智慧电子商城设计与实

关键词：微信平台；智能推荐的前台客户系统，数据可视化后台管理系统；辅助决策。

1. ：绪论

1.1 课题背景

移动电商、微信平台、社交电商

......

在国家大力推进互联网+的大背景下，随着科技变革、产业革命、消费升级，我国电商的发展步入了规模持续增长、业态不断创新结构日趋优化的新阶段。

1.2 研究现状和发展趋势

APP、WebApp、微信推广、社交电商

......

1.3 研究内容

系统整体架构、以模块为单位的微服务设计

系统与微信平台的信息交互

商品推荐、数据分析统计、可视化展现

......

1.4 论文组织结构

第二章：相关理论与技术。

本章节针对电商项目的特点，对项目的架构设计和技术选型进行详细阐述。后端项目的开发语言以kotlin为主、java8为辅；采用社区活跃度很高的SpringCloud微服务架构；数据库采用互联网行业最常用的mysql；前端框架采用当前比较流行的React框架；前端商城借助微信公众平台作为入口。

2.1 Spring Cloud微服务架构

在当下互联网大环境下，互联网产品的需求变化快，必然要求产品迭代更新也要快，然而现实情况是很多的产品都面临着迭代速度慢的问题。分析其中原因，就是随着产品功能的不断累积、变更，应用实现越来越复杂，代码规模也越来越大，开发团队工作在一个逻辑复杂、模块耦合度高的单块架构应用上，从而导致应用难于维护和更新。

2.1.1传统的单体架构

所谓单体应用就是项目所有的功能归档在一个项目中的应用。单体架构在小微企业比较常见，典型代表就是一个应用、一个数据库、一个web容器就可以运行起来。单体应用是最早的应用形态，不需要太关注整体性能，项目规模中小型时，开发和部署都相对方便。



  随着业务的不断扩大，需求功能的持续增加，单体架构已经很难满足业务快速变化的需要，容易造成如下问题，当项目规模变大时，代码容易产生冲突，造成版本管理困难；局部功能模块出现问题，可能会影响整体，因此稳定性也较差；在系统维护性和扩展性方面，随着项目规模的扩大，需求的不断增加，系统复杂性直线上升，容易造成系统不易理解、扩展需求的开发困难等问题；从技术选型和兼容性上来看，单一构架应用兼容的技术相对单一，初始的技术选型严重限制了其将来采用不同语言或者框架的能力，在一定程度上限制了项目技术的革新。

2.1.2微服务架构

从业界的讨论来看，微服务本身并没有一个严格的定义，这里借用ThoughtWorks的首席科学家——马丁·福勒（Martin Fowler）先生，对微服务的这段描述：

微服务架构是一种架构模式，它是提倡将单一应用程序划分成一组小的服务，服务之间互相协调、互相配合，为用户提供最终价值。每个服务运行在其独立的进程中，服务与服务件采用轻量级的通信机制进行沟通（通常采用HTTP的RESTful API）。每个服务都围绕着具体业务进行构建，并且能够被独立的部署到生产环境、类生产环境等。另外，应尽量避免统一的、集中式的服务管理机制，对具体的一个服务而言，应根据业务上下文，选择合适的语言、工具对其进行构建。



总结下来，微服务就是由多个以单一应用构成的小服务，自己拥有自己的进程与轻量化处理，服务依业务功能设计，以全自动的方式部署，并通过各自暴露的API接口相互通讯。同时服务会使用最小的规模的集中管理能力，服务可以用不同的编程语言与数据库等技术实现。

相比于传统的单体架构，微服务架构有很多优点。首先，通过将巨大单体应用分解为多个服务，解决了复杂性问题。在功能不变的情况下，应用被分解为多个可管理的分支或服务。每个服务都有一个用RPC-或者消息驱动API定义清楚的边界。微服务架构模式给采用单体式编码方式很难实现的功能提供了模块化的解决方案，由此，单个服务很容易开发、理解和维护。  
 第二，这种架构使得每个服务都可以有专门开发团队来开发。开发者可以自由选择开发技术，提供API服务。这就意味着开发者不需要被迫使用某项目开始时采用的过时技术，而是可以随着技术的更新换代而选择最新的技术。甚至于，因为服务都是相对简单，即使用最新技术重写以前代码也不是很困难的事情。  
 第三，微服务架构是每个微服务独立部署。开发者不再需要协调其它服务部署对本服务的影响。这种改变可以加快部署速度。微服务架构使得持续化部署成为可能。  
 最后，微服务架构模式使得每个服务独立扩展。开发者可以根据每个服务的规模来部署满足需求的规模。甚至可以使用更适合于服务资源需求的硬件。

微服务设计原则

。。。。。。。。。

2.1.3 Spring Cloud微服务架构

Spring Cloud是一个基于SpringBoot实现的微服务架构开发工具。它为微服务架构中涉及的配置管理、服务治理、断路器、智能路由、微代理、控制总线、全局锁、决策竞选、分布式会话和集群状态管理等操作提供了一种简单的开发方式。



2.2 SpringBoot框架

Spring Boot是由Pivotal团队提供的全新框架，其设计目的是用来简化新Spring应用的初始搭建以及开发过程。该框架使用了特定的方式来进行配置，从而使开发人员不再需要定义样板化的配置。其实spring boot其实不是什么新的框架，它默认配置了很多框架的使用方式。Spring Boot的出现大大简化了Spring应用程序的开发，其中最重要的是以下四个核心：

（1）自动配置：针对很多Spring应用程序常见的应用功能，Spring Boot能自动提供相关配置。

（2）起步依赖：告诉Spring Boot需要什么功能，它就能引入需要的库。

（3）命令行界面：这是Spring Boot的可选特性，借此你只需写代码就能完成完整的应用程序，无需传统项目构建。

（4）Actuator：让你能够深入运行中的Spring Boot应用程序，一探究竟。

2.3 Kotlin语言（可选）+JAVA8

Kotlin是由 [JetBrains](https://baike.baidu.com/item/JetBrains) 开发的一个用于现代多平台应用的静态编程语言，它可以编译成Java字节码，也可以编译成JavaScript，方便在没有JVM的设备上运行。

它是一种针对Java平台的新编程语言。Kotlin简洁、安全、务实，并且专注于与Java代码的互操作性，几乎可以用在现在Java使用的任何地方：服务器端开发、Android应用等。它可以很好的和所有现存的Java库和框架一起工作，而且性能水平和Java旗鼓相当。

2.4 数据库技术

关系型数据库 + 范式

[MySQL](https://baike.baidu.com/item/MySQL/471251)是一种[开放源代码](https://baike.baidu.com/item/%E5%BC%80%E6%94%BE%E6%BA%90%E4%BB%A3%E7%A0%81" \t "https://baike.baidu.com/item/MySQL%E6%95%B0%E6%8D%AE%E5%BA%93/_blank)的关系型[数据库管理](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93%E7%AE%A1%E7%90%86)系统（RDBMS），使用最常用的数据库管理语言--[结构化查询语言](https://baike.baidu.com/item/%E7%BB%93%E6%9E%84%E5%8C%96%E6%9F%A5%E8%AF%A2%E8%AF%AD%E8%A8%80)（SQL）进行数据库管理。

MySQL因为其速度、可靠性和适应性而备受关注。它提供 [TCP/IP](https://baike.baidu.com/item/TCP/IP)、ODBC 和 [JDBC](https://baike.baidu.com/item/JDBC)等多种[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93" \t "https://baike.baidu.com/item/mySQL/_blank)连接途径，同时通过优化的 [SQL](https://baike.baidu.com/item/SQL)查询算法，能有效地提高查询速度；而且mysql支持多线程，能充分利用CPU资源；MySQL还提供用于管理、检查、优化数据库操作的各种[管理工具](https://baike.baidu.com/item/%E7%AE%A1%E7%90%86%E5%B7%A5%E5%85%B7" \t "https://baike.baidu.com/item/mySQL/_blank)，这里我们采用轻量级的可视化管理工具SQLyog。本项目选择使用mysql作为数据库，一方面是由于其代码开源免费。

结构化查询语言（SQL）是数据库查询和程序设计语言，是关系数据库的操作语言。不同的数据库对于SQL语言的支持和标准存在细微的区别，同时根据不同数据库的需求，数据库开发商对SQL语言也做了不同程度的扩展。

MySQL是由原MySQL AB公司自主研发的，是目前IT行业最流行的开放源代码的数据库管理系统，同时它也是一个支持多线程高并发多用户的关系型数据库管理系统。MySQL数据库是基于Linux操作系统开发出来的数据库，是开放源码的免费数据库。

MySQL能够满足通用的商业需求，并提供足够强大的服务。在权威的第三方评测机构多次测试比较各种数据库TPCC值的过程中，MySQL始终都有优异的表现，而且在其他所有商用的通用数据库管理系统中，只有Oracle数据库能够与其一较高下。

2.5 React 前端框架

React是facebook在2013年开源的Github上的JavaScript库，它把用户界面抽象成一个组件，如按钮组件Button、对话框组件Dialog、日期组件Calendar。开发者通过组合这些组件，最终得到功能丰富、可交互的页面。通过引入JSX语法，服用组件变得非常容易，同时也能保证组件结构清晰。有了组建这层抽象，React把代码和真实渲染目标隔离出来，除了可以在浏览器端渲染到DOM来开发网页外，还能用于开发原生移动应用。

和Angular、Ember等框架不同，React并不是完整的MVC/MVVM框架，他专注于提供清晰、简洁的View层解决方案。又与模板引擎不同，它不仅专注于解决View层的问题，又是一个包括View和Controller的库。对于复杂的应用，可以根据应用场景自行选择业务层框架，并根据需要搭配Flux、Redux来使用。在开发难度上，React不像其他框架那样提供了许多复杂的概念和繁琐的API，它是以Minimal APIInterface 为目标，只提供组件化相关的非常少量的API。同时为了保持灵活性，它没有自创一套规则，而是尽可能地让用户使用原生的javascript进行开发。所以只要熟悉原生javascript并了解相关概念后，就可以很容易上手React的开发。

2.6微信公众平台

微信公众平台是运营者通过公众号为微信用户提供资讯和服务的平台，而公众平台开发接口则是提供服务的基础，开发者在公众平台网站中创建公众号、获取接口权限后，可以通过微信公众开发平台接口文档来帮助开发[15]微信用户与任何一个公众号之间都会生成一个唯一的 OpenID 来识别微信用户，这样公众号就可以通过 OpenID 对微信用户进行管理，并与之交互。如果要在多个公众号或应用之间实现用户互通，则需要用到

UnionID，这时需要将这些公众号或应用绑定到同一个开放平台账号下，微信用户与开放平台账号之间会生成唯一的 UnionID[15]。当公众号调用接口时，开发者必须先获取

access\_token，它是调用接口的唯一凭据，可以通过 Get 请求来获取，access\_token

目前的有效期是 7200 秒，需要定时刷新重新获取[16]。

微信公众号提供服务的方式主要有两种，分别是公众号消息会话和公众号内网页。消息会话是公众号与微信用户进行交互的基础，消息会话主要包括四种类型，分别是群发消息、被动回复消息、客服消息和模板消息。而公众号内网页则用于实现比较复杂的业务场景，可以通过网页授权的方式获取用户基本信息，也可以通过微信 JS-SDK 在网页上使用微信原生功能，如微信支付、监听微信分享等功能[15]。

1. ：需求分析

3.1 系统功能性需求

前台微信端：商品查看：搜索商品、分类查看、查看推荐店铺+商品

购物车管理：加入购物车、删除购物车商品、清空购物车；

购买商品：下单、付款、取消订单、退款、收货、评价；

个人信息管理：查看和编辑个人信息（包括收货地址管理）；

后台店主端：创建店铺：开店申请；

商品管理：商品分类管理、库存管理、商品添加、提交审核、信息编辑、上架、下架；

管理店铺：查看和编辑店铺基本信息；

订单管理：接单、收款、发货；

收支统计：按日、月、季度统计店铺各类商品的成交量、营业额、毛利收入；

管理店铺公告

平台管理端：平台店铺管理：申请认证、敕令关闭店铺

店铺商品管理：商品申请审核

店铺数据统计：分布、

用户数据统计：分布

购物车、订单统计：推荐商品

3.2 系统非功能性需求

3.2.1 系统基本要求

1. 系统稳定性

要求系统软硬件整体及其功能模块具有稳定性。

2. 系统可靠性

要求系统数据维护、查询、分析、计算的准确性。

3. 易于维护性

要求系统的数据、业务以及终端应用的维护方便、快捷。

4. 可扩展性

要求系统从规模上、功能上易于扩展和升级，应制定可行的解决方案，预留相应的接口。除此之外，本系统建设还需要实现与其他系统的集成，能够从其他系统中提取相应专业数据，为本系统服务。

5. 兼容性

要求系统充分兼容主流及不同版本的浏览器和移动终端，保证系统在不同主流浏览器和移动终端上能正常使用。

6. 易用性

本系统界面布局、菜单及用户操作等方面的设计，要遵循界面友好、直观，菜单要简洁，菜单格式、快捷键等要充分考虑用户习惯。

7. 安全性

要求保障系统数据安全、不易被侵入、干扰、窃取信息或破坏。

3.2.2系统主要性能要求

1. 访问速度方面

简单类查询速度小于5秒，复杂类查询速度小于15秒。

2. 稳定性方面

在利用本系统正常的工作中，不应出现妨碍工作顺利进行的系统错误或意外中止的情况。

3. 资源利用方面

在进行数据的查询、统计、分析的过程中，允许CPU和内存的占用率提升及网络带宽占有量的加大，但在操作结束后，应该及时释放所占用的资源，以保证工作人员利用电脑顺利进行其它的工作。

4. 使用灵活性方面

当系统与其它软件的接口发生变化，用户的操作方式、运行环境、对信息获取方式以及对期望获取的信息结果发生变化时，软件系统要做到易于调整，拥有高度的灵活性。

3.3 可行性分析

3.3.1技术可行性

系统开发必须要有技术保障，比如计算机硬件技术、软件开发人员的职业素养以及用户管理使用能力。

1. 开发硬件的技术可行性通过市场调研，开发本系统的硬件技术是完全可行的。本系统对于计算机内存、外存、主频等都有一定要求，完全可以满足当前市场最先进的硬件水平，不会影响本系统的流畅运行。
2. 开发如那件的技术可行性当前市场比较主流的电商网站开发在实际应用中，大多使用分布式架构进行开发，本系统选用当前活跃程度比较高的分布式架构——Spring Cloud，它下面提供多个开源项目能够为分布式微服务应用提供比较完善的解决方案；同时采用与SpringCloud能够完美契合的SpringBoot框架来搭建微服务，在很大程度上能简化项目的搭建和开发。因为开发软件的技术也是可行的。

3.3.2 经济可行性

经济可行性研究是通过估算待开发软件的成本及进行成本效益分析，得出待开发软件的开发价值，是对待开发软件价值的一个综合评价。

硬件设备：需要阿里云服务器两台

软件：Linux操作系统

PhotoShop图像处理软件、 墨刀原型设计工具

IntelliJ IDEA后端开发工具、WebStorm前端开发工具

Mysql5.6数据库软件等

3.3.3 用户使用可行性

该系统的操作非常简单，所开发的软件系统对于一般操作手机和计算机的用户来说并不陌生。系统管理人员应当掌握相应的计算机专业知识。开展系统管理员的专业培训，提升职业素养，使其能够规范化、标准化的管理本系统。

1. ：系统设计

4.1 系统总体设计

系统总体设计阶段需要确定各个模块之间的相互关系，以及程序是由什么模块组成的。总体设计的目标是解决系统如何实现系统功能的问题。软件定义使其最重要的任务是对系统物理元素进行分配和对软件进行结构设计。进行系统总体设计的依据是系统分析产生的分析结果。分析结构是确定系统模块构成以及模块与系统结合模式的要素，是保证最优实现系统模块功能的重要保障。系统总体设计的最终目标是使系统完美实现预期功能的同时，满足高效、可靠、人性化和可修改特性，最大限度保证掌握容易和使用便捷。

本系统在设计时采用了微服务的设计原理，将系统所有功能模块按业务关联性划分成若干个微服务。目的是为了提高软件的可行性和可修改性，清晰的掌握软件结构，使软件测试和调试更加方便快捷。即把整个系统按照功能独立且相互关联的原则分解为一个个相对较小的模块，每个模块搭载一个特定子功能，然后再将若干个模块作为一个微服务进行单独开发，然后向外暴露接口，各个微服务之前通过API接口的相互调用组成整个系统。通过这种方式将纷乱复杂的问题简单化，化整为零，变向降低了问题的难度，提升了解决问题的效率，也就提高了系统开发效率，这种基于模块化和微服务的设计对软件开发的组织和管理更有益处。

4.1.1 系统定义

本系统有前台和后台两个功能，前台目前是接入到微信公众号内，通过移动端进行进入，前台的实现主要是对于普通用户来说，用户的注册登录，用户浏览商品、店铺，下单购买等等。而后台主分为两块，一个是在平台入驻的店主的管理端，该管理端的实现主要是对于在平台上开通店铺的店主来说，店主进行店铺的商品、订单、营业额进行管理；另一个是平台自身的管理员端，该管理端的实现主要是对平台的管理员来说，店铺开通申请审核、平台入驻店铺的管理、平台数据分析与统计。

4.1.2 系统功能描述与分析

本系统主要分成13个功能模块，如下图所示，分别为“会员管理”、“店铺管理”、“权限管理”、“商品管理”、“购物车管理”、“订单管理”、“商品分类管理”、“评价管理”、“公告管理”、“财务管理”、“系统用户管理”、“数据分析”、“数据统计”。

根据分布式系统设计原则，按照业务相关性分为“用户中心”、“商品和分类中心”、“订单和物流中心”、“通知和消息中心”、“推荐子系统”和“辅助决策子系统”等6个微服务。



4.2 数据库设计（ER图）

作为一种高效的数据管理技术，数据库技术在保持系统数据完整性和共享性

方面表现的尤为出色。它是基于文件系统的相关数据的集合，将相关数据有组织

的存储于一定结构内，并提供相应的访问机制，使这些数据的存储方式更趋于合

理，形成最小的数据冗余，提升了数据查询效率。

由于数据存储和数据库技术的应用是相互独立的，其中的数据管理提供了极

大的便利，更有利的应用编程和调试。数据库这种仓储式的数据管控模式在高效

实现数据共享的同时又能高度的维护数据一致性，并且自身就可以完成对数据的

增、删、改。数据是系统各类信息的载体，因此数据库的设计直接影响系统整体

开发过程、运行效率质量和维护成本，是系统设计的重要内容。

4.1.1 数据库设计概述

本系统中管理数据庞大、使用频次高且繁杂重复，数据管理难度较大。为满

足系统数据管理需求、正确反应用户现实环境、充分利用现行系统平台、降低数

据维护难度、提升数据维护管理效率，本系统采用了“数据库管理系统”。

数据库中根据系统实现功能不同，记录间的联系结构各有不同，这种数据结构又

被称为数据模型。数据库的功能核心是数据模型，怎样去实现不同的数据管

理模式呢？这就需要搭建不同的数据模型了，相应数据模型的搭建与完善过程就

是数据库的设计。目前数据库管理体系中，数据模型按其构成可分为关系、层次、

网状三种模式。三种模型各有特点，关系模型以其数据处理高端独立著称，基于

这种数据模型的数据库实现增删、修改、统计、显示和打印功能，使这些功能更

加方便，且有利于数据查询和快速定位，所以本系统设计使用关系型数据库。本

系统数据库设计说明如下：

（1）数据库设计的综合因素数据流程图。

（2）管理信息系统是数据库应用的扩展，对核心系统的设计是不是一个数

据库。

（3）本系统的数据库设计尽量满足 3NF（第三范式）的要求。

（4）提升系统运行效率是本系统数据库设计遵循的最基本原则。

一个好的数据库会减少其存储量，我们这里采用的是MySQL，数据有完整性和一致性， 支持Web 技术，系统的响应速度也非常的快，数据存储和维护更加直观。好的数据库 设计时表、减和字段个数都遵循量少原则，键越少表就越简单，数据库的设计就越完美。 既然我们知道好的数据库有多么重要，那么我们就要了解数据库是如何进行维护的，首 先结构要非常的合理，不止在数据库中，结构永远是一个整体的框架，是非常重要的。 数据库中数据的建立要有自己的独立性，这样避免了很多麻烦。当进行相关操作的时候 一定要速度快而且要准确。

E-R 图是直观表示概念模型的工具，它有三个基本成分：矩形框表示实体型

（框内注明实体名），菱形框架表示实体类型间的联系（菱形框内注明联系名），

椭圆框表示实体的属性（用于与相应的实体类型的连接线）。

4.1.2 系统总体E-R图

在建立数据库之前，首先需要分析系统包含的实体以及实体之间的外键约束关系。系统涉及到的主要实体有：会员、店主、店铺、商品、商品一级分类、商品二级分类、购物车、订单、评论、通知、消息、系统用户、公告等。

下面是系统中所有关键实体的E-R图：



4.1.3 系统关键表结构设计

商品这一实体相对其他实体比较复杂，属性极多，比如：商品名、价格、库存、销量、商品图片（各种尺寸）、商品参数、商品规格等许许多多属性。考虑到后面数据库操作的效率，我们将商品实体垂直拆分成下面6个实体，

商品主表：

商品banner表：

商品规格表：

商品参数表：

商品细节表：

商品扩展表：



4.1.4 关键数据表设计

（1）会员用户表：tb\_buyer\_user

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 长度 | 是否允许空 | 字段说明 |
| id | int | 11 | 否 | 会员用户ID |
| usercode | varchar | 100 | 否 | 会员用户编号 |
| nickname | varchar | 100 | 是 | 昵称 |
| header\_url | varchar | 200 | 是 | 头像 |
| name | varchar | 100 | 是 | 姓名 |
| phone | varchar | 20 | 是 | 手机号 |
| sex | int | 1 | 是 | 性别 |
| age | int | 10 | 是 | 年龄 |
| idcard\_no | varchar | 20 | 是 | 身份证号 |
| nation | varchar | 50 | 是 | 籍贯 |
| country | varchar | 50 | 是 | 国家 |
| province | varchar | 50 | 是 | 省份 |
| city | varchar | 50 | 是 | 城市 |
| district | varchar | 50 | 是 | 地区 |
| street | varchar | 50 | 是 | 街道 |
| specific\_address | varchar | 200 | 是 | 详细地址 |
| longitude | decimal | 12,6 | 是 | 所在经度 |
| latitude | decimal | 12,6 | 是 | 所在经度 |
| last\_login\_time | datetime |  | 是 | 最后一次登陆时间 |
| last\_login\_ip | varchar | 50 | 是 | 最后一次登录IP |
| state | int | 1 | 是 | 状态 |
| add\_time | datetime |  | 是 | 添加时间 |
| opt\_time | datetime |  | 是 | 操作时间 |
| opt\_usercode | varchar | 100 | 是 | 操作者 |

（2）用户总表：tb\_user

（3）店铺表：tb\_shop

（4）店主用户：tb\_shop\_user

（5）商品主表：tb\_product

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 长度 | 允许空 | 字段说明 |
| id | int | 11 | 否 | 商品主键 |
| product\_code | varchar | 100 | 否 | 商品编号 |
| name | varchar | 200 | 是 | 商品名称 |
| pic\_small\_url | varchar | 200 | 是 | 商品小图片 |
| pic\_middle\_url | varchar | 200 | 是 | 商品中等尺寸图片 |
| pic\_big\_url | varchar | 200 | 是 | 商品大图片 |
| product\_desc | varchar | 1000 | 是 | 商品描述 |
| state | int | 11 | 是 | 商品状态 |
| origin\_price | decimal | 10,2 | 是 | 原价 |
| sell\_price | decimal | 10,2 | 是 | 售价 |
| stock | int | 11 | 是 | 库存 |
| sale\_volume | int | 11 | 是 | 销量 |
| add\_time | datetime |  | 是 | 添加时间 |
| opt\_time | datetime |  | 是 | 操作时间 |
| opt\_usercode | varchar | 100 | 是 | 操作人 |

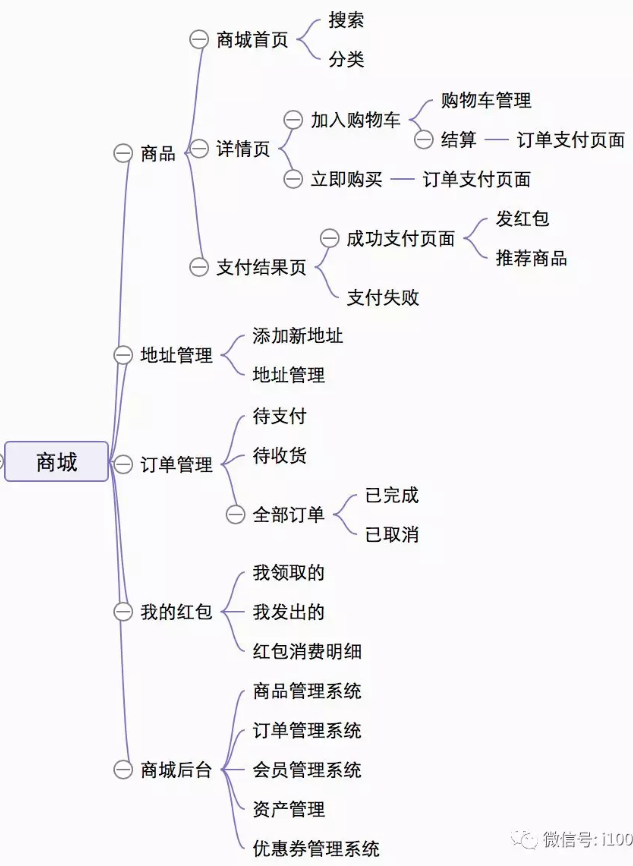
4.3 功能模块设计

（模块图为主）

会员模块、店铺模块、商品中心、订单模块、购物车模块、评价模块、物流中心

4.4 基于微型平台前台客户系统

商品展示、商品推荐系统、店铺推荐系统

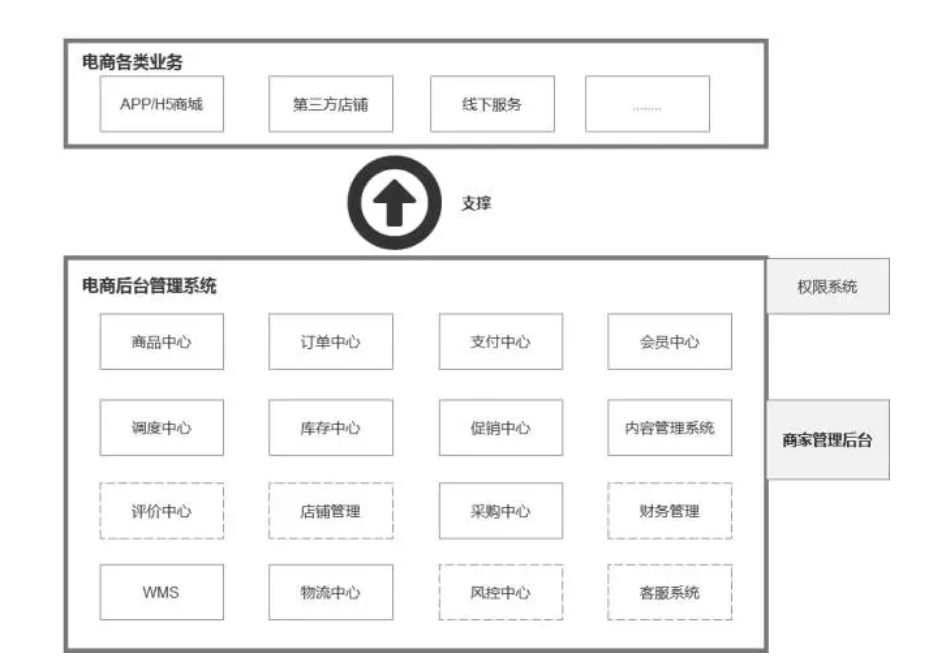


4.5 数据可视化的后台管理系统

（UML图为主）

会员管理、店铺管理、商品管理、订单管理、购物车管理、评价管理、物流中心

数据分析统计、可视化、货源地域分析





1. ：系统的实现与测试(实验设计与结果分析)。

5.1 系统详细设计

5.2 系统程序流程图

5.3 系统界面设计

5.4 系统测试

1. ：总结与展望。

6.1 工作总结

6.2 未来展望

# 参考文献

# 致谢