

2-4: OOMALL 售后模块设计说明书

1. 前言	2
1.1 目的	2
1.2 预期读者	2
1.3 项目背景	2
2. 系统概述	2
2.1 系统目标	2
2.2 运行环境	2
2.2.1 硬件平台	2
2.2.2 软件环境	2
2.3 系统总体结构	2
2.4 包	3
3. 对象模型	4
3.1 领域对象	4
3.2 对象状态	5
3.2.1 售后单状态	5
3.2.2 仲裁单状态	6
4. 数据库	6
5. 程序逻辑	7
5.1 售后	7
5.1.1 静态模型	7
5.2 仲裁	10
5.2.1 静态模型	10
5.2.2	10

1. 前言

1.1 目的

描述 OOMALL 系统售后模块的设计思路以及代码的结构，方便协调开发工作和后期维护。

1.2 预期读者

相关模块的开发人员。

1.3 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。支付模块的设计和实现开始于 2022 年。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。

2. 系统概述

2.1 系统目标

实现系统中售后模块的相关功能。

2.2 运行环境

2.2.1 硬件平台

华为云耀云服务器集群。

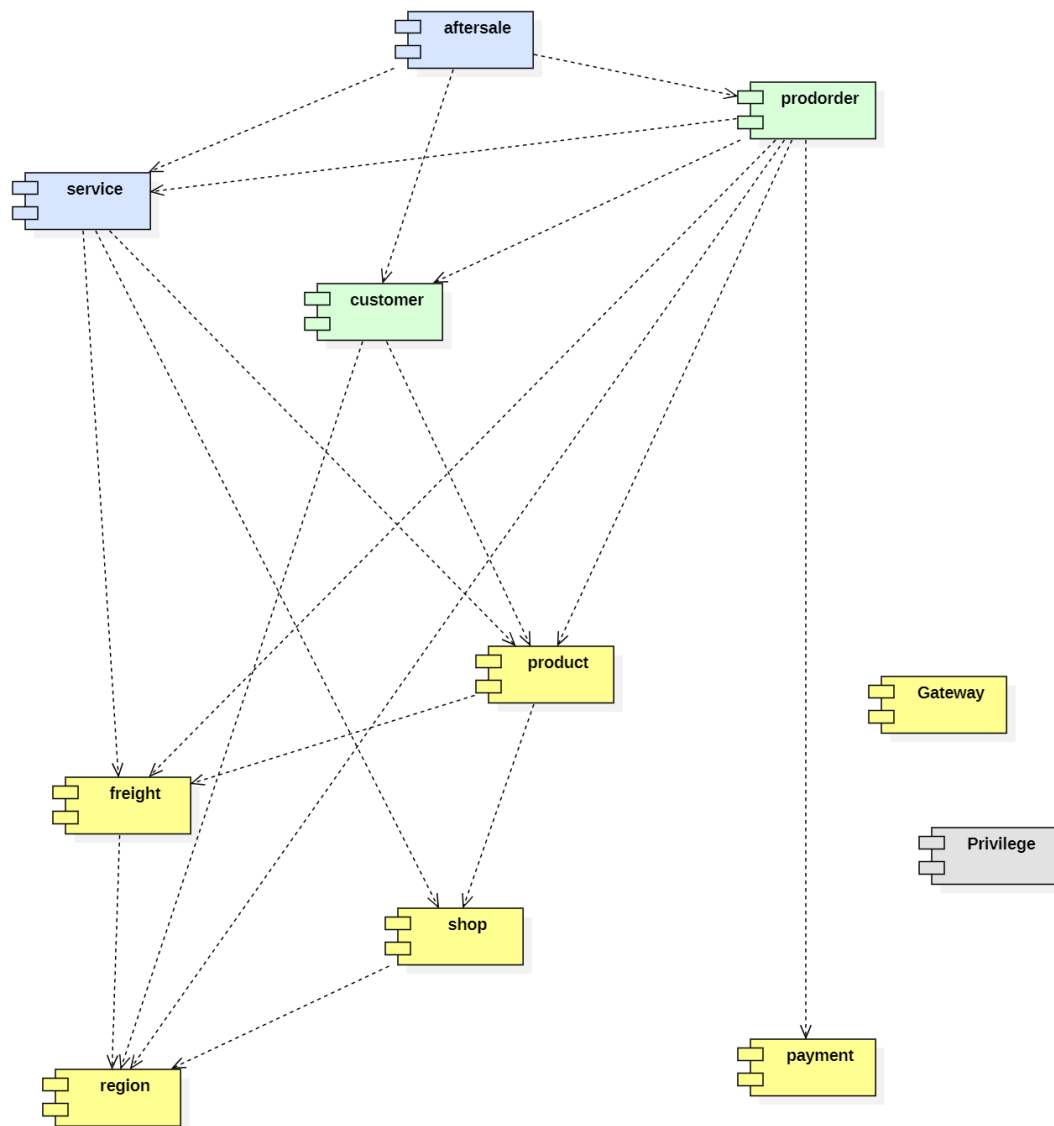
2.2.2 软件环境

操作系统：Ubuntu 22.04

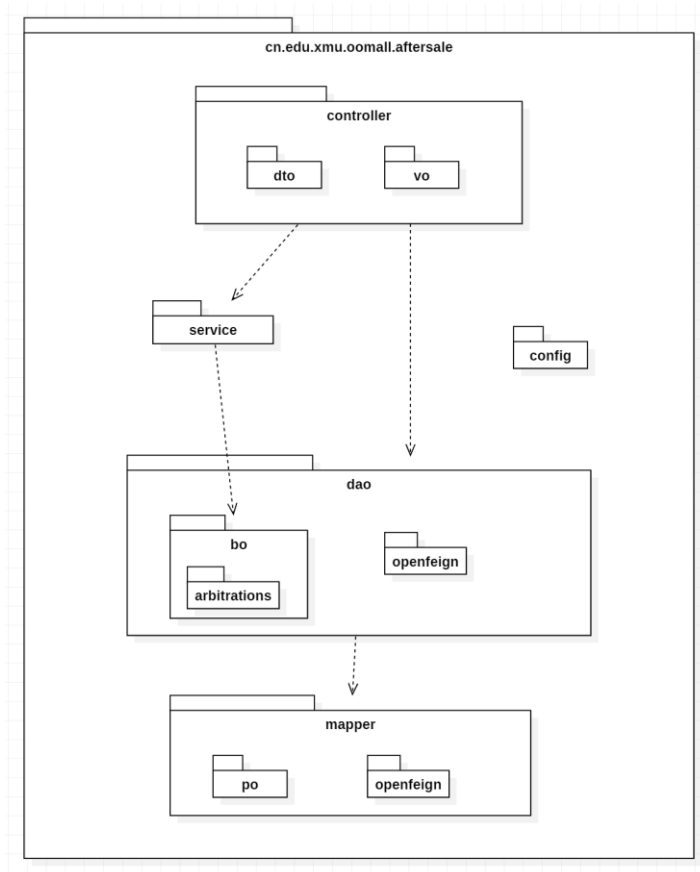
支持环境：Docker Swarm, Tomcat 10.0.23

数据库：MySQL, Mongo

2.3 系统总体结构

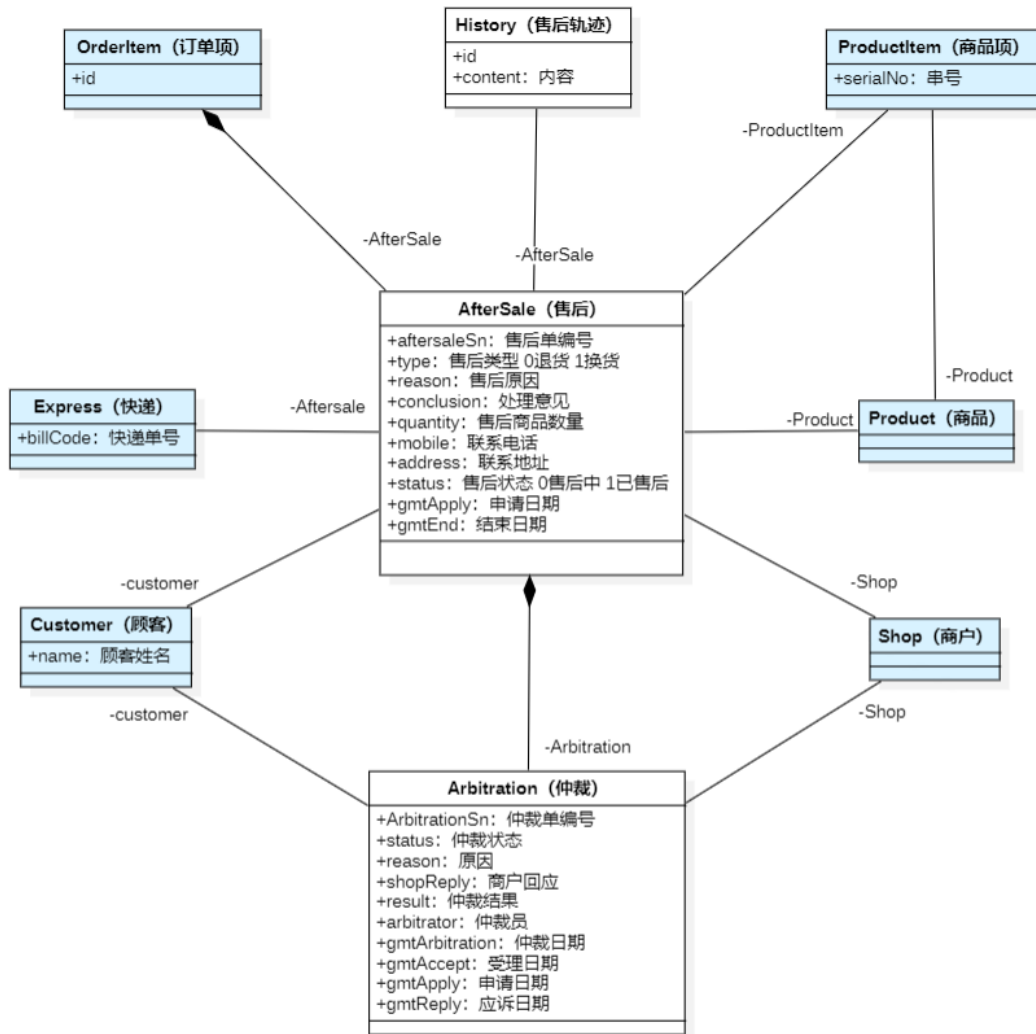


2.4 包



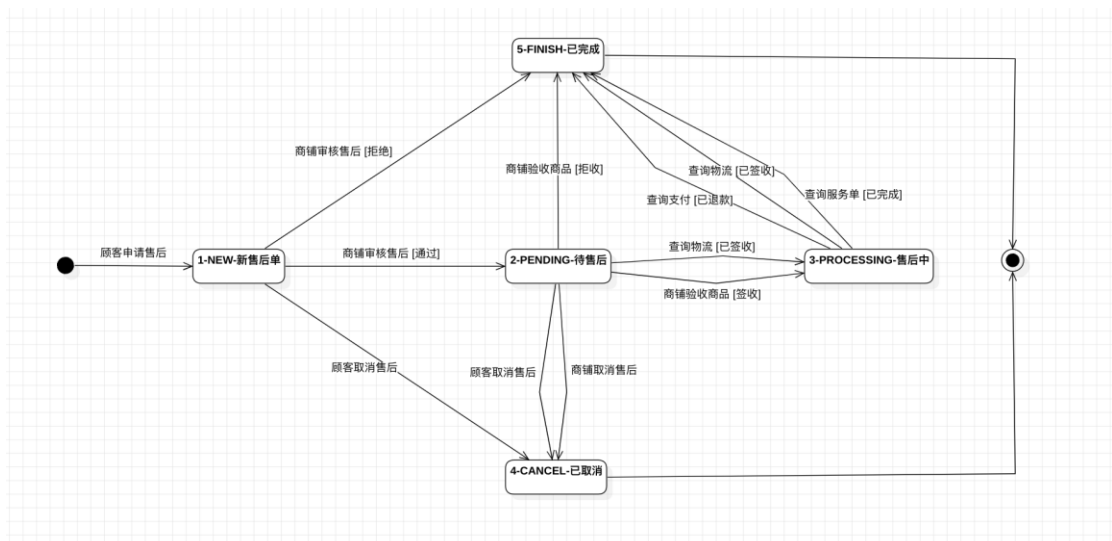
3. 对象模型

3.1 领域对象



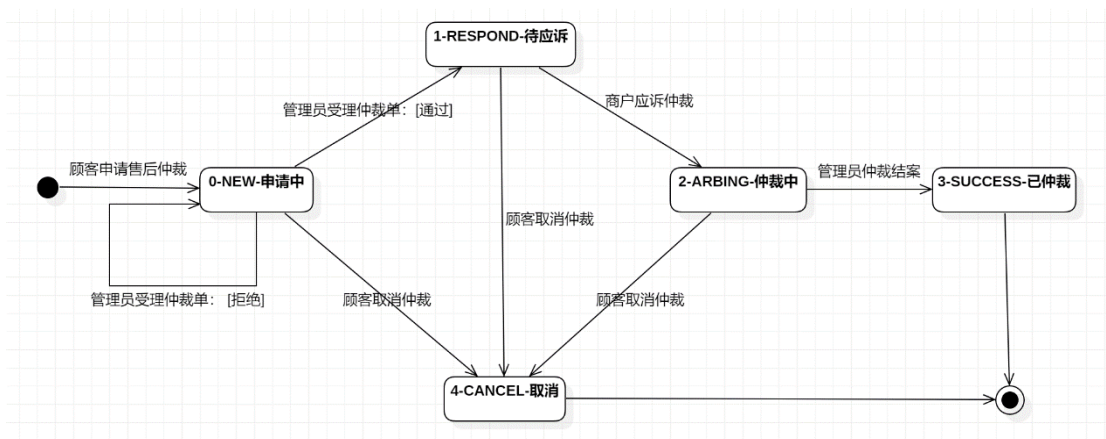
3.2 对象状态

3.2.1 售后单状态

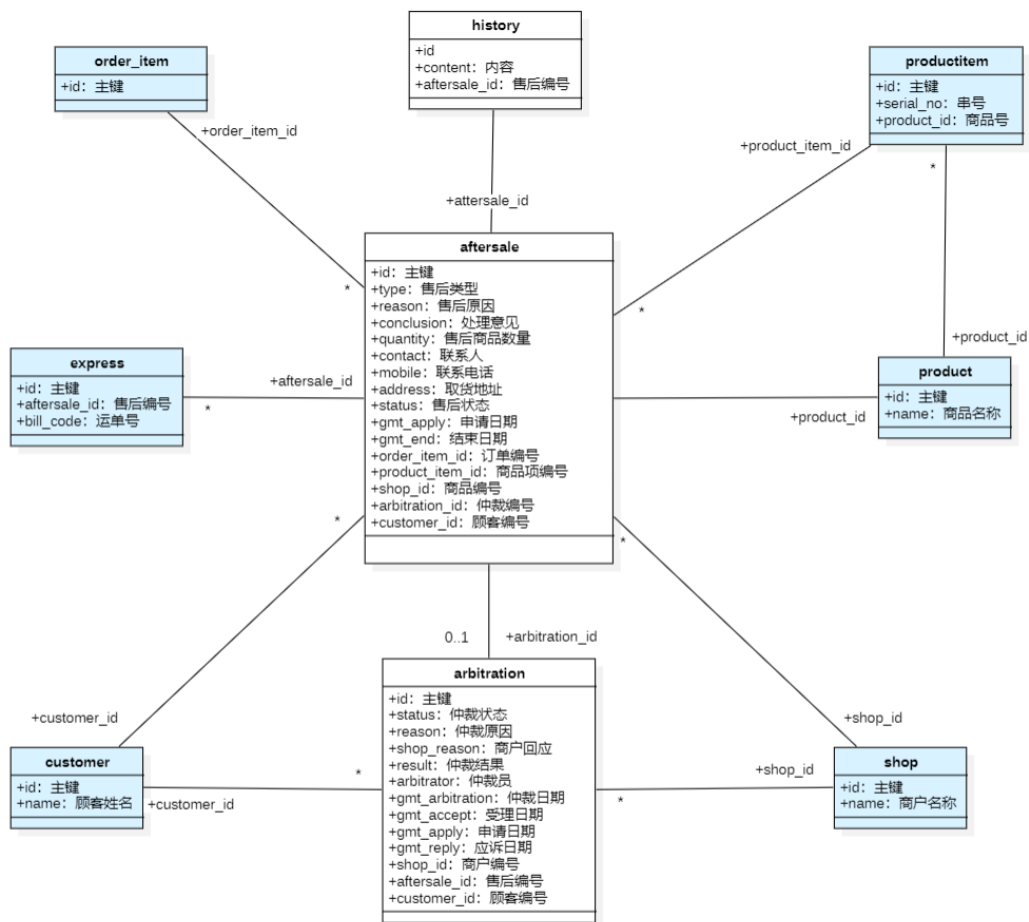


1. 顾客提交申请，创建一个状态为 1-NEW 的售后单；
2. 若商铺审核通过售后，售后单变为 2-PENDING 状态，等待物流上门取件；
若商铺审核拒绝售后，售后单则变为 5-FINISH 已完成状态，顾客可以申请平台介入，即申请仲裁。
3. 若是退换货售后，且商铺验收通过商品，售后单变为 3-PROCESSING 状态；
若是退换货售后，且商铺验收不通过，则将商品寄回，售后单变为 5-FINISH；
若是维修售后，且签收成功，售后单变为 3-PROCESSING 状态。
4. 若是退货售后，且退款支付成功，售后单变为 5-FINISH；
若是换货售后，且换货相应的物流单顾客已签收，售后单变为 5-FINISH 状态；
若是维修售后，且查询服务单完成，售后单变为 5-FINISH 状态。
5. 顾客可以在 1-NEW 和 2-PENDING 状态中取消售后，一旦商户签收售后商品或服务商上门后，售后不可取消；
商户可以在 2-PENDING 状态中取消售后，如果联系不上顾客上门取件和上门维修的话。

3.2.2 仲裁单状态



4. 数据库



5. 程序逻辑




5.1 售后

5.1.1 静态模型

5.1.1.1 顾客申请售后

POST

/order/{oid}/orderitem/{id}/aftersales 顾客提交售后申请*



- 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

Parameters

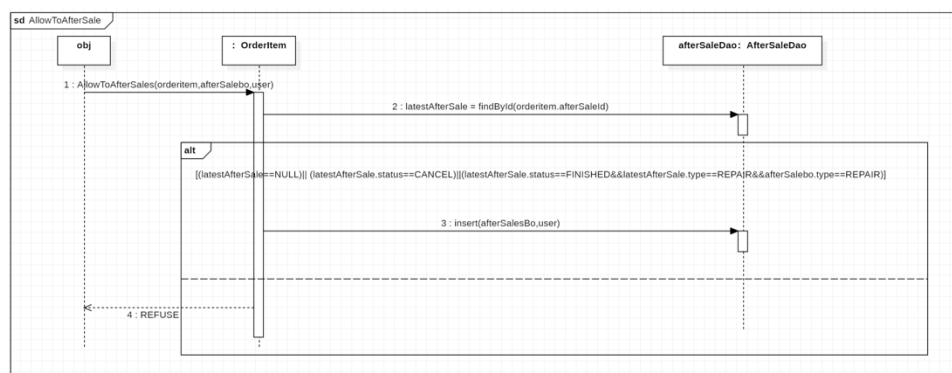
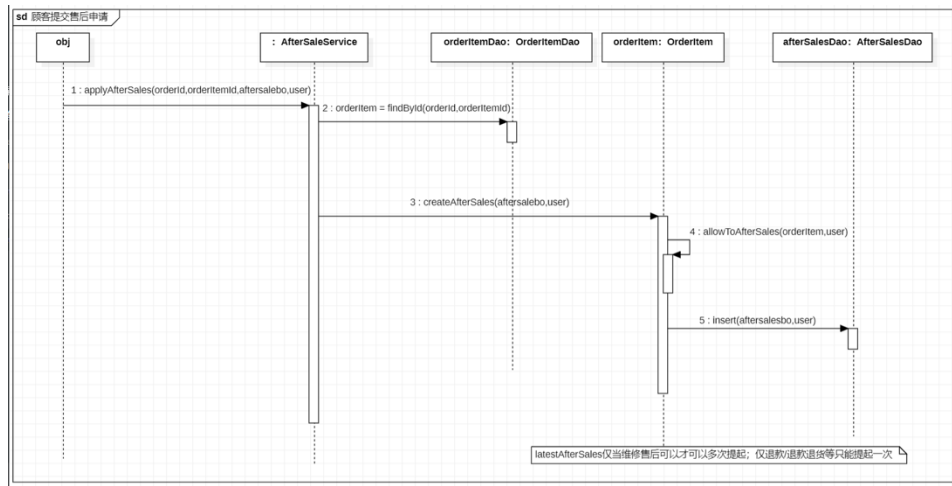
Try it out

Name	Description
authorization * required string <i>(header)</i>	用户token <div>authorization</div>
oid * required integer <i>(path)</i>	订单id <div>oid</div>
id * required integer <i>(path)</i>	商品id <div>id</div>
body * required <i>(body)</i>	售后服务信息 <div>Example Value Model</div> <div><pre>{ "product_id": 0, "type": 0, "reason": "string", "consignee": { "name": "string", "mobile": "string", "regionId": 0, "address": "string" } }</pre></div> <div>Parameter content type application/json</div>

Responses

Response content type application/json

Code	Description
default	成功 <div>Example Value Model</div> <div><pre>{ "errno": {}, "errmsg": "string", "data": { "id": 0, "type": 0, "reason": "string", "conclusion": "string", "quantity": 0, "consignee": { "name": "string", "mobile": "string", "regionId": 0, "address": "string" } }, "status": 0, "gmt_apply": "string", "gmt_end": "string", "order_item_id": 0, "product_item_id": 0, "product_id": 0, "shop_id": 0, "inArbitrated": 0, }</pre></div>



在一个订单 order 里有一到多个 orderItem，顾客提出售后申请需要针对某一 order 里的 orderItem 进行申请(orderItem.id=product.id)，一个 orderItem 里有数量 quantity，我们的设计是对一个 orderItem 提出售后申请时，对应的是所有的 quantity，因此售后申请字段里不需要填写数量，默认是 orderItem 里的数量。

API 路径上的 oid 和 id 对应到 applyAfterSale 里的参数 orderId 和 orderItemId。

(1) 首先通过 orderItem.findById(orderId,orderItemId)才能找到唯一的 orderItem，根据创建者原则，由 orderItem.createAfterSale(afterSalebo)创建 afterSale。

(2)在函数 createAfterSale 里需要通过 allowToAfterSale 判断顾客能否提出售后申请：首先在对象模型中 orderItem 和 afterSale 是一对多的组合关系，在业务需求上 orderItem 记录一个 afterSaleId 表示当前 orderItem 里最新的售后单。

我们的设计是首先根据 afterSale.findById(orderItem.afterSaleId)找到当前最新的售后单 latestAfterSale：

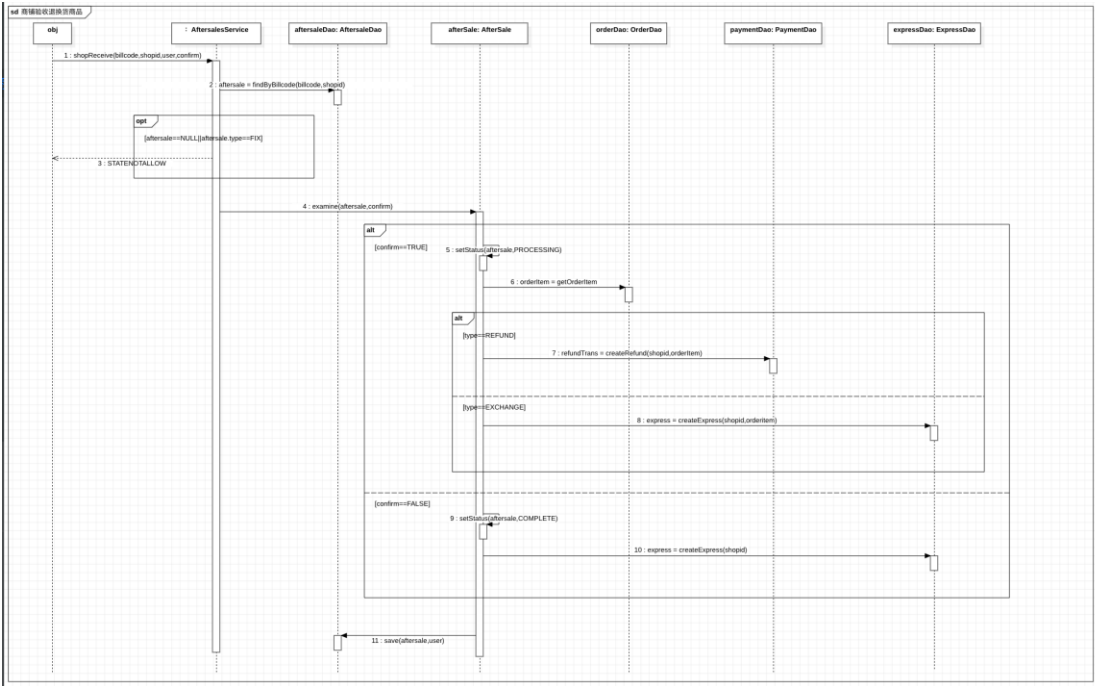
若 orderItem 没有售后单，即 latestAfterSale 为空，则可以允许提出申请

若 orderItem 有售后单但是该售后单是被取消，则可以提出申请

若 orderItem 有已完成的维修类型的售后单，则允许其提出的只能是售后维修申请

其他情况不允许提出多次售后，售后不满意只能走仲裁

5.1.1.2 商铺验收退换货商品



1. findByBillCode: 首先通过 billcode 找到对应的售后单对象 aftersale，并通过 shopid 和 aftersale.shopid 判断是否为当前商户的售后单，如果不是，则返回 NULL。
2. 如果返回为 NULL，则不是这个商户的售后单，不能对其进行处理，拒收物流；如果是维修售后，也不能继续使用此 api，维修不需要验收。
3. 调用 aftersale 的 examine 方法，confirm 为商户拆包后对商品验收的结果：
 - (1) 如果验收通过，则将售后单状态设为处理中。
如果是退货售后，则创建退款单；
如果是换货售后，则将新的商品发出，创建一个物流单。
 - (2) 如果验收不通过，则将商品重新打包发回，创建一个新的物流单。
4. 最后，将对 aftersale 对象的处理存入数据库中。

5.2 仲裁

5.2.1 静态模型

5.2.2 顾客申请售后仲裁

POST

/aftersales/{id}/arbitrations 顾客申请售后仲裁*

←

^

- 只允许顾客提出的仲裁申请
- 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，出705错误

Parameters

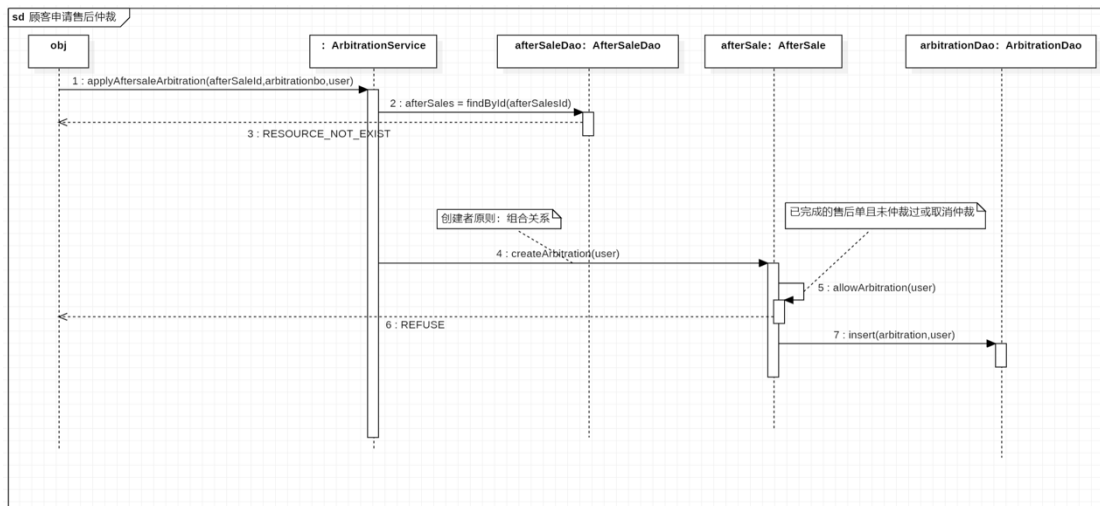
Try it out

Name	Description
authorization * required string (header)	用户token <div>authorization</div>
id * required integer (path)	售后单id <div>id</div>
body * required (body)	信息 <div>Example Value Model</div> <div><pre>{ "reason": "string" }</pre></div> <div>Parameter content type <div>application/json</div></div>

Responses

Response content type application/json

Code	Description
default	成功 <div>Example Value Model</div> <div><pre>{ "errno": 0, "errmsg": "成功", "data": { "id": 0, "status": 0, "reason": "string", "shop_reply": "string", "result": "string", "arbitrator": "string", "gmt_arbitration": "string", "gmt_accept": "string", "gmt_apply": "string", "gmt_reply": "string", "shop_id": 0, "aftersale_id": 0, "customer_id": 0 } }</pre></div>

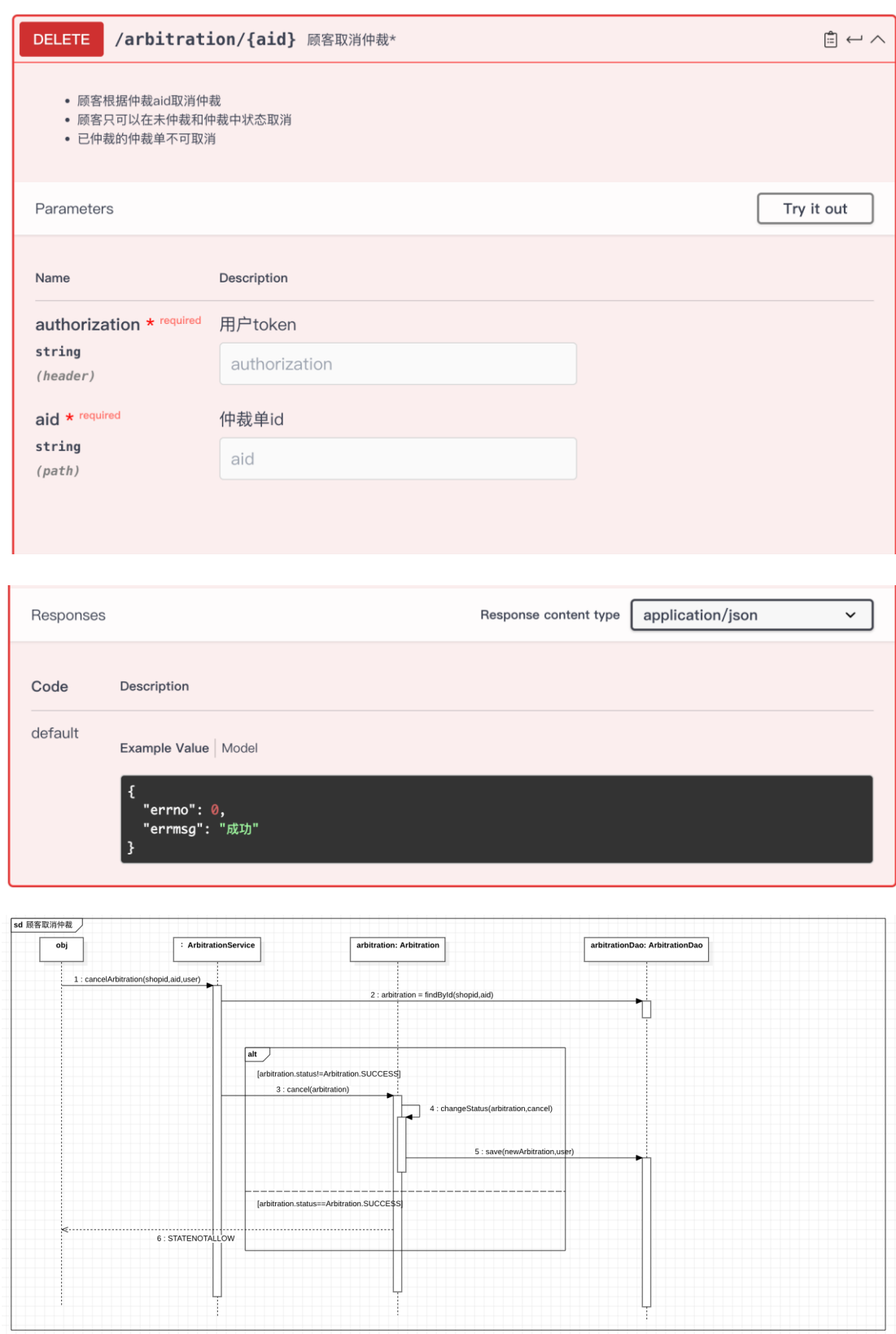


在顾客申请售后仲裁里，顾客必须针对一个售后单申请一次已完成的仲裁，API 路径上的 id 表示 aftersaleId，在 applyAftersaleArbitration 里，传入 afterSaleId 作为参数，arbitrationbo 是通过 POST 请求发送的仲裁单内容在 Controller 层处理完成传入 Service 层，首先通过 afterSelsDao.findByld(afterSaleId) 找到 afterSale，只能针对已完成的售后单提起仲裁
若不存在售后单 afterSale 或 afterSale.status!=FINISHED，则返回错误码 RESOURCE_NOT_EXIST

若存在已完成的售后单 afterSale，由于 Arbitration 与 AfterSale 是组合关系，根据创建者原则应该由整体创建局部，因此调用 afterSale.createAbitration(arbitrationbo) 创建仲裁单
若该售后单没有仲裁单或上次仲裁单状态为取消则允许提出新的仲裁，并将仲裁内容插入到数据库

注意点：在需求上实现的是一个售后单只允许拥有一个已仲裁的仲裁单；若要提出多次仲裁，则需要提出多次售后单

5.2.2 顾客取消仲裁



sd 顾客取消仲裁

obj

: ArbitrationService

arbitration: Arbitration

arbitrationDao: ArbitrationDao

1 : cancelArbitration(shopid,aid,user)

2 : arbitration = findById(shopid,aid)

alt

[arbitration.status!=Arbitration.SUCCESS]

3 : cancel(arbitration)

4 : changeStatus(arbitration,cancel)

5 : save(newArbitration,user)

[arbitration.status==Arbitration.SUCCESS]

6 : STATENOTALLOW

1. 首先通过 shopid 和仲裁 id 找到 arbitration 对象

2. 已结束的仲裁不能取消，若仲裁单状态为已结束，则报错 STATENOTALLOW;
3. 若不是已结束的仲裁，则调用 arbitration 对象的 cancel 方法：首先修改 arbitration 的状态为 cancel，再将修改存入数据库中