厦門大學

电子商城系统 oomall

必做+售后、服务模块概要设计说明书

组　名　　空白对照组

组　别　　2-4 组

学　院　　信息学院

专　业　　软件工程

成　员　　李嘉琪、高凯琪、郭宇阳、胡雨璇、黄勖

日　期　　2023 年 11 月 24 日

目录

1. 引言

1.1 编写目的

在进入 OOMALL 正式开发阶段后，我们将重心转向系统的详细设计，以明确系统的数据结构和软件结构。在需求分析的基础上，我们将致力于形成软件的整体概貌，重点解决实现系统需求的程序模块设计问题。这包括但不限于将系统划分为多个模块、确定各模块之间的接口、定义模块间传递的信息，以及进行数据结构和模块结构的设计。

通过分析和讨论，我们主要基于以下目的编写此说明书：

- 对 OOMALL 概要设计的阶段任务成果形成文档，以便阶段验收、评审和最终的文档验收；
- 对 OOMALL 需求阶段文档的再次确认过程，对前一阶段需求没有充分或错误的地方进行调整和修改；
- 明确整个系统的功能框架和数据库结构，为下一阶段的详细设计、编码和测试提供参考依据；
- 明确编码规范和命名规范，统一程序界面。

预期读者对象：概要设计评审小组、详细设计人员、开发人员、项目经理。

1.2 项目背景

本项目是厦门大学信息学院软件工程专业《软件工程》 《面向对象分析与设计》和《JavaEE 平台技术》三门课程的联合课程设计。2022 年秋季学期由软件工程专业 2020 级学生完成第一次迭代，2023 年秋季学期由软件工程专业 2021 级学生开始第二次迭代。

1.3 定义

- 本报告采用的术语符合国家标准《软件工程术语(GB/T11475-2006) 》 。
- DDoS 攻击：分布式拒绝服务攻击，利用客户或服务器技术，通过将多台计算机联合起来形成攻击平台，对一个或多个目标发动攻击，以成倍提高拒绝服务攻击的威力。DDoS 攻击通过大量合法请求占用网络资源，致力于瘫痪目标网络。
- 数据分片：一种将大型数据集分解成较小数据块并分散存储在多个节点上的技术。每个数据块分配到不同节点上，实现分布式存储和处理。
- 分布式数据库：一种数据库，其中数据存储在不同物理位置。数据可能存储在同一地点的多台计算机上，也可能分布在互连计算机网络中。
- SSL：安全套接层，以及其继任者传输层安全，用于为网络通信提供安全性和数据完整性的安全协议。
- TLS:传输层安全,用于在两个通信应用程序之间提供机密性和数据完整性。
- 访问令牌：是 Windows 操作系统安全性的概念。用户登录时，系统创建一个包含登录进程返回的 SID 以及本地安全策略分配给用户和用户安全组特权列表的访问令牌。
- 刷新令牌：用于获取访问令牌的凭据。授权服务器颁发刷新令牌给客户端，用于在当前访问令牌失效或过期时获取新的访问令牌，或者获取具有相同或更窄范围的附加访问令牌。

1.4 参考资料

- 《软件工程术语(GB/T11475-2006) 》
- 《需求规格说明书》
- 中国人民银行办公厅.
  关于进一步加强无证经验支付业务整治工作的通知. 银办发[2017]217 号文
- 中国人民银行.

关于规范支付创新业务通知. 银办发[2017]281 号文

- 中国人民银行. 关于印发 < 条码支付业务规范（试行）>的通知. 银办发[2017]296 号文
- 郑志成. 京东到家支付平台的高可用性架构计. https://www.zhihu.com/question/527868488/answer/2438919186
- 微信支付. https://pay.weixin.qq.com/wiki/doc/apiv3/apis/chapter8_1_1.shtml
- 支付宝互联网平台交易查询接口 https://opendocs.alipay.com/open/02e7gm?ref=api

2. 任务概述

2.1 目标

OOMALL 系统是一款综合性电商平台，旨在为用户提供便捷的在线购物体验。系统的主要功能包括顾客在线购买商品、商户管理销售商品、服务商提供服务以及平台管理员管理平台运营等多个方面。

2.2 运行环境

2.2.1 硬件平台

服务端：

- 服务器：华为云/耀云/服务集群

客户端：

- 主机：内存 256M 以上，显示分辨率 1920*1080 以上

2.2.2 软件环境

服务端：

- 操作系统：Ubuntu 22.04
- 支持环境：Docker Swarm，Tomcat 10.0.23
- 数据库：MySQL，MongoDB

客户端：

- 操作系统： Windows 7 及以上

2.3 需求概述

2.3.1 购买商品

1. 浏览商品：顾客可以通过搜索、分类查看等方式浏览商品；
2. 加入购物车：顾客可以将喜欢的商品加入购物车并管理购物车商品；
3. 提交订单：顾客可以在商品详情页直接下单商品，也可以在购物车中批量下单商品。

2.3.2 支付管理

1. 配置支付渠道：管理员会为平台配置一些支付渠道，顾客可以通过这些渠道支付；
2. 签约账户：商户可以选择平台已有的支付渠道并且与其签约，这将会作为顾客在本店铺支付时的支付渠道；
3. 支付：顾客确定要购买的商品后，可以选择支付渠道并支付自己的订单；
4. 退款：顾客申请退款并成功后，将会收到退款。
5. 分账：由于顾客支付后的金额会分摊给商户和平台，因此在支付后需要分账；在需要退款时，也要通过分账确定商户和平台各要退款多少；

2.3.3 服务业务

1. 发布服务：服务商在平台上发布售前、售后服务，当服务经过平台管理人员审核后，商户可以为他们的商品绑定这些服务；
2. 选择服务：商户可以为他们的商品绑定服务，以便用户在购买后能够享受到售前、售后服务；
3. 审核服务：当服务商发布服务后，平台管理人员会审核服务的描述详情是否合法、合规；

4. 接受服务单：服务商可以从服务单列表中选择服务单接受，以为其进行服务；
5. 撤销服务单：当服务商判断当前服务单无法完成或者不符合服务单条件时，服务商可以在平台上取消已接受的服务单；
6. 完成服务单：当服务商完成当前服务单的服务内容时，服务商可以在平台上完成服务单。

2.3.4 售后业务

1. 售后申请：顾客可以对相关订单提出售后申请，供商户进行审核；
2. 审核售后：商户可以对顾客提出的售后申请进行审核，判断售后是否合理，将审核结果反馈给顾客；
3. 取消售后：商户如果拒绝顾客提出的售后申请，商户可以取消相关售后，并将拒绝理由反馈给顾客；
4. 售后收件：退货或换货收取原件时需要进行检查，如果原件检查通过，则正常退款或更换新货，若原件检查失败将原件返回；
5. 完成售后：如果售后类型为服务类型，待服务完成之后需要结束本次售后，如果售后类型为退货，待退款完成之后结束本次售后，如果售后为换货售后，在换货订单产生时售后也需结束。

2.3.5 仲裁业务

1. 仲裁申请：顾客可以对不满意的售后提出仲裁申请，等待管理员后续的审核；
2. 审核仲裁：管理员需要对顾客的仲裁申请审核是否同意顾客的申请；
3. 应诉纠纷：顾客的仲裁申请通过审核后，商户需要针对相关的售后仲裁提出自己的理由应诉；
4. 仲裁结案：管理员最后需要再次考虑顾客和商户的理由，给出最后的仲裁结案结果。

2.4 条件与限制

- **技术限制：**系统设计采用的是 Docker 容器化技术，以实现负载均衡。容器编排工具将被配置，确保容器的自动伸缩、网络的安全性以及对持久化存储的有效管理。为了确保系统的高效、安全和可扩展的运行，监控系统将被集成，以实时监测系统各个方面的运行情况。

- **资源限制：**后端部分将受到严格的资源约束。系统将运行在最多三台华为云耀服务器上，每台服务器的配置为 2 核 CPU、2GB 内存、40GB 存储。在开发和测试阶段，每个服务器的每月流量包限制在 400GB 以内，以控制资源使用的成本和合理分配流量。

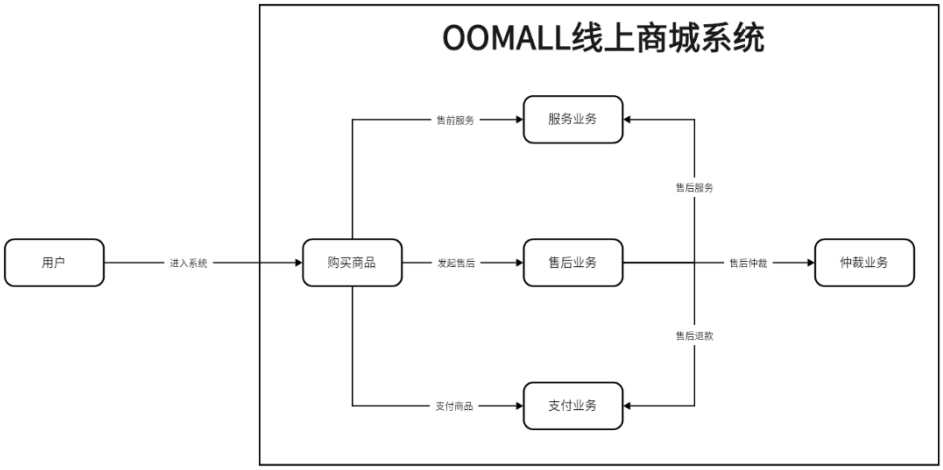
- **安全限制：**要求使用网关实施访问控制，确保不同角色仅能访问其授权的资源。

- **可扩展性限制：**由于采用模块化的设计原则，将整个系统划分为独立的模块或组件。每个模块都应当能够独立进行开发、测试、部署和扩展，从而实现系统的高度可维护性和可扩展性。

- **性能和效率限制：**系统应当能够支持大规模的并发用户。系统设计要求能够在高流量时同时处理数千个用户的访问请求，确保系统能够在高负载情况下依然保持高效、稳定的运行状态。性能测试和优化将是系统设计的重要组成部分，以满足用户的使用需求。

3. 总体设计

3.1 处理流程



OOMALL 系统处理流程图

通过上图中呈现的流程图，我们详细描述了用户在 OOMALL 系统中的处理流程。从用户成功登录 OOMALL 系统开始，用户将正式进入系统的处理流程，同时在系统中生成了持久化的数据记录。

在浏览商品阶段，用户可以选择购买商户上架的商品，随后进入支付管理模块完成支付操作。一旦顾客成功购买商品，商户将通过物流系统将商品送达顾客手中。对于提供售前服务的商品，服务提供商将在购买前为顾客提供相应的服务。

在顾客收到商品后，如遇到问题，用户可以启动售后服务流程，并与商户进行协商。这一过程涉及服务提供、退款等操作，为了保证公平，用户对于不满意的售后结果可以申请仲裁，由平台管理员进行仲裁决定。这一仲裁过程旨在解决争议，确保系统运作的公正性和顾客满意度。

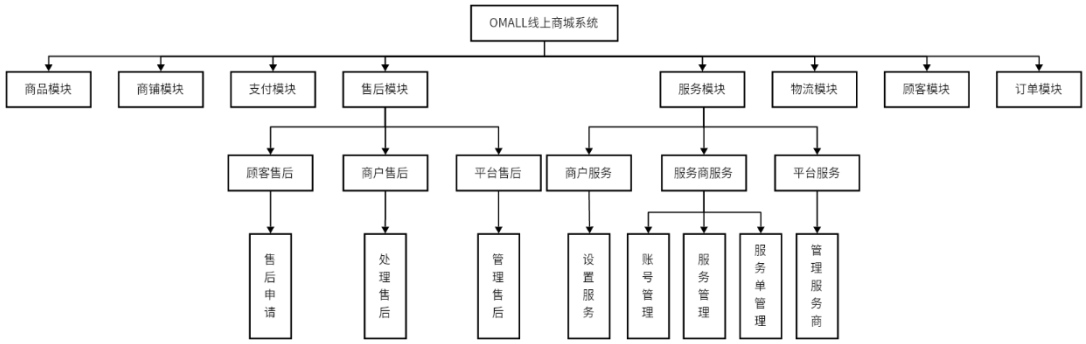
这一系统设计考虑了用户交互和数据持久化，以确保整个流程的顺畅性和可追踪性。每个步骤都有相应的模块和数据记录，使得 OOMALL 系统能够高效、透明地处理用户与商户之间的交互与交易。

3.2 总体结构和模块外部设计

3.2.1 总体结构

在软件设计中，模块扮演着至关重要的角色，因为软件结构的质量在很大程度上由模块的属性所决定。将软件进行模块化的目的在于降低软件的复杂性，从而使得软件设计、测试、调试和维护等工作变得更加简易和可管理。

为了确定 OOMALL 系统的功能和模块结构，根据上述的购买商品、支付管理、服务、售后、仲裁五个业务和 OOMALL 系统的处理流程，我们将 OOMALL 系统划分为了商品模块、商铺模块、支付模块、物流模块、顾客模块、订单模块、售后模块、服务模块八个模块来进行总体结构设计。



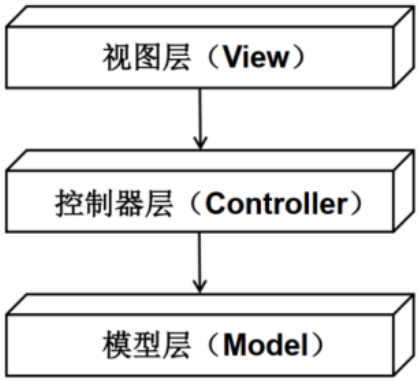
OOMALL 系统结构图

这里我们利用系统结构图来刻画 OOMALL 系统的总体主要介绍小组负责的售后模块和服务模块。OOMALL 系统的系统结构图明确刻画了售后模块和服务模块，这两个模块是我们小组负责的内核领域。在售后模块中，顾客、商户和平台管理员共同参与售后申请、处理和管理工作。与此同时，在服务模块中，服务商、商户和平台管理员分别承担账号管理、服务管理、服务单管理、服务设置和服务商管理等职责。这一结构的清晰划分为系统提供高效、可维护的售后和服务功能奠定了坚实的基础。

对于每一个模块内部，项目使用了层次体系结构，使用 MVC 的方式进行设计，将模块内部在总体上划分为模型、视图、控制器三个层次，分别用于实现不同的功能:

- 模型(Model)：包含所有应用特定内容和处理逻辑

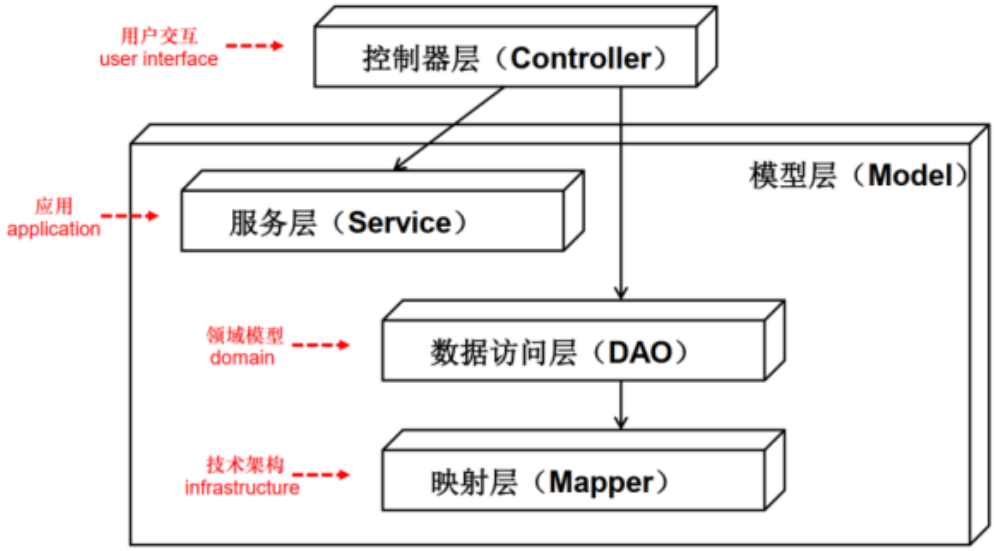
- 视图(View)：包含所有接口特定功能并能够显示终端用户所需的内容和操作逻辑
- 控制器(Controller)：管理对模型和视图的访问并协调它们之间的数据流



MVC 层次体系结构

结合后端所使用的 JavaEE 和 Spring 相关技术，将模型层内部再一次细化为 Service 层、DAO 层和 Mapper 层，各层的主要功能如下：

- Service 层（服务层）：Service 层负责封装业务逻辑和处理业务相关的操作。接收来自控制器层的请求，并调用合适的 DAO 层方法来获取数据，进行业务处理和转换。Service 层包含事务管理、权限校验、数据验证等功能，是控制器层与其他层之间的桥梁，负责协调不同的模块和组件。
- DAO 层（数据访问对象层）：DAO 层是与数据存储交互的接口层，负责封装对数据库或其他数据存储的访问操作。提供了一系列的接口和方法，用于数据的增删改查操作。DAO 层设计采用面向对象的方式，将数据操作封装为对象，使得业务逻辑层可以通过调用这些对象来进行数据的访问和处理。
- Mapper 层（映射器层）：Mapper 层是 DAO 层的具体实现，通过配置文件或注解的方式，将数据查询和持久化操作与数据库进行映射（DAO 层是对象模型，而使用 MySQL 数据库是关系数据库，需要提供对应的转换映射），此外 Mapper 层使用查询语言（如 SQL 等）来定义数据操作的细节，并提供与数据库交互的底层方法。



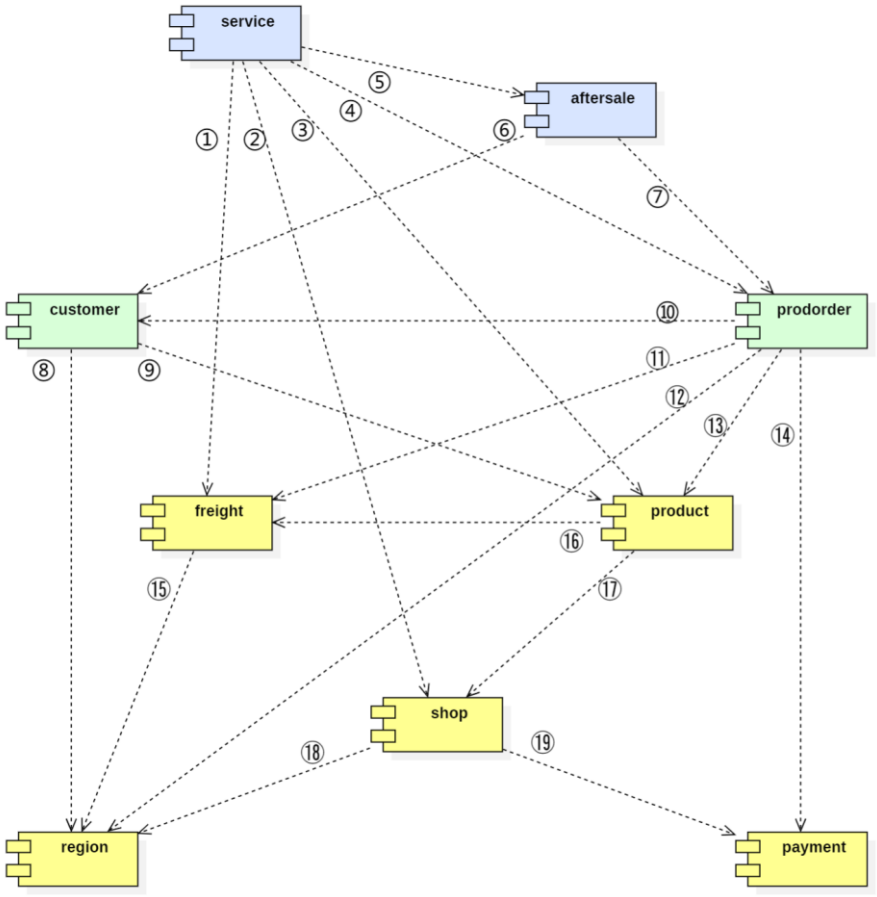
基于 SpringMVC 的模型层细化

3.2.1 模块外部设计

根据对 OOMALL 处理流程的分析和总体结构的整理，我们计划将 OOMALL 系统划分为八个模块，以便进行详细的分模块分析与设计。首要考虑的是模块之间的外部设计，以确保这八个模块之间的关系能够在后续的具体开发中进行有效的功能调用。

这个外部设计的重点在于明确每个模块的职责和功能，并创建它们之间清晰的接口和交互机制。通过合理的模块划分，我们能够实现模块的独立开发、测试和部署，同时确保它们能够协同工作，构建起一个完整、高效的系统。

在这里我们针对模块外部的设计分析八个模块互相之间的关系是怎样的，便于我们后续详细开发时的功能调用。



① 服务模块调用物流模块：服务中的寄修服务会涉及物流业务；

② 服务模块调用商铺模块：商铺要为每个商品设置服务，服务单中需要绑定商铺信息；

③ 服务模块调用商品模块：商铺要为每个商品设置服务，服务单中也需要绑定商品信息；

④ 服务模块调用订单模块：顾客为订单申请服务，服务单需要绑定订单信息；

⑤ 服务模块调用售后模块：售后服务通过售后业务申请，需要绑定售后单信息；

⑥ 售后模块调用顾客模块：售后和仲裁需要绑定申请顾客的信息；

⑦ 售后模块调用订单模块：售后单需要绑定订单信息；

⑧ 顾客模块调用地区模块：顾客需要按照地区设定收货地址；

⑨ 顾客模块调用商品模块：顾客需要浏览商品，查看商品详情；

⑩ 订单模块调用顾客模块：订单需要绑定下单顾客信息；

11 订单模块调用物流模块：订单会涉及物流业务，如物流轨迹查询等；

12 订单模块调用地区模块：订单需要绑定收货地区及发货地区信息；

13 订单模块调用商品模块：订单需要绑定购买的商品项；

14 订单模块调用支付模块：订单的创建需要调用支付模块的支付功能，订单的退款需要调用支付模块的退款功能；

15 物流模块调用地区模块：物流设定在不同地区，物流运费计算也依赖于地区；

16 商品模块调用物流模块：商品需要绑定运费计算模板；

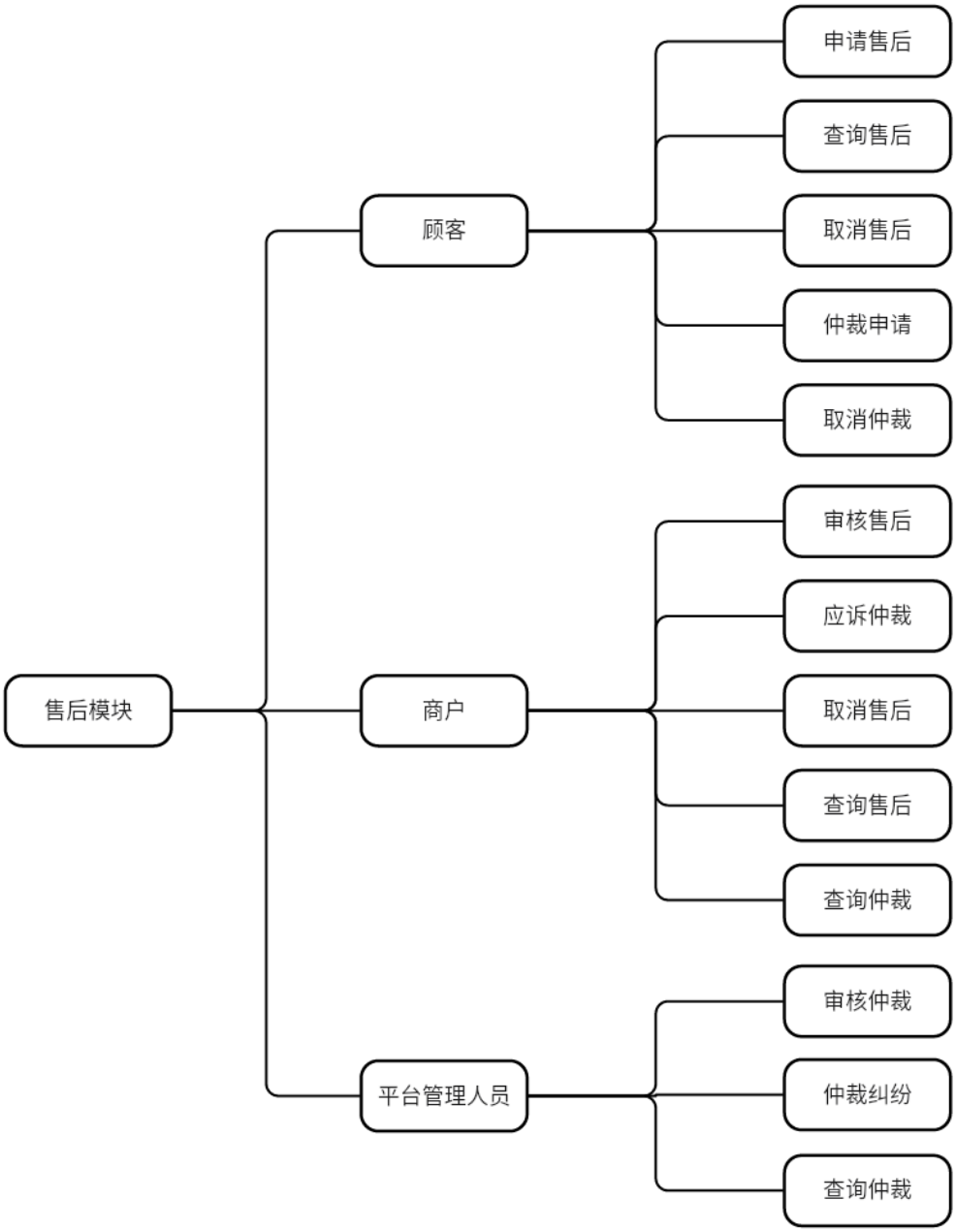
17 商品模块调用商铺模块：商品需要绑定商铺信息；

18 商铺模块调用地区模块：商铺模块在新增仓库和选择服务时需要用到地区；

19 商铺模块调用支付模块：商铺的分账等业务会设计到支付模块；

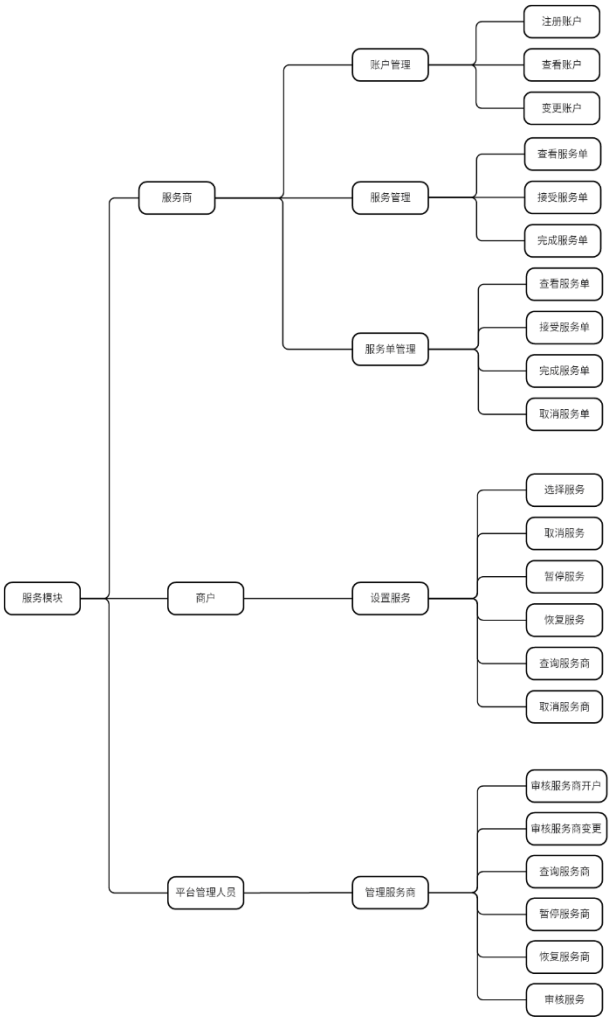
3.3 功能分配

- 商品模块：商品的新增、修改、审核和查看，销售活动的新增、修改、审核和删除；
- 商铺模块：管理商品，管理商铺账号，管理账户，设置物流模板功能；
- 支付模块：调用微信、支付宝等支付平台 API 完成付款、退款等功能，平台、商户分账功能；
- 物流模块：商户管理仓库和管理物流部分的功能；
- 顾客模块：顾客管理个人信息和地址、管理购物车和优惠券，以及平台管理员管理顾客；
- 订单模块：顾客提交、修改、取消以及查看订单，商铺发货，商铺接受、取消以及查看订单；
- 售后模块：顾客申请、查询及取消售后，顾客申请、取消仲裁，商户审核、查询及取消售后，商户应诉、查询仲裁，平台管理人员审核仲裁及处理纠纷；



售后模块系统结构图

- 服务模块：服务商管理账户、服务和服务单，商户为商品设置服务，平台管理员管理服务商


服务模块
服务商
账户管理
注册账户
查看账户
变更账户
服务管理
查看服务单
接受服务单
完成服务单
服务单管理
查看服务单
接受服务单
完成服务单
取消服务单
商户
设置服务
选择服务
取消服务
暂停服务
恢复服务
查询服务商
取消服务商
平台管理人员
管理服务商
审核服务商开户
审核服务商变更
查询服务商
暂停服务商
恢复服务商
审核服务

服务模块功能结构图

4. 接口设计

4.1 外部接口

4.1.1 用户界面

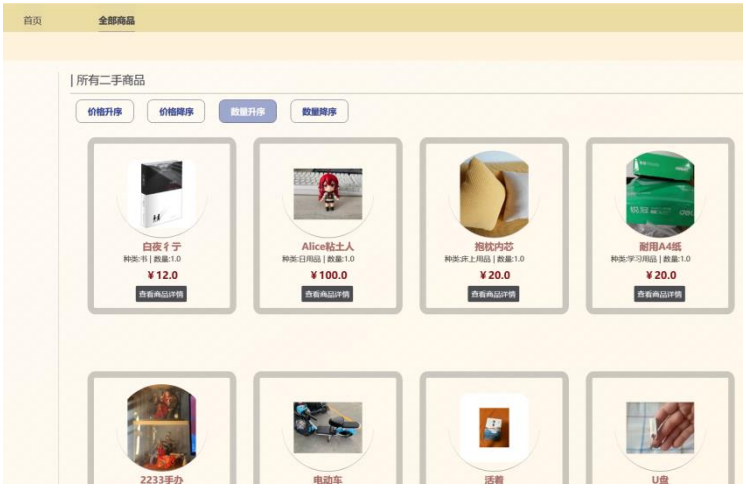
在用户界面设计方面，根据需求分析和总体设计的结果，OOMALL 系统的用户界面将遵循 B/S 结构，可通过浏览器端在移动设备和电脑网页上使用。为了提供符合要求的用户体验，我们采用了传统的 HTML＋CSS 前端界面框架。相较于其他框架，该框架的优势在于只需掌握基础的 HTML 知识即可着手前端开发，无需深入研究，而后续我们将探索更优化的开发方式。

OOMALL 用户界面主要划分为三个模块：主界面、个人界面和管理员界面。主界面负责商品及其信息的展示，使顾客能够轻松浏览商品并进行购买。个人界面则专注于顾客、商户和服务商的功能和服务管理，提供个性化的用户体验。最后，管理员界面致力于执行管理员功能，确保系统的平稳运行。通过这样的模块划分，我们能够实现清晰的界面结构，使各用户群体能够便捷地使用系统的不同功能。

4.1.1.1 主界面

主界面-主页面

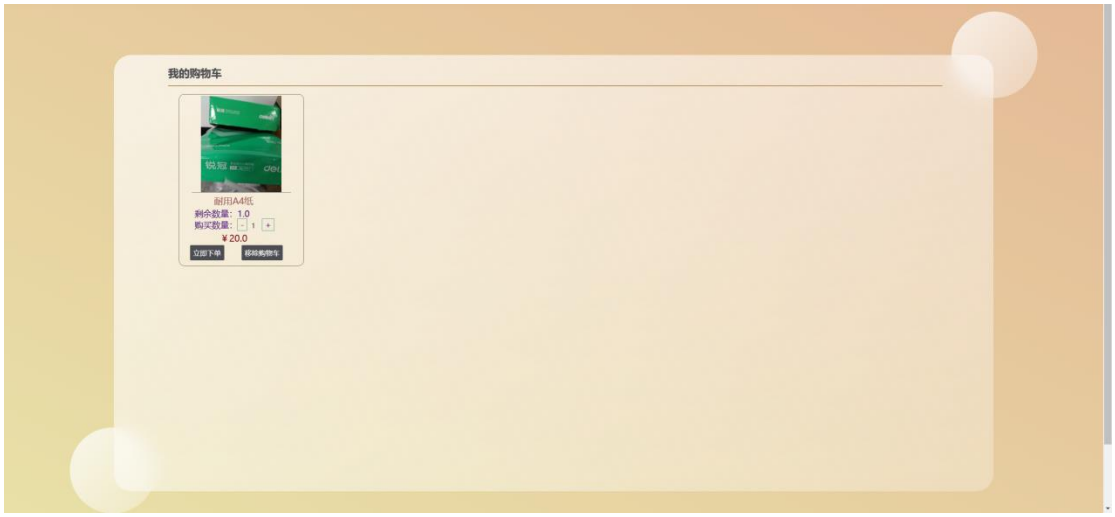
主界面-登录界面



主界面-商品界面

4.1.1.2 个人界面

个人界面-用户个人信息



个人界面-顾客购物车

个人界面-顾客订单管理

个人界面-商户上架商品

个人界面-商户管理订单

4.1.1.3 管理员界面

管理员界面-用户管理

4.1.2 软件接口

4.1.2.1 Web 服务接口

确保系统能够与 Web 服务进行交互，提供网站服务：

- 接口类型： RESTful API
- 协议： HTTPS
- 身份验证： OAuth 2.0

4.1.2.2 支付网关接口

确保系统与支付网关进行集成，以处理支付请求：

- 接口类型： RESTful API
- 协议： HTTPS
- 身份验证： API 密钥

4.1.2.3 物流服务接口

确保系统与物流服务进行交互，以处理订单配送：

- 接口类型： SOAP
- 协议： HTTPS
- 身份验证： API 密钥

4.1.2.4 用户认证接口

确保系统能够与用户认证服务进行集成，以验证用户身份：

- 接口类型： OAuth 2.0
- 协议： HTTPS
- 身份验证： 访问令牌 token

4.1.2.5 消息队列接口

确保系统能够与消息队列服务进行交互，以实现异步处理：

- 接口类型： AMQP
- 协议： RabbitMQ
- 身份验证： 用户名密码

4.1.2.6 数据库接口

确保系统与不同类型的数据库进行交互，包括 MySQL、MongoDB、Redis 等：

- MySQL 数据库接口：
  - 接口类型： JDBC
  - 协议： MySQL 协议

- 身份验证： 用户名密码

- MongoDB 数据库接口：
  - 接口类型： MongoDB 驱动程序
  - 协议： MongoDB 协议
  - 身份验证： 用户名密码
- Redis 缓存接口：
  - 接口类型： Redis 客户端
  - 协议： Redis 协议
  - 身份验证： 无

4.1.3 硬件接口

无

4.2 内部接口

4.2.1 物流模块

接口名称	输入	输出	接口描述
商户创建运单	商铺 ID，运单信息	运单号	商户创建一个物流单（内部接口）
商户或平台管理员获得运单信息	商铺 ID，运单号	运单信息	商户或平台管理员查看指定单号的物流单详情（内部接口）
顾客获得运单信息	顾客 ID，运单号	运单信息	顾客查看指定单号的物流单详情（内部接口）
商户验收快递	商铺 ID，运单号，验收状态	操作是否成功	商户验收一个快递，并反馈验收状态
商户取消运单	商铺 ID，运单号	操作是否成功	商户取消指定的物流单
商户新增物流合作	商铺 ID，物流渠道信息	操作是否成功、商铺物流信息	商户新增一个物流合作
商户获得物流合作信息	商铺 ID	该商铺的物流合作信息列表	商户获得它拥有的物流合作信息
商户更新物流合作信息	商铺 ID，商铺物流 ID，更新的物流合作信息	操作是否成功	商户更新一个物流合作
商户停用物流合作	商铺 ID，商铺物流 id	操作是否成功	商户停用一个物流合作
商户恢复物流合作	商铺 ID，商铺物流 id	操作是否成功	商户恢复一个物流合作
查询物流公司	用户 ID，运单 ID	运单信息	根据指定的运单号查询该运单属于哪家物流公司

4.2.2 支付模块

GET	/channels	顾客获得有效的支付渠道
PUT	/shops/{shopId}/channels/{id}/valid	有效平台支付渠道
PUT	/shops/{shopId}/channels/{id}/invalid	无效平台支付渠道
GET	/shops/{shopId}/shopchannels	获得商铺的所有支付渠道
POST	/shops/{shopId}/channels/{id}/shopchannels	签约支付渠道
GET	/shops/{shopId}/shopchannels/{id}	获得商铺支付渠道
DELETE	/shops/{shopId}/shopchannels/{id}	解约店铺的账户
PUT	/shops/{shopId}/shopchannels/{id}	修改支付渠道
PUT	/shops/{shopId}/shopchannels/{id}/valid	有效支付渠道
PUT	/shops/{shopId}/shopchannels/{id}/invalid	无效支付渠道
GET	/shops/{shopId}/shopchannels/{id}/payments	查询支付信息
GET	/shops/{shopId}/payments/{id}/refunds	管理员查询支付单的退款信息
GET	/shops/{shopId}/refunds/{id}	管理员查询退款信息
GET	/shops/{shopId}/shopchannels/{id}/divrefundtrans	商铺查询自己的退款分账信息
GET	/shops/{shopId}/shopchannels/{id}/divpaytrans	商铺查询自己的分账信息

GET /shops/{shopId}/ledgers 商铺查询自己的对账信息

GET /shops/{shopId}/ledgers/{id} 商铺查询自己的分账信息

PUT /shops/{shopId}/ledgers/{id} 调账

POST /notify/payments/alipay 支付宝支付回调

POST /notify/payments/wepay 微信支付回调

POST /internal/payments 买家创建支付单

PUT /internal/payments/div 分账

POST /internal/shops/{shopId}/payments/{id}/refunds 管理员创建退款信息，需检查Payment是否是此商铺的payment

PUT /internal/shops/{shopId}/ledgers/check 对账

GET /internal/shops/{shopId}/payments/{id} 查询支付信息

DELETE /internal/shops/{shopId}/payments/{id} 取消支付

GET /internal/shops/{shopId}/refunds/{id} 查询退款信息

DELETE /internal/shops/{shopId}/refunds/{id} 取消退款

接口名称	输入	输出	接口描述
顾客获得有效的支付渠道	顾客 ID	支付渠道信息列表	顾客获得当前有效的平台支付渠道或商铺支持的有效支付渠道、综合商铺的 shopChannel 和平台的 channel 的结果
管理员恢复平台支付渠道	商铺 ID（只能为 0），支付渠道 ID	操作是否成功	平台管理员恢复一个平台支付渠道，使其从无效变为有效
管理员暂停平台支付渠道	商铺 ID（只能为 0），支付渠道 ID	操作是否成功	平台管理员暂停一个平台支付渠道，使其从有效变为无效
商户获得支付渠道	商铺 ID	支付渠道信息列表	商户获得商铺的所有支付渠道，包括有效和无效的
商户签约支付渠道	商铺 ID，支付渠道 ID，支付信息	操作是否成功、商铺支付渠道信息	商户签约一个支付渠道（刚签约时为无效态）
商户查询支付渠道	商铺 ID，支付渠道 ID	商铺支付渠道信息	商户查询指定渠道号的商铺支付渠道信息

商户解约支付渠道	商铺 ID，支付渠道 ID	操作是否成功	商户节约一个支付渠道（无效态时才能解约）
商户修改支付渠道	商铺 ID，支付渠道 ID，修改的支付信息	操作是否成功	商户修改一个支付渠道
商户恢复平台支付渠道	商铺 ID，支付渠道 ID	操作是否成功	商户恢复一个商铺支付渠道，使其从无效变为有效
商户暂停平台支付渠道	商铺 ID，支付渠道 ID	操作是否成功	商户暂停一个商铺支付渠道，使其从有效变为无效
查询支付单信息	商铺 ID，支付渠道 ID，渠道交易 ID，起始和结束时间	支付单信息列表	查询支付单信息
管理员查询支付单的退款信息	商铺 ID（只能为 0），支付单 ID	支付单退款信息列表	管理员查询指定支付单号的平台支付单的退款信息
管理员查询退款信息	商铺 ID（只能为 0），退款单 ID	退款单信息	管理员查询指定退款单号的退款单信息
商户查询退款分账信息	商铺 ID，商铺支付渠道 ID，起始和结束时间	退款分账信息列表	商铺查询自己的退款分账信息
商户获得分账信息	商铺 ID，商铺支付渠道 ID，起始和结束时间	分账信息列表	商铺查询自己的分账信息
商户查询对账信息	商铺 ID，交易类型，商铺支付渠道 ID，起始和结束时间	对账信息列表	商铺查询自己的对账信息
商户查询分账信息	商铺 ID，台账 ID	分账信息	商铺查询指定台账 id 分账信息
商户调帐	商铺 ID，台账 ID	操作是否成功	商户调帐
支付宝支付回调	支付宝支付信息	操作是否成功	支付宝支付回调
微信支付回调	微信支付信息	操作是否成功	微信支付回调
顾客创建支付单	顾客 ID，支付单信息	支付单 ID、预支付 ID	顾客创建一个支付单
顾客取消支付	顾客 ID，支付单 ID	操作是否成功	顾客取消指定单号的支付
分账	支付渠道 ID，起始和结束时间	操作是否成功	对指定支付渠道的指定时间段内的所有账单进行分账操作
管理员创建退款信息	商铺 ID，支付 ID	操作是否成功	对指定账单进行退款处理，同时对相应店铺的分账进行退款处理

对账	商铺 ID，起始和结束时间	操作是否成功	对指定支付渠道的指定时间段内的所有账单进行对账操作
商户查询支付信息	商铺 ID，支付单 ID	支付单信息	商户查询指定单号的支付单信息
商户取消支付	商铺 ID，支付单 ID	操作是否成功	商户取消指定单号的支付单
商户查询退款信息	商铺 ID，退款单 ID	退款单信息	商户查询指定单号的退款单信息
商户取消退款	商铺 ID，退款单 ID	操作是否成功	商户取消指定单号的退款单

4.2.3 地区模块

接口名称	输入	输出	接口描述
获得地区的所有状态	无	地区状态信息	获得地区的所有状态
管理员新增子地区	商铺 ID（只能为 0），地区 ID，子地区信息	子地区 ID	管理员在指定 ID 的地区下新增子地区

管理员查询地区的下级地区	商铺 ID（只能为 0），地区 ID	子地区信息列表	管理员查询该地区的下级地区
顾客查询地区的下级地区	地区 ID	子地区信息列表	顾客查询该地区的下级地区
顾客查询地区	地区 ID	地区信息	顾客查询指定 ID 地区的信息
查询地区的上级地区	地区 ID	父地区 ID	查询该地区的上一级地区
管理员修改地区	商铺 ID（只能为 0），地区 ID，修改的地区信息	操作是否成功	管理员修改某个地区
管理员废弃地区	商铺 ID（只能为 0），地区 ID	操作是否成功	管理员废弃某个地区，下级地区一并废弃
管理员停用地区	商铺 ID（只能为 0），地区 ID	操作是否成功	管理员停用某个地区，下级地区一并停用
管理员恢复地区	商铺 ID（只能为 0），地区 ID	操作是否成功	管理员恢复某个地区，下级地区一并恢复

4.2.4 商铺模块

接口名称	输入	输出	接口描述
获得商铺的所有状态	无	商铺的所有状态	获得商铺的所有状态
用户申请商铺	商铺名称、类型、免邮金额	商铺 ID	用户申请商铺，一个用户只能开一家商铺，申请的商铺在审核态
顾客查询商铺信息	商铺类型、名称	商铺信息列表	顾客查询店铺信息，只返回上线和下线状态的商铺
商户修改商铺信息	商铺 ID、修改的商铺信息	操作是否成功	商户修改其商铺信息
平台管理员或商户关闭商铺	商铺 ID	操作是否成功	只有下线的商铺才能关闭
获得商铺信息	商铺 ID	商铺信息	商户获得其商铺信息，需检查 shopId 是否与访问者的一致
管理员查询店铺信息	商铺 ID（只能为 0）、类型、状态、名称	商铺信息列表	管理员查询店铺信息，会返回所有状态的商铺

管理员审核店铺信息	商铺 ID（只能为 0），审核结果	操作是否成功	管理员审核店铺信息
管理员上线店铺	商铺 ID（只能为 0）	操作是否成功	管理员上线店铺
管理员下线店铺	商铺 ID（只能为 0）	操作是否成功	管理员下线店铺
管理员定义运费模板	商铺 ID（只能为 0），运费模板资料	运费模板 ID	管理员定义运费模板
获得运费模板	商铺 ID，模板名称	运费模板信息列表	获得指定名称的运费模板
管理员克隆运费模板	商铺 ID（只能为 0），模板 ID	操作是否成功	管理员克隆运费模板
获得运费模板详情	商铺 ID，模板 ID	运费模板信息	获得指定模板 ID 的运费模板
管理员修改运费模板	商铺 ID（只能为 0），模板 ID，运费模板资料	操作是否成功	管理员修改指定模板 ID 的运费模板
删除运费模板	商铺 ID，模板 ID	操作是否成功	管理员删除 ID 的运费模板，同步删除与商品的关系
管理员定义重量模板明细	商铺 ID，运费模板 ID，地区 ID，重量模板资料	操作是否成功	管理员定义重量模板明细
管理员修改重量模板明细	商铺 ID，运费模板 ID，地区 ID，重量模板资料	操作是否成功	管理员修改重量模板明细
管理员删除地区模板	商铺 ID，运费模板 ID，地区 ID	操作是否成功	管理员删除地区模板
管理员定义件数模板明细	商铺 ID，运费模板 ID，地区 ID，件数模板资料	操作是否成功	管理员定义件数模板明细
管理员修改件数模板明细	商铺 ID，运费模板 ID，地区 ID，件数模板资料	操作是否成功	管理员修改件数模板明细
计算一批商品的运费	订单商品的订货详情	运费价格	为商品计算运费

4.2.5 商品模块

GET	/products/states	获得货品的所有状态
GET	/categories/{id}/subcategories	查询商品分类关系
GET	/shops/{shopId}/orphoncategories	查询没有一级分类的二级分类
POST	/shops/{shopId}/categories/{id}/subcategories	管理员新增商品类目
PUT	/shops/{shopId}/categories/{id}	管理员修改商品类目信息
DELETE	/shops/{shopId}/categories/{id}	管理员删除商品类目信息
GET	/products	查询正式商品
GET	/products/{id}	获得Product有效详细信息
GET	/onsales/{id}	获得Product的指定onsale历史信息
POST	/shops/{shopId}/draftproducts	商铺管理员申请增加新的Product
GET	/shops/{shopId}/draftproducts	店家查看草稿货品
DELETE	/shops/{shopId}/draftproducts/{id}	管理员或店家物理删除审核中的Products
PUT	/shops/{shopId}/draftproducts/{id}	店家修改审核态货品信息
GET	/shops/{shopId}/draftproducts/{id}	店家查看草稿货品信息详情
GET	/shops/{shopId}/products/{id}	店家查看货品信息详情
PUT	/shops/{shopId}/products/{id}	店家修改货品信息
POST	/shops/{shopId}/products/{id}/relations	店家将两个商品设置为相关
DELETE	/shops/{shopId}/products/{id}/relations	店家取消商品的相关

PUT /shops/{shopId}/draftproducts/{id}/publish 货品发布

PUT /shops/{shopId}/products/{id}/allow 平台管理员解禁商品

PUT /shops/{shopId}/products/{id}/prohibit 平台管理员禁售商品

POST /shops/{shopId}/products/{id}/onsales 管理员上架商品销售

GET /shops/{shopId}/products/{id}/onsales 管理员查询特定商品的销售

GET /shops/{shopId}/onsales/{id} 管理员查询特定价格浮动的详情

DELETE /shops/{shopId}/onsales/{id} 删除销售

PUT /shops/{shopId}/onsales/{id}/cancel 取消销售

GET /groupons 查询所有上线态的团购活动

GET /groupons/{id} 查看团购活动详情

GET /shops/{shopId}/groupons 管理员查询商铺的所有状态团购

POST /shops/{shopId}/onsales/{pid}/groupons 管理员新增团购活动

GET /shops/{shopId}/groupons/{id} 管理员查看特定团购详情

PUT /shops/{shopId}/groupons/{id} 管理员修改团购活动

DELETE /shops/{shopId}/groupons/{id} 管理员终止团购活动

POST /shops/{shopId}/couponactivities 管理员新建己方优惠活动

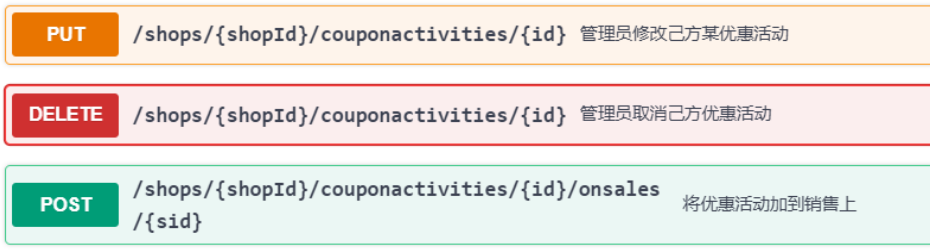
GET /shops/{shopId}/couponactivities 查看店铺的优惠活动列表

POST /couponactivities/{id}/caculate 计算优惠

GET /couponactivities 查看所有的上线优惠活动列表

GET /activities/{id}/onsales 查看活动中的商品

GET /shops/{shopId}/couponactivities/{id} 查看优惠活动详情



接口名称	输入	输出	接口描述
获得货品的所有状态	无	货品的所有状态	获得货品的所有状态
查询商品分类关系	种类 ID	分类信息	根据分类 ID 获取商品下一级分类信息
查询没有一级分类的二级分类	无	分类信息	查询没有一级分类的二级分类
管理员新增商品类目	种类 ID，类目详细信息，商铺 ID（只能为 0）	操作是否成功	管理员新增商品类目某些分类不允许增加新的下级分类
管理员修改商品类目信息	种类 ID，类目详细信息，商铺 ID（只能为 0）	操作是否成功	管理员修改商品类目信息
管理员删除商品类目信息	种类 ID，商铺 ID（只能为 0）	操作是否成功	管理员删除商品类目信息，分类下的所有商品变成孤儿商品，categoryId = 0
查询正式商品	店铺 ID，条形号，页码、每页数目	商品信息	查询正式商品，禁售态的商品不返回商品信息
获得商品有效详细信息	商品 ID	商品详细信息	无需登录即可获得商品有效详细信息
获得商品指定售后的历史信息	商品 ID	指定售后的历史信息	无需登录即可查询指定售后历史信息
商铺申请增加新商品	商铺 ID，商品信息	操作是否成功	增加的新商品被列入草稿商品，商品不允许加入到一级分类
商铺查看草稿商品	商铺 ID，页码，每页数目	草稿商品 ID	shopId 为 0 表示查看所有商铺的草稿
删除审核中的商品	商铺 ID，商品 ID	操作是否成功	管理员或商铺物理删除审核中的商品
修改审核态商品信息	商铺 ID，商品 ID，可修改的货品信息	操作是否成功	商铺修改审核态商品信息，部分商品不允许加入到一级分类
查看草稿商品信息详情	商铺 ID，商品 ID	草稿商品信息详情	商铺查看草稿商铺详细信息，且只能查看自身

			店铺的草稿商品
查看商品信息详情	商铺 ID，	商品详细信息	商铺查看商品信息详情
修改商品信息	商铺 ID，商品 ID，可修改的商品信息	操作是否成功	商铺或管理员修改商品信息。如果把修改的信息需要审核，则产生草稿，否则直接修改货品信息，管理员可修改分账比例
商铺设定商品关系	商铺 ID，关联商品 ID，可修改的商品信息	操作是否成功	根据某个商品的 ID 设定另一个商品的 ID
商铺取消商品关系	商铺 ID，商品 ID	操作是否成功	将 goodsId 设为 0
发布商品	商铺 ID（只能为 0），商品 ID	操作是否成功	管理员发布货品，审核后才能发布商品
管理员解禁商品	商铺 ID（只能为 0），商品 ID	操作是否成功	禁售状态的货品才允许解禁
管理员禁售商品	商铺 ID（只能为 0），商品 ID	操作是否成功	下架和上架状态的货品才允许禁售
管理员上架商品销售	商铺 ID，商品 ID，可修改的信息	操作是否成功	管理员上架销售时不能与已有的同产品的销售时间重合
管理员查询商品销售	商铺 ID，商品 ID，页码，每页数目	商品销售信息	查询的是商品销售类型：普通、秒杀、团购和预售
管理员查询特定价格	商铺 ID，销售 ID	价格浮动详细信息	管理员查询特定价格浮动的详情
删除销售	商铺 ID，销售 ID	操作是否成功	如果存在关联的订单则不能删除
取消销售	商铺 ID，销售 ID	操作是否成功	取消销售时，如果结束时间晚于当前时间，将结束时间改为当前时间
查询所有上线态的团购活动	商品 ID，商铺 ID，页码，每页数目	团购活动信息	查询时仅显示上线状态的团购活动
查看团购活动详情	团购活动 ID	团购活动详情	管理员或商铺查看团购活动详情
管理员查询商铺团购	商铺 ID，商品 ID，页码，每页数目	团购活动列表	所有状态的团购均会显示
管理员新增团购活动	商铺 ID，销售 ID，可修改的团购活动	操作是否成功	团购与优惠和预售活动不能并存
管理员查看特定团购详情	商铺 ID，团购活动 ID	团购活动详情	所有状态的团购均会显示
管理员修改团购活动	商铺 ID，团购活动	操作是否成功	管理员修改团购活动

	ID，可修改的团购活动信息		
管理员终止团购活动	商铺 ID，团购活动 ID	操作是否成功	管理员调用 delOnsale 提前终止团购活动，通过消息服务器全额退款
管理员新建优惠活动	商铺 ID，可修改的优惠活动信息	操作是否成功	管理员新建己方优惠活动
查看店铺优惠活动列表	商铺 ID，商品 ID，页码，每页数目	优惠活动列表	管理员查看店铺优惠活动列表
计算优惠	活动 ID，订单信息	订单数量，订单价格，订单优惠金额	计算商品参与活动的优惠价格（实际价格）
查看所有上线优惠活动列表	商铺 ID，晚于开始时间，早于结束时间，页码，每页数目	优惠活动列表	无需登录也可查看所有上线优惠活动列表
查看活动中的商品	活动 ID，页码，每页数目	商品详细信息	商品优惠卷的有效期取所有销售中最早结束的
查看优惠活动详情	商铺 ID，活动 ID	优惠活动详细信息	管理员查看某个商铺优惠活动详情
管理员修改某优惠活动	商铺 ID，活动 ID，可修改的优惠活动信息	操作是否成功	管理员修改某优惠活动
管理员取消优惠活动	商铺 ID，活动 ID	操作是否成功	管理员删除活动与所有售后的关系
将优惠活动加到销售上	商铺 ID，活动 ID，销售 ID	操作是否成功	管理员将某个优惠活动与销售关联

4.2.6 售后模块

POST	/orderitems/{id}/aftersales	顾客提交售后申请*
GET	/aftersales	顾客查询售后列表（根据售后状态分类查询）
GET	/aftersales/orderitems	顾客查询所有的可申请售后的订单明细
GET	/aftersales/{id}	顾客根据售后单id查询售后单信息
PUT	/aftersales/{id}	顾客修改售后单信息（只能在申请态）
DELETE	/aftersales/{id}	顾客取消售后
POST	/aftersales/{id}/arbitrations	顾客申请售后仲裁*
POST	/aftersales/{id}/second_arbitrations	顾客申请二次售后仲裁*
DELETE	/aftersales/{id}/arbitration/{aid}	顾客取消仲裁*
GET	/aftersales/{id}/express	顾客或商铺查询售后单对应的物流单号*

PUT	/shops/{shopid}/aftersales/{id}/confirm	商铺审核售后
PUT	/shops/{shopid}/aftersales/{id}/receive	商铺验收售后商品
DELETE	/shops/{shopid}/aftersale/{id}/cancel	商铺取消售后
PUT	/shops/{shopid}/arbitrations/{id}/response	商铺应诉仲裁*
PUT	/shops/{shopid}/arbitrations/{id}/accept	管理员受理仲裁单
PUT	/shops/{shopid}/arbitrations/{id}/close	管理员仲裁结案
GET	/shops/{shopid}/aftersales	商铺或管理员查看所有售后单（可根据售后类型和状态选择）
GET	/shops/{shopid}/aftersales/{id}	商铺或管理员根据售后单id查询售后单信息
GET	/shops/{shopid}/arbitrations	商铺或管理员查询顾客申请的仲裁单信息（根据仲裁单状态）
GET	/shops/{shopid}/arbitrations/{id}	商铺或管理员查询仲裁单详情信息

接口名称	输入	输出	接口描述
顾客提交售后申请	订单 ID，售后服务信息	售后服务信息	顾客进入售后申请界面提交售后单，设置售后单状态为未售后
顾客查询售后列表	售后状态，页码，每页数目	售后列表	默认所有售后单列表，选择按售后单状态作为条件查询，顾客只能查询到自己的售后单
顾客查询订单明细	页码，每页数目	可售后的订单详细信息	顾客只能查询到自己的订单明细，已经申请过退货或者换货的订单明细不会返回
顾客查询售后单信息	售后单 ID	售后单详细信息	顾客通过这个 API 只能查询到自己的售后单
顾客修改售后单信息	售后单 ID，可修改的信息：地址，售后商品的数量，申请售后的原因，联系人以及联系电话	操作是否成功	顾客只能在处于申请态的售后单上进行修改，一旦售后服务进入其他状态（处理中、已完成等状态），顾客将无法再对售后单进行修改
顾客取消售后	售后单 ID	操作是否成功	顾客只可以在未售后和售后中的状态取消
顾客申请售后仲裁	售后单 ID，售后信息	售后仲裁信息	只允许顾客提出仲裁申请，需设置售后为仲裁中，如果已经在仲裁中的售后不允许再仲裁，则出现 705 错误
顾客申请二次仲裁	售后单 ID，仲裁详细信息	售后二次仲裁信息	只允许顾客提出的仲裁申请，且有一次的仲裁记录，需设置售后为仲裁中，如果已经在仲裁中的售后不允许再仲裁，出 705 错误
顾客取消仲裁	售后单 ID，仲裁单 ID	操作是否成功	顾客根据仲裁 ID 取消仲裁，顾客只可以在未仲裁和仲裁中状态取消，已仲裁的仲裁单不可取消
查询售后单对应的物流单号	售后单 ID	物流单号	根据售后 id 查询到对应的物流订单，已经申请过退货或者换货的订单明细不会返回
商铺审核售后	商铺 ID，售后单	操作是否成功	商铺审核顾客提出的售

	ID，处理意见		后请求并设置售后状态为售后中或已售后
商铺验收售后商品	商铺 ID，售后单 ID，处理意见	操作是否成功	维修商品不可调用此接口
商铺取消售后	商铺 ID，售后单 ID	操作是否成功	商铺取消售后，需要设置相应售后为正常
商铺应诉仲裁	商铺 ID，仲裁单 ID，仲裁详细信息	仲裁单详情	顾客提出仲裁，商铺应诉仲裁
管理员受理仲裁单	商铺 ID（只能为 0），仲裁单 ID，仲裁员信息	操作是否成功	仅申请中的仲裁单才能被受理
管理员仲裁结案	商铺 ID（只能为 0），仲裁单 ID，仲裁结果信息	操作是否成功	仅仅受理态才可以结案，负责仲裁的仲裁员才可以结案，设置相应售后为正常
查看所有售后单	商铺 ID（管理员为 0），开始时间，结束时间，页码，每页数目，售后类型，售后状态	售后单列表	商铺或管理员可通过售后单 ID 查看所有售后单
查询售后单信息	商铺 ID（管理员为 0），售后单 id	售后单详细信息	商铺或管理员根据售后单 id 查询售后单信息
查询顾客申请的仲裁单信息	商铺 ID（管理员为 0），页码，每页数目	仲裁单列表	商铺或管理员查询顾客申请的仲裁单信息（根据仲裁单状态），按照申请时间从近到远排序
查询仲裁单详情	商铺 ID（管理员为 0），仲裁单 ID	仲裁单详细信息	商铺或管理员根据仲裁单 ID 查询仲裁单详情信息

4.2.7 服务模块

POST	/shops/{did}/products/{id}/maintainers/{mid}/service/{sid}	商户选择服务商提供的地区服务，与具体商品关联
PUT	/shops/{did}/product_services/{id}	商户修改服务商在某个地区的商品服务*(暂停：从有效变为无效；恢复：从无效变为有效)
DELETE	/shops/{did}/product_services/{id}	商户取消服务商在某个地区的商品服务*
GET	/shops/{did}/products/{id}/region/{rid}	商户通过商品和地区找到服务商
GET	/shops/{did}/maintainers/{mid}	商户通过服务商id找到服务商*
DELETE	/shops/{did}/maintainers/{mid}	商户取消服务商的所有服务
GET	/services/states	获得服务单的所有状态
POST	/internal/shops/{did}/products/{id}//region/{rid}/serviceOrders	创建服务单
GET	/shops/{did}/serviceOrders	商户查询服务单信息
GET	/shops/{did}/serviceOrder/{id}	店铺根据服务单id查询服务单信息
DELETE	/shops/{did}/serviceOrder/{id}	平台和商家取消服务单
POST	/maintainers	服务商注册账号*
PUT	/maintainers/{mid}	服务商申请变更账号信息*
DELETE	/maintainers/{mid}	服务商注销账号*
GET	/maintainers/{mid}	服务商查看账户信息*
GET	/adminusers/{aid}/maintainers	管理员查看服务商*
PUT	/adminusers/{aid}/maintainers/{mid}/account	平台管理员审核服务商开户*
PUT	/adminusers/{aid}/maintainers/{mid}/modify	平台管理员审核服务商变更*
PUT	/adminusers/{aid}/maintainers/{mid}/suspend	平台管理员暂停服务商*
PUT	/adminusers/{aid}/maintainers/{mid}/resume	平台管理员恢复服务商*
PUT	/adminusers/{aid}/services/{sid}/service	平台管理员审核服务*

接口名称	输入	输出	接口描述
商户选择商品服务	商铺 ID，商品 ID，服务商 ID	商品服务 ID	商户选择服务商提供的地区服务，与具体商品关联
商户修改商品服务	商铺 ID，商品服务 ID	操作是否成功	商户修改商品服务(暂停：从有效变为无效；恢复：从无效变为有效)
商户取消商品服务	商铺 ID，商品服务 ID	操作是否成功	商户取消商品服务
商户通过商品和地区查询服务商	商铺 ID，商品 ID，地区 ID	服务商信息	商户通过商品和地区找到服务商
商户通过服务商 id 查询服务商	商铺 ID，服务商 ID	服务商信息	商户通过服务商 id 找到服务商
商户取消服务商的所有服务	商铺 ID，服务商 ID	操作是否成功	商户取消指定 ID 的服务商的所有服务

获得服务单的所有状态	无	服务单的所有状态	获得服务单的所有状态
商户创建服务单	商铺 ID，商品 ID，地区 ID	服务单 ID	商户创建服务单
获得服务单信息	商铺 ID	服务单信息列表	管理员或商户获得服务单信息
商户根据 id 查询服务单	商铺 ID，服务单 ID	服务单信息	商户查询指定 ID 的服务单
取消服务单	商铺 ID，服务单 ID	操作是否成功	管理员或商户取消服务单
服务商注册账号	服务商名称、密码、联系人、保证金等	服务商 ID	服务商注册账号
服务商申请变更账号信息	服务商 ID，服务商信息	操作是否成功	服务商申请变更账号信息
服务商注销账号	服务商 ID	操作是否成功	服务商注销账号，完成所有服务单后才可注销
服务商查看账户信息	服务商 ID	服务商信息	服务商查看账户信息，需检查 maintainerId 是否与访问者的一致
管理员查询服务商	管理员 ID，服务商类型，状态，名称	服务商信息列表	管理员查询服务商
管理员审核服务商开户	管理员 ID，服务商 ID，审核结果	操作是否成功	平台管理员审核服务商开户
管理员审核服务商变更	管理员 ID，服务商 ID，审核结果	操作是否成功	平台管理员审核服务商变更
管理员暂停服务商	管理员 ID，服务商 ID	操作是否成功	平台管理员暂停服务商，修改服务商状态为暂停
管理员恢复服务商	管理员 ID，服务商 ID	操作是否成功	平台管理员恢复服务商，修改服务商状态为有效
管理员审核服务	管理员 ID，服务 ID，审核结果	操作是否成功	平台管理员审核服务
服务商查询服务单信息	服务商 ID	服务单信息列表	服务商通过这个 API 查询关联商家派发的服务单和已完成的服务单
服务商根据 id 查询服务单信息	服务商 ID，服务单 ID	服务单信息	服务商查询指定 ID 的服务单信息

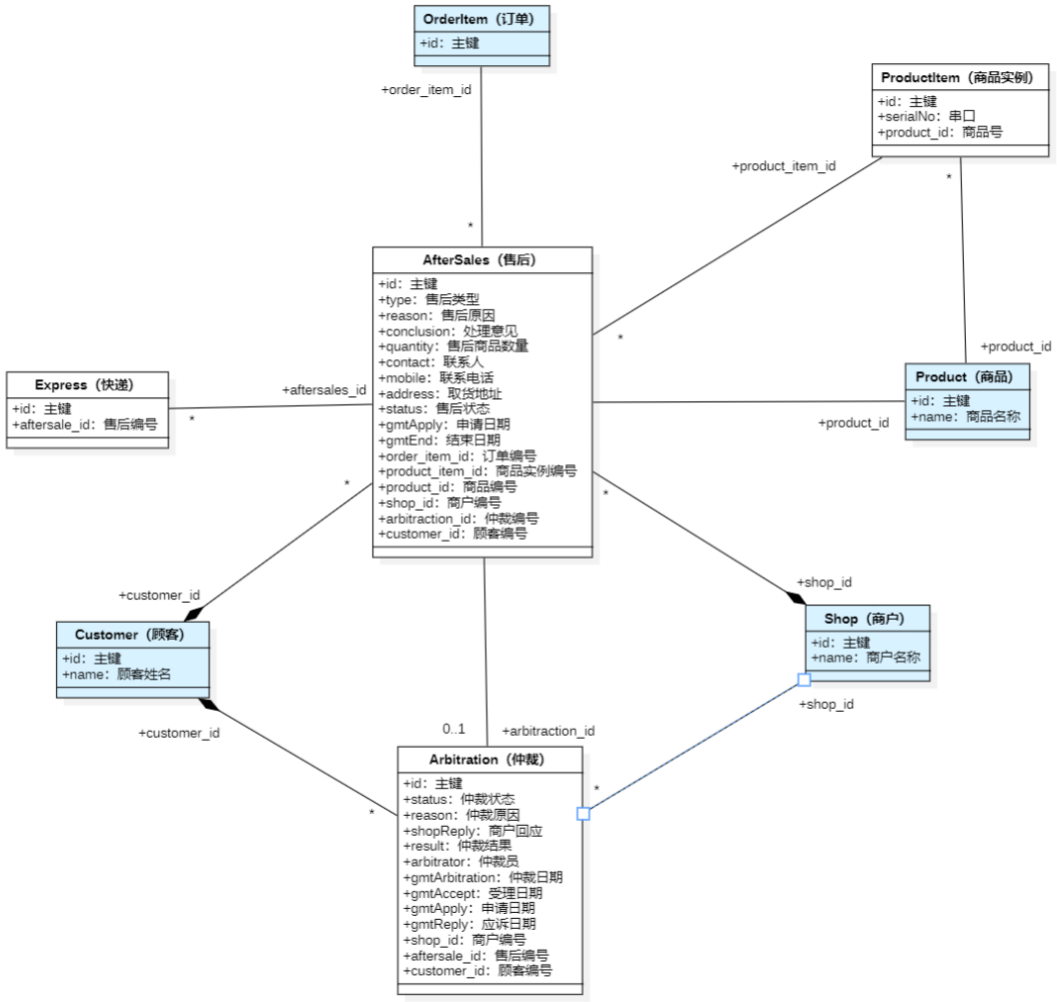
服务商接受服务单	服务商 ID，服务单 ID	操作是否成功	服务商接受服务单
服务商接受服务单	服务商 ID，服务单 ID	操作是否成功	服务商接受服务单
服务商收到寄出商品	服务商 ID，服务单 ID	操作是否成功	服务商验收一个商品，并反馈验收状态
服务商完成服务单	服务商 ID，服务单 ID	操作是否成功	服务商完成一个服务单，并反馈服务状态
服务商取消服务单	服务商 ID，服务单 ID	操作是否成功	服务商取消一个服务单
服务商定义服务	服务商 ID，商品类别 ID，地区 ID	服务 ID	服务商定义在某个或多个地区为某类商品的服务
服务商取消服务	服务商 ID，服务 ID	操作是否成功	服务商取消某一服务
服务商修改服务范围	服务商 ID，服务 ID，地区 ID	操作是否成功	服务商修改某服务的服务范围
服务商查询服务	服务商 ID，服务状态，服务地区	服务信息列表	服务商查询自己注册的服务
服务商根据 id 查询服务	服务商 ID，服务 ID	服务信息	服务商根据服务 id 查询服务信息

5. 数据结构设计

5.1 逻辑结构设计

5.1.1 售后模块

5.1.1.1 数据库图



- 每个售后对应多个快递物流，因为可能有寄出和寄回两次物流；
- 每个售后对应 0 或 1 个仲裁，因为顾客可能不会申请仲裁，也可能重复申请仲裁；

- 每个售后对应多个售后订单；
- 每个顾客对应多个售后，也对应多个仲裁；
- 每个商品实例对应多个售后，可以多次申请售后；

5.1.1.2 数据表

aftersales 售后表			
字段名称	字段类型	约束	说明
id	INTEGER	PRIMARY KEY	售后ID
type	SMALLINT	NOT NULL	售后类型
reason	VARCHAR (200)	NOT NULL	售后原因
conclusion	VARCHAR (200)	NOT NULL	处理原因
quantity	INTEGER	NOT NULL	售后数量
region_id	INTEGER	FOREIGN KEY	取货地区ID
address	VARCHAR (200)	NOT NULL	取货地址
contact	VARCHAR (20)	UNIQUE NOT NULL	联系人
mobile	VARCHAR (20)	UNIQUE NOT NULL	联系电话
gm_apply	DATE	NOT NULL	申请日期
gm_end	DATE	NOT NULL	结束日期
status	SMALLINT	NOT NULL	售后状态
product_id	INTEGER	FOREIGN KEY	商品ID

product_item_id	INTEGER	FOREIGN KEY	商品实例ID
order_item_id	INTEGER	FOREIGN KEY	订单ID
name	VARCHAR (50)	UNIQUE NOT NULL	产品名称
arbitration_id	INTEGER	FOREIGN KEY	仲裁ID
custormer_id	INTEGER	FOREIGN KEY	顾客ID
shop_id	INTEGER	FOREIGN KEY	商铺ID

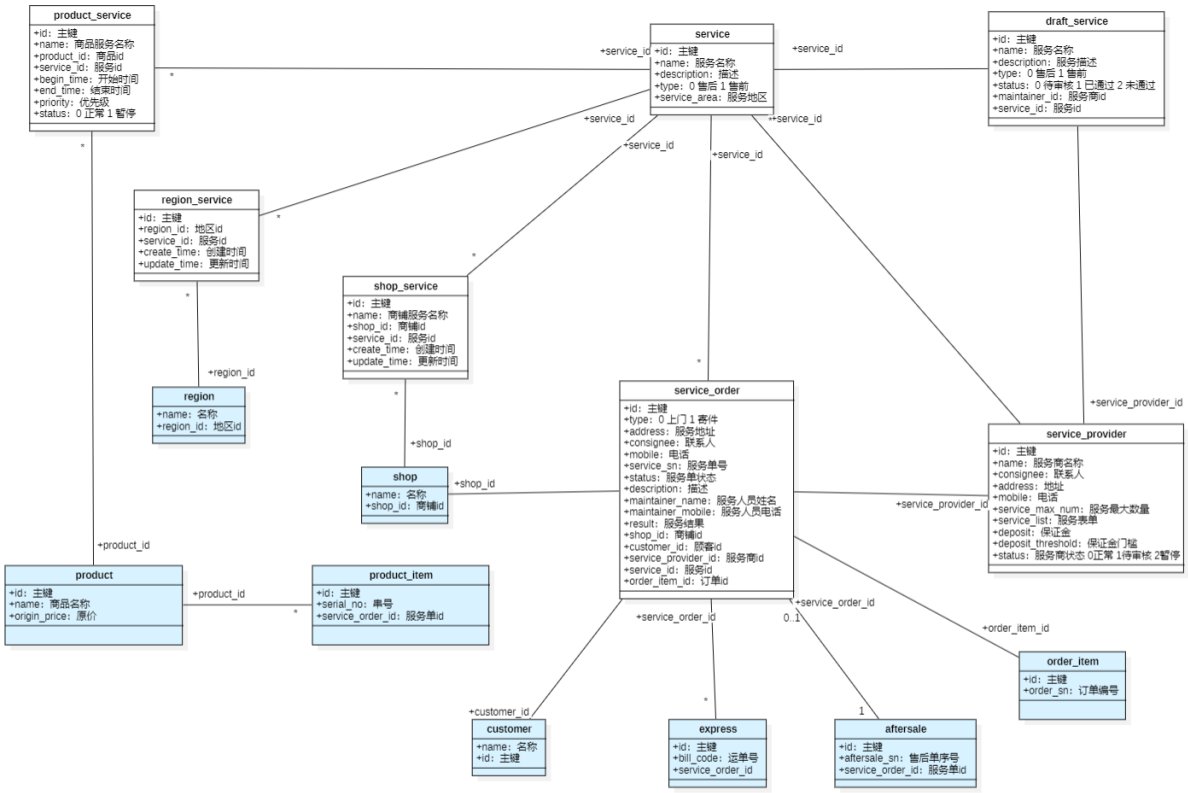
product_item 商品实例表			
字段名称	**字段类型**	**约束**	**说明**
id	INTEGER	PRIMARY KEY	商品实例ID
serial_no	INTEGER	UNIQUE NOT NULL	串号
product_id	INTEGER	FOREIGN KEY	商品ID

arbitration 仲裁表			
字段名称	**字段类型**	**约束**	**说明**
id	INTEGER	PRIMARY KEY	仲裁ID
status	SMALLINT	NOT NULL	仲裁状态
result	VARCHAR (200)	NOT NULL	仲裁结果
reason	VARCHAR (200)	NOT NULL	仲裁原因
shop_reply	VARCHAR (200)	NOT NULL	商户回应

aftersale_id	INTEGER	FOREIGN KEY	售后ID
shop_id	INTEGER	FOREIGN KEY	商铺ID
customer_id	INTEGER	FOREIGN KEY	顾客ID
arbitrator	VARCHAR (20)	NOT NULL	仲裁员
gmt_arbitration	DATE	NOT NULL	仲裁日期
gmt_accept	DATE	NOT NULL	受理日期
gmt_apply	DATE	NOT NULL	申请日期
gmt_reply	DATE	NOT NULL	应诉日期

5.1.2 服务模块

5.1.2.1 数据库图



- 每个服务商对应多个服务，服务商可以注册管理多个服务；
- 每个服务商对应多个草稿服务，服务商的信息可能进行多次修改；
- 每个服务对应一个草稿服务，草稿服务保存的是未经审核的服务；
- 每个服务会产生多个服务单，每个服务单对应一个服务；
- 每个服务商对应多个服务单，服务商可以接受多个服务单；
- 每个服务对应多个地区服务，一个地区服务中有多个地区；
- 每个服务对应多个商品服务，每个商品可以被设置多个服务；
- 每个服务对应多个商铺，可以为多个商铺提供服务，每个商铺也拥有多个服务；
- 每个服务单对应 1 到多个商品项，每个服务单可以服务多个商品；
- 每个服务单对应多个物流，一个服务单可能对应寄出和寄回物流单；
- 每个服务单对应一个售后，每个售后对应 0 个或 1 个服务单；
- 每个服务单对应一个订单，每个订单对应一个服务单；
- 每个服务单对应一个顾客，每个顾客有多个服务单。

5.1.2.2 数据表

region_service 地区服务表			
字段名称	**字段类型**	**约束**	**说明**
id	INTEGER	PRIMARY KEY	服务地区ID
service_id	INTEGER	FOREIGN KEY	服务ID
region_id	INTEGER	FOREIGN KEY	地区ID

create_time	DATE	NOT NULL	创建时间
update_time	DATE	NOT NULL	更新时间

product_service 商品服务表			
字段名称	**字段类型**	**约束**	**说明**
id	INTEGER	PRIMARY KEY	商品服务ID
product_id	INTEGER	FOREIGN KEY	商品ID
service_id	INTEGER	FOREIGN KEY	服务ID
name	VARCHAR (20)	NOT NULL	商品服务名称
begin_time	DATE	NOT NULL	开始时间
end_time	DATE	NOT NULL	结束时间
priority	SMALLINT	NOT NULL	优先级
status	SMALLINT	NOT NULL	0正常 1暂停

shop_service 商铺服务表			
字段名称	**字段类型**	**约束**	**说明**
id	INTEGER	PRIMARY KEY	商铺服务ID
shop_id	INTEGER	FOREIGN KEY	商铺ID
service_id	INTEGER	FOREIGN KEY	服务ID
name	VARCHAR (20)	NOT NULL	商铺服务名称
create_time	DATE	NOT NULL	创建时间
update_time	DATE	NOT NULL	更新时间

draft_service 草稿服务表			
字段名称	**字段类型**	**约束**	**说明**
id	INTEGER	PRIMARY KEY	草稿服务ID
maintainer_id	INTEGER	FOREIGN KEY	服务商ID
service_id	INTEGER	FOREIGN KEY	服务ID
name	VARCHAR (20)	NOT NULL	服务名称
description	VARCHAR (200)	NOT NULL	服务描述
status	SMALLINT	NOT NULL	0待审核 1通过 2拒绝
type	SMALLINT	NOT NULL	0售后 1售前

service 服务表			
字段名称	**字段类型**	**约束**	**说明**
id	INTEGER	PRIMARY KEY	服务ID
name	VARCHAR (20)	NOT NULL	服务名称
description	VARCHAR (200)	NOT NULL	服务描述
type	SMALLINT	NOT NULL	服务类型（售前、售后）
maintainer_id	INTEGER	FOREIGN KEY	服务商ID

service_area	VARCHAR (20)	NOT NULL	服务地区

service_provider 服务商表			
字段名称	**字段类型**	**约束**	**说明**
id	INTEGER	PRIMARY KEY	服务商ID
name	VARCHAR (20)	NOT NULL	服务商名称
consignee	VACHAR (20)	NOT NULL	联系人姓名
address	VACHAR (50)	NOT NULL	地址
mobile	VACHAR (20)	UNIQUE NOT NULL	电话
service_max_num	INTEGER	NOT NULL	服务最大数量
service_list	INTEGER	FOREIGN KEY	服务表单
deposit	INTEGER	NOT NULL	保证金
status	INTEGER	NOT NULL	0正常 1待审核 2暂停

service_order 服务单表			
字段名称	**字段类型**	**约束**	**说明**
id	INTEGER	PRIMARY KEY	服务单ID
type	SMALLINT	NOT NULL	0上门 1寄件
status	SMALLINT	NOT NULL	服务单状态描述

description	VARCHAR (200)	NOT NULL	服务描述
region_id	VARCHAR (50)	FOREIGN KEY	服务地址ID
address	VARCHAR (50)	NOT NULL	详细地址
consignee	VARCHAR (20)	NOT NULL	联系人
mobile	VARCHAR (20)	NOT NULL	电话
maintainer_name	VARCHAR (20)	NOT NULL	服务人员姓名
maintainer_mobile	VARCHAR (20)	NOT NULL	服务人员电话
result	VARCHAR (200)	NOT NULL	服务结果
shop_id	INTEGER	FOREIGN KEY	商铺ID
customer_id	INTEGER	FOREIGN KEY	顾客ID
service_id	INTEGER	FOREIGN KEY	服务ID
service_provider_id	INTEGER	FOREIGN KEY	服务商ID
product_id	INTEGER	FOREIGN KEY	商品ID
serial_no	INTEGER	UNIQUE NOT NULL	串号
aftersales_id	INTEGER	FOREIGN KEY	售后单ID
service_sn	INTEGER	UNIQUE NOT NULL	服务单号
order_item_id	INTEGER	FOREIGN KEY	订单id

5.2 物理结构设计

数据库的物理结构设计是一个关键阶段，涉及两个关键步骤：

首先，需要明确**定义数据库的物理结构**，这主要包括确定存取方法和存储结构。在关系数据库中，存取方法涉及如何有效地检索和更新数据，而存储结构则关注数据在物理媒体上的实际组织方式。这一步的关键目标是确保数据库在物理层面上能够高效地支持系统的功能需求，并且能够满足性能和可维护性的要求。

其次，对物理结构进行**评价**，着重考察时间和空间效率。时间效率涉及数据库在处理查询、更新等操作时的速度和响应时间，而空间效率则关注数据库在物理存储上的利用率。评价的过程需要考虑各种因素，包括硬件资源、系统负载以及数据的增长趋势。通过评估物理结构的时间和空间效率，可以优化数据库的性能，确保系统在运行时表现出良好的响应速度，并且能够高效地利用存储资源。

这两个步骤共同构成了数据库的物理结构设计过程，为数据库系统的稳定性、高效性和可维护性奠定了坚实的基础。这一设计过程需要综合考虑数据库管理系统的特性、应用需求以及硬件环境，以确保数据库在物理层面上能够充分发挥其优势。

5.2.1 存取方法选择

5.2.1.1 B+树索引存取方法的选择

- 为每个表的主键添加 B+树索引：主键是表中的唯一标识，而且在查询条件中主键经常被用于检索特定行。通过为主键添加 B+树索引，可以加速对该属性的查询操作，提高检索效率。
- 为商品销售表的价格添加 B+树索引：商品销售表中的价格属性可能需要进行范围查询，例如查找特定价格范围内的商品销售记录。为价格属性添加 B+树索引将支持高效的范围查询，提高查询操作的性能。
- 为订单表的创建时间添加 B+树索引：订单表中的创建时间属性通常用于按

时间排序，例如按照订单创建时间查找最新的订单记录。通过为创建时间添加 B+树索引，可以在排序和检索方面提供更高的效率，尤其在需要按时间顺序查询或检索最新订单时。

5.2.1.2 Hash 索引存取方法的选择

- 为商品明细表的串号添加 Hash 索引：串号属性在查询条件中频繁出现，为了加速对该属性的查询，采用 Hash 索引是一种有效的选择。Hash 索引能够提供快速的等值查询，适用于串号这种常用于检索的属性。

5.2.1.3 聚簇存取方法的选择

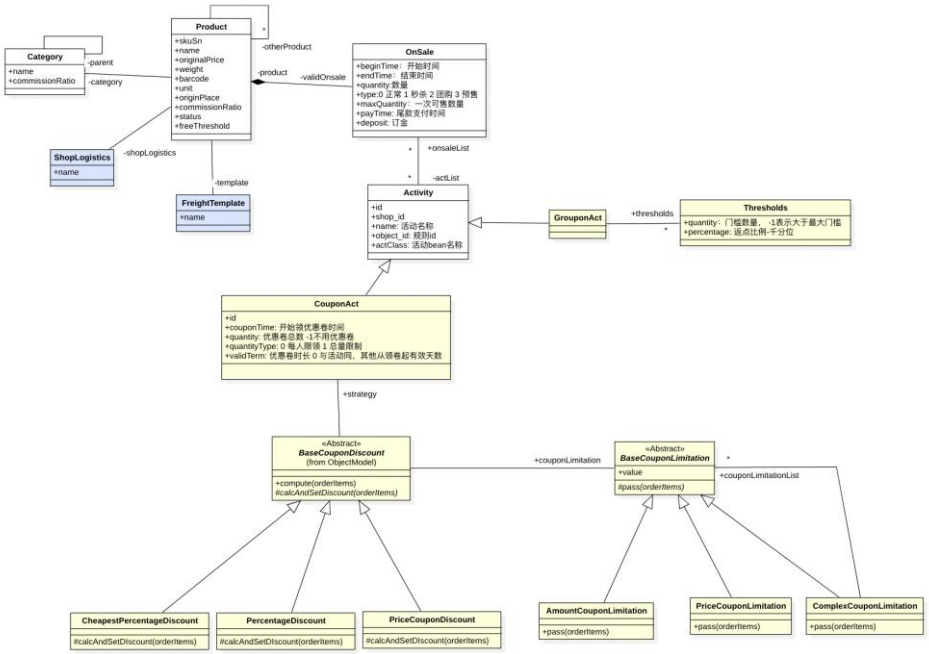
- 为数据表中出现的相关状态和类型属性添加聚簇索引：聚簇索引可以将相同值的元组集中存储在一起，便于对相关状态和类型进行分类展示和处理。通过聚簇索引，相同状态或类型的数据将在物理存储上更加紧密，提高了相关数据的查询效率。

5.2.2 存放位置和存储结构

5.2.2.1 确定数据的存放位置

为了提高系统性能，我们需要根据应用情况将数据的易变部分与稳定部分、经常存取部分和存取频率较低部分等分开存放。

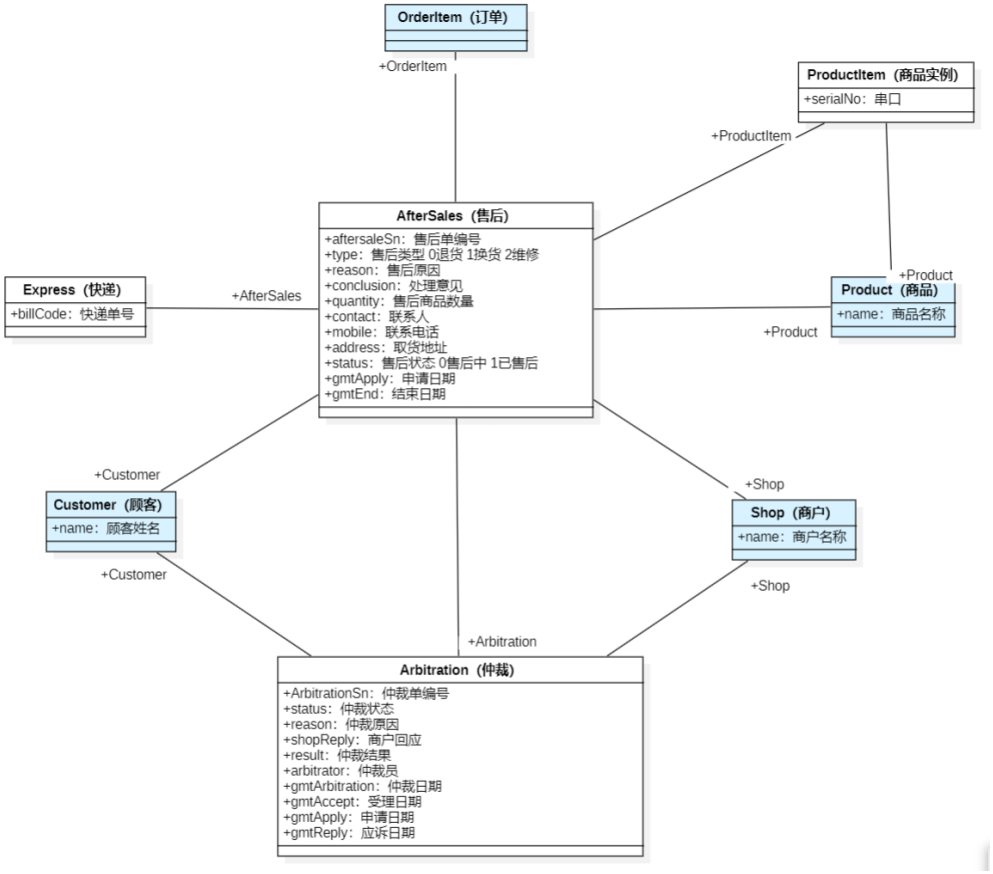
- 利用关系数据库 MySQL 存储关系型的对象，而使用 MongoDB 来存储非关系存储的对象或者关系很复杂，难以用关系数据库来进行存储的对象。



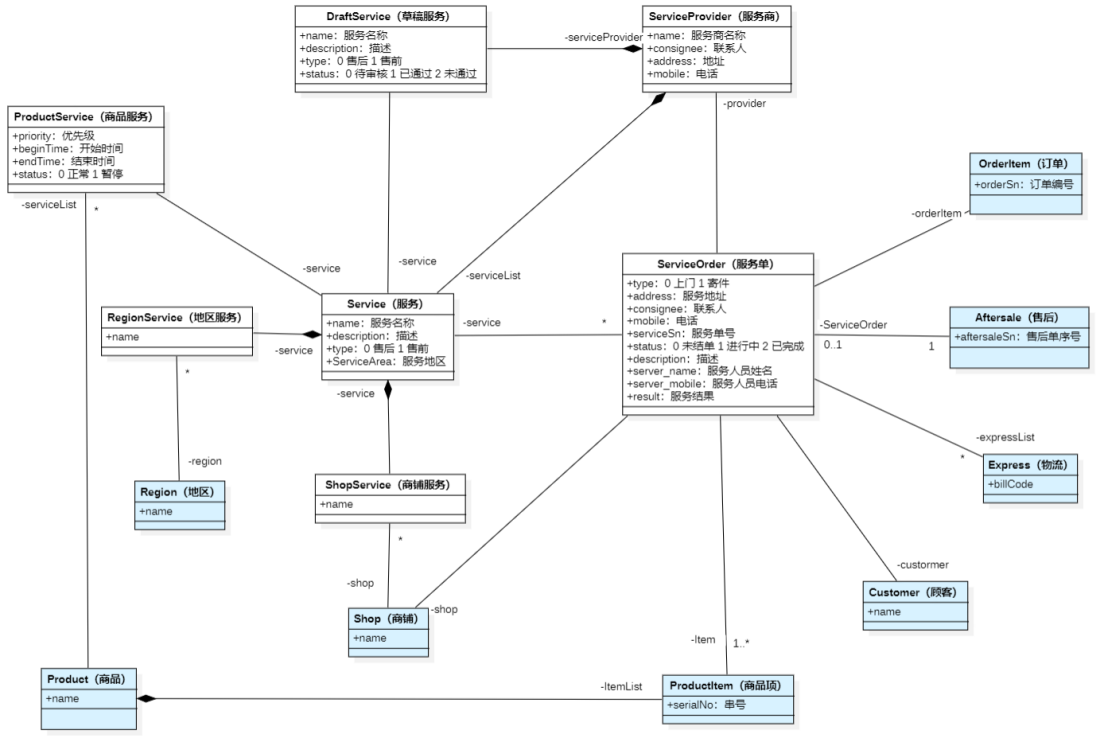
商品模块对象模型

比如在商品模块的对象模型设计中，我们非黄色部分使用 MySQL 关系数据库来存储，但是黄色部分中有关团购和优惠活动的部分关系复杂，我们使用 MongoDB 数据库来直接存储对象。

在我们小组负责的服务和售后模块，我们也利用对象模型进行来分析。



售后模块对象模型



服务模块对象模型

售后模块和服务模块这里我们通过构建对象模型发现它们并不需要像商品模块那样有部分利用 MongDB 存储。

- 可以将比较大的表分别放在两个磁盘上来加快存取速度，这在 OOMALL 这样的多用户环境下特别有效。

OOMALL 预期在商品表、订单表、售后单表、服务单表等表都会存在数据较多的情况，我们可以将他们放在不同磁盘上进行后续存取。

- 将 OOMALL 系统的日志文件与数据库对象（表、索引等）放在不同的磁盘上，以改进系统的性能。

5.2.2.2 确定数据的存储结构

在确定数据的存储结构方面，我们需要综合考虑数据的访问和使用需求，以选择最适合的存储结构。常见的存储结构包括顺序方式、链式方式、哈希、树和图等：

1. **顺序方式：**适用于有序数据，能够提供快速的顺序遍历和检索。特别适用于对数据进行顺序分析或按照某一顺序进行检索的场景。

2. **链式方式：** 链表结构适用于需要频繁插入和删除操作的场景。链式存储结构通过指针相连，支持高效的插入和删除，但在随机访问时可能性能较差。

3. **哈希：** 适用于等值查询的场景，通过哈希函数将关键字映射到存储位置，提供快速的等值查询。哈希在提供高效检索的同时，需要注意解决冲突的机制。

4. **树：** 树状结构如 B+树适用于范围查询和有序数据的存储。B+树具有平衡性，支持范围查询和有序遍历，是关系数据库中常用的存储结构。

5. **图：**图结构适用于表达复杂的关系，例如社交网络或网络拓扑。图数据库能够有效地表示和处理节点之间的关联关系。

在选择存储结构时，需要根据具体的业务需求和数据访问模式进行权衡。不

同的存储结构具有不同的优势和劣势，选择适合场景的存储结构能够提高系统的性能和效率。

5.2.3 评价物理结构

- 存储空间利用率：采用分布式数据库，如 MongoDB，以提高存储空间的灵活性和利用率。分布式数据库允许数据分布在多个节点上，有效地分担数据存储压力，提高整体的存储空间利用率；
- 存取时间：采用单一数据库服务器可能在高峰时间段出现性能瓶颈，影响系统响应速度。为了解决这个问题，引入 Redis 作为缓存，实现数据分片和负载均衡。通过 Redis，可以将热门数据缓存在内存中，减轻数据库服务器的压力，并通过数据分片和负载均衡技术，分散访问压力，提高存取效率；
- 维护代价：为了降低维护成本，选择使用开源数据库 Redis，并结合使用华为 codearts 优惠券，能够有效减少相关成本。Redis 作为一种高性能的开源缓存数据库，可以提供快速的数据访问，而华为 codearts 优惠券则能够降低采用相关服务的费用，从而在维护代价上取得一定的经济优势。

5.3 数据结构与程序的关系

数据结构与程序之间存在密切的关系，尤其在数据库应用中：

1. **数据操作语言(SQL)与程序的交互：**
    - 程序通过特定的数据操作语言(SQL)与数据结构进行交互。SQL 作为一种标准的数据库查询语言，允许程序通过执行 SQL 语句来操作数据库中的数据结构。
2. **SQL 语句与数据结构的操作：**
    - 通过 SQL 语句，程序能够创建、查询、更新或删除数据库中的数据结构。SQL 提供了丰富的语法和命令，允许程序以声明性的方式描述对数据结构的操作，包括表的创建、索引的创建、数据的查询和更新等。

3. **编写 SQL 语句进行交互：**

- 程序员使用特定的数据操作语言(SQL)编写相应的 SQL 语句，通过执行这些语句与数据库的数据结构进行交互。编写 SQL 语句需要考虑数据的组织结构、关系模式以及具体的操作需求，从而实现对数据结构的有效操作。

6. 运行设计

6.1 运行模块的组合

OOMALL 系统中体现了一系列有序而相互协作的模块，形成了整体的数据处理流程：

1. **启动接收数据模块（VIEW）：**
    - OOMALL 系统在有输入时启动接收数据模块，这是整个流程的起点。通过该模块，系统能够接收外部输入的数据，并对输入进行格式化，以确保数据的一致性和正确性。
2. **调用网络传输模块（HTTP）：**
    - 当接收数据模块得到充分的数据时，系统将调用网络传输模块。该模块负责将数据通过网络送到服务器，并等待接收服务器返回的信息。在服务器端，接收网络数据模块需要保持活动状态，以及时响应客户端的请求。
3. **调用数据处理/查询模块（MyBatis + MySQL + MongoDB）：**
    - 一旦服务器接收到数据，系统将调用数据处理/查询模块，该模块对数据库进行访问和处理。在这一过程中，系统可以充分利用 MyBatis 进行数据库操作，结合 MySQL 和 MongoDB 等数据库，实现对不同类型数据的高效处理和查询。
4. **调用网络发送模块：**
    - 数据处理/查询模块完成后，系统将调用网络发送模块，将处理后的信息返回客户机。这一过程确保了及时的数据传递和客户端的响应。
5. **调用数据输出模块（VIEW）：**
    - 最后，接收到返回信息的客户机将调用数据输出模块，对信息进行处理并生成相应的输出。这个模块将负责展示最终结果给用户，形

成系统的可视化输出。

通过这种模块的组合方式，OOMALL 系统实现了流畅的数据处理流程，各个模块之间协同工作，确保了数据的正确传递和处理，以及用户能够获得合适的输出。这种模块组合的设计使得系统具有良好的可维护性和可扩展性。

6.2 运行控制

针对运行控制，系统将严格按照各模块间的函数调用关系进行实现，并在关键模块中进行正确的判断，选择适当的运行控制路径，确保系统的正常运行和事务的正确执行：

1. **事务中心模块中的运行控制：**
    - 在各事务中心模块中，系统需要对运行控制进行正确的判断，以选择正确的运行控制路径。这确保了在处理不同事务时，系统能够根据具体的业务逻辑和运行条件，选择适当的执行路径，以保证事务的正确性和完整性。
2. **网络传输方面的运行控制：**
    - 客户机在发送数据后，等待服务器的确认收到信号。收到确认信号后，再次等待服务器发送回答数据，并对数据进行确认。服务器在接收到数据后，发送确认信号。在对数据进行处理和访问数据库后，将返回信息送回客户机，并等待客户机的确认。这一运行控制策略确保了在网络传输中的双向通信中，数据的可靠传递和双方的同步操作。

6.3 运行时间

OOMALL 系统采用 Redis 内存服务器作为高频率数据存储和访问的解决方案。由于 Redis 具备亚毫秒级的响应时间，能够迅速处理频繁请求，从而提高系统的响应速度和性能。

针对高负载、多并发的开发需求，OOMALL 系统设定了用户响应时间的目

标。在正常情况下，系统的用户响应时间被控制在 1-2 秒左右，确保用户在正常使用过程中能够获得迅速的反馈。

考虑到可能出现用户访问量巨大的极端情况，系统在设计中预留了一定的响应时间余地。即便在极端情况下，用户响应时间也被控制在 2.5 秒以内，以确保系统在高压力情境下仍能提供相对较好的用户体验。

7. 出错处理设计

7.1 出错输出信息

OOMALL 系统具备详细的错误处理机制，确保在出错时能够提供清晰的错误码、错误描述以及相应的提示。出错信息分为处理错误、设定错误和系统错误三种情况：

1. **处理错误：**
   - 处理错误主要涉及输入错误信息超出或不符合预定格式的情况。在操作成功判断及输入数据验证模块中，系统进行数据分析，判断错误类型，并生成相应的错误提示语句。这些错误信息将被送到输出模块，同时返回给客户端用户，确保用户能够理解错误的具体原因。
2. **设定错误：**
   - 设定错误主要涉及系统预设不能执行的错误，例如权限问题等。在开始提交信息类别时，系统依据权限等判定错误类别，并生成相应的错误信息语句，输出到输出模块。这样的处理方式保证了在系统层面能够识别和处理各类设定错误，同时向客户端提供相应的提示信息。
3. **系统错误：**
   - 系统错误涉及网络传输超时、服务器响应超时等导致的问题。系统根据服务器的响应内容，判断错误类别并输出。这保障了在系统级别发生错误时，能够及时识别问题并提供明确的错误描述。

此外，所有的出错信息都必须给出相应的出错原因，以提高用户对错误信息的理解。例如，“该地区不可达”这样的具体原因有助于用户了解问题的本质。系统通过自定义返回码避免与 HTTP 自带返回码冲突，确保错误码的唯一性。整体而言，OOMALL 系统的出错信息设计符合良好的用户体验和系统可维护性的原则。

7.1.1 通用错误处理

状态码	含义	对策

404	路径上的 id 不存在	修改路径 id 后重试
400	请求参数错误	语法错误导致服务器无法处理请求参数，修改请求参数后重试
401	请求时未登录	登录后重试
403	没有权限访问	更改请求账户后重试

7.1.2 业务错误处理

状态码	对应代码	含义	对策
601	AUTH_INVALID_JWT	JWT 不合法	检查登录名和密码是否正确
602	AUTH_EXPIRED_JWT	JWT 过期	重新登录后重试
603	FIELD_NOTVALID	字段不合法	检查字段的合法性
608	IMG_FORMAT_ERROR	图片格式不正确	更换图片格式
609	IMG_SIZE_EXCEED	图片大小超限	压缩图片大小
605	RESOURCE_ID_OUTSCOPE	操作的资源 id 不是自己的对象	检查权限是否正确

7.1.3 数据库错误处理

状态码	含义	对策
130	文件格式不正确	重新设置文件格式
145	文件无法打开	检查文件
1005	创建表失败	检查创建表的语句
1006	创建数据库失败	检查数据库或重启数据库
1007	数据库已存在，创建数据库失败	检查数据库
1008	数据库不存在，删除数据库失败	检查删除数据库是否正确
1009	不能删除数据库文件导致删除	查看错误信息来解决

	数据库失败	
1020	该记录已被其他用户更改	查看修改记录，回滚
1021	硬盘剩余空间不足	增加硬盘容量
1022	关键字重复，更改记录失败	修改关键字或记录的其他字段
1036	数据表只读，不可修改	停止修改数据表
1037	系统内存不足	重启数据库服务器
1038	用于排序的内存不足	增大排序缓冲区
1040	已达到数据库的最大连接数	增大数据库的最大连接数
1045	连接数据库账户错误	重新输入用户名和密码
1048	字段不能为空	重新输入字段
1116	打开的数据表过多	关闭部分数据表
1158	网络错误，出现读错误	检查网络连接状况
1159	网络错误，出现读超时	检查网络连接状况
1160	网络错误，出现写错误	检查网络连接状况
1161	网络错误，出现写超时	检查网络连接状况
1203	当前用户和数据库建立的连接已到达数据库的最大连接数	增大可用的数据库连接数或重启数据库

7.1.4 服务器错误处理

状态码	含义	对策
500	服务器内部错误	检查服务器或重启
501	服务器不具备完成请求的功能	修改请求
502	网关错误	增加 php-cgi 的进程数
503	服务器目前无法请求	等待服务器可用
504	错误的网关超时	检查 nginx.conf 的配置
505	服务器不支持请求中使用的 HTTP 协议版本	更换 HTTP 协议版本

7.2 出错处理对策

- 为确保数据的完整性，所有服务端应保持持续开机，以防止停电或电压不稳定造成数据丢失。在服务器意外断电的情况下，可以根据 SQL Server 的日志文档，使用 ROLLBACK 机制进行恢复，确保系统在断电后能够尽可能恢复到稳定的状态。
- 在返回错误码时，系统不仅将出错信息返回给客户端，而且输出在服务端控制台以便查看。同时，系统记录错误码、出错信息、以及出错时间到日志文档中，以便后续维护和调整。这种全面记录的做法有助于分析和解决问题，并提高系统的可维护性。
- 为了保持网关与外部网络的通信畅通，系统可以创建一条成本较低的后备网络。这样，即使主网络意外中断，备份网络能够保证通信的继续进行，确保系统在网络异常情况下的稳定性。
- 在硬件方面，需要保证服务器的数量，以确保系统能够在充分的硬件资源支持下稳定运行。增加服务器的数量可以提高系统的负载能力，降低单点故障的风险，确保系统的可用性。

  以下为部分出错及处理对策：

  1. 用户无法登录

  解决方式：验证用户信息，检查用户记录以及测试程序。

  2. 用户无法查看订单或运单信息

  解决方式：核对数据库订单或运单信息，调试程序。

  3. 服务器宕机

  解决方式： 所有服务端应保持持续开机， 以防止停电或电压不稳定造成数据丢失。若服务器意外断电，可根据日志文件，使用回滚机制进行恢复。

  4. 数据库非法修改

解决方式：数据库管理人员定期检查数据库，并且做好数据库备份，防止数据库记录出错造成巨大损失。

5. 网络传输错误

在网络传输方面，保持网关与外部网络的通信畅通。可考虑为其建立一条成本较低的后备网络，以保证在主网络意外中断时仍然能维持正常通信。

6. 硬件设备故障

保证服务器数量不少于已划分模块数量，以确保每个模块能相对独立的运行环境，减少相互干扰。同时，设置备用服务器，以保证主服务器崩溃时可使用备用服务器继续运行系统。

8. 安全保密设计

- **会话管理与单点登录：**OOMALL 系统采用会话管理机制，确保用户在系统中的安全会话。同时，引入单点登录（SSO）机制，使用户在一次登录后能够访问系统中的多个服务，提高用户体验。通过有效的会话管理，系统能够及时终止不活跃的会话，防范会话劫持和其他安全威胁。

- **网关拦截：**OOMALL 电子商城系统设置 GATEWAY 微服务网关模块，微服务网关可以实施身份验证机制，确保只有经过身份验证的用户才能访问系统，并且在微服务架构中对微服务进行鉴权，以保证只有授权的服务才能调用另一个服务。GATEWAY 网关可以检测和防止对 API 的滥用，控制访问的流量的大小，可以防止流入 DDos 攻击或恶意爬虫。

- **权限控制与审计记录：**系统的系统用户管理保证了只有授权的用户才能进入系统进行数据操作，而且对一些重要数据，系统设置为只有更高权限的人员方可读取或是操作。系统安全保密性较高。引入安全审计机制，对系统中的重要操作和事件进行监控和审计。所有的安全事件和操作都记录在安全日志中，以便后续的安全分析和溯源。这有助于及时发现潜在的安全威胁，并进行及时的响应和处理。

- **数据加密与脱敏：**系统在数据传输和存储阶段采用 SSL/TLS 协议进行数据加密，确保用户与服务器之间的通信安全。同时，在存储敏感数据时，采用脱敏技术，对于不需要明文展示的敏感信息进行适度的隐藏，保护用户隐私。

8.1 物理层安全

物理层的安全主要体现在通信线路的可靠性，即线路备份、网管软件、传输介质；软硬件设备安全性，即替换设备、拆卸设备、增加设备；设备的备份；防灾害能力、防干扰能力；设备的运行环境，即温度、湿度、烟尘；不间断电源保障，等等。

在物理层的安全设计中，着重考虑主机的物理安全，包括机柜、硬盘的安全性，内部存储着重要的业务数据。同时应当考虑周边环境，相关的网络设备等，避免周边的环境因素对网络通信设备造成损伤。

系统在实施部署时将采用数据库集群、应用服务器集群技术。数据库有备份并可以恢复。

8.2 网络层安全

系统的应用服务器和数据库服务器将放置在防火墙后方基于用户现有网络安全设备来保障系统的网络安全。

由于数据的传输上需要通过网络传输，为了将资料进行保密，需要在网络的传输过程中对数据进行加密。在加密算法选择上将使用 MD5 加密算法。系统的硬件，软件及其系统中的数据受到保护，不会因为偶然的或者恶意的原因而遭到破坏，更改和泄露，系统连续可靠正常的运行，网络服务不会中断。

监控与入侵检测系统作为系统端的监控进程，负责接受进入系统的所有信息，并对信息包进行分析和归类，对可能出现的入侵及时发出报警信息，同时如发现有合法用户的非法访问和非法用户的访问，监控系统及时断开访问连接，并进行追踪检查。

8.3 系统层安全

软件部署的所有操作系统都将采用 Linux 操作系统。Linux 操作系统病毒少， 本身安全性较高，系统部署时将会对操作系统、中间件、数据库服务进行安全漏洞扫描，根据扫描结果安装相应的补丁，保障系统层的安全。

8.4 应用层安全

软件对用户的信息需保密，不可暴露给不需要的接口，对数据库中信息的操作要有记录并定期检查处理，使用和记录用户信息时，要符合法律法规并告知用户。

1. 所有涉及用户身份信息，订单详情信息，在网络中传输时均需通过加密传输。

2. 数据库中的用户信息需加密存储。

3. 每次用户要求进入系统时，由系统核对账号密码，通过鉴定后才提供系统的使用权。通过用户权限定义和合法权检查确保只有合法权限的用户访问数据库，所有未被授权的人员无法存取数据。

4. 为不同的用户定义视图，通过视图机制把要保密的数据对无权存取的用户隐藏起来，从而自动地对数据提供一定程度的安全保护。

5. 加入其他的附加逻辑判断条件，采用正则表达式配合 SpringSecurity 反 SQL 注入。当发生数据泄露时，及时对数据库的信息进行修补、更新与查询，设计一个安全防护模块，保障数据的安全。

6. 软件应有安全性检查，对于外部 API，使用网关验证，对非法访问内部 API 的请求进行拦截，对系统负荷有动态负载均衡和限制，防止对服务器性能的冲击攻击。

8.5 接口层安全

1. 外部接口：

使用 MySQL, MongoDB 和 Redis，进行对数据库的所有访问，做到对数据的保存。在网络软件接口方面，使用一种无差错的传输协议，采用滑动窗口方式对数据进行网络传输及接受，保证了过程的安全性。

2. 内部接口：

各个模块之间使用内部接口，采用函数调用、参数传递、返回值的方式进行信息传递。接口传递的信息以数据结构封装了的数据，保证其准确性及安全性。

8.6 访问控制

防止对任何资源进行未授权的访问，从而使计算机系统在合法的范围内使用。通过用户身份及其所归属的某项定义组来限制用户对某些信息项的访问，或限制对某些控制功能的使用。对于网络资源保持有限访问的原则，信息流向可根据安全需求实现单向或双向控制。采用防火墙进行访问控制，安置在不同安全区域出入口，对进出网络的 IP 信息包进行过滤并按设定的安全策略进行信息流控制。

9. 维护设计

- **集成开发生产线管理：**OOMALL 系统采用华为 CODEARTS 进行一站式、全流程软件开发生产线管理。这确保了系统的开发流程在看板上可视化，便于团队成员随时了解项目进展。系统的错误和维护信息都被适当地记录和溯源，使得在维护过程中能够更加迅速、准确地定位问题，提高了维护的效率。

- **模块化设计和团队负责：**OOMALL 电子商城系统采用模块化设计，系统维护时仅需对相应模块进行维护。不同模块由团队成员各自负责，确保了维护设计的高效性。这种分工明确的模块化设计降低了维护的复杂性，使得团队能够更加有针对性地处理各个模块的问题，提高了系统整体的可维护性。

- **数据库维护策略：**对于数据库的维护，OOMALL 系统采取定期备份数据库的内容的策略，确保在发生意外事件时能够及时恢复数据。定期检测数据库的一致性，防止数据异常导致系统故障。同时，定期查看操作日志，以监控数据库的操作情况，发现潜在问题并及时进行处理。这些维护措施有效降低了数据库风险，保障了数据的安全性和完整。

- **用户业务操作维护：**对于用户业务操作出错，系统对所有用户输入数据、系统接口调用参数进行检查，如不符合系统规则则返回错误信息，有用户交互界面的需要在界面上显示出错信息。

- **代码维护：**为方便后续代码的修改，设计中尽量保持开闭原则，以减少修改造成的影响， 同时减轻修改代码所需的工作量。 代码采用 Git 方式进行保存，建立统一的 git 仓库，编程人员可建立分支 clone 到本地，修改完毕后发起 merge request，通过审核后并入主仓库。主仓库保留修 改记录，方便回滚与历史查看。

- **系统程序异常维护：**对于系统程序异常，程序模块需要记录错误日志，调用模块需要记录 try catch 得到的异常信息，包括堆栈调用信息，以方便问题的定位和原因查找。 各个程序调用统一的日志模块将错误信息记录到日志文件中，日志文件保存在文件夹中，便于统一管理。