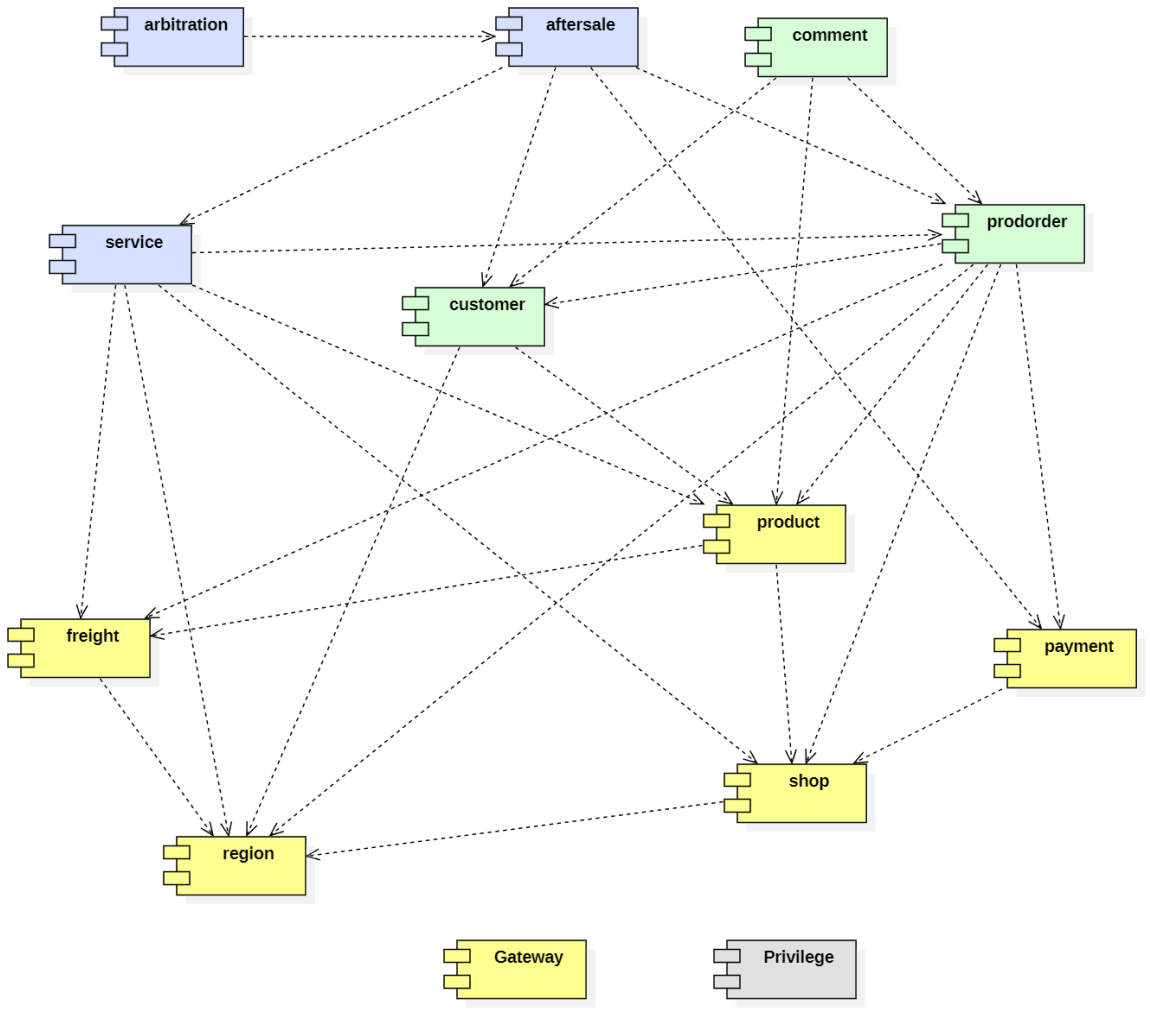
详细设计文档 3-13 组

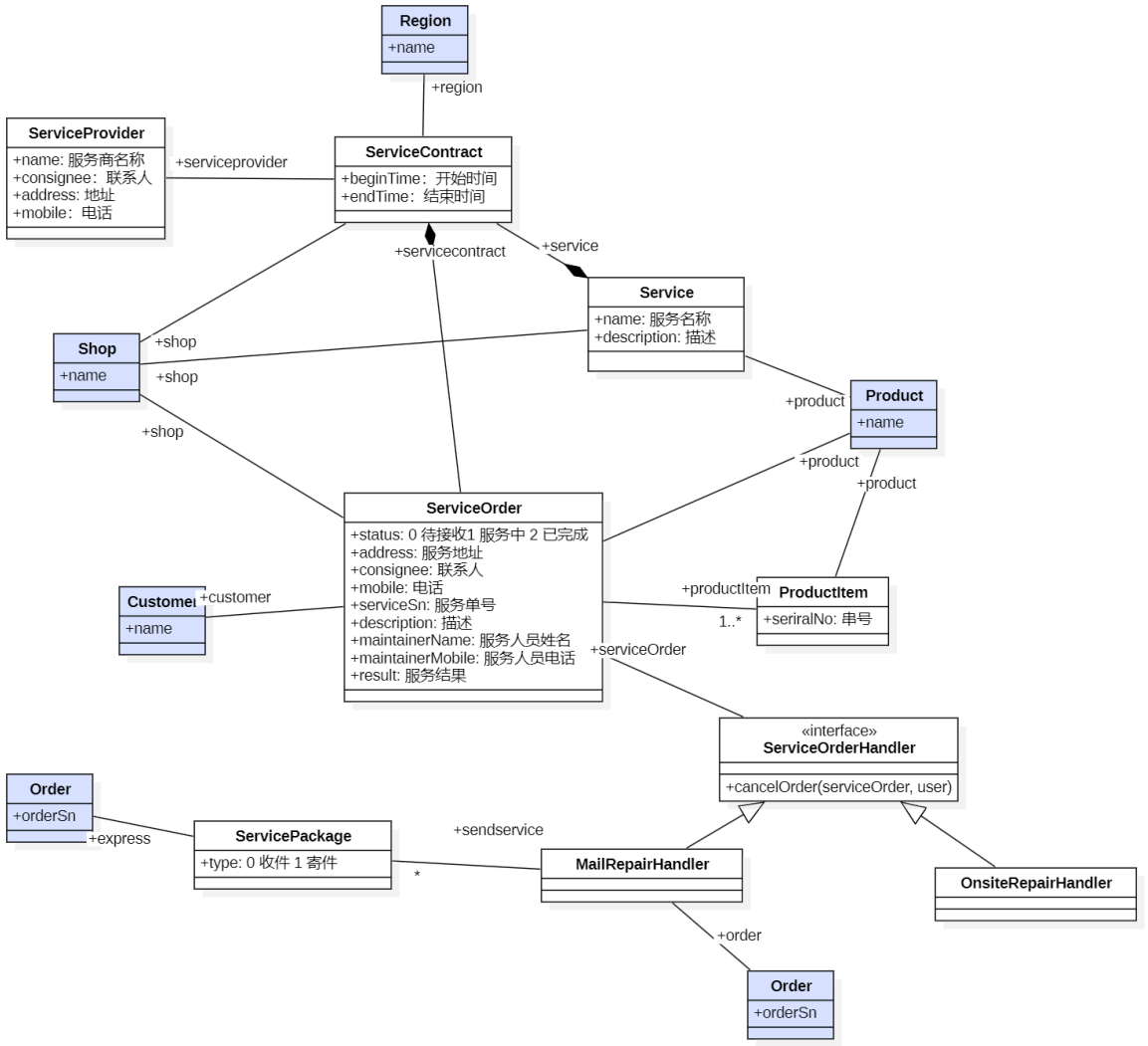
API URL: https://app.swaggerhub.com/apis/KNIGHTKIRIUS/OOMALL/1.0.0#/

组件图

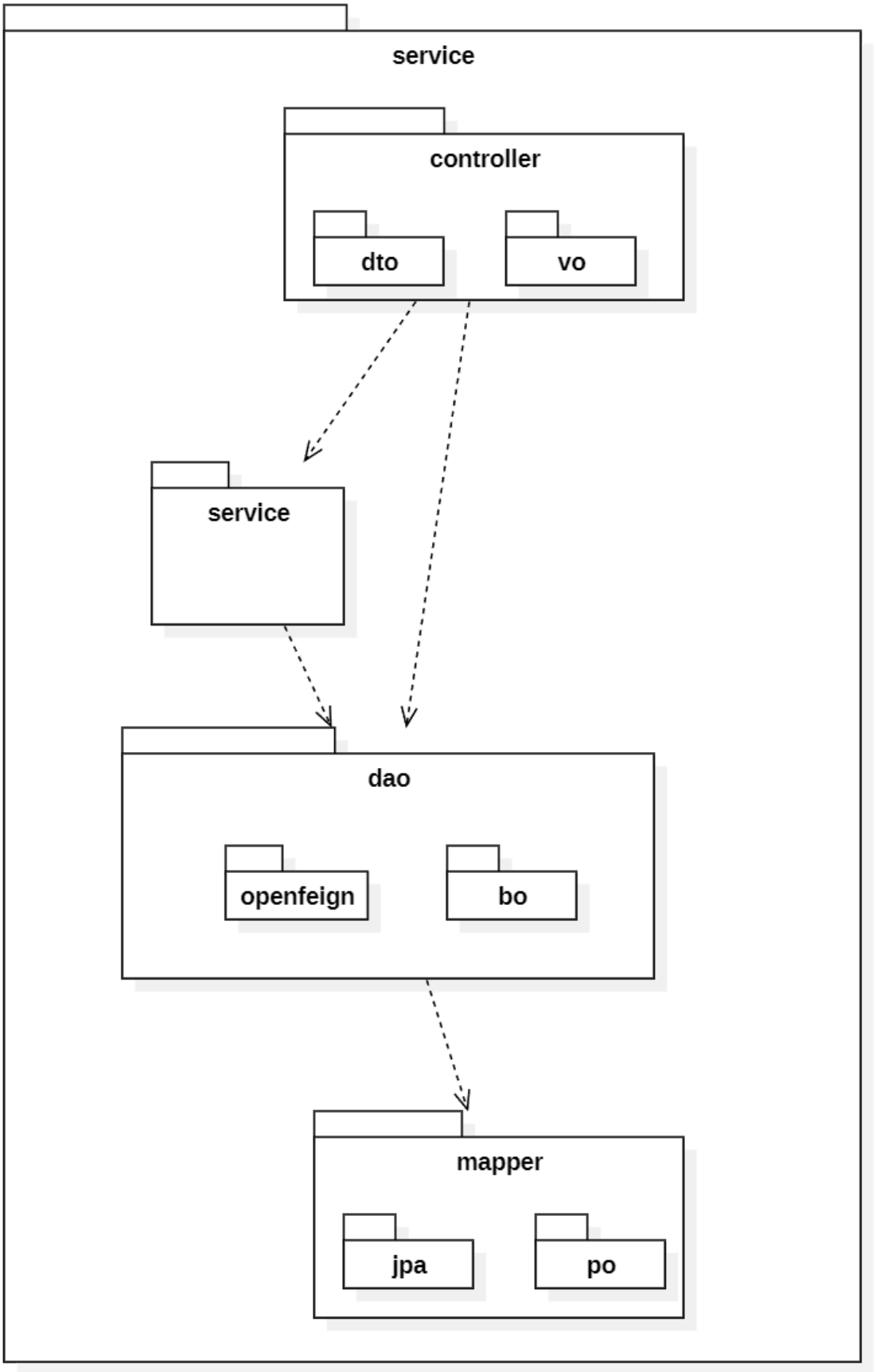


一、服务模块

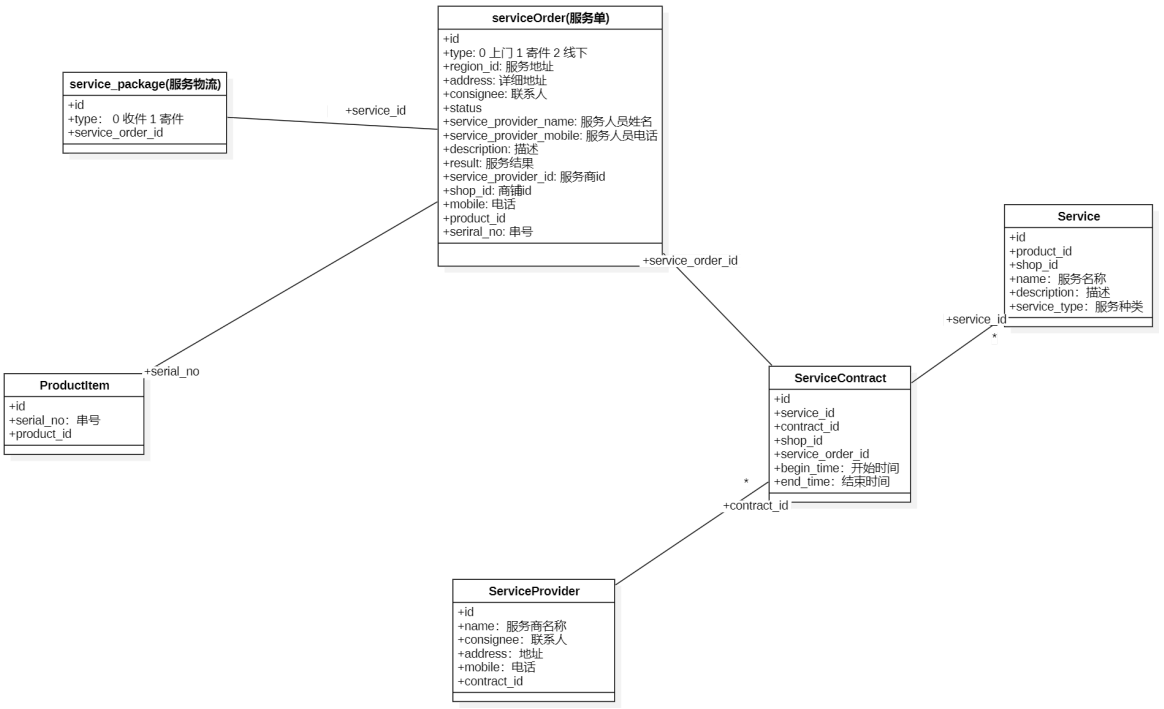
1 对象模型


Region
+name
+region
ServiceProvider
+name: 服务商名称
+consignee：联系人
+address: 地址
+mobile：电话
+serviceprovider
ServiceContract
+beginTime：开始时间
+endTime：结束时间
+servicecontract
+service
Service
+name: 服务名称
+description: 描述
Shop
+name
+shop
+shop
+shop
+product
Product
+name
+product
+product
ServiceOrder
+status: 0 待接收1 服务中 2 已完成
+address: 服务地址
+consignee: 联系人
+mobile: 电话
+serviceSn: 服务单号
+description: 描述
+maintainerName: 服务人员姓名
+maintainerMobile: 服务人员电话
+result: 服务结果
Customer
+name
+customer
+productItem
1..*
ProductItem
+seriralNo: 串号
+serviceOrder
«interface»
ServiceOrderHandler
+cancelOrder(serviceOrder, user)
Order
+orderSn
+express
ServicePackage
+type: 0 收件 1 寄件
+sendservice
*
MailRepairHandler
OnsiteRepairHandler
+order
Order
+orderSn

2 包图


service
controller
dto
vo
service
dao
openfeign
bo
mapper
jpa
po

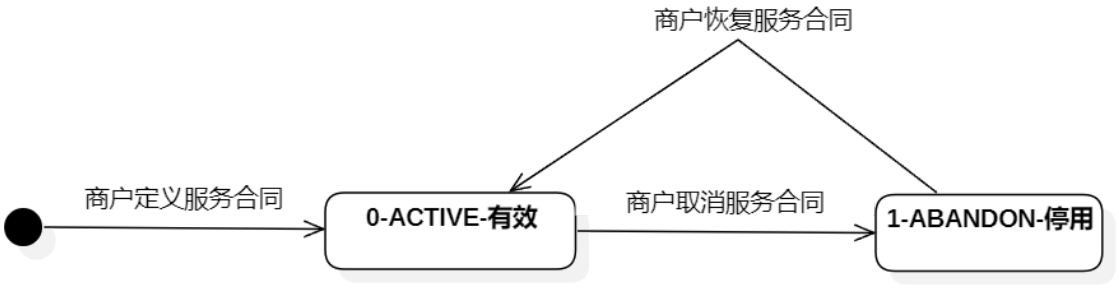
3 数据库



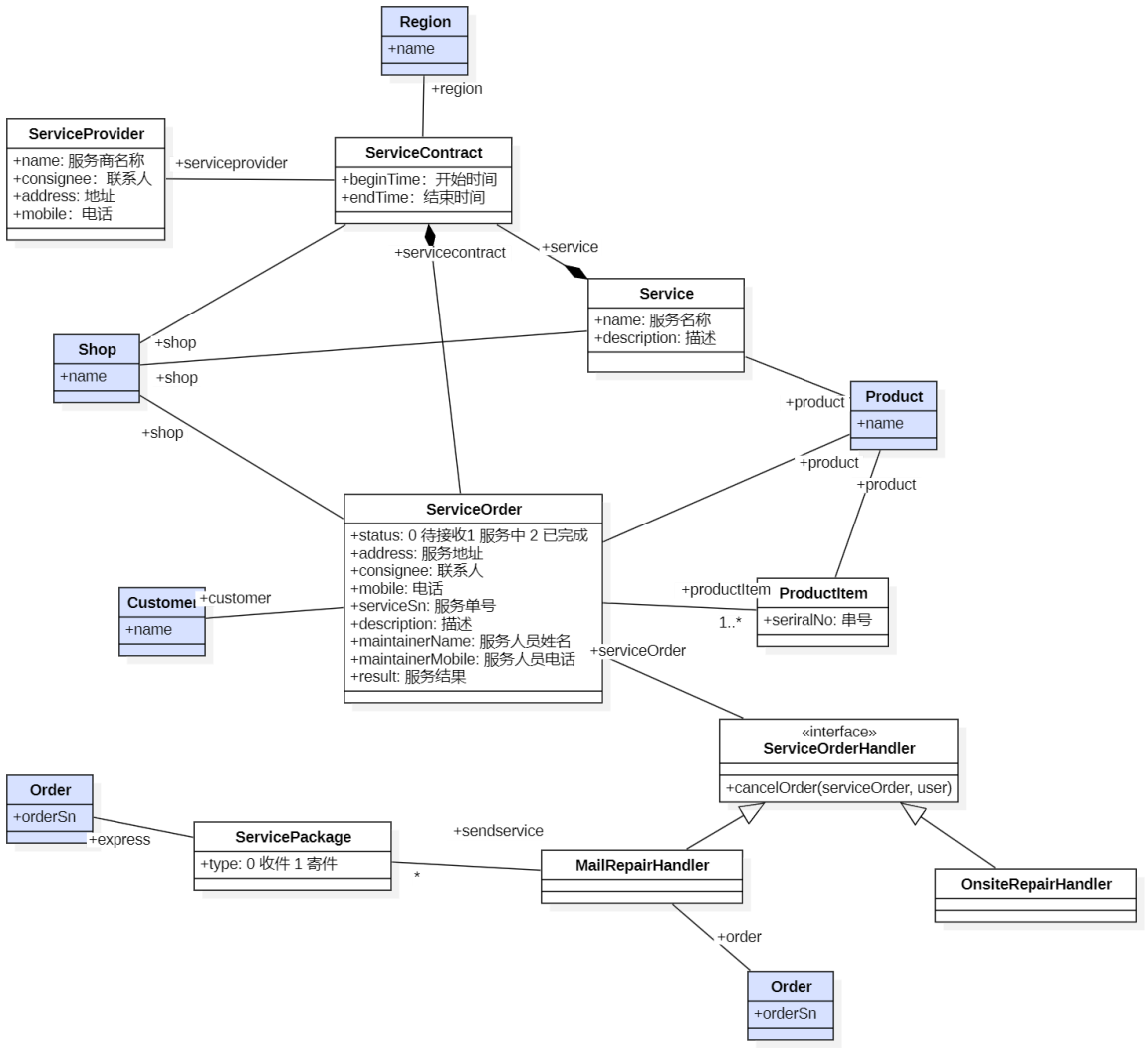
4 状态图

4.1服务单状态图

4.2服务合同状态图



5 服务类图



6 顺序图

6.1取消服务单

界面设计：

API：

DELETE `/shops/{did}/services/{id}` 服务商取消服务单

取消服务单

Parameters — Try it out

Name	Description
authorization * required string *(header)*	用户token authorization
did * required string *(path)*	店铺id did
id * required integer *(path)*	服务单id id

Responses — Response content type: application/json

Code	Description
default	成功

Example Value | Model

```
{
  "errno": 0,
  "errmsg": "成功"
}
```

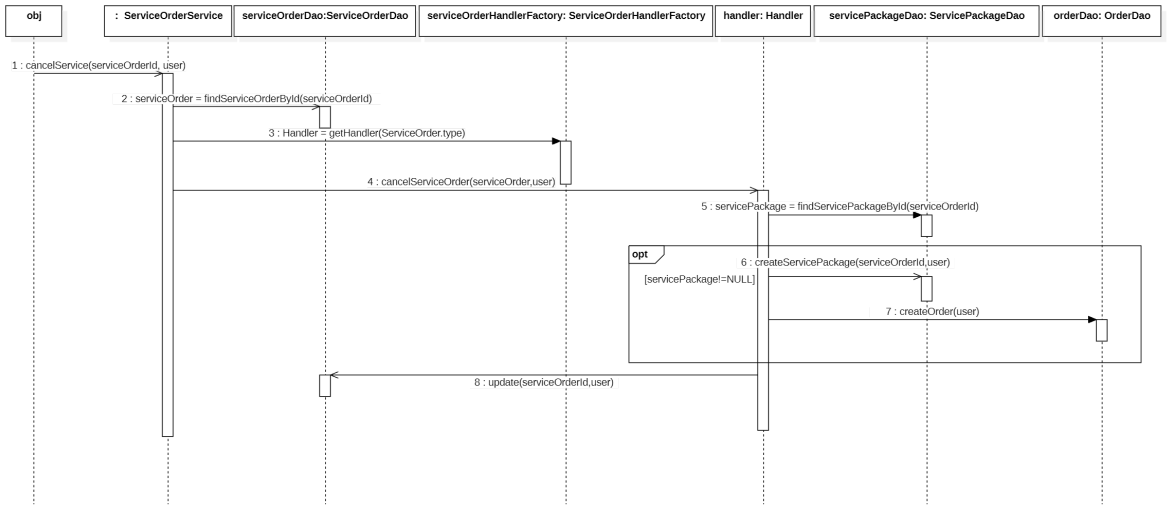
取消服务单的业务流程根据服务类型的不同，后续的行为分为取消服务单以及如果取消运单的时候货物已经发出，商家需要收到货后再给顾客寄回，从而创建返程的运单、订单两种行为，这一特征体现了多态模式。最后再取消服务单，将取消服务单的职责分配给服务单操纵对象，体现信息专家模式。

取消服务单时，若服务单对应的是上门维修服务，服务单操纵对象的行为不同在，直接进行服务单的取消。

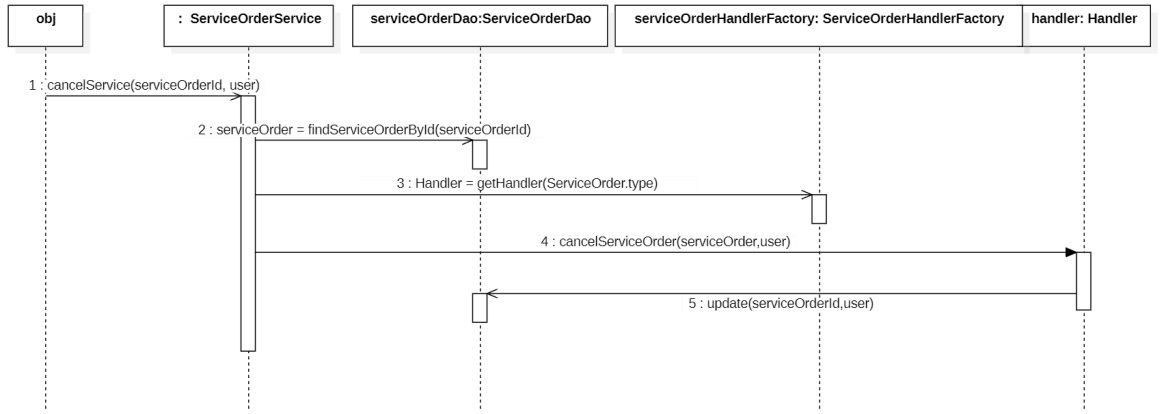
取消服务单时，若服务单对应的是寄修，服务单操纵对象的行为不同在，当寄修已经开始时，服务单对象会创建返程的订单与运单。

这体现了软件设计方法中的开闭原则（使得便于进行其他服务单类型处理方法的拓展）和liskov替换原则（父类【接口】只使用抽象方法，子类具有父类的性质）。

6.1.1取消寄修



6.1.2取消上门服务



6.2定义服务合同

界面设计：


OOMALL
个人中心
我的服务单
我的服务
帮助中心
服务中心
服务合同
取消服务
查看服务
服务列表
服务名称
服务描述
服务类型
操作
服务1
服务描述
类型1
创建合同
服务2
服务描述
类型2
创建合同
服务3
服务描述
类型3
创建合同
服务4
服务描述
类型4
创建合同
共4条
1


OOMALL
个人中心
我的服务单
我的服务
帮助中心
服务中心
服务合同
取消服务
查看服务
服务名称
服务描述
服务类型
服务1
服务描述
类型1
开始日期
结束日期
服务地区
服务商
返回
提交

API:

POST /shops/{shopId}/service/{id}/createContract 定义服务合同

服务商通过此API创建服务合同

Parameters Try it out

Name	Description
authorization * required any *(header)*	用户token authorization
serviceId * required any *(path)*	服务ID serviceId

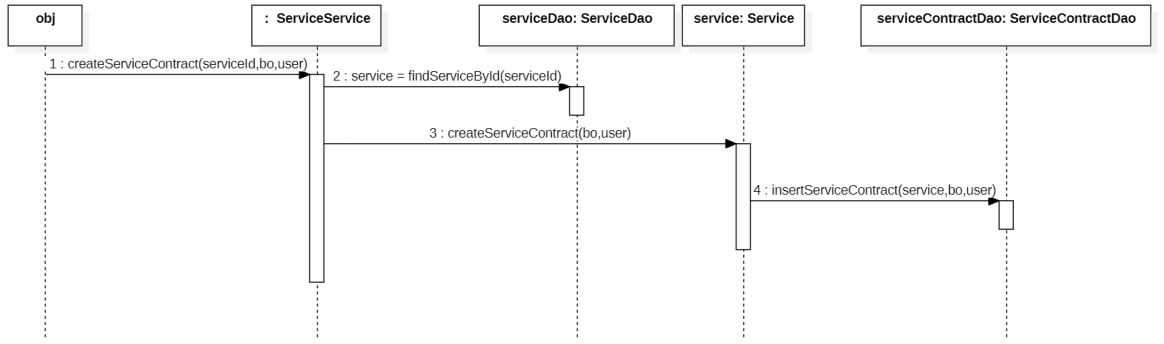
body * required
object
(body)
	服务合同信息
	Example Value | Model
	{
	"serviceId": "string",
	"details": "string"
	}
	Parameter content type
	application/json

Responses Response content type application/json

Code	Description

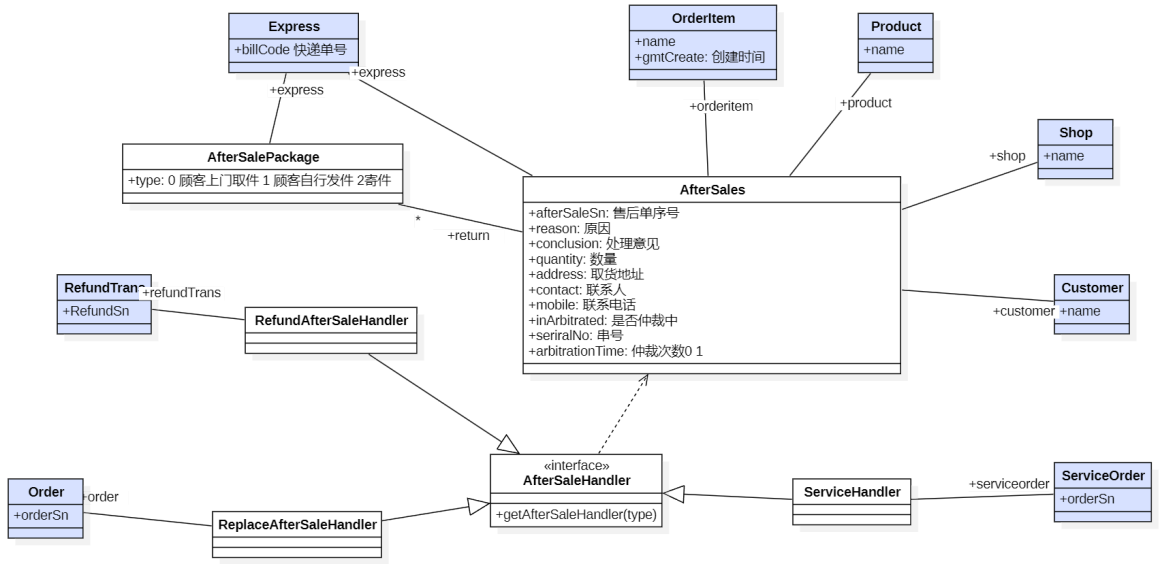
200	成功
	Example Value | Model
	{
	"errno": 0,
	"errmsg": "成功"
	}

使用了创建者模式。服务对象包含服务合同对象，且含有初始化服务合同的主要信息。将创建服务合同对象的职责分配给了服务对象。

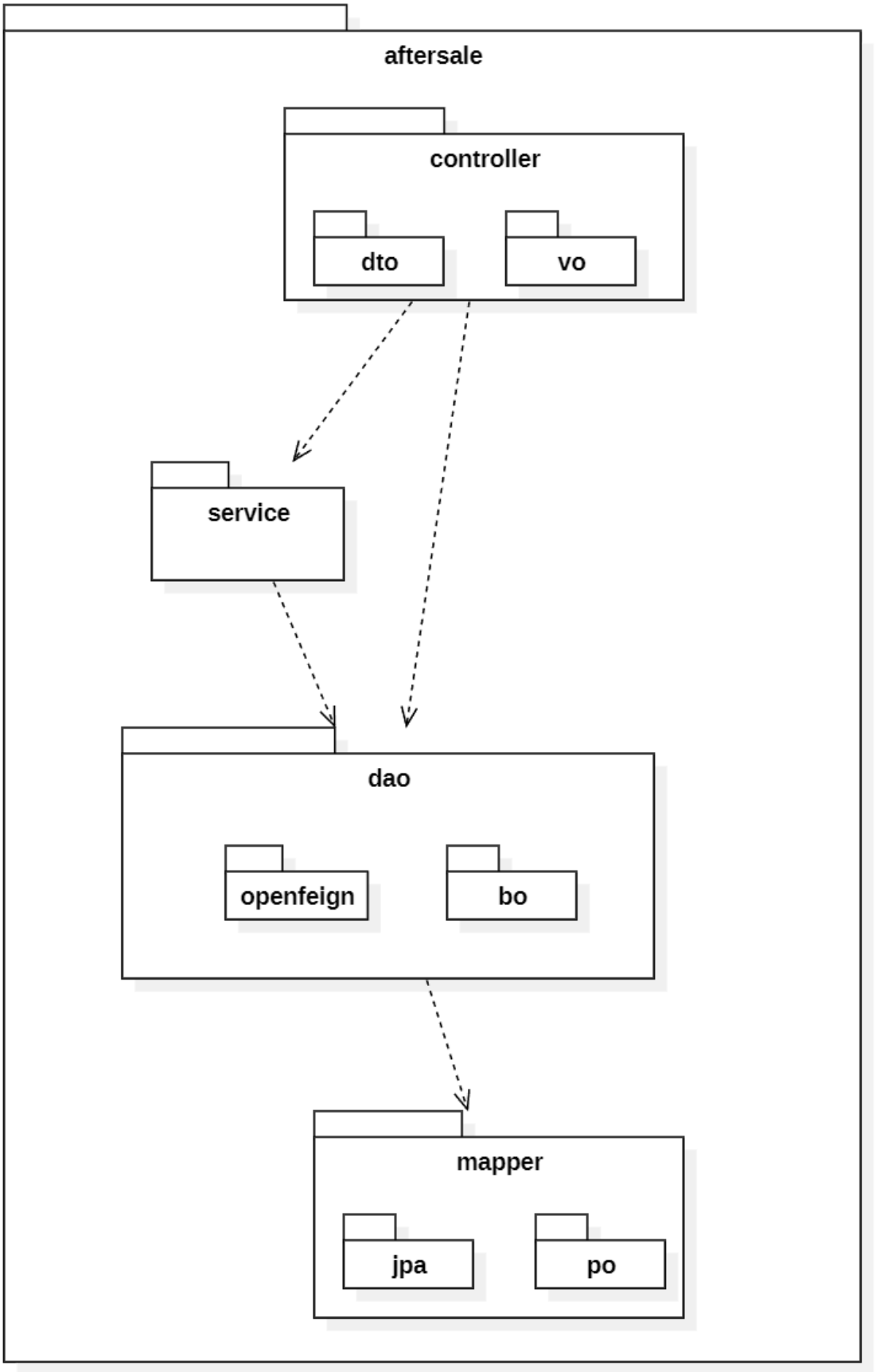


二、售后模块

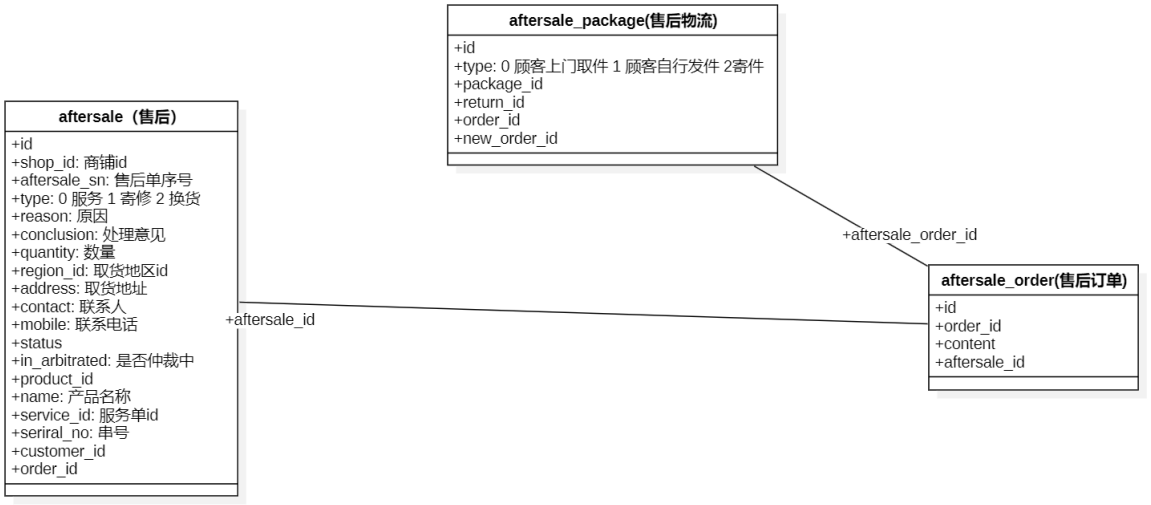
1 对象模型


Express
+billCode 快递单号
+express
+express
OrderItem
+name
+gmtCreate: 创建时间
+orderitem
Product
+name
+product
Shop
+name
+shop
AfterSalePackage
+type: 0 顾客上门取件 1 顾客自行发件 2寄件
*
+return
AfterSales
+afterSaleSn: 售后单序号
+reason: 原因
+conclusion: 处理意见
+quantity: 数量
+address: 取货地址
+contact: 联系人
+mobile: 联系电话
+inArbitrated: 是否仲裁中
+seriralNo: 串号
+arbitrationTime: 仲裁次数0 1
Customer
+name
+customer
RefundTrans
+RefundSn
+refundTrans
RefundAfterSaleHandler
«interface»
AfterSaleHandler
+getAfterSaleHandler(type)
Order
+orderSn
+order
ReplaceAfterSaleHandler
ServiceHandler
+serviceorder
ServiceOrder
+orderSn

2 包图

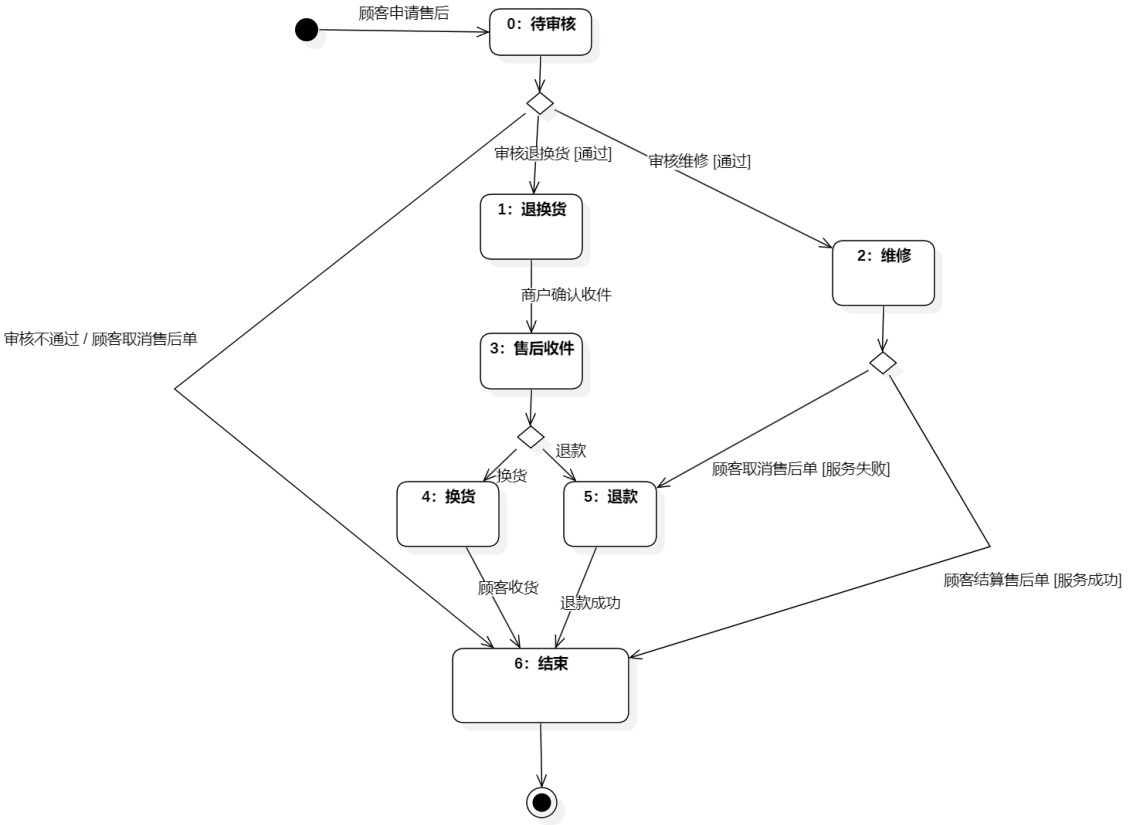

aftersale
controller
dto
vo
service
dao
openfeign
bo
mapper
jpa
po

3 数据库

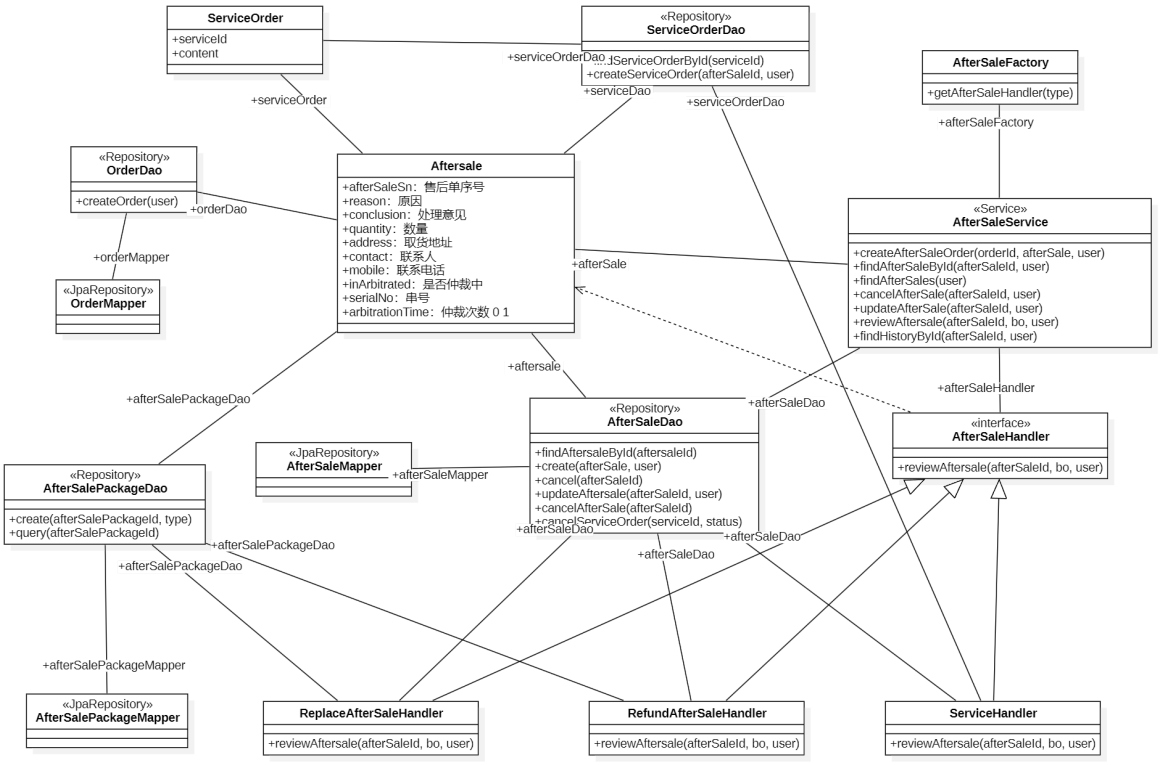


4 状态图

售后单状态图



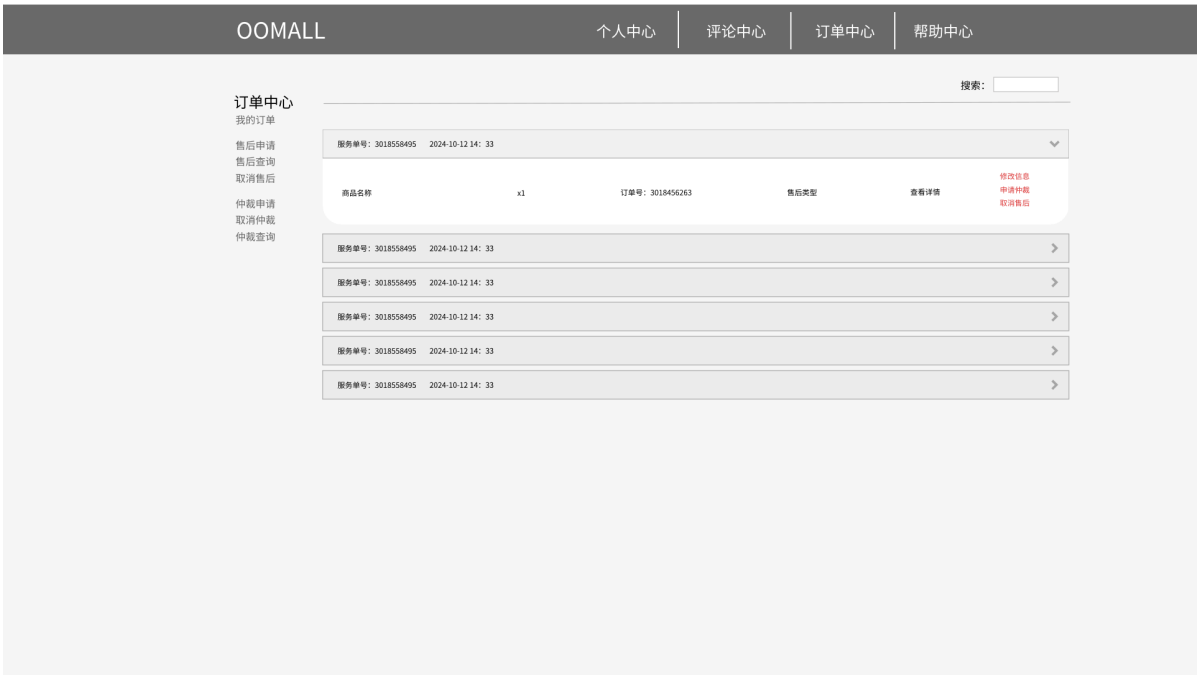
5 类图

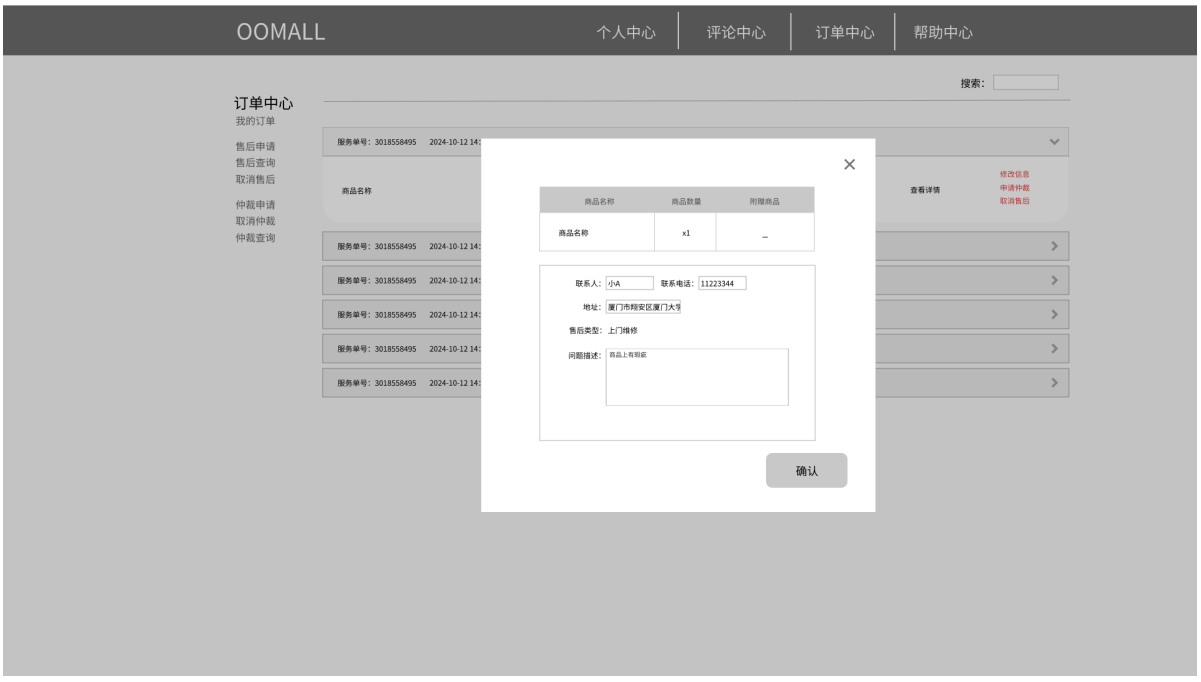


6 顺序图

6.1修改售后单

界面设计：





API：

PUT `/aftersales/{id}` 顾客修改售后单信息

- 顾客只能修改处于申请态的售后单

Parameters — Try it out

Name	Description
authorization * required `string` *(header)*	用户token authorization
id * required `string` *(path)*	售后单id id
body * required `object` *(body)*	买家可修改的信息：地址，申请售后的原因，联系人以及联系电话 **Example Value** | Model

```
{
  "reason": "string",
  "consignee": {
    "name": "string",
    "mobile": "string",
    "regionId": 0,
    "address": "string"
  }
}
```

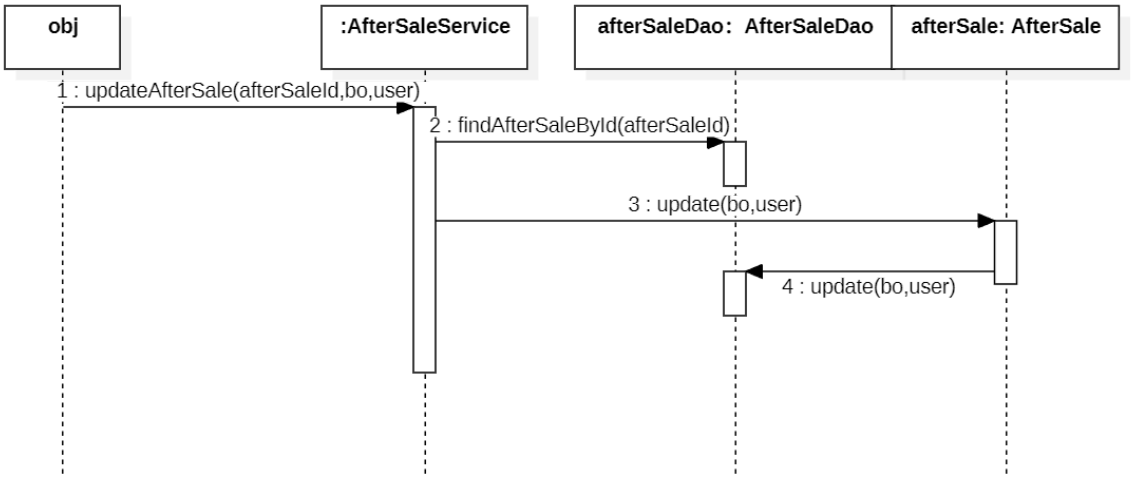
Parameter content type: application/json

Responses — Response content type: application/json

Code	Description
default	成功 **Example Value** | Model

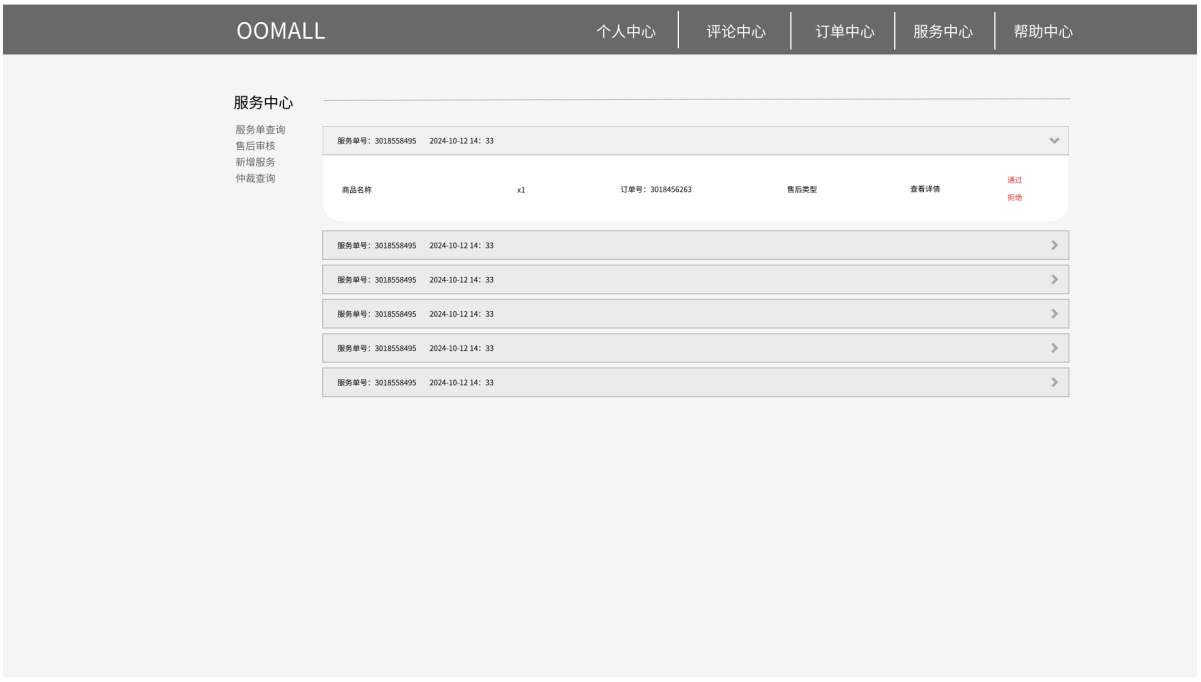
```
{
  "errno": 0,
  "errmsg": "成功"
}
```

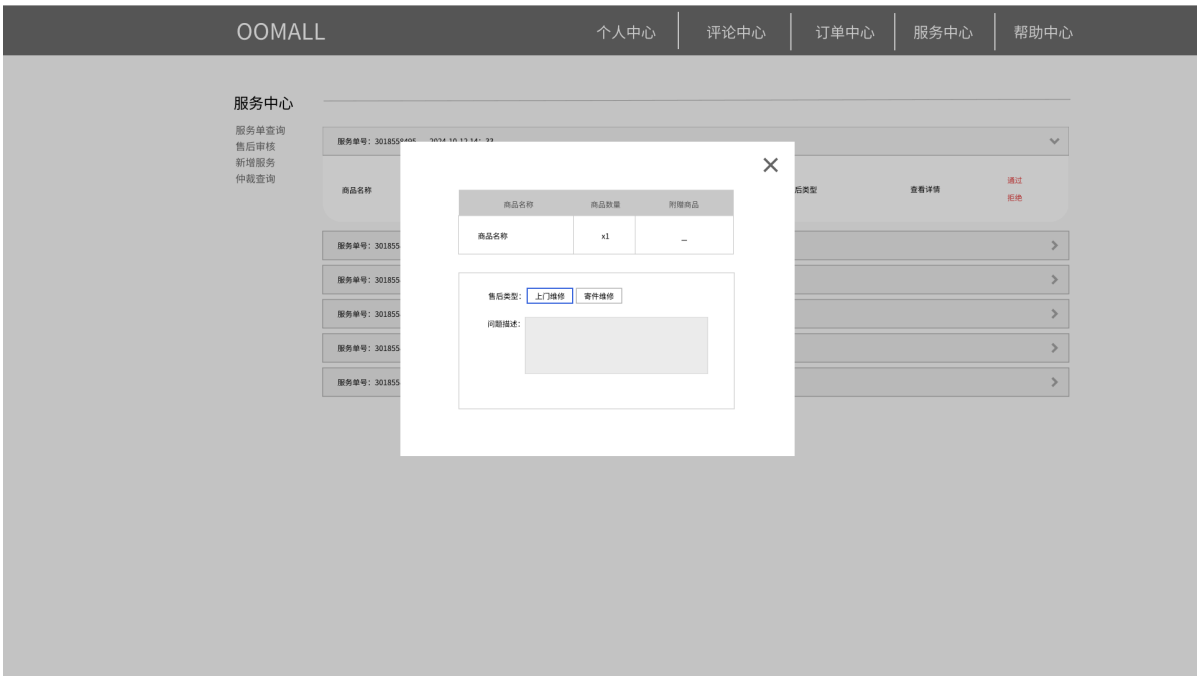
使用了信息专家模式。售后单对象包含判断售后单能否修改所需的信息。将修改售后单的职责分配给售后单对象。



6.2审核售后单

界面设计：





API：

PUT /shops/{shopId}/aftersales/{id}/confirm 管理员审核同意/不同意

Parameters Try it out

Name	Description
authorization * required string *(header)*	用户token authorization
id * required integer *(path)*	售后单id id

body * required
object
(body)
	处理意见
	Example Value | Model
	{
	"confirm": true,
	"conclusion": "string",
	"type": 0
	}
	Parameter content type
	application/json

Responses Response content type application/json

Code	Description

default	成功
	Example Value | Model
	{
	"errno": 0,
	"errmsg": "成功"
	}

审核售后的业务流程根据售后单类型的不同，后续的行为分为是否创建运单两种行为。这一特征体现了多态模式。并且都更新了售后单状态，将更新售后单的职责分配给售后单操纵对象，体现信息专家模式。

当售后单类型为服务时，对象的行为不同在，只需要修改售后单状态即可，后续创建服务单的行为由其他API完成。

当售后单类型为退款或换货时，对象的行为不同在，修改售后单状态后需要产生一个寄件运单。

这体现了软件设计方法中的开闭原则（使得便于进行其他售后单类型处理方法的拓展）和liskov替换原则（父类【接口】只使用抽象方法，子类具有父类的性质）。