厦門大學

电子商城系统 oomall

售后、服务模块详细设计说明书

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 12 月 8 日

目录

1. 引言.....3
1.1 编写目的.....3
1.2 项目背景.....3
1.3 定义.....3
1.4 参考资料.....4
2. 总体设计.....4
2.1 需求概述.....5
2.2 软件结构.....6
3. 程序描述.....11
3.1 功能.....11
3.2 性能.....13
3.3 输入项目.....14
3.4 输出项目.....16
3.5 算法.....19
3.6 程序逻辑.....20
3.7 接口.....23
3.8 存储分配.....28
3.9 限制条件.....32
3.10 测试要点.....32

1. 引言

1.1 编写目的

本项目为厦门大学信息学院软件工程专业软件工程系大三上学期《面向对象分析与设计》、《软件工程导论》、《JavaEE 平台技术》课程大作业。本文档编写目的在于介绍课程大作业中的售后模块、服务模块的详细设计。

项目的总目标为开发一个能支持高并发、大负载的电子商城系统（OOMALL）的订单模块。用户可以使用此系统完成和目前主流电商平台相似的购物流程，如购物、发货、售后等。本小组负责其中售后、服务模块的开发。项目计划选用 Java 代码，由五人进行敏捷开发，并进行软件功能，性能和安全性等方面的测试。

OOMALL 电子商城系统的售后、服务模块详细设计是设计的第三阶段，本阶段主要任务是在概要设计说明书的基础上，对概要设计中产生的功能模块进行过程性的描述，设计功能模块的内部细节及内部功能。

在此详细说明书中，针对本小组负责的售后、服务模块进行了进一步的系统架构细化，即过程或描述。本文档展示了软件结构的图表、物理设计、数据结构设计及算法设计，同时详细介绍了三个模块的实现过程。包括涉及的算法，逻辑流程等。

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.shtm

- 支付宝互联网平台交易查询接口。

  https://opendocs.alipay.com/open/02e7gm?ref=api

2. 总体设计

2.1 需求概述

2.1.1 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。

2.1.2 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。

2.2 软件结构

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。

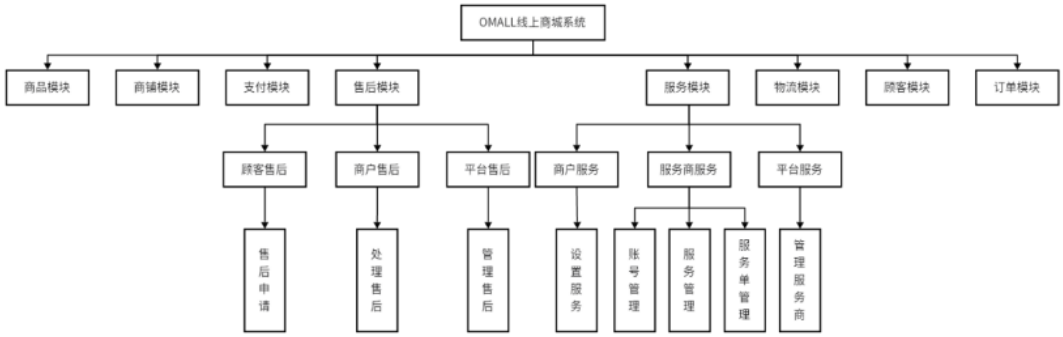

图 1 OOMALL 系统结构图

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：

1. 模型(Model)：包含所有应用特定内容和处理逻辑。
2. 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。
3. 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。

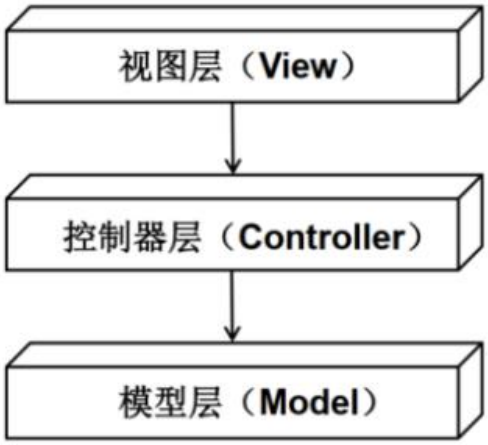

图 2 MVC 层次体系结构

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：

1. 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。

2. 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。

3. 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。

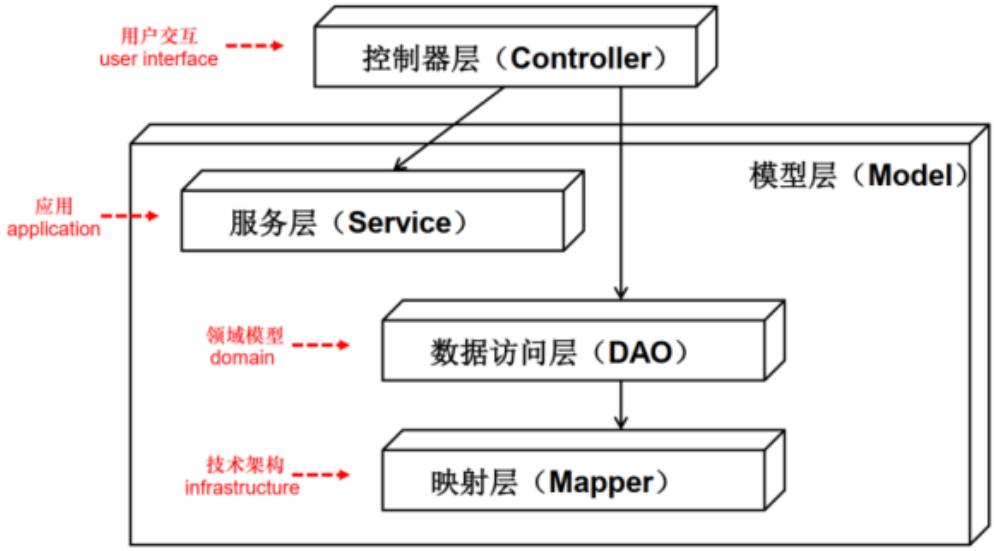

图 3 基于 SpringMVC 的模型层细化

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。

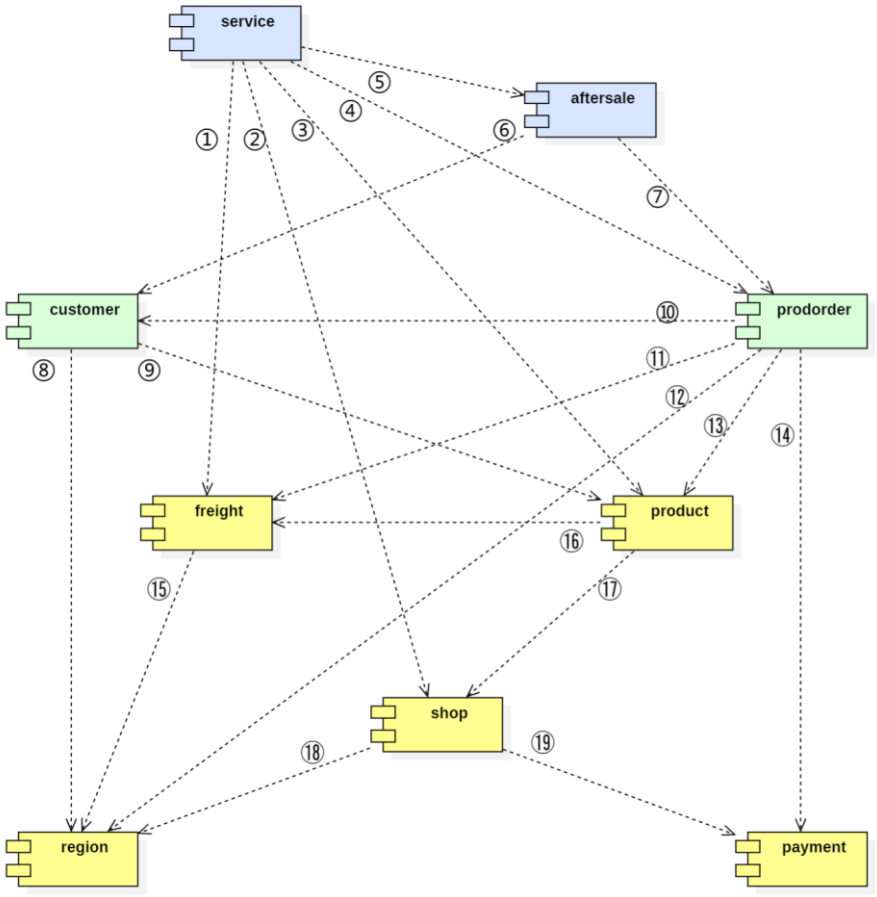

图 4 软件结构示意图

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. 程序描述

3.1 功能

3.1.1 售后模块

售后模块主要包含以下功能：

1. 顾客提交售后申请：顾客可以通过相关 API 进入售后申请界面提交售后单，设置售后单状态为未售后；
2. 顾客查询售后列表：顾客可以通过相关 API 查询所有售后单列表，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；
3. 顾客查询订单明细：顾客可以通过相关 API 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；
4. 顾客查询售后单信息：顾客可以通过 API 查询到自己的售后单详细信息；
5. 顾客修改售后单信息：顾客可以通过相关 API 将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；
6. 顾客取消售后：顾客可以通过相关 API 将未售后和售后中的状态的售后单取消；

7. 顾客申请售后仲裁：顾客可以通过相关 API 提出仲裁申请，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；
8. 顾客申请二次仲裁：顾客可以通过相关 API 提出二次仲裁申请（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；
9. 顾客取消仲裁：顾客可以通过相关 API 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；
10. 顾客查询售后单对应的物流单号：顾客可以通过相关 API 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；
11. 商铺审核售后：商铺可以通过相关 API 审核顾客提出的售后请求并设置售后状态为售后中或已售后；
12. 商铺验收售后商品：商铺可以通过相关 API 将售后商品确认为已验收，维修商品不可调用此接口；
13. 商铺取消售后：商铺可以通过相关 API 取消售后，需要设置相应售后为正常；
14. 商铺应诉仲裁：顾客可以通过相关 API 提出仲裁，商铺应诉仲裁；
15. 管理员受理仲裁单：管理员可以通过相关 API 受理相关的仲裁单，仅申请中的仲裁单才能被受理；
16. 管理员仲裁结案：管理员可以通过相关 API 将仲裁单结案，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；
17. 查看所有售后单：商铺或管理员可通过相关 API 根据售后单 ID 查看所有售后单；
18. 查询售后单信息：商铺或管理员可通过相关 API 根据售后单 ID 查询售后单信息；
19. 查询顾客申请的仲裁单信息：商铺或管理员可通过相关 API 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；
20. 查询仲裁单详情：商铺或管理员可通过相关 API 根据仲裁单 ID 查询仲裁单详情信息

3.1.2 服务模块

服务模块主要包含以下功能：

1. 商户选择商品服务：商户可以通过相关 API 选择服务商提供的地区服务，与具体商品关联；
2. 商户修改商品服务：商户可以通过相关 API 修改商品服务(暂停：从有效变为无效；恢复：从无效变为有效)；
3. 商户取消商品服务：商户可以通过相关 API 取消商品服务；
4. 商户通过商品和地区查询服务商：商户可以通过相关 API 通过商品和地区找到服务商；
5. 商户通过服务商 ID 查询服务商：商户可以通过相关 API 用服务商 ID 找到服务商；
6. 商户取消服务商的所有服务：商户可以通过相关 API 取消指定 ID 的服务商的所有服务；

7. 获得服务单的所有状态：可以通过相关 API 获得服务单的所有状态；
8. 商户创建服务单：商户可以通过相关 API 创建服务单；
9. 获得服务单信息：管理员或商户可以通过相关 API 获得服务单信息；
10. 商户根据 id 查询服务单：商户可以通过相关 API 查询指定 ID 的服务单；
11. 取消服务单：管理员或商户可以通过相关 API 取消服务单；
12. 服务商注册账号：服务商可以通过相关 API 注册账号；
13. 服务商申请变更账号信息：服务商可以通过相关 API 申请变更账号信息；
14. 服务商注销账号：服务商可以通过相关 API 注销账号，完成所有服务单后才可注销；
15. 服务商查看账户信息：服务商可以通过相关 API 查看账户信息，需检查服务商 ID 是否与访问者的一致；
16. 管理员查询服务商：管理员可以通过相关 API 查询服务商；
17. 管理员审核服务商开户：平台管理员可以通过相关 API 审核服务商开户；
18. 管理员审核服务商变更：平台管理员可以通过相关 API 审核服务商变更；
19. 管理员暂停服务商：平台管理员可以通过相关 API 暂停服务商，修改服务商状态为暂停；
20. 管理员恢复服务商：平台管理员可以通过相关 API 恢复服务商，修改服务商状态为有效；
21. 管理员审核服务：平台管理员可以通过相关 API 审核服务；
22. 服务商查询服务单信息：服务商可以通过相关 API 查询关联商家派发的服务单和已完成的服务单；
23. 服务商根据 ID 查询服务单信息：服务商可以通过相关 API 查询指定 ID 的服务单信息；
24. 服务商接受服务单：服务商可以通过相关 API 接受服务单；
25. 服务商接受服务单：服务商可以通过相关 API 接受服务单；
26. 服务商收到寄出商品：服务商可以通过相关 API 验收一个商品，并反馈验收状态；
27. 服务商完成服务单：服务商可以通过相关 API 完成一个服务单，并反馈服务状态；
28. 服务商取消服务：服务商可以通过相关 API 取消一个服务单；
29. 服务商定义服务：服务商可以通过相关 API 定义在某个或多个地区为某类商品的服务；
30. 服务商取消服务：服务商可以通过相关 API 取消某一服务；
31. 服务商修改服务范围：服务商可以通过相关 API 修改某服务的服务范围；
32. 服务商查询服务：服务商可以通过相关 API 查询自己注册的服务；
33. 服务商根据 ID 查询服务：服务商可以通过相关 API 根据服务 ID 查询服务信息

3.2 性能

服务和售后模块要求实现高负载大并发，在高峰时段可以支持多顾客频繁发起售后单的操作。要实现多并发，满足 500 个用户同时下售后\服务单，同

时保证售后单信息和服务单信息的一致性和对应关系；要支持高负载，满足 1 秒实现 1000 个下单操作的性能。

3.3 输入项目

3.3.1 售后模块

功能	用户输入信息
顾客提交售后申请	订单 ID，售后服务信息
根据售后状态分类查询售后列表	售后状态，页码，每页数目
顾客查询所有的可申请的订单明细	页码，每页数目
顾客查询售后单信息	售后单 ID
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
顾客取消售后	售后单 ID
顾客申请售后仲裁	售后单 ID，售后信息
顾客申请二次仲裁	售后单 ID，仲裁详细信息
顾客取消仲裁	售后单 ID，仲裁单 ID
查询售后单对应的物流单号	售后单 ID
商铺审核售后	商铺 ID，售后单 ID，处理意见
商铺验收售后商品	商铺 ID，售后单 ID，处理意见
商铺取消售后	商铺 ID，售后单 ID
商铺应诉仲裁	商铺 ID，仲裁单 ID，仲裁详细信息
管理员受理仲裁单	商铺 ID（只能为 0），仲裁单 ID，仲裁员信息
管理员仲裁结案	商铺 ID（只能为 0），仲裁单 ID，仲裁结果信息
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

查询售后单信息	商铺 ID（管理员为 0），售后单 ID
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
查询仲裁单详情	商铺 ID（管理员为 0），仲裁单 ID

3.3.2 服务模块

功能	用户输入信息
商户选择商品服务	商铺 ID，商品 ID，服务商 ID
商户修改商品服务	商铺 ID，商品服务 ID
商户取消商品服务	商铺 ID，商品服务 ID
商户通过商品和地区查询服务商	商铺 ID，商品 ID，地区 ID
商户通过服务商 id 查询服务商	商铺 ID，服务商 ID
商户取消服务商的所有服务	商铺 ID，服务商 ID
获得服务单的所有状态	无
商户创建服务单	商铺 ID，商品 ID，地区 ID
获得服务单信息	商铺 ID
商户根据 id 查询服务单	商铺 ID，服务单 ID
取消服务单	商铺 ID，服务单 ID
服务商注册账号	服务商名称，密码，联系人，保证金等
服务商申请变更账号信息	服务商 ID，服务商信息
服务商注销账号	服务商 ID
服务商查看账户信息	服务商 ID
管理员查询服务商	管理员 ID，服务商类型，状态，名称
管理员审核服务商开户	管理员 ID，服务商 ID，审核结果
管理员审核服务商变更	管理员 ID，服务商 ID，审核结果
管理员暂停服务商	管理员 ID，服务商 ID

管理员恢复服务商	管理员 ID，服务商 ID
管理员审核服务	管理员 ID，服务 ID，审核结果
服务商查询服务单信息	服务商 ID
服务商根据 id 查询服务单信息	服务商 ID，服务单 ID
服务商接受服务单	服务商 ID，服务单 ID
服务商接受服务单	服务商 ID，服务单 ID
服务商收到寄出商品	服务商 ID，服务单 ID
服务商完成服务单	服务商 ID，服务单 ID
服务商取消服务单	服务商 ID，服务单 ID
服务商定义服务	服务商 ID，商品类别 ID，地区 ID
服务商取消服务	服务商 ID，服务 ID
服务商修改服务范围	服务商 ID，服务 ID，地区 ID
服务商查询服务	服务商 ID，服务状态，服务地区
服务商根据 id 查询服务	服务商 ID，服务 ID

3.4 输出项目

3.4.1 售后模块

功能	正常输出	异常输出
顾客提交售后申请	系统提示操作成功并返回售后单号	
根据售后状态分类查询售后列表	系统提示操作成功并按页返回售后信息列表	
顾客查询订单明细	系统提示操作成功并返回可售后的订单详细信息	
顾客查询售后单信息	系统提示操作成功并返回售后单详细信息	

顾客修改售后单信息	系统提示操作成功	
顾客取消售后	系统提示操作成功	
顾客申请售后仲裁	系统提示操作成功并返回售后仲裁信息	系统返回错误码 705（已经在仲裁中的售后不允许再仲裁）
顾客申请二次仲裁	系统提示操作成功并返回售后二次仲裁信息	系统返回错误码 705（已经在仲裁中的售后不允许再仲裁）
顾客取消仲裁	系统提示操作成功	
查询售后单对应的物流单号	系统提示操作成功并返回物流单号	
商铺审核售后	系统提示操作成功	
商铺验收售后商品	系统提示操作成功	
商铺取消售后	系统提示操作成功	
商铺应诉仲裁	系统提示操作成功并返回仲裁单详情	
管理员受理仲裁单	系统提示操作成功	
管理员仲裁结案	系统提示操作成功	
查看所有售后单	系统提示操作成功并按页返回售后单列表	
查询售后单信息	系统提示操作成功并返回售后单详细信息	
查询顾客申请的仲裁单信息	系统提示操作成功并按页返回仲裁单列表	
查询仲裁单详情	系统提示操作成功并返回仲裁单详细信息	

3.4.2 服务模块

功能	异常输出	异常输出
商户选择商品服务	系统返回错误码 801（商品服务已经存在，无法重复选择）	系统返回错误码 801（商品服务已经存在，无法重复选择）
商户修改商品服务	系统提示操作成功	系统返回错误码 504（修改商品服务时发现该服务商被禁止或者服务商并不存在
商户取消商品服务	系统提示操作成功	

商户通过商品和地区查询服务商	系统提示操作成功并按页返回服务商信息	
商户通过服务商 id 查询服务商	系统提示操作成功并返回服务商信息	
商户取消服务商的所有服务	系统提示操作成功	
获得服务单的所有状态	系统提示操作成功并返回服务单的所有状态	
商户创建服务单	系统提示操作成功并返回服务单 ID	
获得服务单信息	系统提示操作成功并按页返回服务单信息列表	
商户根据 id 查询服务单	系统提示操作成功并返回服务单信息	
取消服务单	系统提示操作成功	
服务商注册账号	系统提示操作成功并返回服务商 ID	
服务商申请变更账号信息	系统提示操作成功	
服务商注销账号	系统提示操作成功	
服务商查看账户信息	系统提示操作成功并返回服务商信息	
管理员查询服务商	系统提示操作成功并按页返回服务商信息列表	
管理员审核服务商开户	系统提示操作成功	
管理员审核服务商变更	系统提示操作成功	
管理员暂停服务商	系统提示操作成功	
管理员恢复服务商	系统提示操作成功	
管理员审核服务	系统提示操作成功	
服务商查询服务单信息	系统提示操作成功并按页返回服务单信息列表	
服务商根据 id 查询服务单信息	系统提示操作成功并返回服务单信息	
服务商接受服务单	系统提示操作成功	
服务商接受服务单	系统提示操作成功	
服务商收到寄出商品	系统提示操作成功	
服务商完成服务单	系统提示操作成功	

服务商取消服务单	系统提示操作成功	
服务商定义服务	系统提示操作成功并返回服务 ID	
服务商取消服务	系统提示操作成功	
服务商修改服务范围	系统提示操作成功	
服务商查询服务	系统提示操作成功并按页返回服务信息列表	
服务商根据 id 查询服务	系统提示操作成功并返回服务信息	

3.5 算法

在售后和服务模块中，我们主要用到的是 AES（Advanced Encryption Standard）对称加密算法，它被广泛用于保护敏感数据的安全性。在售后和服务模块中，AES 加密算法可以发挥关键作用，确保用户的敏感信息在存储和传输过程中得到有效保护。

以下是该算法可以在我们的模块中运用的部分：

1. 用户数据保护

   将用户的个人信息、密码、电话号码、电子邮件等敏感数据进行加密，确保这些数据在数据库中存储时不容易被未经授权的人访问。提高用户数据的安全性，防止潜在的数据泄露风险。

2. 安全认证和授权

   在用户身份验证和授权阶段使用加密算法，确保用户的身份信息在传输和存储过程中不容易被伪造或冒用。提高系统对合法用户的辨识能力，防止身份伪造等攻击，保障系统的安全性。

3. 数据传输安全

   在与外部系统或服务进行数据交互时，通过 AES 加密算法对数据进行加密，确保数据在传输过程中不易被窃取或篡改。保护用户敏感信息在不同系统之间的传输安全，防范数据劫持、中间人攻击等风险。

3.6 程序逻辑

3.6.1 售后模块

3.6.1.1 业务流程图

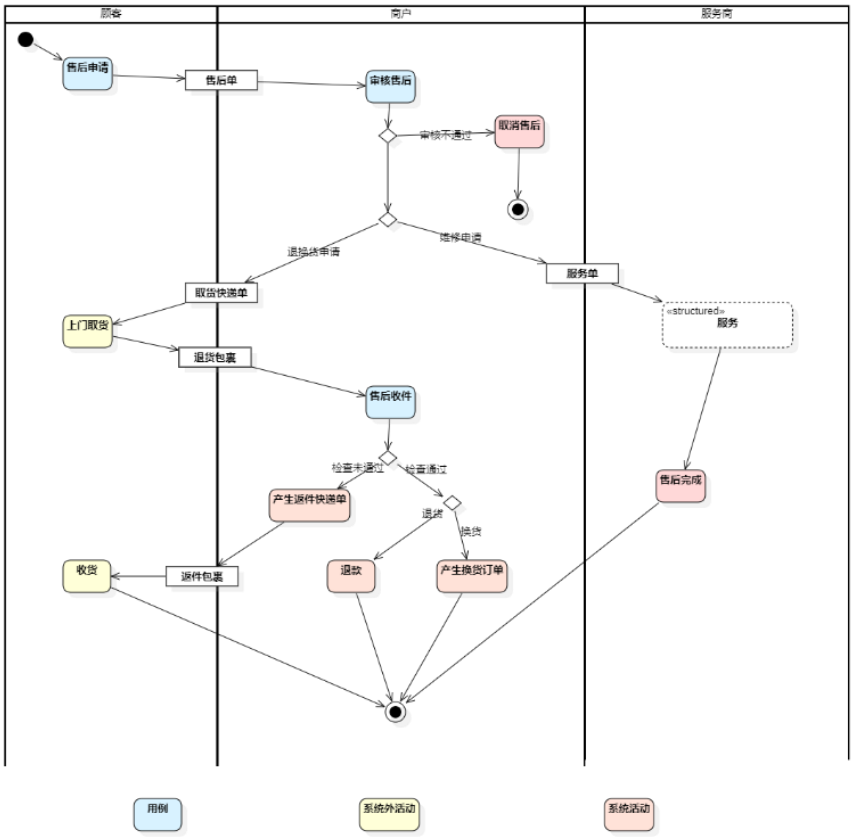

图 售后业务流程图

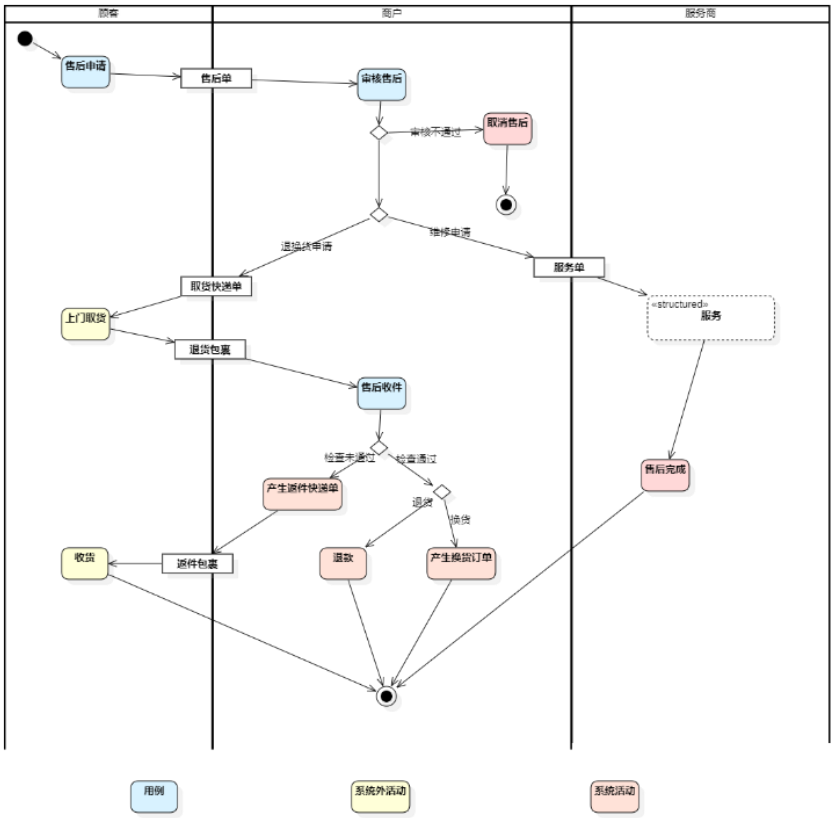

图 售后仲裁业务流程图

3.6.1.2 判断表

顾客修改售后单信息

		1	2	3
条件	售后单状态	=0	=1	=2
操作	允许修改售后单	√		
	拒绝修改售后单		√	√

（售后单状态取值说明：未售后 0，售后中 1，已售后 2）

顾客申请二次仲裁

		1	2	3	4

条件	仲裁记录数量	= 0	= 1	= 1	> 1
	当前仲裁状态	=0	<=2	= 3	<=4
操作	允许二次仲裁			√	
	不允许二次仲裁	√	√		√

(仲裁状态取值说明：未仲裁 0，仲裁中 1，取消仲裁 2，已仲裁 3，二次仲裁 4)

3.6.2 服务模块

3.6.2.1 业务流程图

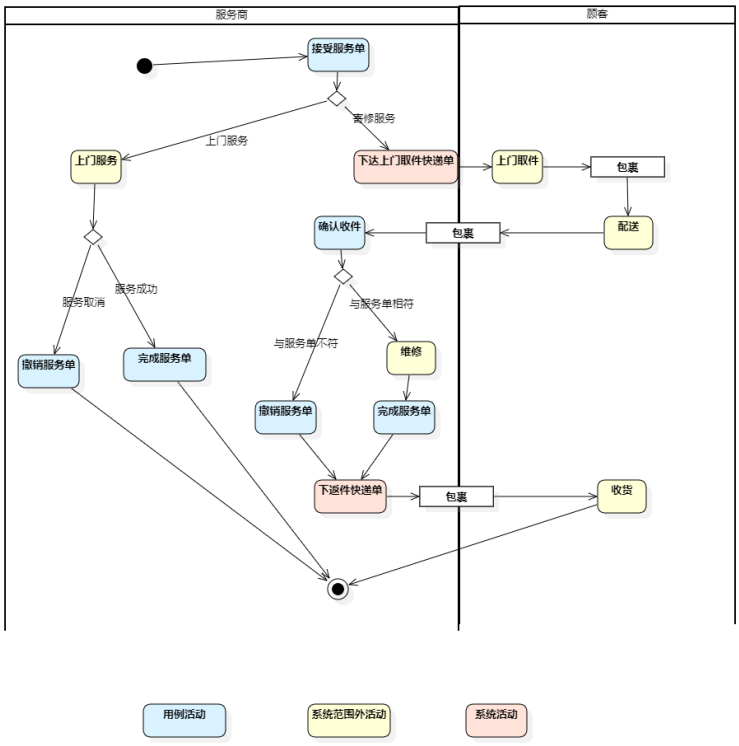

图 服务业务流程图

3.6.2.2 判断表

服务商处理服务单

		1	2	3	4	5
条件	服务单状态	0	1	2	3	4
	登录状态	1	1	1	1	1
操作	接收服务单	√				
	查看服务单	√	√	√	√	√
	取消服务单			√		
	完成服务单			√		

（服务单状态取值说明：待分配 0，未处理 1，处理中 2，已处理 3，取消 4）

（登录状态说明：未登录 0，已登录 1）

3.7 接口

3.7.1 售后模块

功能	类型	API 接口	接口描述
顾客提交售后申请	POST	/orderitems/{id}/aftersales	用户 token 验证当前为顾客身份操作资源，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
顾客或者商铺根据售后状态分类查询售后列表	GET	/aftersales	默认所有售后单列表，选择按售后单状态作为条件查询。 用户 token 验证身份，顾客和商铺只能查询到自己的售后单
顾客查询所有的可申请售后的订单明细	GET	/aftersales/orderitems	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
顾客根据售后单 id 查询售后单信息	GET	/aftersales/{id}	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	PUT	/aftersales/{id}	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	DELETE	/aftersales/{id}	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	POST	/aftersales/{id}/arbitrations	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	POST	/aftersales/{id}/second_arbitrations	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	DELETE	/aftersales/{id}/arbitrations	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
顾客、商铺、管理员查询售后单对应的物流单号	GET	/aftersales/{id}/express	顾客、商铺、管理员通过 token 验证. 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	PUT	/shops/{shopid}/aftersales/{id}/confirm	商铺审核顾客提出的售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	PUT	/shops/{shopid}/aftersales/{id}/receive	维修商品不可调用此 API
商铺取消售后	DELETE	/shops/{shopid}/aftersales/{id}/cancel	商铺取消售后，需要设置相应售后为正常
商铺应诉仲裁	PUT	/shops/{shopid}/arbitrations/{id}/response	顾客提出仲裁，商铺应诉仲裁
管理员受理仲裁单	PUT	/shops/{shopid}/arbitrations/{id}/accept	仅申请中的仲裁单才能被受理
管理员仲裁结案	PUT	/shops/{shopid}/arbitrations/{id}/close	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
商铺或管理员查看所有售后单	GET	/shops/{shopid}/aftersales	商铺或管理员可通过售后单 ID、售后类型和状态选择查看所有售后单
商铺或管理员查询售后单信息	GET	/shops/{shopid}/aftersales/{id}	商铺或管理员根据售后单 id 查询售后单信息
商铺或管理员查询顾客申请的仲裁单信息	GET	/shops/{shopid}/arbitrations	商铺或管理员查询顾客申请的仲裁单信息（根据仲裁单状态分类查询），按照申请时间从近到远排序
商铺或管理员查询仲裁单详情	GET	/shops/{shopid}/arbitrations/{id}	商铺或管理员根据仲裁单 ID 查询仲裁单详情信息

3.7.2 服务模块

功能	类型	API 接口	接口描述
商户选择服务商提供的地区服务	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。 需要登录，服务的状态默认为无效，priority 默认为 1000
商户修改服务商在某个地区的商品服务	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)
商户取消服务商在某个地区的商品服务	DELETE	/shops/{did}/product_services/{id}	商户取消商品服务； 需要登录
商户通过商品和地区查询服务商	GET	/shops/{did}/products/{id}/region/{rid}	商户通过商品和地区找到服务商，可以用下级地区查找，返回结果按照优先级排序（已关闭的服务商不返回）
商户通过服务商 id 查询服务商	GET	/shops/{did}/maintainers/{mid}	商户通过服务商 id 找到服务商，已经关闭的服务商不返回
商户取消服务商的所有服务	DELETE	/shops/{did}/maintainers/{mid}	商户取消指定 ID 的服务商的所有服务； 需要登录

获得服务单的所有状态	GET	/services/states	获得服务单的所有状态
商户创建服务单	POST	/internal/shops/{did}/products/{id}//region/{rid}/serviceOrders	商户创建服务单
商户查询服务单信息	GET	/shops/{did}/serviceOrders	商户通过这个 API 只能查询到自己的服务单，平台管理人员可以查询所有的服务单
商户根据服务单 id 查询服务单信息	GET	/shops/{did}/serviceOrder/{id}	商户查询指定 ID 的服务单，商户通过这个 API 只能查询到自己的服务单，平台管理人员可以查询所有的服务单
管理员或商家取消服务单	DELETE	/shops/{did}/serviceOrder/{id}	商家只能取消自己的服务单，管理员可以取消所有的服务单
服务商注册账号	POST	/maintainers	服务商注册账号时无需登录，注册的账号在审核态
服务商申请变更账号信息	PUT	/maintainers/{mid}	服务商申请变更账号信息，在除关闭态以外的状态均可以修改
服务商注销账号	DELETE	/maintainers/{mid}	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	GET	/maintainers/{mid}	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	GET	/adminusers/{aid}/maintainers	管理员查询服务商，会返回所有符合状态的服务商
管理员审核服务商开户	PUT	/adminusers/{aid}/maintainers/{mid}/account	平台管理员修改服务商状态为开户状态的服务商
管理员审核服务商变更	PUT	/adminusers/{aid}/maintainers/{mid}/modify	平台管理员修改服务商状态为变更状态的服务商
管理员暂停服务商	PUT	/adminusers/{aid}/maintainers/{mid}/suspend	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	PUT	/adminusers/{aid}/maintainers/{mid}/resume	平台管理员恢复服务商，修改服务商状态为恢复服务

管理员审核服务	PUT	/adminusers/{aid}/services/{sid}/service	平台管理员审核服务
服务商查询服务单信息	GET	/maintainers/{mid}/serviceOrders	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	GET	/maintainers/{mid}/serviceOrders/{id}	服务商查询指定 ID 的服务单信息，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
服务商接受服务单	PUT	/maintainers/{mid}/serviceOrders/{id}/accept	服务商接受服务单，当类型为寄件时，需要产生一个上门取件的物流单与之关联
服务商拒绝服务单	PUT	/maintainers/{mid}/serviceOrders/{id}/refuse	服务商拒绝服务单后，重新派单， 派给下一优先级的服务商
服务商收到寄出商品	PUT	/maintainers/{mid}/serviceOrders/{id}/receive	服务商验收一个商品，并反馈验收状态
服务商完成服务单	PUT	/maintainers/{mid}/serviceOrder/{id}/finish	服务商完成一个服务单，并反馈服务状态，当类型为寄件时，需要产生一个寄件的物流单与之关联
服务商取消服务单	PUT	/maintainers/{mid}/serviceOrders/{id}/cancelOrder	服务商取消一个服务单
服务商定义在某个或多个地区为某种商品的服务*	POST	/maintainers/{mid}/categories/{id}/service	服务商定义在某个或多个地区为某类商品的服务，需要登录； 服务默认为无效， priority 默认为 1000
服务商取消服务	PUT	/maintainers/{mid}/services/{id}/cancel	服务商取消某一服务
服务商修改服务范围	PUT	/maintainers/{did}/services/{id}/changeRegion	服务商修改某服务的服务范围
服务商查询服务	GET	/maintainers/{mid}/services	服务商通过这个 API 查询自己注册的服务
服务商根据 id 查询服务	GET	/maintainers/{mid}/service/{id}	服务商根据服务 id，通过这个 API 查询自己注册的服务

3.8 存储分配

3.8.1 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。

3.8.2 存取方法选择

3.8.2.1 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。

- 为售后表的售后类型添加 B+树索引：售后表中的售后类型属性可能需要进行查询。为售后类型属性添加 B+树索引将支持高效的范围查询，提高查询操作的性能。
- 服务单表的服务状态字段添加 B+树索引：服务单表中的服务状态属性字段较常使用。通过为服务状态添加 B+树索引，可以在排序和检索方面提供更高的效率，尤其在需要查询可接服务单和查看未结单服务单时。

3.8.2.2 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。

3.8.2.3 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。

3.8.3 存放位置和存储结构

3.8.3.1 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。

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。

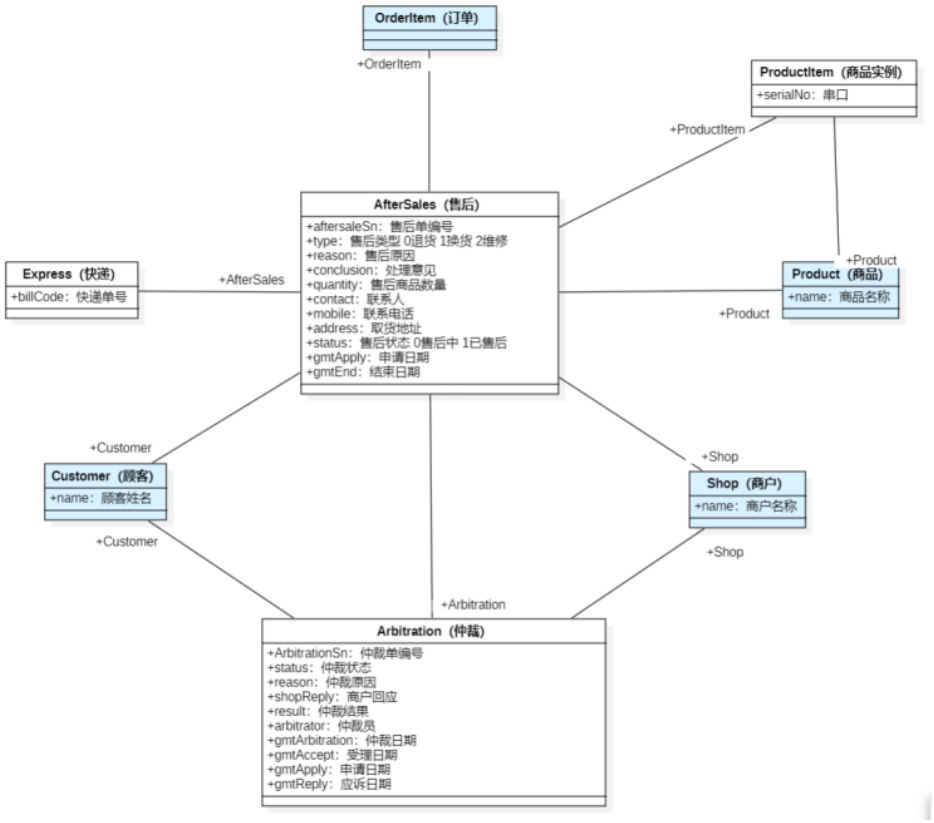

图 5 售后模块对象模型图

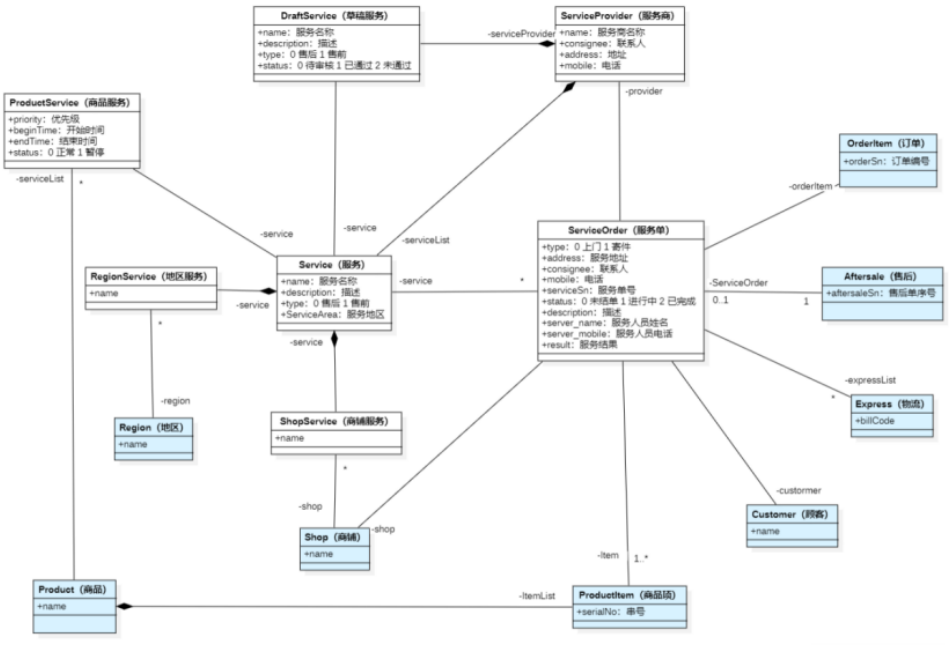

图 6 服务模块对象模型图

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。

3.8.3.2 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。

3.8.4评价物理结构

1. 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；
2. 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；
3. 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。

3.9 限制条件

1. 服务器必须正常启动，服务器有连上各服务之间能访问。
2. 网络状况必须良好。
3. 对于大部分 API，需要用户登录，进行身份检查。

3.10 测试要点

单元测试对每个功能模块进行测试，采用各种测试用例对模块接口、模块内部数据结构、逻辑路径、出错处理和边界条件进行测试。如模块内部数据结构的测试，在程序编写过程中就要考虑数据的类型、范围等方面，测试时就要对这些方面进行测试。最终目的是保证每个模块单独运行正确。

单元测试完成后，还需要进行集成测试。将所有模块按照设计要求（如根据结构图）组装成为子系统或系统，进行集成测试。

1. 测试要求：

（1）要求所有 API 返回的数据正确；

（2）要求 API 能够正确的处理异常；

（3）要求 API 能够较快的响应请求。

2. 测试用例：根据等价类划分法，选择测试用例。

3. 测试方法：使用白盒测试方法，基本路径测试方法。

3.10.1 售后模块

1. 检验售后模块在顾客申请服务后，是否能正确产生服务单。
2. 检验售后模块在申请仲裁后，是否能正确产生对应仲裁信息。
3. 对于售后单的增删改查举例测试并观察测试结果是否符合逻辑规律。
4. 检验售后单状态有无实时确认并更新当前状态。

3.10.2 服务模块

对服务单的增删改查举例测试并观察测试结果是否符合逻辑规律。