

Litemall系统

详细设计说明书

厦门大学软件学院2017级

普通OO是AD而且还是OOAD这样的J2EE你喜欢吗 小组

组长：郑钦文

组员：颜倩 刘佳硕 蔡郭平 叶颖津

目录

[一、组件图 3](#_Toc28003486)

[二、部署图 4](#_Toc28003487)

[三、Discount模块： 4](#_Toc28003488)

[1.Coupon部分 4](#_Toc28003489)

[1.1 coupon需求 4](#_Toc28003490)

[1.2 couponAPI说明 5](#_Toc28003491)

[1.3 Coupon类图 8](#_Toc28003492)

[1.4 coupon时序图 9](#_Toc28003493)

[1.5coupon状态图 10](#_Toc28003494)

[1.6coupon设计说明 11](#_Toc28003495)

[2.Groupon部分 12](#_Toc28003496)

[2.1 Groupon需求 12](#_Toc28003497)

[2.2GrouponAPI说明 13](#_Toc28003498)

[2.3Groupon类图 15](#_Toc28003499)

[2.4 Groupon时序图 16](#_Toc28003500)

[2.5 GrouponRule状态图 17](#_Toc28003501)

[2.6Groupon设计说明 17](#_Toc28003502)

[3.presale部分 18](#_Toc28003503)

[3.1 presale需求说明 18](#_Toc28003504)

[3.2 presaleAPI说明 18](#_Toc28003505)

[3.3presale类图 20](#_Toc28003506)

[3.4presale时序图 21](#_Toc28003507)

[3.5presale状态图 21](#_Toc28003508)

[3.6 presale设计说明 21](#_Toc28003509)

[4.discount部分 22](#_Toc28003510)

[四． UserInfo模块 22](#_Toc28003511)

[1. user部分 22](#_Toc28003512)

[1.1 user需求说明 22](#_Toc28003513)

[1.2 userAPI说明 23](#_Toc28003514)

[1.3 user类图 25](#_Toc28003515)

[1.4 user时序图 26](#_Toc28003516)

[1.5 设计思路 27](#_Toc28003517)

[2 admin部分 27](#_Toc28003518)

[2.1 admin需求说明 27](#_Toc28003519)

[2.2 adminAPI说明 28](#_Toc28003520)

[2.3 admin类图 32](#_Toc28003521)

[2.4 admin时序图 33](#_Toc28003522)

[2.5 admin 设计思路 33](#_Toc28003523)

[3 zuul网关部分 34](#_Toc28003524)

[3.1 zuul类图 34](#_Toc28003525)

[五．address模块 35](#_Toc28003526)

[1. address部分 35](#_Toc28003527)

[1.1 address需求说明 35](#_Toc28003528)

[1.2 addressAPI说明 35](#_Toc28003529)

[1.2.2 address内部API 36](#_Toc28003530)

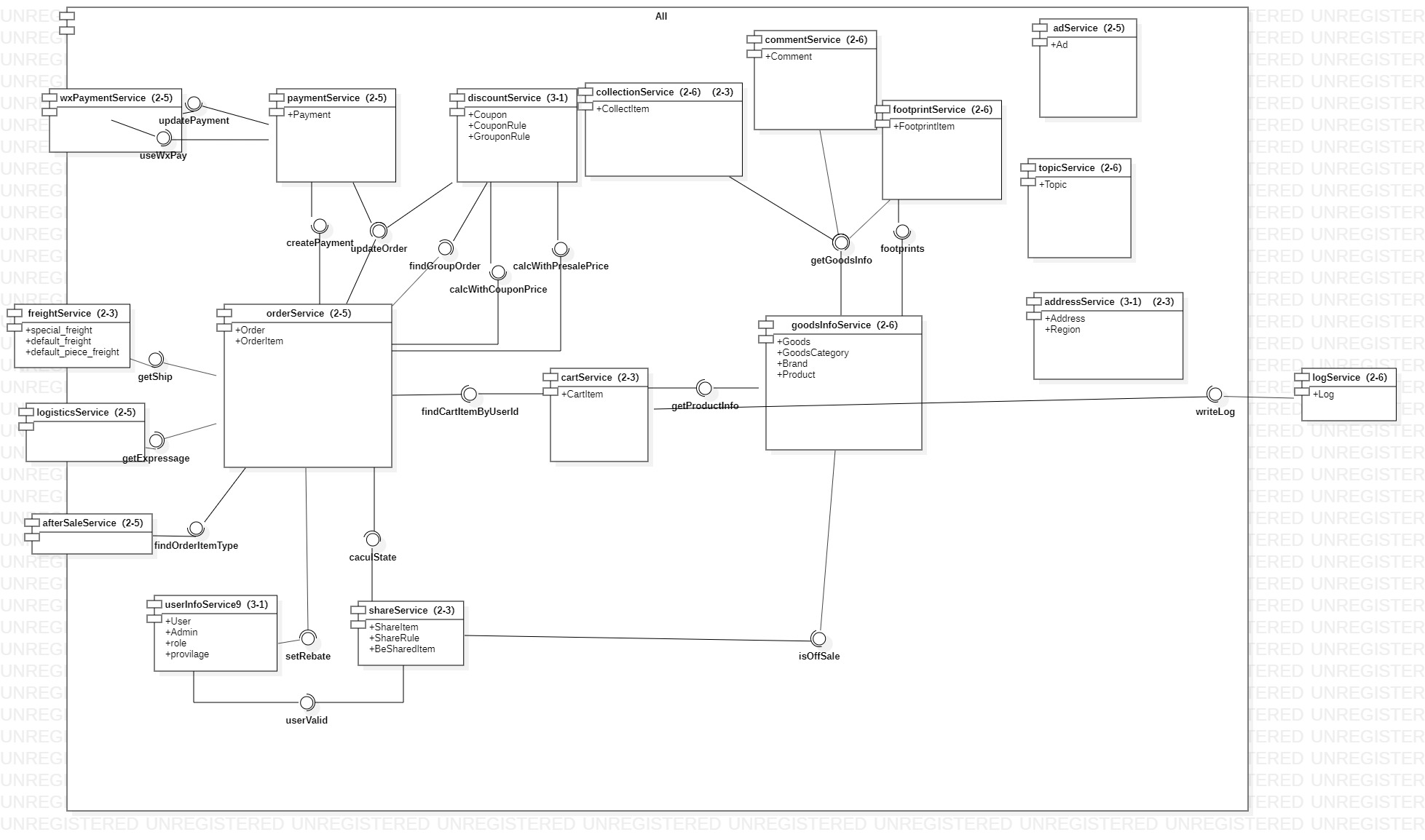
[1.3 address类图 37](#_Toc28003531)

[1.4 address时序图 38](#_Toc28003532)

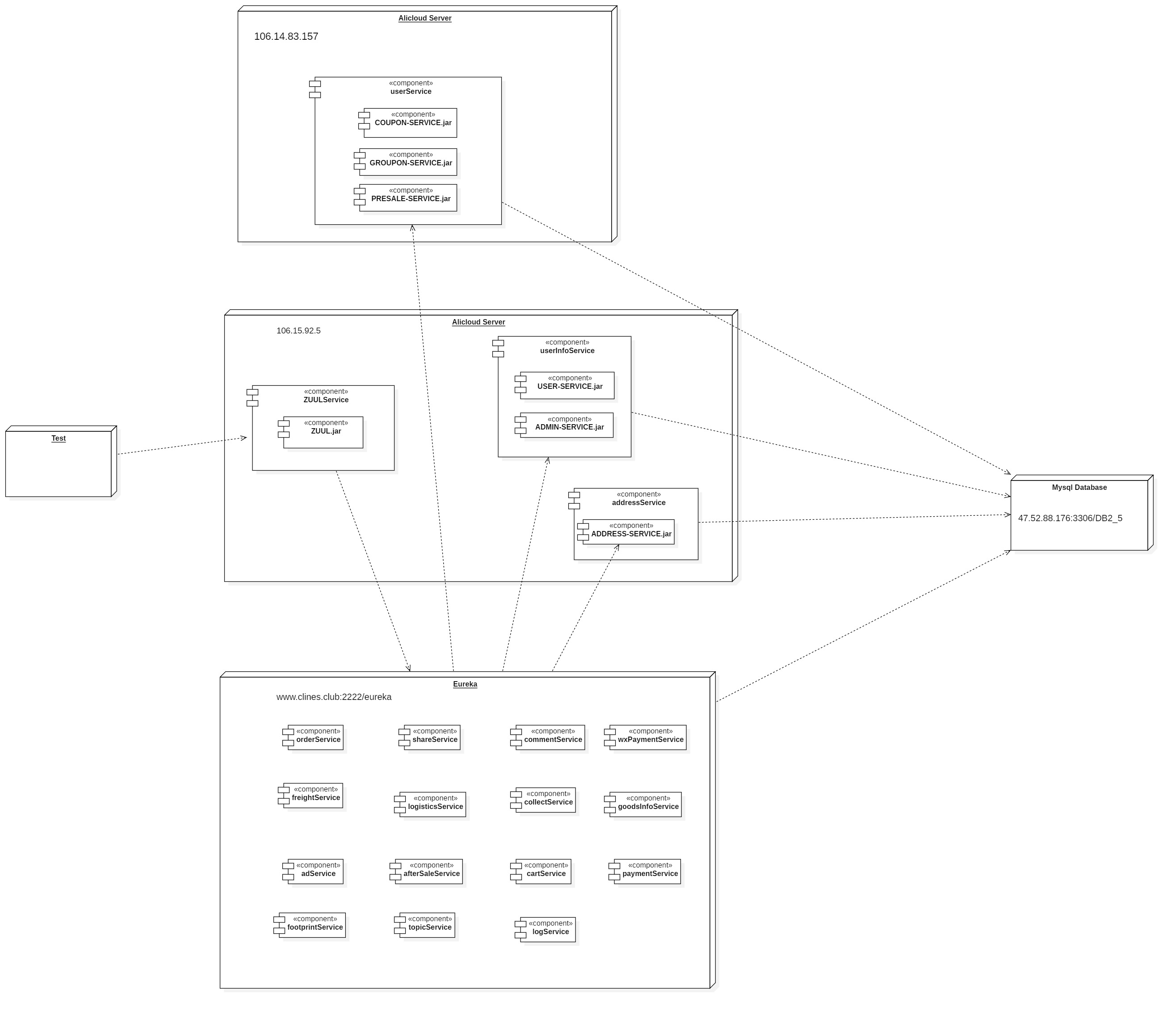
[1.5 状态机图 39](#_Toc28003533)

[1.6 address模块设计说明 39](#_Toc28003534)

# 一、组件图



# 二、部署图



# 三、Discount模块：

## 1.Coupon部分

### 1.1 coupon需求

管理员不能修改正在起止期限内的优惠券规则。 限制每个用户同类优惠券只能有一张。

优惠券应支持分类和指定商品使用。

可支持各种不同类型的优惠券规则。

优惠券规则需设定起止期限，在 起止期限中才可以领用或使用优惠券。

管理员可以作废优惠券规则，优惠券规则作废后，所有领出未用的优惠券也一并作废，已经使用的优惠券不受影响。

管理员不能修改正在起止期限内的优惠券规则。 限制每个用户同类优惠券只能有一张。

### 1.2 couponAPI说明

#### 1.2.1外部API

外部接口：

GET //用户在商城首页查看可领取的优惠劵规则

/couponRules

参数：

page: Integer 第几页

limit: Integer 一页多少

返回值：List<couponRulePo>

GET //用户查看自己的优惠券

/coupons

//查看不同状态（0未使用，1已使用，2已失效，3已过期）优惠劵

参数：@requestParam showType

page: Integer 第几页

limit: Integer 一页多少

0未使用，1已使用，2已失效，3已过期

返回值：List<coupon>

POST //用户领取一张优惠券

/coupons

参数：Requestbody(couponPo)

前端可传参数：

userId

couponRuleId

couponSn

beginTime

endTime

usedTime

name

picUrl（这是第一次上传图片产生的）

statusCode

返回值：couponPo

POST//根据用户id和商品id返回可用的优惠券列表

/coupons/availableCoupons

参数：Requestbody(List<cartItem>)

返回值：List<coupon>

POST//管理员下架优惠券规则

/couponRules/{id}/invalid

PathVariable：id

Return：responseUtil.ok()

GET //管理员查找所有优惠券规则

/admin/couponRules

参数：

page: Integer 第几页

limit: Integer 一页多少

返回值：List<couponRulePo>

POST //管理员添加一个优惠券规则

/couponRules

参数：Requestbody(couponRulePo)

前端可传参数：

name

brief

beginTime

endTime

statusCode

picUrl（这是第一次上传图片产生的）

validPeriod

strategy

total

collectedNum

goodsList1

goodsList2

返回值：couponRulePo

GET //查看一种优惠券规则

/couponRules/{id}

参数：Pathvariable(id)

返回值：couponRulePo

PUT //管理员修改优惠券规则

/couponRules/{id}

参数：Pathvariable(id)Requestbody(couponRulePo)

前端可传参数：

name

brief

beginTime

endTime

statusCode

picUrl（这是第一次上传图片产生的）

validPeriod

strategy

total

collectedNum

goodsList1

goodsList2

返回值：couponRulePo

Delete //管理员删除优惠券规则

/couponRules/{id}

参数：Pathvariable(id)

返回值：无

#### 1.2.2内部API

内部接口：

Post //根据优惠券id使用一张优惠券

/coupons/{id}/update

参数：id, type

返回值：状态

Post:// 根据order和优惠券id计算总折扣

/coupon/calcDiscount

参数：order

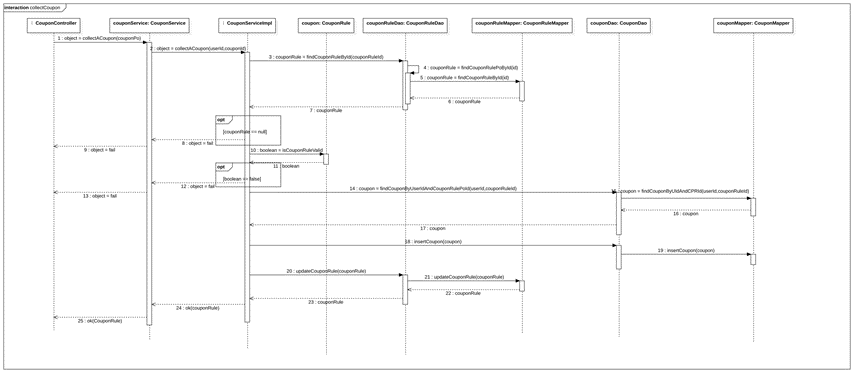
返回值：payment

### 1.3 Coupon类图

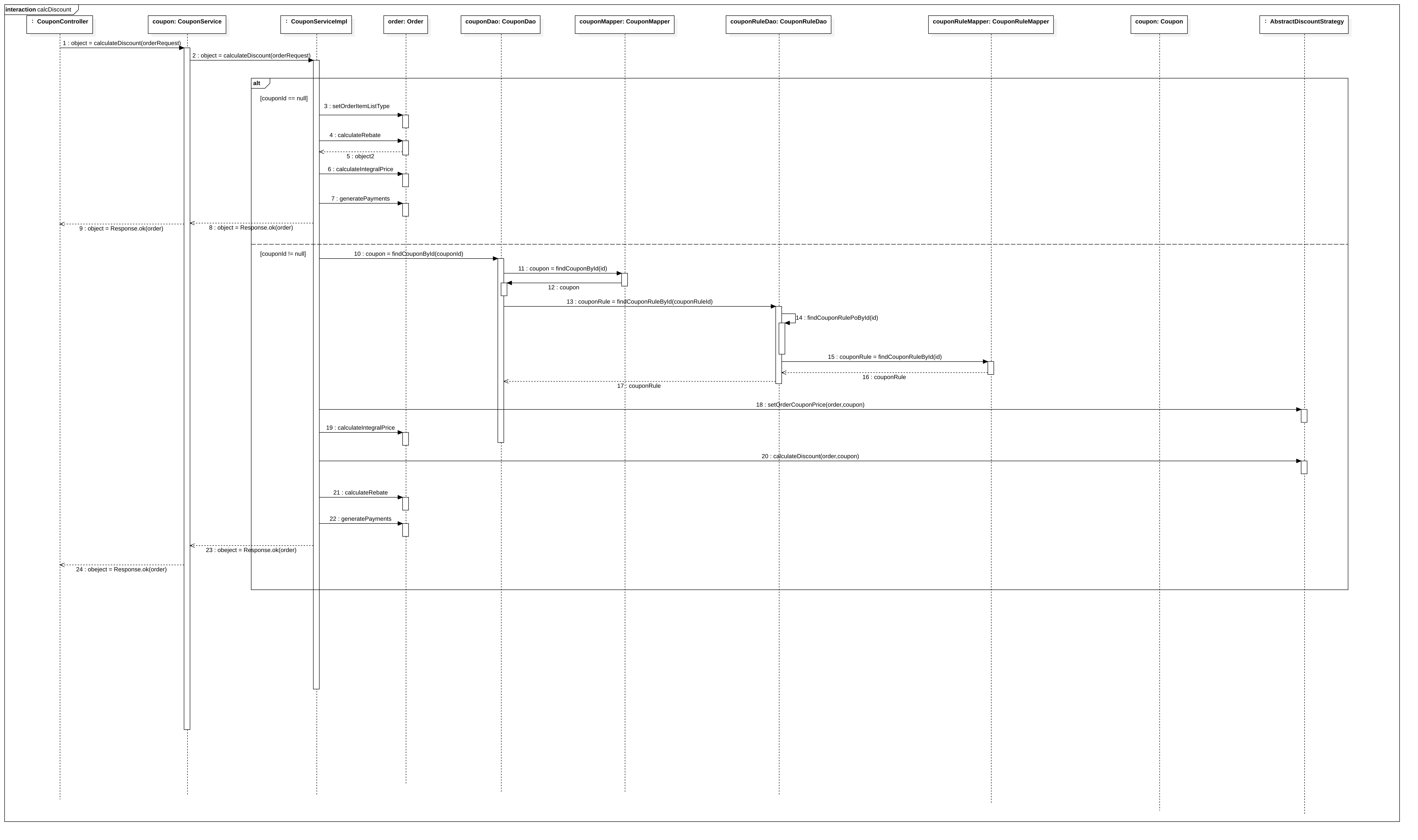


### 1.4 coupon时序图

#### **1.4.1collectCoupon**

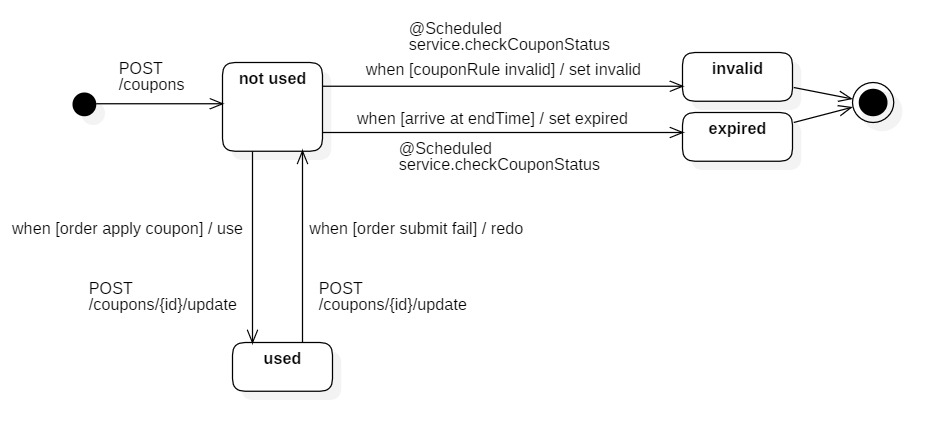


#### 1.4.2 calcDiscount

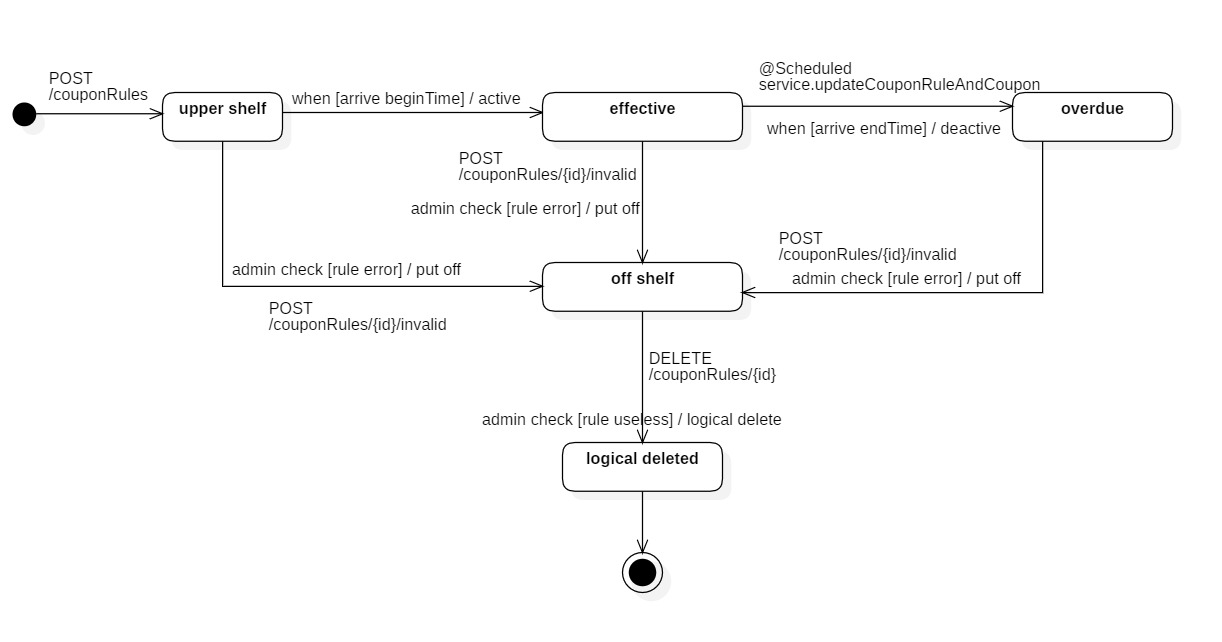
****

### 1.5coupon状态图

CouponStatus:



CouponRuleStatus:



### 1.6coupon设计说明

coupon模块的设计说明

coupon模块的设计主要有以下重点：

1. 优惠券规则采用策略模式（strategy）。具体的优惠券规则策略类只有在算法或行为上稍有不同，优惠券使用的策略随时会发生变化。

2. 采用严格的MVC架构。

3. 优惠券和优惠券规则有各自的service、dao、mapper、domain层。

4. 确定优惠券和优惠券规则各自的功能边界。

5. 优惠券和优惠券规则的定时器设计。

coupon模块如何设计以满足需求

优惠券应支持分类和指定商品使用。

数据库层面：coupon\_rule表中通过JSON字符串存放商品列表，举例如：{"gIDs":[1,4,6,7,9]}。支持分类即将分类下所有商品ID加入JSON字符串。

代码层面：CouponRule类中有非数据库属性`List<Integer> goodsList`，装配couponRule时首先从数据库读取Po类CouponRulePo，然后将string类型的goodsList1和goodsList2从JSON字符串解析为goodsList的List对象。

可支持各种不同类型的优惠券规则。

采取策略模式（strategy）。将不同的优惠券规则类当作一组相似的算法，通过总的strategy抽象类AbstractDiscountStrategy作为外部接口。具体的装配行为在CouponRule中实现，实际使用时，只能通过CouponRule的AbstractDiscountStrategy属性调用具体方法。

优惠券规则需设定起止期限，在起止期限中才可以领用或使用优惠券。

优惠券和优惠券规则的生命周期设计成不完全重叠。优惠券规则中的validPeriod属性的正负值决定其优惠券的声明周期。

为优惠券规则设置定时器，每天晚上24点服务器检查所有上架优惠券是否过期，过期则下架。

为优惠券设置定时器，同样每晚24点检查。

管理员可以作废优惠券规则，优惠券规则作废后，所有领出未用的优惠券也一并作废，已经使用的优惠券不受影响。

管理员可以在任意上架状态下架优惠券规则。

管理员不能修改正在起止期限内的优惠券规则。 限制每个用户同类优惠券只能有一张。

这些通过逻辑判断实现

计算订单中的优惠券折扣

涉及订单类型的二次判断和优惠券折扣金额、返点金额的分摊，以及保留两位小数带来的误差的消除。

## 2.groupon部分

### 2.1 groupon需求

对于特定商品设定不同等级的团购数量（如 1-100 件 9 折，101-200 件 8 折，201件以上 7 折）。用户在提交团购订单时按照商品的正常价格支付订单，在规定时间内达到不同等级团购数量则团购成功，按照不同的团购等级原渠道退回用户多支付的金额（团购不支持使用优惠卷和返点）。未达到团购门槛，用户可以选择继续订单，若用户未确认则全额退款。管理员可以选择让正在进行中团购下线，下线的团购即可终止团购活动，无论是否成团都不能享受折扣，按照未达到成团门槛处理。

### 2.2 grouponAPI说明

#### 2.2.1 groupon外部API说明

POST 下架团购规则

/grouponRules/{id}/invalid

PathVariable(id)

Return responseUtil.ok()

//管理员查看一个goodsId的所有团购规则GET

/grouponRules

RequestParam (goosId) ， RequestParam (page), RequestParam (limit)

return List<GrouponRuleVo>

创建一条团购规则

POST

/grouponRules

Requestbody(GrouponRulePo)

return GrouponRulePo

管理员根据grouponId查看规则

GET

/admin/grouponRules/{id}

Pathvariable(id)

return GrouponRuleVo

修改规则

PUT

/grouponRules/{id}

Pathvariable(id) Requestbody(GrouponRulePo)

return GrouponRulePo

删除规则

DELETE

/grouponRules/{id}

Pathvariable(id)

返回值：无（ResponseUtil.ok()即可）

管理员查看所有没有被删除的groupon

GET

/admin/grouponGoods

RequestParam (page), RequestParam (limit)

return List<GrouponRuleVo>

用户根据grouponId查看一条规则

GET

/grouponRules/{id}

Pathvariable(id)

return GrouponRuleVo

用户查看所有能查看的规则

GET

/grouponGoods

RequestParam (page), RequestParam (limit)

return List<GrouponRuleVo>

#### 2.2.2 groupon内部API说明

给goods模块调用，查询某个goodsId的当前团购规则

GET

/goods/{id}/grouponRule

Pathvariable(id)

return GrouponRule

被discount中心调用，进行订单处理

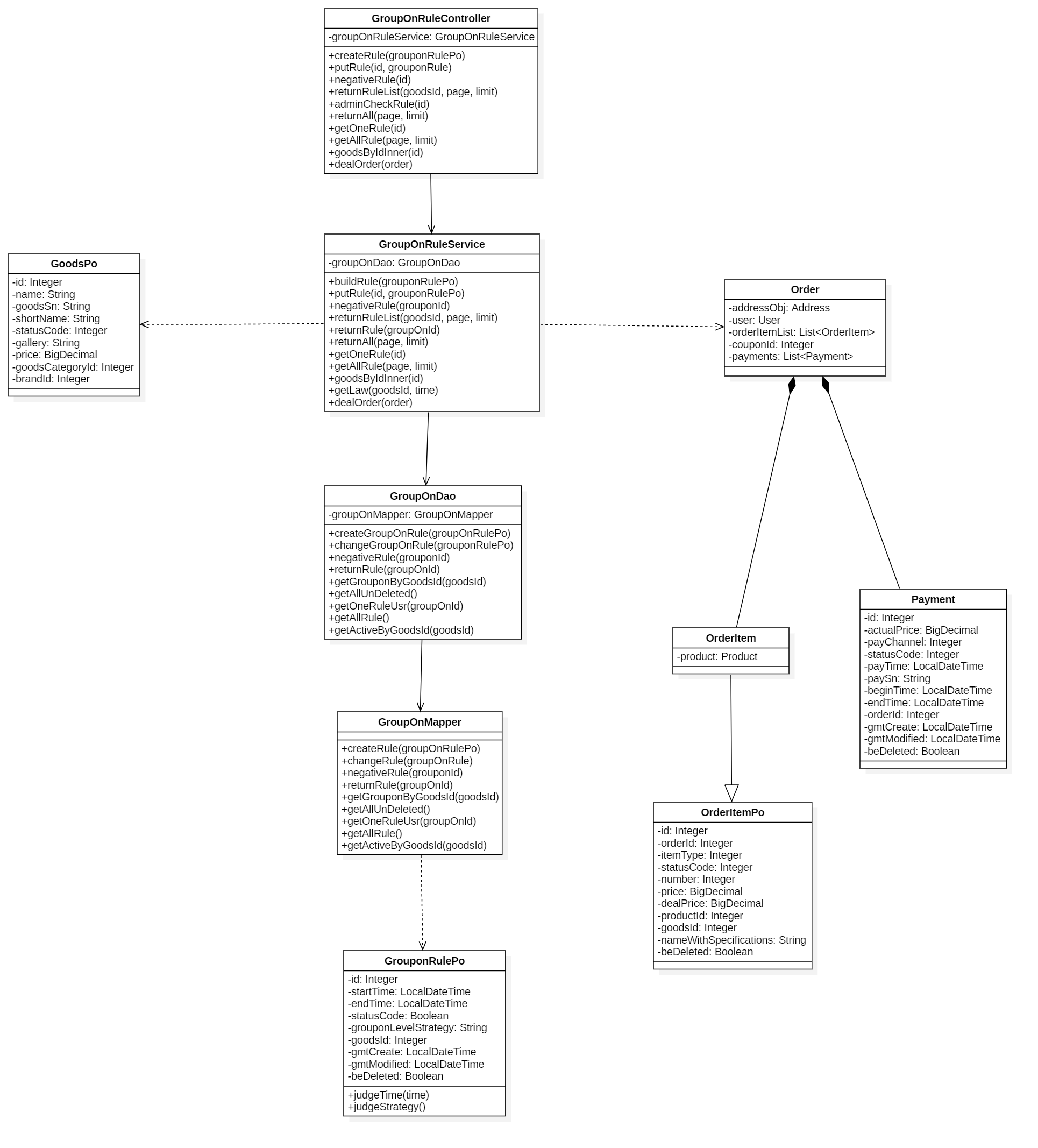
POST

/order

@RequestBody (Order)

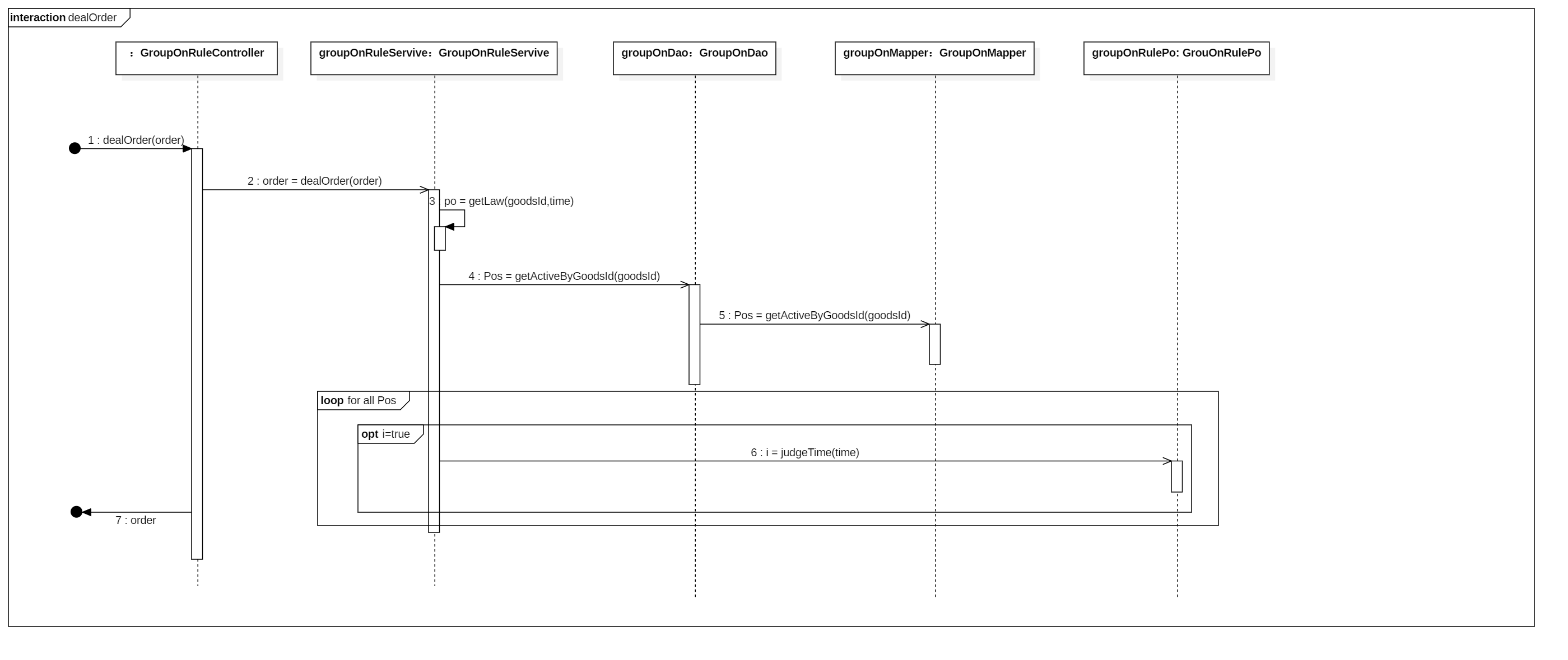
return ResponseUtil.ok()

### 2.3 Groupon类图

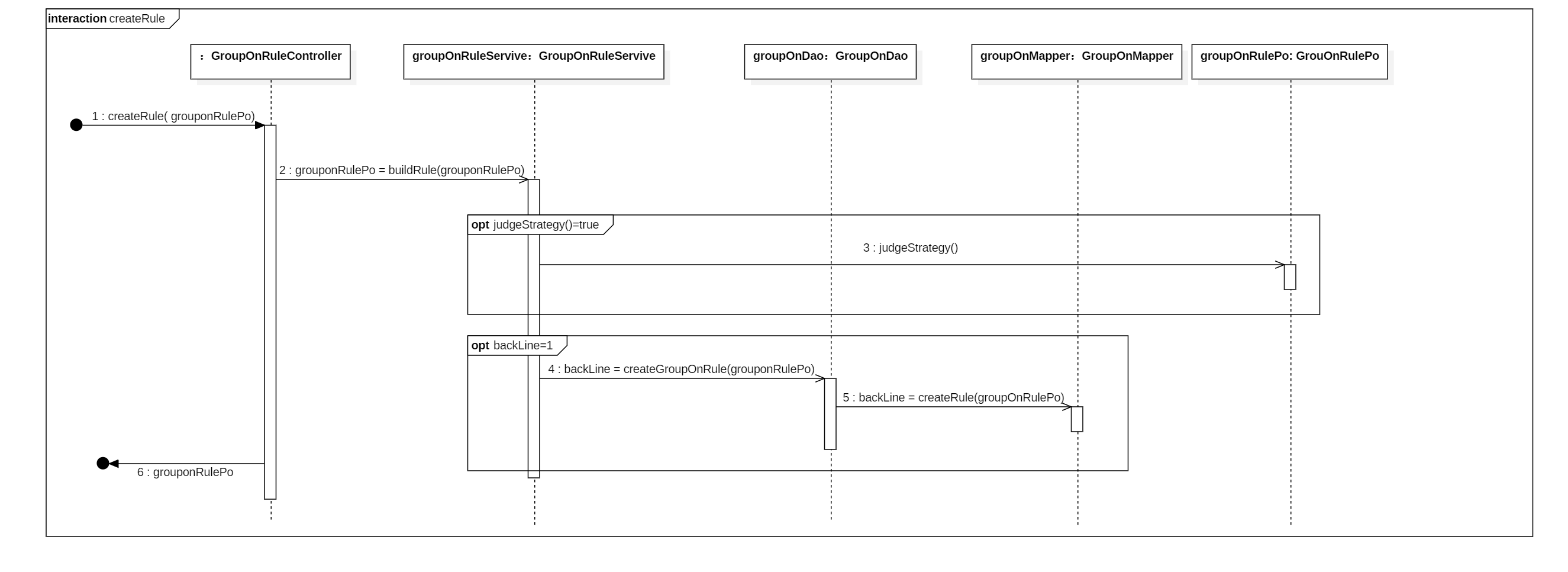


### 2.4 Groupon时序图

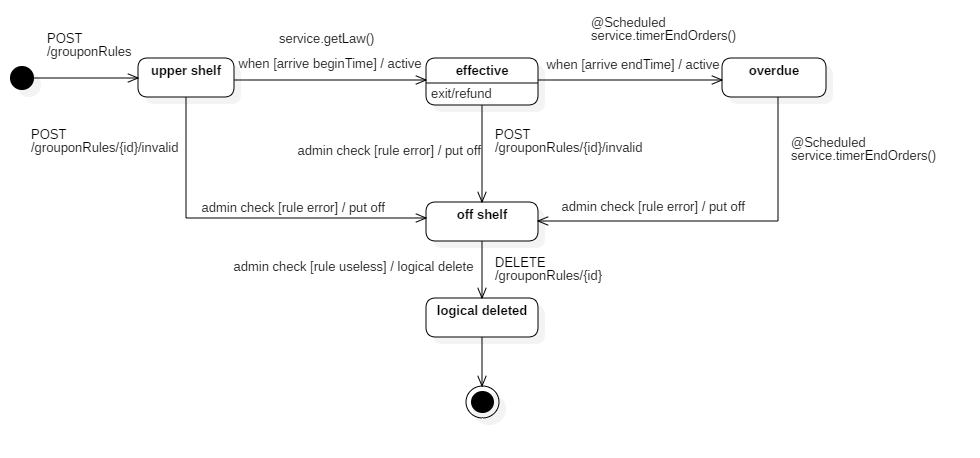
#### 2.4.1 dealOrder



#### 2.4.2 createRule



### 2.5 GrouponRule状态图



### 2.6 Groupon设计说明

分级团购需求：

在domain层中的GrouponRulePo.java有grouponLevelStrategy属性，这是一个json格式字符串，里面分别存储了多级别团购的策略，而其子类GrouponRule.java中有strategyList属性，其为由grouponLevelStrategy字符串解析而来，分级别存储策略的List<Strategy>结构体。为了体现面向对象设计的特点，我在domain层实现了这两个属性相互转化、形成的方法。

进行团购原价下单，结束返点的需求

按照v10的需求文档，团购活动的完成过程，是先按原价下单，待团购活动结束之后，按照团购分级策略，进行返点。为了完成这个需求，我们为团购设置了一个状态码，来记录一个团购的上下架情况。当一个团购活动被创建的时候，我们默认将其设置为已上架。当团购活动结束的时候，我们就会将其状态设置为已下架，并且调用Order模块的接口进行针对订单的返点。

团购不支持优惠券和返点支付的需求

当order传入discount中心的时候，discount中心会优先判断其是不是团购订单。若order的goodsId，下单时间，有符合的合法团购规则，则认为这是一份团购订单。但如果这份订单同时带有couponId或者rebate，那则会认为是非法订单，抛出报错。

## 3.presale部分

### 3.1 presale需求说明

商品的付款分为预售款和尾款两部分支付。商品在同一时刻预售和团购只能有一个活动生效，在一个团购或预售订单中只能有一条商品规格。预售不支持使用优惠卷和返点。管理员可以选择让正在进行中预售下线，下线的预售全额退款（包括定金）。

用户在提交团购订单时按照商品的定金价格支付订单，在尾款开始后，按照设置好的payment支付尾款（预售不支持使用优惠卷和返点）。

管理员可以选择让正在进行中预售下线，但只有已经下架的预售才可以删除。

### 3.2 presaleAPI说明

#### 3.2.1 presale外部API

用户查询所有预售商品

GET

/presaleRules

RequestParam (page)

RequestParam (limit)

RequestParam (beDeleted)

Return responseUtil.ok(List<PresaleRule>)

管理员查询所有预售商品

GET

/admins/presaleRules

RequestParam (page)

RequestParam (limit)

RequestParam (beDeleted)

return responseUtil.ok(List<PresaleRule>)

管理员创建一个预售商品

POST

/presaleRules

Requestbody(PresaleRule)

return ResponseUtil.ok(PresaleRule)

管理员根据presaleRuleId查看规则

GET

/admin/presaleRules/{id}

Pathvariable(id)

return ResponseUtil.ok(PresaleRule)

管理员下架规则

POST

/presaleRules/{id}/invalid

Pathvariable(id) Integer id

return ResponseUtil.ok()

管理员逻辑回复规则

POST

/{id}

Pathvariable(id) Integer id

return ResponseUtil.ok()

管理员修改规则

PUT

/presaleRules/{id}

Pathvariable(id) Requestbody(PresaleRule)

return ResponseUtil.ok(PresaleRule)

管理员删除规则

DELETE

/presaleRules/{id}

Pathvariable(id) Integer id

返回值：ResponseUtil.ok()即可

#### 3.2.2 presale内部API

给goods模块调用，查询某个goodsId的当前预售规则

GET

/goods/{id}

Pathvariable(id) Integer id

return ResponseUtil.ok(PresaleRule)

给goods模块调用，级联删除某个goodsId的当前预售规则

DELETE

/goods/{id}

Pathvariable(id) Integer id

return ResponseUtil.ok()

被discount中心调用，进行订单处理

POST

/order

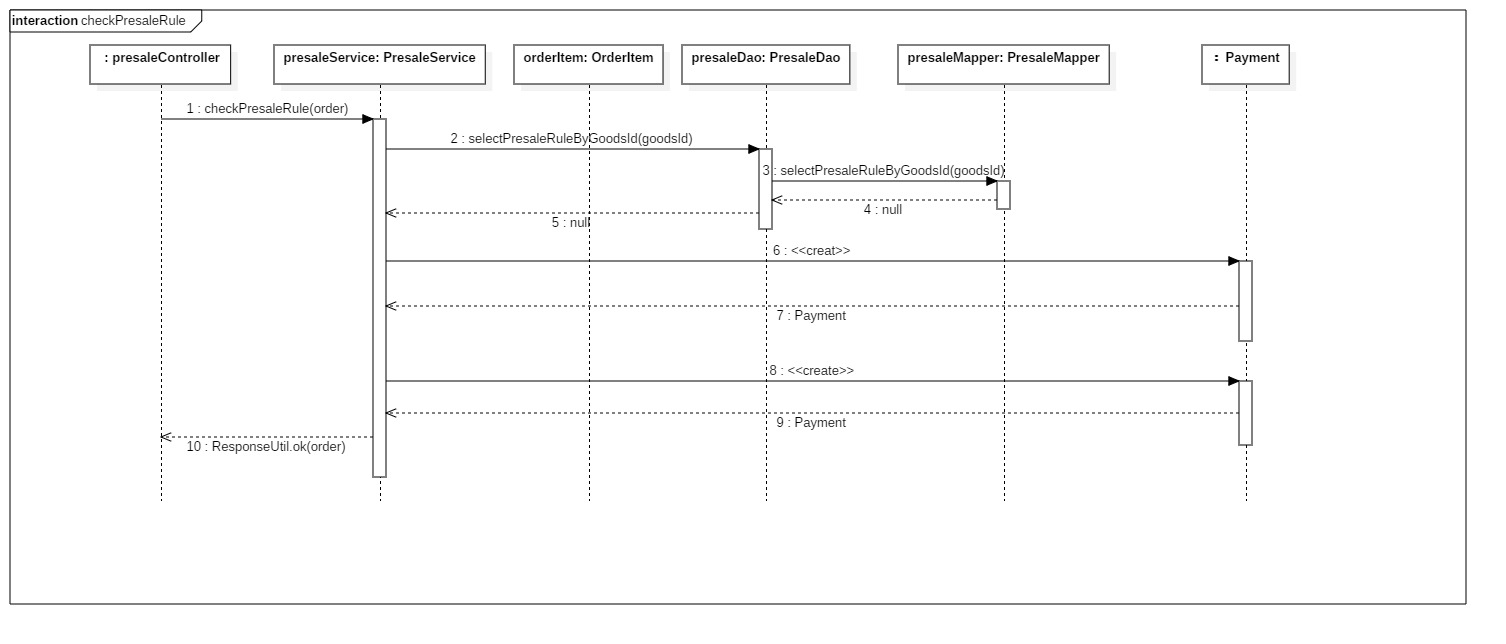
@RequestBody (Order)

return ResponseUtil.ok(Order)

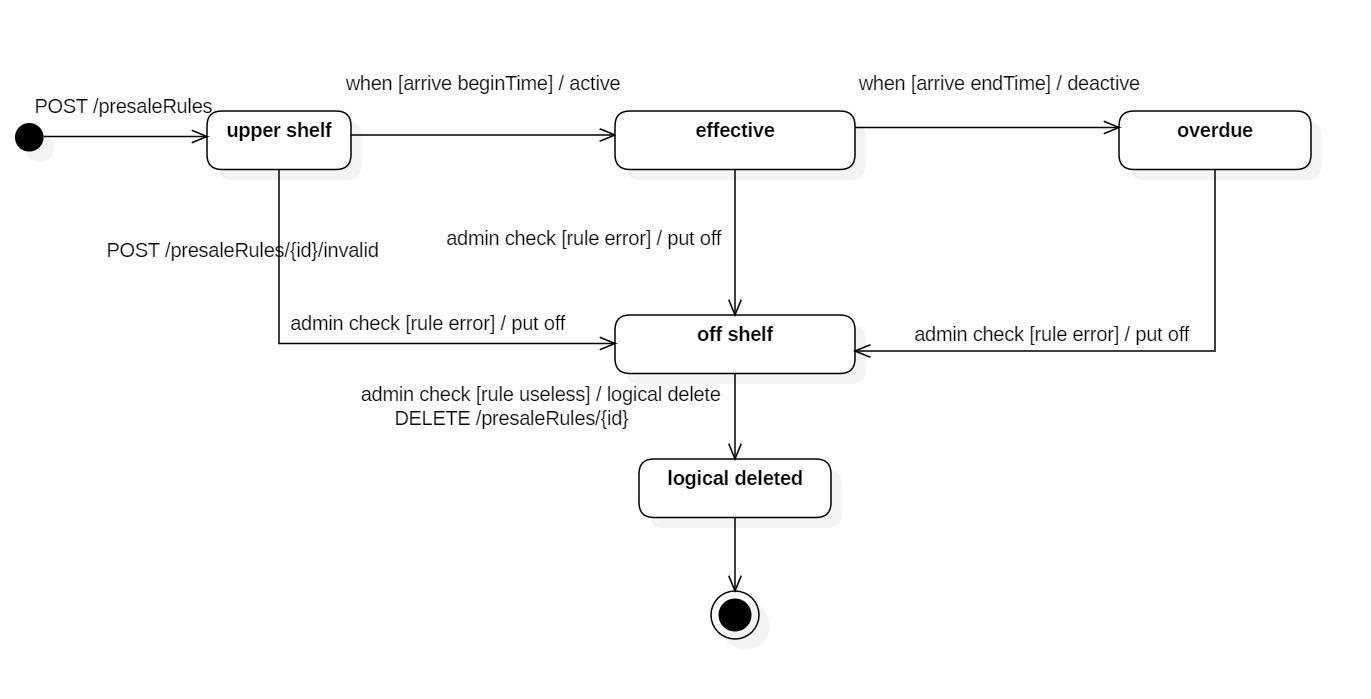
### 3.3 presale类图



### 3.4 presale时序图



### 3.5 presale状态图



### 3.6 presale设计说明

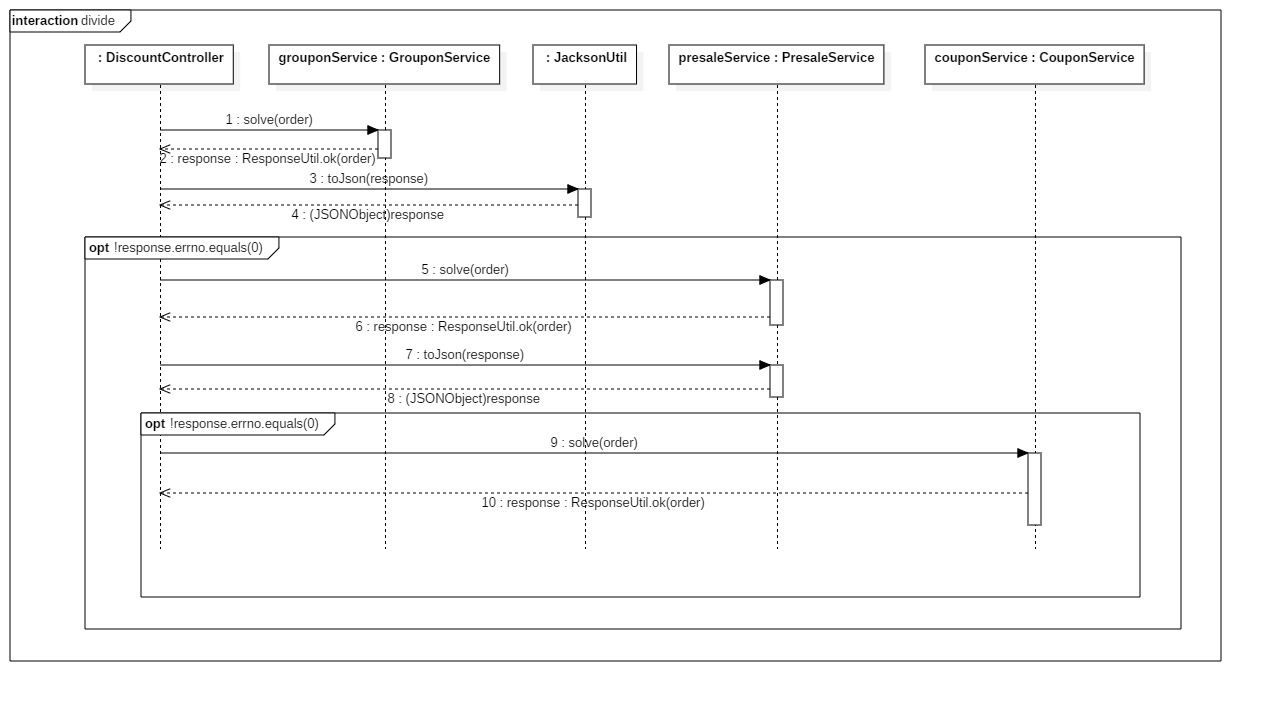
进行预售，按定金价下单

按照需求文档，想要完成一个预售订单，一定是在定金部分才下单。如果判断合法，则产生两个payment，设置金额为定金和尾款，结束

预售不支持优惠券和返点支付的需求

当 discount中心收到团购的回复的时候，如果discount中心发现order不是团购订单，则转交presale服务。若order的goodsId，下单时间，有符合的合法预售商品，则认为这是一份预售订单。但如果这份订单的couponId == null或者rebate == 0，那则会认为是非法订单，回复订单处理失败。

# 4.discount部分



# 四． UserInfo模块

## 1. user部分

### 1.1 user需求说明

主要分为浏览商城、购买、信息管理、售后服务和其他功能：

（1）用户在模块购物模块主要达到的目标为：浏览商品、下订单、 购物车管理。

（2）用户在账户管理主要达到的目标为：优惠券管理、收藏夹管理、订单管理、查看足迹、地址管理。

（3）用户在售后服务主要达到的目标为：退货、换货、联系客服、意见反馈。

（4）在其他功能模块主要达到的目标为：登陆、密码找回、注册、退出。

### 1.2 userAPI说明

#### 1.2.1 user外部API

用户登录

POST

/login

RequestBody UserDTO userDTO

return ResponseUtil.ok(User)

用户注册

POST

/register

RequestBody ConfirmDTO confirmDTO

return ResponseUtil.ok(User)

用户获得自己的信息

GET

/user

return ResponseUtil.ok(User)

用户修改自己的信息

PUT

/user

RequestBody User user

return ResponseUtil.ok(User)

用户修改自己的密码

PUT

/password

RequestBody ConfirmDTO confirmDTO

return ResponseUtil.ok()

用户修改自己的手机号

PUT

/mobile

RequestBody ConfirmDTO confirmDTO

return ResponseUtil.ok()

获得用户返点

GET

/rebate

return ResponseUtil.ok(rebate)

获得用户id

GET

/id

return ResponseUtil.ok(id)

获得用户等级

GET

/level

return ResponseUtil.ok(level)

获得验证码

POST

/captcha

RequestBody String moblie

return ResponseUtil.ok(code)

管理员查看用户信息

GET

/users/{id}

PathVariable("id") Integer userId

return ResponseUtil.ok(User)

管理员分页查看用户信息

GET

/users

RequestParam String page

RequestParam String limit

return ResponseUtil.ok(List<User>)

#### 1.2.2 user内部API

修改用户返点

PUT

/users/{id}/rebate

PathVariable("id") String userId

RequestBody String rebate

return ResponseUtil.ok(rebate)

获得用户信息

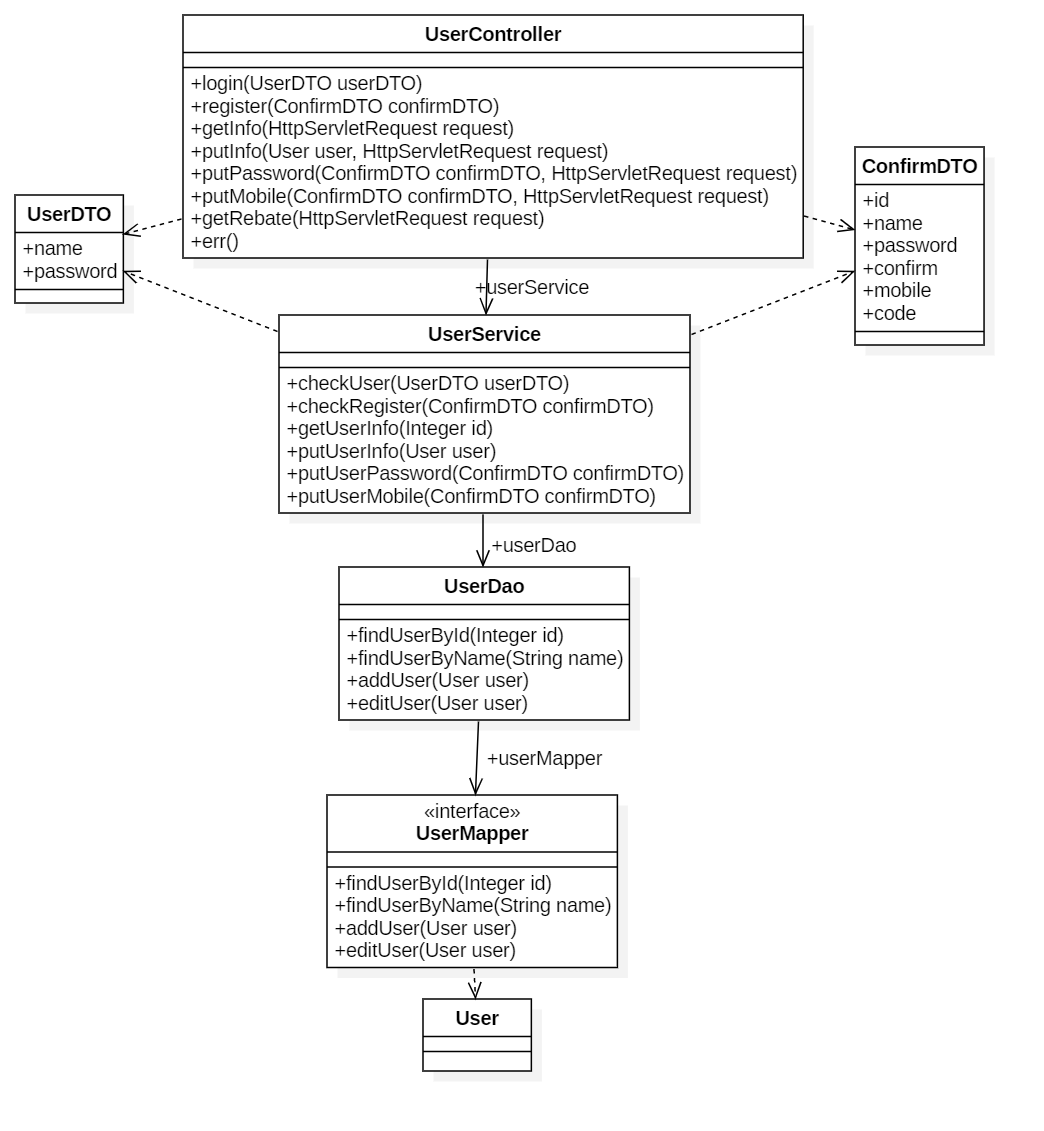
GET

/users/{id}/in

PathVariable("id") Integer userId

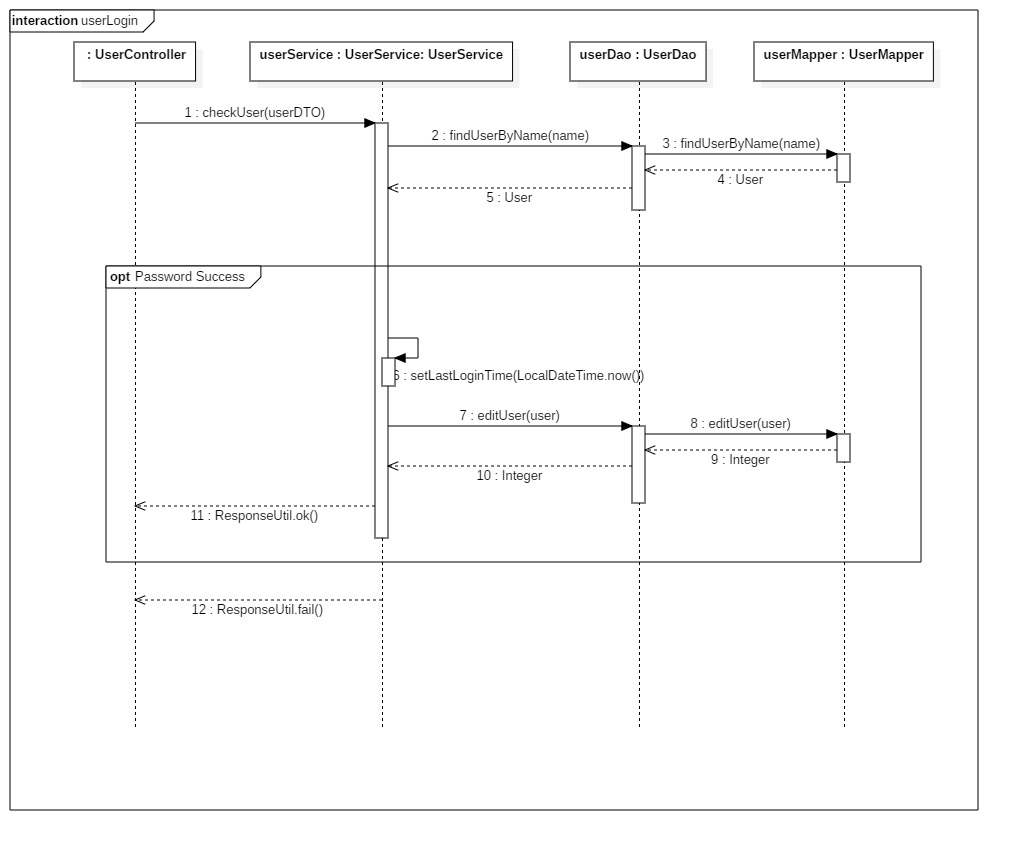
return ResponseUtil.ok(User)

### 1.3 user类图

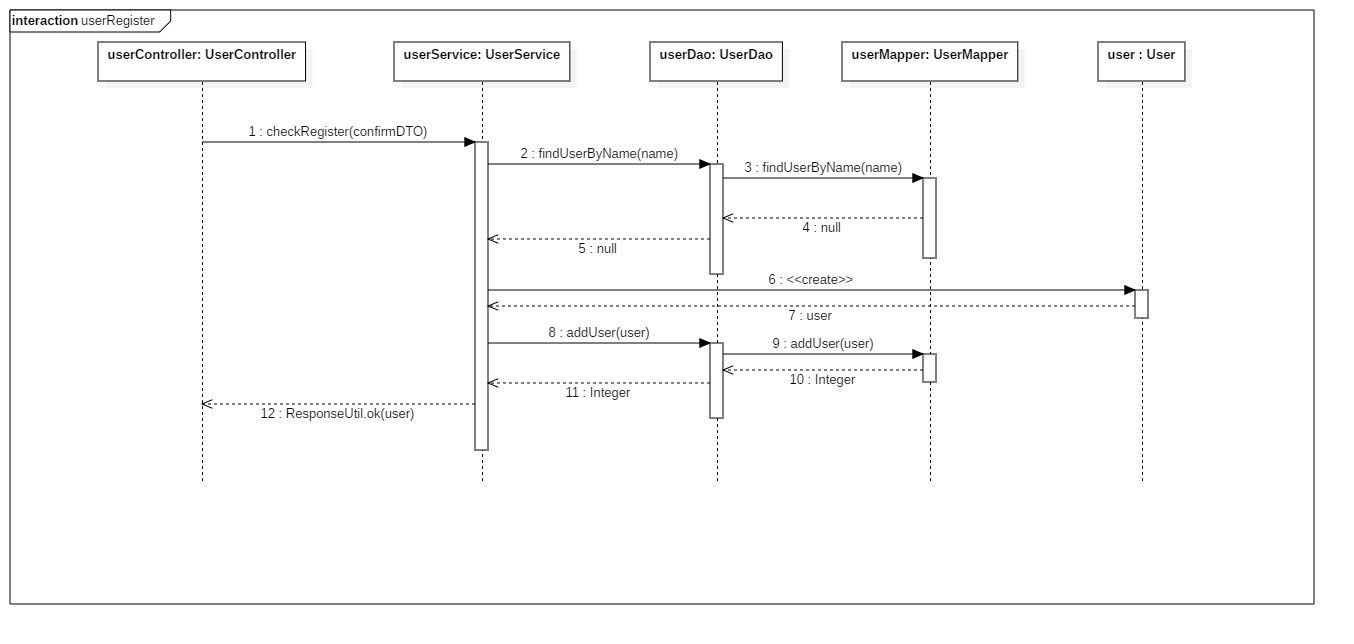


### 1.4 user时序图

#### 1.4.1 login



#### 1.4.2 register



### 1.5 设计思路

1. 明确基本功能：
   1. 增（注册）
   2. 不允许删
   3. 改
   4. 查（用户本人查询、管理员查询）
   5. 登录
2. 画出逻辑描述图片
   1. 类图
   2. 时序图
   3. 状态机图
3. 书写具体逻辑：
   1. 判空，防止空指针错误
   2. 判定数据合法性，防止脏数据
   3. 若是管理员操作，在具体操作前先写日志
   4. 修改数据库

## 2 admin部分

### 2.1 admin需求说明

主要分为用户管理、商场管理、商品管理、推广管理、系统管理、配置管理、统计报表和其他功能：

（1）用户管理模块主要达到的目标为：查看和导出用户、用户的收货地址、收藏、足迹、搜索历史和意见反馈。

（2）商场管理模块主要达到的目标为：查看行政区域，查看、添加和导出品牌制造商、订单，查看、添加、编辑和删除商品类目、通用问题、关键词。

（3）商品管理模块主要达到的目标为：添加和导出商品列表，编辑、删除商品，查看商品详情，管理商品上架，查看、审核、回复、删除商品评论。 （4）推广管理模块主要达到的目标为：添加、编辑广告，设置广告的上线时间和下线时间，将广告下线，优惠券、专题、团购规则、团购活动的管理。

（5）系统管理模块主要达到的目标为：查看管理员、操作日志，角色的授权、对象存储。

（6）配置管理模块主要达到的目标为： 配置商场、运费、订单、小程序信息。

### 2.2 adminAPI说明

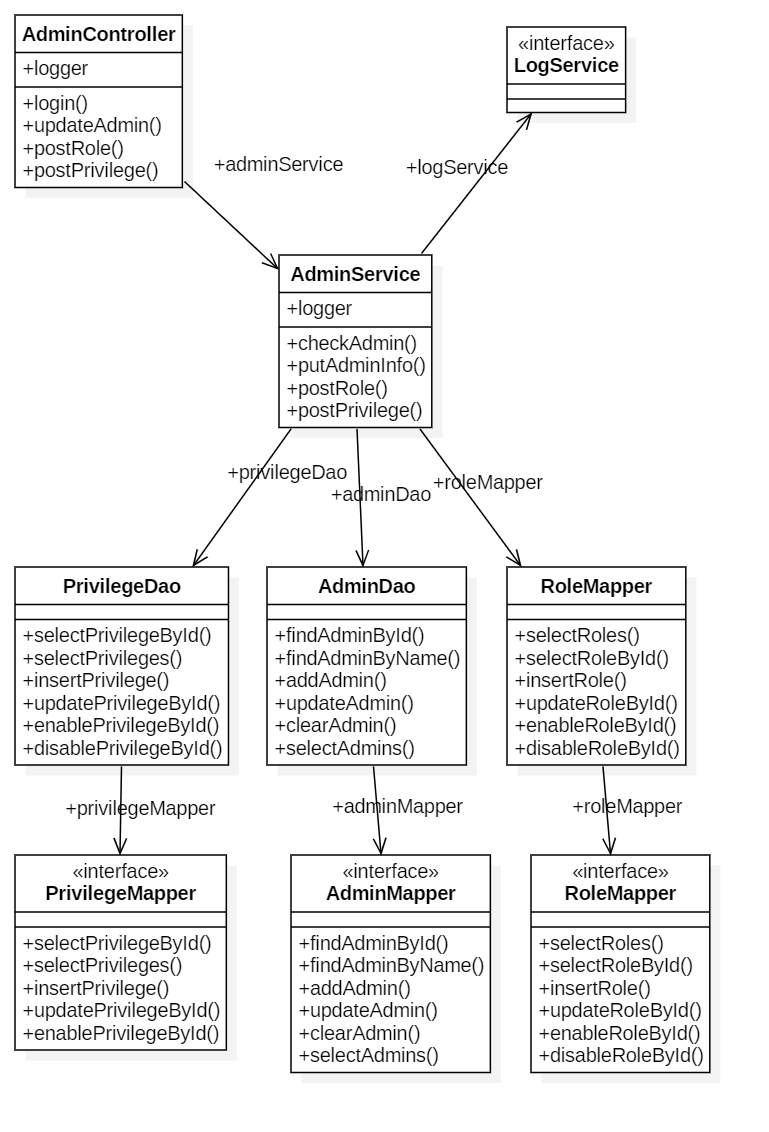
#### 2.2.1 admin外部API

管理员获得自己的信息  
/info  
GET  
return ResponseUtil.ok(Admin)  
  
高级管理员查看所有管理员（无姓名约束）  
/admins  
GET  
RequestParam String page   
RequestParam String limit  
RequestParam String baDeleted  
return ResponseUtil.ok(List<Admin>)  
  
高级管理员查看所有管理员（有姓名约束）  
/  
GET   
RequestParam String adminName  
RequestParam String page  
RequestParam String limit  
RequestParam String baDeleted  
return ResponseUtil.ok(List<Admin>)  
  
管理员登录  
/login  
POST  
RequestBody AdminDTO adminDTO  
return ResponseUtil.ok()  
  
高级管理员添加一个管理员  
/add  
POST   
RequestBody ConfirmDTO confirmDTO   
return ResponseUtil.ok(Admin)  
  
管理员修改自己的密码  
/password  
POST   
RequestBody ConfirmDTO confirmDTO   
return ResponseUtil.ok()  
  
高级管理员删除一个管理员  
/{id}  
DELETE  
PathVariable(“id”) String id  
return ResponseUtil.ok()  
  
高级管理员修改一个管理员  
/{id}  
PUT  
PathVariable(“id”) String id   
RequestBody AdminDTO adminDTO  
return ResponseUtil.ok()  
  
管理员查看角色列表  
/role  
GET  
RequestParam String page   
RequestParam String limit  
RequestParam String baDeleted  
return ResponseUtil.ok(List<Role>)  
  
管理员查看角色  
/role/{id}  
GET  
PathVariable(“id”) String id  
return ResponseUtil.ok(Role)  
  
管理员新增角色  
/role/{id}  
POST   
RequestBody Role role  
return ResponseUtil.ok(Role)  
  
  
管理员修改角色  
/role/{id}  
PUT  
PathVariable(“id”) String id   
RequestBody Role role  
return ResponseUtil.ok(Role)   
  
管理员使角色生效  
/role/{id}  
GET  
PathVariable(“id”) String id  
return ResponseUtil.ok()  
  
管理员使角色失效  
/role/{id}  
GET  
PathVariable(“id”) String id  
return ResponseUtil.ok()  
  
管理员查看权限列表  
/privilege  
GET  
RequestParam String page   
RequestParam String limit  
RequestParam String baDeleted  
return ResponseUtil.ok(List<Privilege>)  
  
  
管理员查看权限  
/privilege/{id}  
GET  
PathVariable(“id”) String id  
return ResponseUtil.ok(Privilege)  
  
管理员新增权限  
/privilege/{id}  
POST   
RequestBody Privilege privilege  
return ResponseUtil.ok(Privilege)  
  
  
管理员修改权限  
/privilege/{id}  
PUT  
PathVariable(“id”) String id   
RequestBody Privilege privilege  
return ResponseUtil.ok(Privilege)   
  
管理员使权限生效  
/privilege/{id}  
GET  
PathVariable(“id”) String id  
return ResponseUtil.ok()  
  
管理员使权限失效  
/privilege/{id}  
GET  
PathVariable(“id”) String id  
return ResponseUtil.ok()

#### 2.2.2 admin内部API

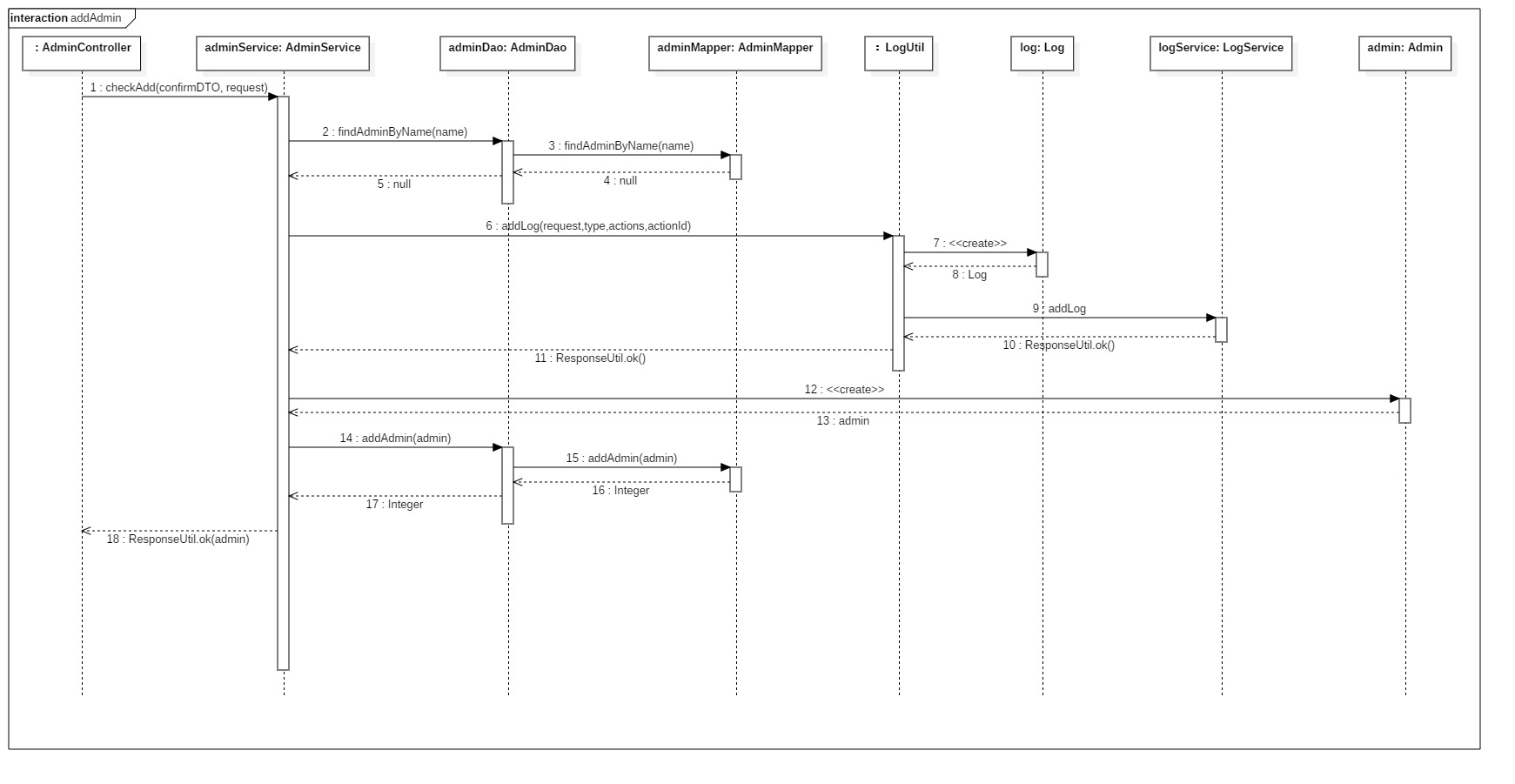
无

### 2.3 admin类图

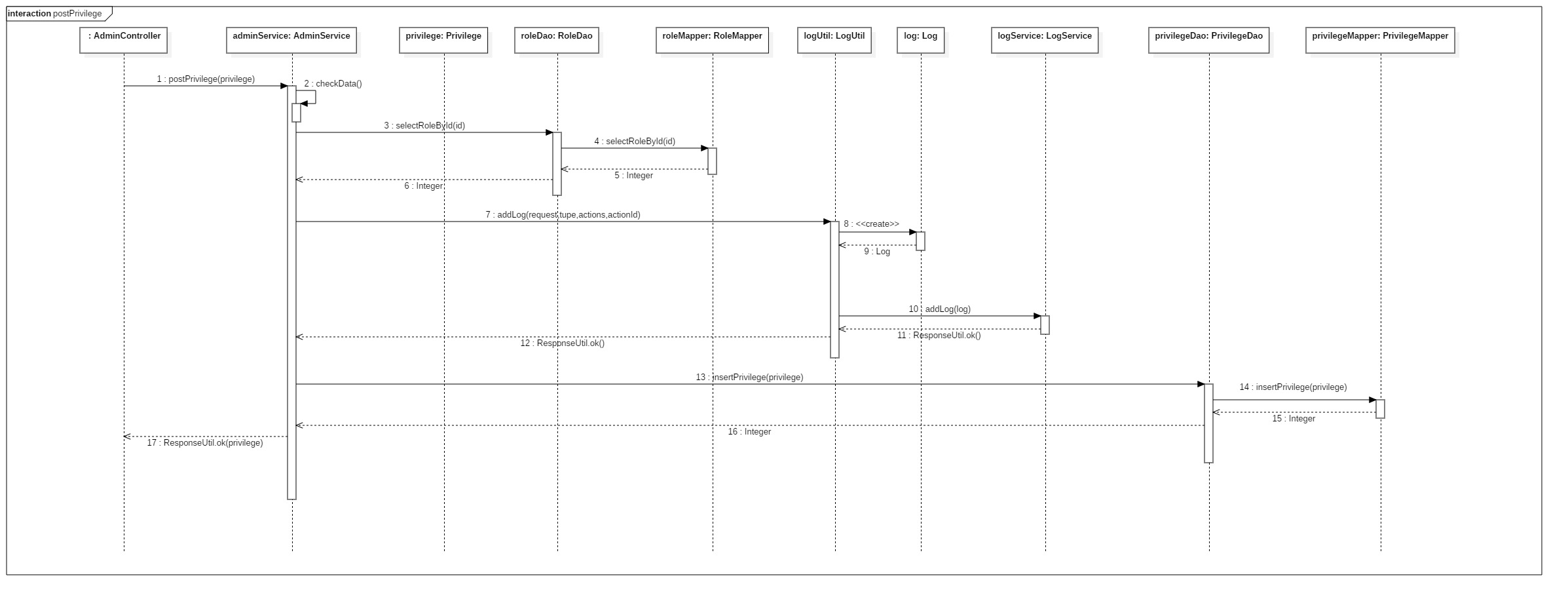


### 3.4 admin时序图

#### 3.4.1 addAdmin



#### 3.4.2 postPrivilege



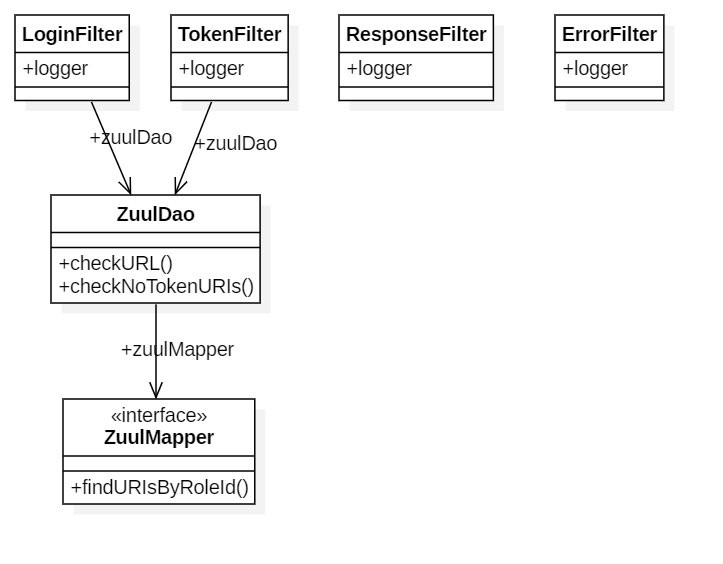
### 4.5 admin 设计思路

Admin设计思路

1. 明确基本功能：
   1. 增（由最高级管理员手动添加）
   2. 不允许删
   3. 改
   4. 查（管理员本人查询、高级管理员查询）
   5. 登录
2. 画出逻辑描述图
   1. 类图
   2. 时序图
   3. 状态机图
3. 书写具体逻辑：
   1. 判空，防止空指针错误
   2. 判定数据合法性，防止脏数据
   3. 在具体操作前先写日志
   4. 修改数据库

## 3 zuul网关部分

### 3.1 zuul类图



# 五．address模块

## address部分

### 1.1 address需求说明

地址模块：管理员对地址没有操作权限，用户可以查看自己的地址列表，查看地址详情，增加一条地址，修改一条地址，删除一条地址。根据带name的地址对象获取带id的地址对象。

**新增和修改地址时：**

1.对地址的省份id、城市id、地区id进行验证，例如：通过查找城市的pid，判断是否等于省份id，若不匹配，说明此城市不属于此省份，则插入失败。

2.查询省份id、城市id、地区id是否存在于数据库，若不存在，则代表不在地址选择范围内，则插入失败。

3.对电话号码进行验证，如果电话号码数小于11位，则插入失败。

4.对基本的地址信息进行判空：收获姓名、收货电话、省市县、地址详情任一数据不能为空。

5.若上述条件都不满足，插入成功。

**查看地址详情时：**

根据id查询出AddressPo对象，根据AddressPo中的省市县id，查询region表，拿到省市县的名称，再将数据填充到Address对象中，返回Address对象。

**删除地址时：**

1.查询数据库中是否存在这条数据，若不存在，则删除失败

2.查询这条地址的is\_deleted属性是否为true，为true代表已被删除，则删除失败

3.若1.2都不满足，则删除地址成功

### 1.2 addressAPI说明

#### 1.2.1 address外部API

外部API设计：

GET 用户获取地址列表

/addresses

参数：HttpServletRequest request, RequestParam (page), RequestParam (limit)

返回值：return List<address>

POST 用户新增一条地址

/addresses

参数：Requestbody(addressPo)

Body中有userId,cityId,provinceId,countyId, addressDetail, mobile, postalCode,consignee, beDefault

返回值：return addressPo//新增的addressPo

GET 用户查看地址详情

/addresses/{id}

参数：Pathvariable(id)

返回值：return address

PUT 用户修改一条地址

/addresses/{id}

参数：Pathvariable(id) Requestbody(addressPo)

Body中有userId,cityId,provinceId,countyId, addressDetail, mobile, postalCode,consignee, beDefault

返回值：return addressPo

DELETE 用户删除一条地址

/addresses/{id}

参数：Pathvariable(id)

返回值：无

### 1.2.2 address内部API

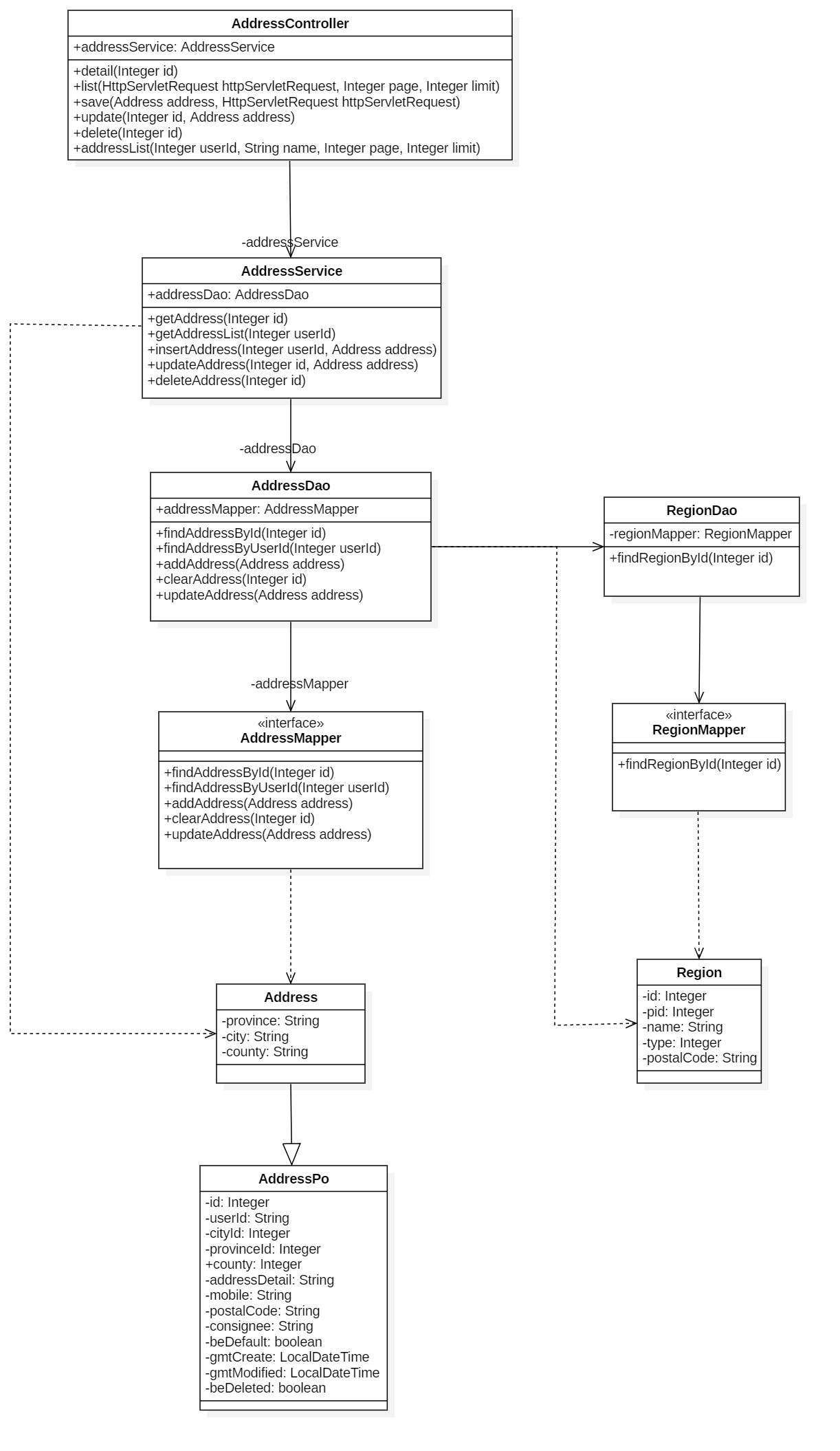
POST//根据只有地区id的地址对象，返回一个带地区id的地址对象

/inner/address

参数：Requestbody（Address）

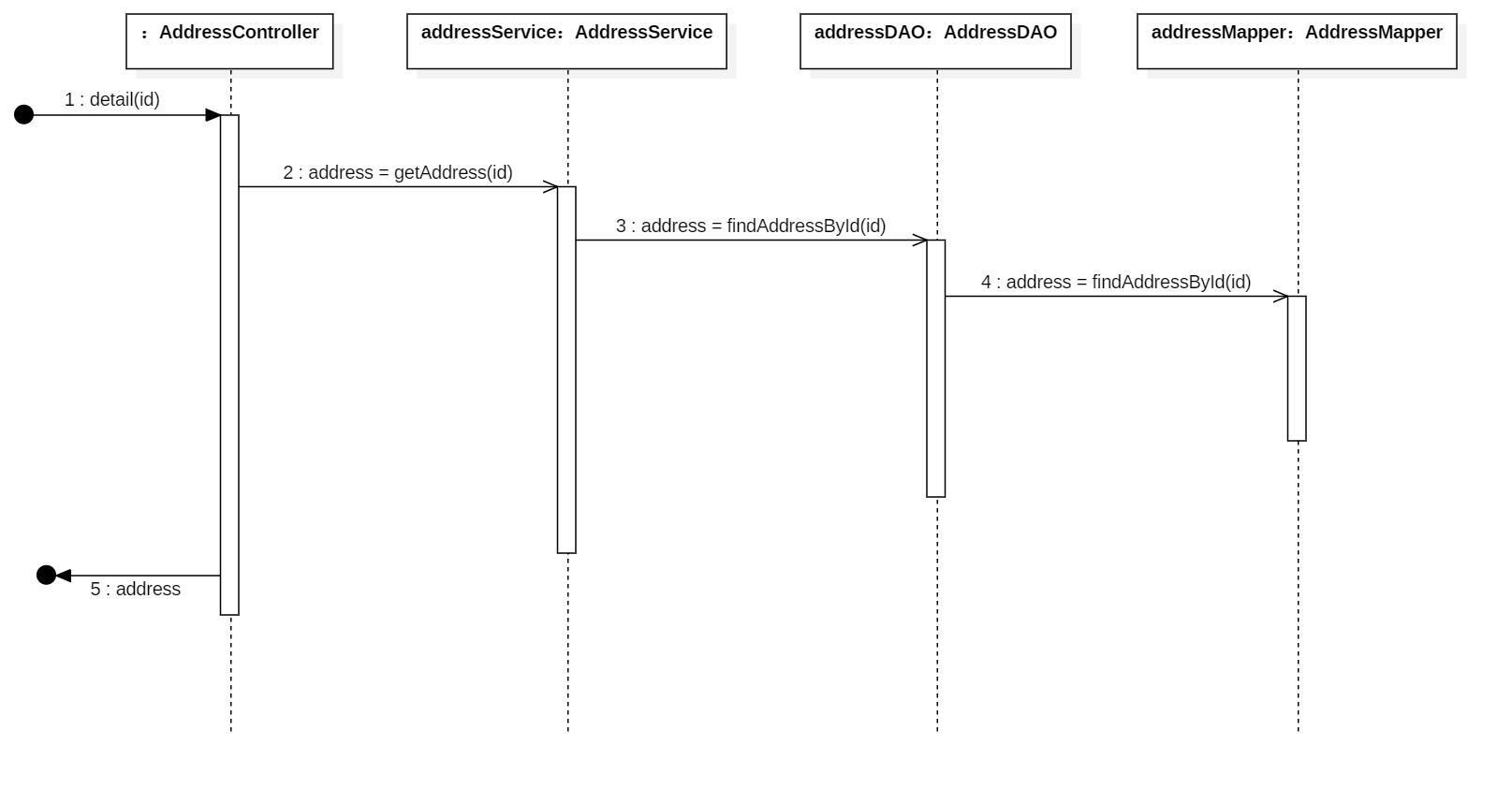
返回值：Address

### 1.3 address类图

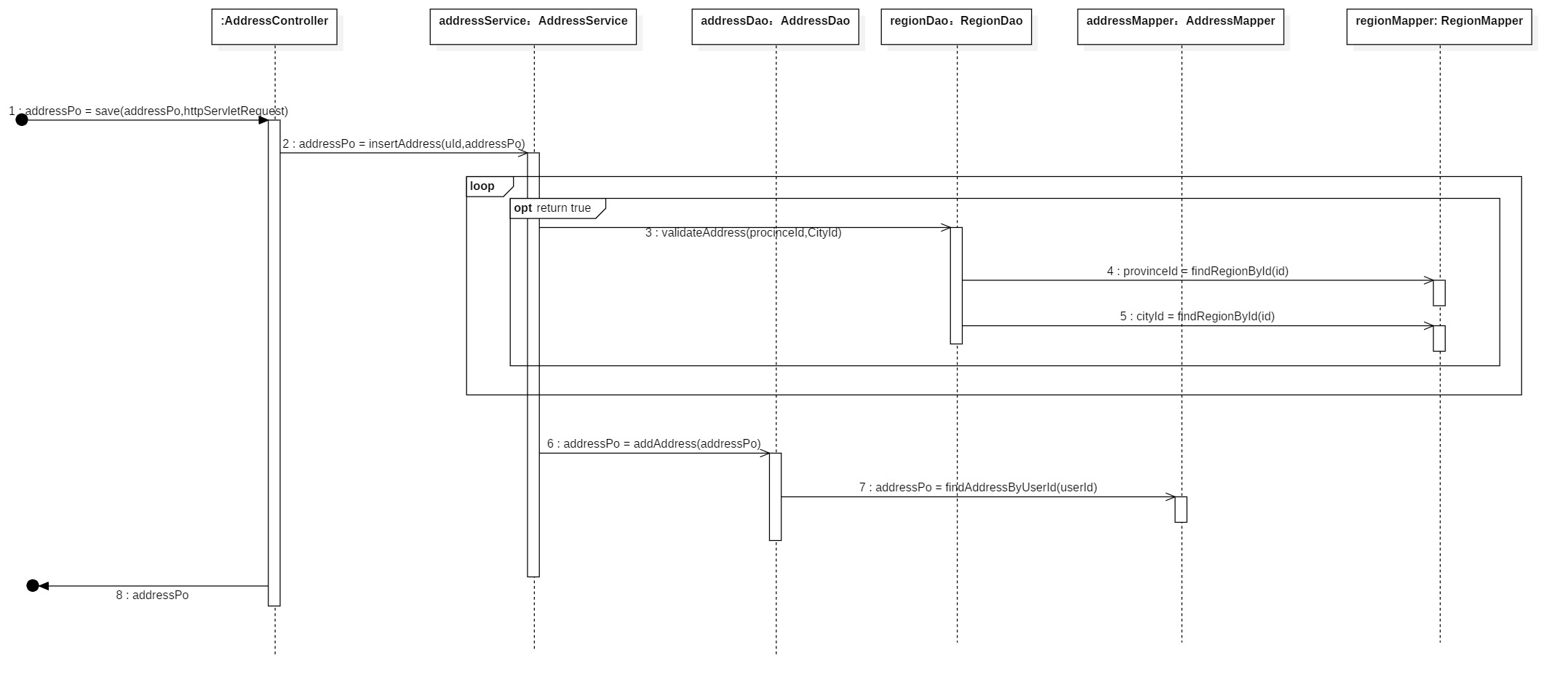


### 1.4 address时序图

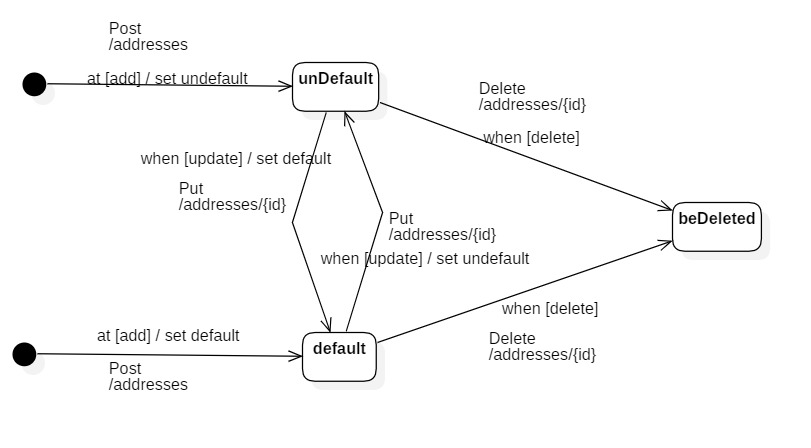
#### getAddress



#### 1.4.2 insertAddress



## 1.5 状态机图



### 1.6 address模块设计说明

地址模块只涉及增删改查，需求不多且不复杂，因此将大部分业务逻辑代码放在Service层，涉及到对对象属性的处理或判断的业务逻辑，放在domain层.

在对地址进行操作的过程中，对传入的数据进行判断放在controller层，对从数据库取出的数据进行验证放在dao层