePassword(id, originPassword, newPassword);
    System.out.println(result);
}

@Test
public void testQueryUserById() {
    Long id = 1002L;
    UserBO userBO = userMapper.queryUserById(id);
    System.out.println(GsonUtil.getGson().toJson(userBO));
}
```

图 8.2 用户相关单元测试截图补充

店铺相关单元测试截图如图 8.3 所示：



```
StoreMapperTest.java x
1 package com.xhban.clothingbusinessdataservice;
2
3 import ...
9
10 @RunWith(SpringRunner.class)
11 @SpringBootTest
12 public class StoreMapperTest {
13     @Autowired
14     private StoreMapper storeMapper;
15
16     @Test
17     public void testCountAllByUserId() {
18         Long userId = 1001L;
19         Long result = storeMapper.countAllByUserId(userId);
20         System.out.println(result);
21     }
22
23 }
24
```

图 8.3 店铺相关单元测试

商品发布相关单元测试如图 8.4 所示

```
ReleasesMapperTest.java x
1 package com.xhban.clothingbusinessdataservice;
2
3 import ...
12
13 @RunWith(SpringRunner.class)
14 @SpringBootTest
15 public class ReleasesMapperTest {
16     @Autowired
17     private ReleasesMapper releasesMapper;
18
19     @Test
20     public void testQueryAllReleases() {
21         ReleasesBO releasesBO = new ReleasesBO();
22         releasesBO.setStoreId(1000L);
23         List<ReleasesBO> releasesBOList = releasesMapper.queryAllReleases(releasesBO);
24         System.out.println(releasesBOList);
25     }
26
27     @Test
28     public void testQueryStoreIdByGoodsId() {
29         Long goodsId = 1002L;
30         Long storeId = releasesMapper.queryStoreIdByGoodsId(goodsId);
31         System.out.println(storeId);
32     }
33 }
34
35
```

图 8.4 商品发布相关单元测试截图

商品属性相关单元测试如图 8.5 所示：

```
PropertyMapperTest.java x
1 package com.xhban.clothingbusinessdataservice;
2
3 import ...
12
13 @RunWith(SpringRunner.class)
14 @SpringBootTest
15 public class PropertyMapperTest {
16     @Autowired
17     private PropertyMapper propertyMapper;
18
19     @Test
20     public void testAdd() {
21         PropertyBO propertyBO = new PropertyBO();
22         propertyBO.setGoodsId(1000L);
23         propertyBO.setName("大小");
24         propertyBO.setValue("20尺");
25         int result = propertyMapper.addProperty(propertyBO);
26         System.out.println(result);
27     }
28
29     @Test
30     public void testQueryPropertiesByGoodsId() {
31         Long goodsId = 1017L;
32         List<PropertyBO> propertyBOList = propertyMapper.queryPropertiesByGoodsId(goodsId);
33         System.out.println(propertyBOList);
34     }
35 }
36
37
38
```

图 8.5 商品属性相关单元测试

商品图片相关单元测试如图 8.6 所示：

```
ImageMapperTest.java
1 package com.xhban.clothingbusinessdataservice;
2
3 import ...
4
5 @RunWith(SpringRunner.class)
6 @SpringBootTest
7 public class ImageMapperTest {
8     @Autowired
9     private ImageMapper imageMapper;
10
11     @Test
12     public void testQueryImageByGoodsId() {
13         Long goodsId = 1001L;
14         List<ImageBO> imageBOList = imageMapper.queryImageByGoodsId(goodsId);
15         System.out.println(imageBOList);
16     }
17 }
```

图 8.6 商品图片相关单元测试

购物行为相关单元测试如图 8.7 所示：

```
ActionMapperTest.java
1 package com.xhban.clothingbusinessdataservice;
2
3 import ...
4
5 @RunWith(SpringRunner.class)
6 @SpringBootTest
7 public class ActionMapperTest {
8     @Autowired
9     private ActionMapper actionMapper;
10
11     @Test
12     public void testAdd() {
13         ActionBO actionBO = new ActionBO();
14         actionBO.setGoodsId(1001L);
15         actionBO.setUserId(1002L);
16         actionBO.setOperationLocation("四川成都");
17         actionBO.setOperationLongitude(new BigDecimal( val: "123.45"));
18         actionBO.setOperationLatitude(new BigDecimal( val: "110.02"));
19         actionBO.setType(1);
20         actionBO.setTypeExplain("浏览");
21         actionBO.setOperationTime(new Date());
22         int result = actionMapper.addAction(actionBO);
23         System.out.println(result);
24     }
25
26     @Test
27     public void testDelete() {
28         Long id = 1000L;
29         int result = actionMapper.deleteAction(id);
30         System.out.println(result);
31     }
32
33     @Test
34     public void testUpdateAction() {
35         Long id = 1L;
36         Long userId = 1001L;
37         Long goodsId = 1100L;
38         Integer type = 2;
39         String typeExplain = "浏览";
40         String location = "四川成都";
41         BigDecimal longitude = new BigDecimal( val: "109.77");
42         BigDecimal latitude = new BigDecimal( val: "33.45");
43         Integer result = actionMapper.updateActionByUserId(goodsId, type, typeExplain, location, longitude, latitude, userId, id);
44         System.out.println(result);
45     }
46 }
```

图 8.7 购物行为相关单元测试

购买记录相关单元测试，如图 8.8 所示：

```
OrderRecordMapperTest.java
1 package com.xhban.clothingbusinessdataservice;
2
3 import ...
4
5 @RunWith(SpringRunner.class)
6 @SpringBootTest
7 public class OrderRecordMapperTest {
8     @Autowired
9     private OrdersRecordMapper ordersRecordMapper;
10
11     @Test
12     public void testAddOrderRecord() {
13         OrdersRecordBO ordersRecordBO = new OrdersRecordBO();
14         ordersRecordBO.setLatitude(new BigDecimal( val: "110.23"));
15         ordersRecordBO.setLongitude(new BigDecimal( val: "95.17"));
16         ordersRecordBO.setUserId(1000L);
17         ordersRecordBO.setLocation("四川成都");
18         ordersRecordBO.setSumMoney(new BigDecimal( val: "148.43"));
19         int result = ordersRecordMapper.addOrdersRecord(ordersRecordBO);
20         System.out.println(result);
21     }
22
23     @Test
24     public void testUpdateSumMoneyById() {
25         int result = ordersRecordMapper.updateSumMoneyById( id: 1000L, new BigDecimal( val: "2323.33"));
26         System.out.println(result);
27     }
28 }
29
30
31
32
33
34
35
36
37
```

图 8.8 购买记录相关单元测试

商品相关单元测试如图 8.9 所示：

```

1 @RunWith(SpringRunner.class)
2 @SpringBootTest
3 public class OrdersMapperTest {
4     @Autowired
5     private OrdersMapper ordersMapper;
6
7     @Test
8     public void testAddOrderRecord() {
9         OrdersBO ordersBO = new OrdersBO();
10         ordersBO.setGoodsId(1000L); //商品ID
11         ordersBO.setUserId(1001L); //用户ID
12         ordersBO.setOrdersRecordId(1002L); //订单记录ID
13         ordersBO.setStatus(EOrderStatus.NEW_ORDER_WAIT_PAY.getStatus()); //状态
14         ordersBO.setStatusExplain(EOrderStatus.NEW_ORDER_WAIT_PAY.getStatusExplain()); //状态说明
15         BigDecimal price = new BigDecimal( val: "3.55");
16         BigDecimal ordersSumMoney = price.multiply(new BigDecimal( val: 7));
17         ordersBO.setSumMoney(ordersSumMoney); //单个订单总价
18         ordersBO.setAmount(12); //订单数量
19         int result = ordersMapper.addOrders(ordersBO);
20         System.out.println(result);
21     }
22
23     @Test
24     public void testQueryOrdersByUserId() {
25         Long userId = 5000L;
26         List<OrdersBO> ordersBOList = ordersMapper.queryOrdersByUserId(userId);
27         System.out.println(ordersBOList);
28     }
29
30     @Test
31     public void testQueryOrdersByGoodsId() {
32         Long goodsId = 1001L;
33         List<OrdersBO> ordersBOList = ordersMapper.queryOrdersByGoodsId(goodsId);
34         System.out.println(ordersBOList);
35     }
36 }
37
```

图 8.9 商品相关单元测试



测试结果：商品添加抛出数据库异常，其他结果正常，经排查是数据库中商品名字段长度不够，后面通过增加其长度，进行回归测试，最后结果正确。

8.2 集成测试

(1) 测试接口：用户注册

接口地址：http://localhost:13000/api/user/register

测试数据：

```
{
  "latitude": "98.77",
  "location": "四川成都",
  "longitude": "100.32",
  "password": "982352.Zp",
  "phone": "18483638748",
  "verifyCode": "419170"
}
```

响应数据：

```
{
  "code": 0,
  "message": "注册成功"
}
```

(2) 测试接口：修改个人信息

接口地址：http://localhost:13000/api/user/updateUser

测试数据：

```
{
  "email": "18483638748@163.com",
  "name": "周月月",
  "receiveAddress": "成都市西华大学",
  "sex": "男",
  "signature": "我是一颗小星星，咿呀咿呀哟"
}
```

响应数据：

```
{
  "code": 0,
  "message": "修改个人信息成功",
  "data": true
}
```



(3) 测试接口：用户开店

接口地址：http://localhost:13000/api/user/addStore

测试数据：

```
{
  "address": "四川成都",
  "entityStore": true,
  "introduce": "成都最新服装店, 品种齐全, 种类繁多",
  "latitude": 100.32,
  "location": "成都市西华大学",
  "longitude": 98.23,
  "name": "金克斯服装",
  "type": 1,
  "typeExplain": "服装店",
  "userId": 1000
}
```

响应数据：

```
{
  "code": 0,
  "message": "申请开店成功",
  "data": true
}
```

(4) 测试接口：用户购买商品

接口地址：http://localhost:13000/api/goods/buyGoods

测试数据：

```
{
  "latitude": 100.23,
  "location": "成都市环球广场",
  "longitude": 98.23,
  "purchaseGoodsParaSubVOList": [
    {
      "amount": 2,
      "goodsId": 1000
    }
  ]
}
```



响应数据:

```
{
  "code": 0,
  "message": "购买成功",
  "data": {
    "ordersRecordId": 1009,
    "ordersVOList": [
      {
        "id": null,
        "goodsId": 1000,
        "userId": 1000,
        "ordersRecordId": 1009,
        "addTime": 1555400153473,
        "deliverTime": null,
        "cancelTime": null,
        "receiveTime": null,
        "status": 1,
        "statusExplain": "新订单,等待支付",
        "commentType": null,
        "commentContent": null,
        "amount": 2,
        "sumMoney": 236,
        "goodsName": "韩都衣舍 2019 夏装新款女装韩版印花宽松圆领情侣短袖 T 恤",
        "goodsPrice": 118,
        "goodsImage": {
          "id": 1000,
          "goodsId": null,
          "imageName": "90fb7588e4774358bdaf48235c04bd8a.jpg"
        }
      }
    ],
    "ordersImage": {
      "id": 1003,
      "goodsId": null,
      "imageName": "e60b1482a04e428799749be289e186f0.jpg"
    },
    "sumMoney": 535.7
  }
}
```

(5) 测试接口: 用户发布商品

接口地址: <http://localhost:13000/api/goods/releaseGoods>

测试数据:



```
{
  "goodsB0": {
    "amount": 34,
    "introduce": "2019 新款男生潮流牛仔裤",
    "name": "潮流仔裤",
    "price": 47,
    "type": 1,
    "typeExplain": "服装"
  },
  "imageB0List": [
    {
      "imageName": "1.jpg"
    },
    {
      "imageName": "2.jpg"
    }
  ],
  "latitude": 110.23,
  "location": "四川绵阳",
  "longitude": 89.92,
  "propertyList": [
    {
      "name": "腰围",
      "propertyExplain": "腰围尺寸",
      "value": "28"
    }
  ]
}
```

响应数据:

```
{
  "code": 1,
  "message": "商品上架失败",
  "data": null
}
```

测试结果: 除了商品上架以为, 其余集成测试均正确。后来经排查, 商品上架接口中的上架失败原因是后台程序代码逻辑出现一个小 Bug, 经修改后, 回归测试, 结果正确。

8.3 功能测试



通过运行前端网页，在浏览器中对各个功能点进行点击测试，从用户登录，到浏览商品、加入购物车、购买商品，再到商品推荐，附近店铺查看。除地图处因网速问题响应较慢，其他页面响应正确、流畅。

8.4 测试结论

通过以上三种方式的测试，结果表明网站具有很好的鲁棒性，数据一致性，预期功能已全部实现，界面以及界面之间的跳转正常，响应速度流畅，完成了需求分析要求的各个功能要求与性能要求。



总结

在这次毕业设计中我得到了许多收获，主要体现在三个方面，一个方面是对电子商务的新认知，一个是技术上的提升，一个是对自己学习能力的提升与毅力的锻炼。

在设计前，我做了大量的调研工作，对中小企业的电商网业务有了更详细的了解，结合之前在企业的实习经验，以一个中小企业项目的理念来进行设计与开发，也是我第一次自主开发电商型的网络产品，让我对一些电子商务企业的业务细节与技术实施手段都有了全新的认识。

在这次毕业设计过程中，让我对高并发、分布式、微服务的架构在电商场景下的运用更加熟练，整个后台开发期间，遇到了许多坑点和问题，一些是曾经困扰我的技术问题，一些是新的技术问题，而本次开发过程中将这些问题都进行了很好的解决，对我的技术综合实力提升了不少！

在设计完成时，我发现软件的研发是一个漫长和复杂的过程。在整个过程中，都需要周密的计划和安排，每一步骤，都需要按照软件流程来实现。在项目的代码开发过程中，我始终遵守开发规范原则，并通过实践来加强理论的学习，通过实习锻炼，我巩固了软件研发的一般过程，加强了对实际问题的处理能力。因此我的自学能力得到了很大提升，同时毅力也得到了很好的锻炼。



致谢

在指导老师和计算机学院老师们的指导与同学们的帮助下，使得此次论文定稿能够顺利的完成。老师们都倾注了大量的心血和汗水，无论是在论文的选题、构思和资料的收集方面，还是在论文的研究方法以及成文定稿方面，我都得到了我的指导老师和计算机学院各位老师悉心细致的教诲和无私的帮助，特别是他们广博的学识、深厚的学术素养、严谨的治学精神和一丝不苟的工作作风使我终生受益，在此表示真诚地感谢。在论文的写作过程中，也得到了许多同学的宝贵建议，同时还得到了许多同学的支持和帮助，在此一并致以诚挚的谢意。感谢所有关心、支持、帮助过我的良师益友，最后感谢能给我提供良好学习范围的与平台的学校！



参考文献

- [1]焦灵. 基于 Web 的购物网站系统设计[J]. 电脑编程技巧与维护. 2018. 12
- [2]刘易. 网上商城及推荐系统的设计与实现[D]. 哈尔滨工业大学. 2017. 07
- [3]张晓红. 基于服装定制功能的电子商务网站设计与实现[J]. 福建电脑. 2017. 07
- [4]孙沈楠. 女性服装购物网站用户界面设计研究[D]. 长春工业大学. 2018. 09
- [5]王朋. 基于专家系统的个性化网上服装商城的设计[D]. 东华大学. 2016. 12
- [6]郭致. 魏银珍. 基于 Spring Cloud 服务调用的设计与应用[J]. 信息技术与网络安全. 2019. 02.
- [7]王方旭. 基于 Spring Cloud 实现业务系统微服务化的设计与实现[J]. 电子技术与软件工程. 2018. 08
- [8]陶璐. 基于微服务的大型网站架构设计[J]. 科技视界. 2018. 10
- [9]张颖, 顾雯, 李浩, 刘晓刚. 基于用户体验的服装定制网站构建维度的探析[J]. 东华大学学报(自然科学版). 2018. 02
- [10]徐士川. 电子商城系统中订单模块与秒杀模块的设计与实现[D]. 南京大学, 2018. 09
- [11]刘彦昌. 二手书交易平台的设计与实现[D]. 北京邮电大学, 2018. 11