五、详细设计说明书

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的其他模块（包括用户模块、优惠券模块、购物车模块、分享模块、售后模块、对账清算模块）中的需求。
此次编写为其他模块第一次迭代，目前项目进度处于实际代码的编写阶段。
根据课程要求，我们小组计划制作一个能支持高并发、大负载的电子商城系统，其命名为OOMALL。其旨在能让用户能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。编写此需求说明书是为此项目做服务，预期读者为开发人员和审核人员。

1.2 项目背景

a. 本项目来源于《JavaEE 平台技术》、《软件工程导论》、《面向对象分析与设计》课程设计大作业需求
b. 项目开发者与设计者：袁佳哲、刘赫昭、程昊天、徐荪睿、谢健祥
c. 客户端用户：有电商系统使用需求的用户，包括买家与卖家（店铺）
d. 后台管理用户：后台管理人员，店铺也可有限参与管理

1.3 定义

客户：Customer，使用电商系统进行购买等功能的用户，是系统的主要用户
卖家：Shop，也称之为店铺，是系统的第二主要用户
管理员：Admin，电商平台的管理员，负责日常维护和监控
最高管理员：ADMIN，电商平台创建之初的用户，最高权限的管理员，拥有所有权限
客户端：客户访问系统的途径，在本项目中客户端无用户界面，用户通过 API 进行访问
服务端：集业务逻辑、数据存储等功能与一身的服务器
MVC：Model View Controller ，是模型 (model)一视图 (view)一控制器（Controller）的缩写

1.4 参考资料

a. OOAD 课程文件《成绩计算办法 V14》
b. 阿里巴巴 Java 开发手册
c. Roger S.Pressman , Bruce R.Maxin. 软件工程 实践者的研究方法[M]：机械工业出版社.2018.8
d. 中国大学慕课厦门大学课程《JavaEE 平台技术》

2. 总体设计

2.1 需求概述

本学期的课程设计内容是采用面向对象的方法设计和实现一个高负载大并发的电子商城的后端系统。系统的需求基本**基于京东系统**，在此基础上增加或简化以下需求：
（黄色底色为其他模块内容， 灰色为必做模块内容）
1. **修改团购功能**：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. **修改商品分享功能**：支持店铺管理员在特定商品上设定推广提成规则（如推广 10 件，可获得 1%的返点，推广 11-50 件可获得 1.3%的返点）。允许用户通过微信用二维码或链接分享商品，如果其他用户通过二维码点击浏览商品，则认为分享成功。如果用户在分享成功后购买了该产品，则在清算时计入分享用户的返点。当商品有多个分享成功的情况，返给在活动有效时间内最早分享成功的用户。每次分享成功只能享受一次返点。可以为店铺定义默认的分享规则
3. **修改管理员的后台管理功能**：管理员后台所有的功能均需要记住由谁在何时新建、修改了数据

4. **支持用户用积点和优惠卷，以及通过微信和支付宝平台支付以及退款**。支付前端确定为小程序。用户在使用返点购买商品时,一点抵扣一分，不限制用户在一笔订单中使用返点的上限。退货时优先退回返点。

5. **添加平台级优惠券和店铺优惠券**

平台级优惠券：如限品类东券：满 300 减 30，第二或第三件半价（选择价钱最低的半价），商品满 3 件 7 折等等，我们要求这一部分的设计可扩展未来的新的优惠券种类,所有商铺都可以使用。我们的优惠券不可叠加

店铺优惠券：种类同平台级优惠券，但只能限定在本店铺使用，优惠券不能叠加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卷。优惠券需设定起止期限，在起止期限中才可以领用或使用优惠卷。管理员可以作废优惠活动，优惠活动作废后，活动所有领出未用的优惠卷也一并作废，已经使用的优惠卷不受影响。

优惠券不支持在团购和预售中使用

6. **增加商品预售功能**：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且只能是单纯预售或团购活动，不能同时附加优惠活动和分享活动。在一个团购或预售订单中只能有一条商品规格。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。

7. **修改商品评论功能**：商品的评论需要经过平台管理员审核后才能显示在商城中。删除商品亦一并删除商品的评论、用户在购物车内的商品、用户所有分享的商品

8. **修改查询功能**：删除关键字查询功能，改为以商品名称查询，只能查询起头的名称，删除查询历史的功能

9. **修改加密功能**：用户的密码、电话、Email 需要用 ASE 算法加密存储在数据库中

10. **定义商铺（商户）功能**：支持多商户的电子商城，每个商户仅仅可以管理自己商户的商品。商品的所有权是各个商户的，但需统一放在电子商城的仓库中进行销售和配送。每个商户需缴纳设定的保证金。保证金低于设定比例时，限制商户的退款交易。商户可定义多个账户，可设定清算时从平台转账优先顺序。

11. **支持用户敏感信息的保护功能**：如对于没有权限的用户，返回的订单信息中，用户的姓名只显示姓，名字用**代替，电话号码仅显示前三位和后四位，其他位用****代替

12. **增加优惠活动**：优惠活动与优惠卷相似，但不需要发行优惠卷，只需指定商品或分类的商品适用于某种活动（如满 300 减 30），同样优惠活动有起止时间

13. **修改发货功能**：在发货之前可以修改配送地址，修改地址不影响运费

14. **修改订单功能**：当订单中有多个店铺的商品时，用户支付订单后，需要分成多个店铺的订单。将支付金额和支付的积点按比例分摊给各子订单。团购和预售商品不能放入购物车中，既不能和其他商品合并在一个订单中。

15. **修改售后功能**：售后支付采用通过虚拟商品订单的方式支付。即允许商铺定义单价为 1 元的虚拟商品，让顾客通过购买虚拟商品的方式支付售后费用。

16. **添加对账清算功能**：

支持第三方支付平台的对账，以及第三方商铺的清算和个人用户的分享返点清算。

对账是每天凌晨都会自动进行的操作。对前一天的账目进行对账。清算是在对账完成后再做的操作，可以定义 T+n 的清算，即在对账完成后 n 天进行清算。

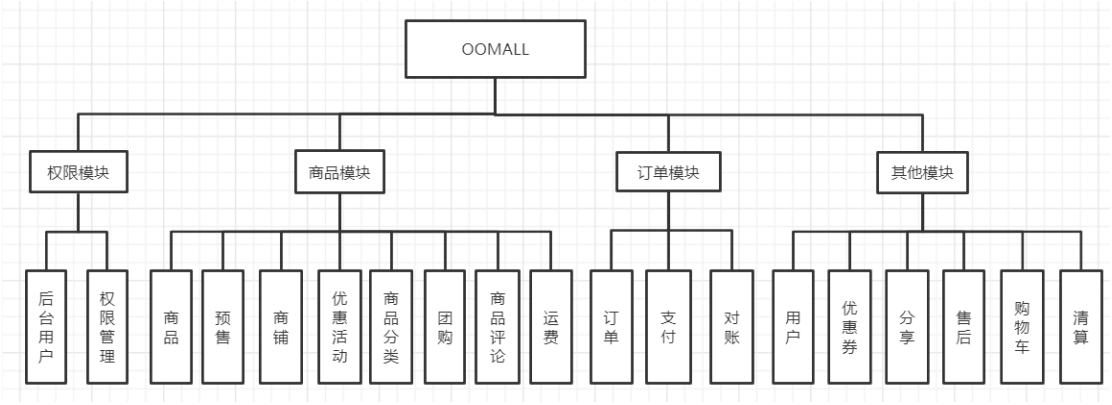
平台按照不同商品的类别抽取支付佣金。由于平台代为管理仓储和配送的商品，平台还需收取商品的物流费用。保证金不需参加清算。

清算时需分别计算商铺的订单收款、订单退款、售后退款和售后支付。订单收款需计算平台的支付佣金以及对分享用户的返点（一点抵扣一分钱）。订单退款和售后退款需根据对应收款情况计算退回的平台佣金、店铺收入和返点。售后支付全额清算给店铺。

如果清算出的转账金额为负数则从商铺保证金中扣除。多余款项汇出到商铺账户。如果商铺保证金低于设定比例时，则会在售后流程中暂停商铺订单的退款。清算的时候，如果商铺保证金低于设定比例，先补足商铺保证金，再把钱转账到商家指定账户。

2.2 软件结构

根据 OOAD 课程的划分，OOMALL 系统主要分为四个部分：权限模块、商品模块、订单模块、其他模块
其中，本组主要参与其他模块的设计与代码编写，以此接下来的程序描述等也主要围绕其他模块的下属模块展开



3. 程序描述

【逐个模块给出以下的说明:】

3.1 用户模块

3.1.1 功能

因考虑到购物车、优惠券、用户三个模块关联较为紧密，所有将这三个模块统一归属为用户模块
用户 Customer 模块的所有功能如下：

1. 管理员查看买家的所有状态：包括状态的对应代号与描述
2. 用户注册：注册信息包括电话号码、邮箱、用户名、密码、真实姓名，对注册信息进行合法性检验，检验完成即完成注册
3. 买家查看自己的信息：可查看的信息包括 id、用户名、真实姓名、电话号码、邮箱、状态值、返点数
4. 买家修改自己的信息：只允许用户修改自己的真实姓名
5. 买家重置自己的密码：需要买家提供用户名或邮箱或电话，然后向用户邮箱发送验证码，在有效期内，用户可以使用验证码修改自己的密码
6. 买家修改自己的密码：用户可以在验证码有效期内修改自己的密码

7. 管理员查看所有已注册的买家的信息：可查看的信息包括用户 id 和真实姓名
8. 买家使用用户名和密码登录：使用用户名和密码登录后，为用户创建唯一的 token，用于后续功能的权限校验
9. 买家登出
10. 管理员根据用户 id 查看任意买家信息：可查看的信息包括 id、用户名、真实姓名、电话号码、邮箱、状态值、返点数
11. 平台管理员根据用户 id 封禁买家
12. 平台管理员根据用户 id 解封买家
13. 买家查看自己的购物车列表：可查看的信息包括商品规格信息、商品数量、商品价格、商品参加的优惠活动
14. 买家将商品加入自己的购物车
15. 买家清空自己的购物车
16. 买家修改已在购物车内的某个商品的数量或规格
17. 买家删除自己购物车中的某个商品
18. 买家新增收货地址：需要提供的信息包括收获地区 id、详细地址、收件人、收件人电话，每个用户最多可有 20 个收货地址
19. 买家查询自己的所有收货地址
20. 买家将自己的某个收货地址设为默认收货地址：需要将原来的默认收货地址改为普通地址
21. 买家修改自己的某个收货地址
22. 买家删除自己的某个收货地址
23. 买家查看自己的优惠券列表并获取简要信息
24. 买家领取活动优惠券：优惠券须处于上线状态才可领取

3.1.2 性能

用户模块的所有功能中，只对领取活动优惠券有性能要求：可完成 1 秒领取 1000 张优惠券，且领取的结果全部（最低要求为 500 次/s）

3.1.3 输入项目

因本系统无前端界面，所有的请求仅可通过 HttpRequest 的方式进行输入，输入可包括四个部分：
method：http 请求方法类型
header：http 请求头部信息
path：http 请求 url 路径中的参数信息
body：http 请求 body 中的信息，采用 json 字符串格式
query：httl 请求 url 路径后附加参数信息

类型	功能	header	path	body	query
GET	管理员查看买家的所有状态				
POST	用户注册			可填写的用户信息 {	

				"mobile": "string", "email": "string", "userName": "string", "password": "string", "name": "string" }	
GET	买家查看自己的信息	用户 token			

PUT	买家修改自己的信息	用户 token		可修改的用户信息
				{
				"name": "string"
				}	

PUT	用户重置密码			邮箱/用户名/电话号码
				{
				"name": "string"
				}	

PUT	用户修改密码			验证码和新密码
				{
				"captcha": "string",
				"newPassword": "string"
				}	

GET	管理员查看所有已注册用户信息	管理员 token	管理员 id		用户名、用户邮箱、用户电话、结果列表起始页码、结果列表每页数量

POST	用户登录			用户名和密码
				{
				"userName": "string",
				"password": "string"
				}	

GET	用户登出	用户 token			
GET	管理员根据 id 查看任意买家信息	管理员 token	管理员 id 目标用户 id		
PUT	平台管理员根据用户 id 封禁买家	管理员 token	管理员 id 目标用户 id		
PUT	平台管理员根据用户 id 解禁买家	管理员 token	管理员 id 目标用户 id		

GET	买家查看自己的购物车列表	用户 token			结果列表起始页码、结果列表每页数量

POST	买家将商品加入自己的购物车	用户 token		要加入的商品的信息
				{
				"productId": 0,
				"quantity": 0
				}	

DELETE	买家清空购物车	用户 token			

PUT	买家修改自己的购物车内某个商品的数量或规格	用户 token	购物车 id	修改购物车单个商品信息
				{
				"productId": 0,
				"quantity": 0
				}	

DELETE	买家删除购物车中商品	用户 token	购物车 id		

POST	买家新增收货地址	用户 token		地址信息
				{
				"regionId": 0,
				"detail": "string",
				"consignee": "string",
				"mobile": "string"
				}	

GET	买家查询自己的所有收货地址	用户 token			结果列表起始页码、结果列表每页数目
PUT	买家将自己的某个收货地址设为默认收货地址	用户 token	地址 id		

PUT	买家修改自己的某个收货地址	用户 token	地址 id	可修改的地址信息
				{
				"regionId": 0,
				"detail": "string",
				"consignee": "string",
				"mobile": "string"
				}	

DELETE	买家删除自己的某个收货地址	用户 token	地址 id		
GET	买家查看自己的优惠券列表	用户 token			优惠券状态、结果列表起始页码、结果列

					表每页数目
POST	买家领取优惠券	用户 token	优惠活动 id		

3.1.4 输出项目

因本系统不设置前端界面，只可通过控制台或终端查看输出结果，结果以 HttpResponse 返回，有效内容包括以下三部分：
errno：操作状态码，操作成功为 0，操作失败为对应错误码
errmsg：操作状态信息，操作成功为“成功”，操作失败为对应错误信息
data：操作成功的返回值，具体返回值由功能定义

功能	正常输出	异常输出（错误码）

管理员查看买家的所有状态	操作成功通知，以及所有的状态和状态描述
	{
	"errno": 0,
	"errmsg": "string",
	"data": [
	{
	"code": 0,
	"name": "string"
	}
	]
	}	

用户注册	操作成功通知，以及自己的用户 id 和真实姓名
	{
	"errno": 0,
	"errmsg": "成功",
	"data": {
	"id": 0,
	"name": "string"
	}
	}
		611——电话已被注册
		612——邮箱已被注册
		613——用户名已被注册

买家查看自己的信息	操作成功通知，以及可以查看到的自己的信息
	{
	"errno": 0,
	"errmsg": "成功",
	"data": {
	"id": 0,	

	"userName": "string", "name": "string", "mobile": "string", "email": "string", "state": 0, "point": 0 } }	

买家修改自己的信息	操作成功通知
	{
	"errno": 0,
	"errmsg": "成功"
	}	

用户重置密码	操作成功通知，以及发送的返回码
	{
	"errno": 0,
	"errmsg": "成功",
	"data": {
	"captcha": "string"
	}
	}
		608——用户名/电话号码/邮箱不存在

用户修改密码	操作成功通知
	{
	"errno": 0,
	"errmsg": "成功"
	}	

管理员查看所有已注册用户信息	操作成功通知，以及所有用户信息列表
	{
	"errno": 0,
	"errmsg": "string",
	"data": {
	"page": 0,
	"pageSize": 0,
	"total": 0,
	"pages": 0,
	"list": [
	{
	"id": 0,
	"name": "string"
	}
	]
	}
	}	

用户登录	操作成功通知，以及用户 token
	{
	"errno": 0,
	"errmsg": "string",
	"data": "{token}"
	}
		609——用户名不存在或密码错误
		610——用户被禁止登录

用户登出	操作成功通知
	{
	"errno": 0,
	"errmsg": "成功"
	}	

管理员根据 id 查看任意买家信息	操作成功通知，以及对应用户信息
	{
	"errno": 0,
	"errmsg": "string",
	"data": {
	"id": 0,
	"userName": "string",
	"name": "string",
	"mobile": "string",
	"email": "string",
	"state": 0,
	"point": 0
	}
	}	

平台管理员根据用户 id 封禁买家	操作成功通知
	{
	"errno": 0,
	"errmsg": "成功"
	}	

平台管理员根据用户 id 解禁买家	操作成功通知
	{
	"errno": 0,
	"errmsg": "成功"
	}	

买家查看自己的购物车列表	操作成功通知，以及购物车信息列表
	{
	"errno": 0,
	"errmsg": "成功",
	"data": {
	"page": 0,
	"pageSize": 0,
	"total": 0,	

	"pages": 0,
	"list": [
	{
	"id": 0,
	"product": {
	"id": 0,
	"name": "string",
	"imageUrl": "string"
	},
	"quantity": 0,
	"price": 0,
	"couponActivity": [
	{
	"id": 0,
	"name": "string",
	"beginTime": "string",
	"endTime": "string"
	}
	]
	}
	]
	}
	}	

买家将商品加入自己的购物车	操作成功通知，以及加入商品的信息
	{
	"errno": 0,
	"errmsg": "成功",
	"data": {
	"id": 0,
	"quantity": 0,
	"price": 0
	}
	}	

买家清空购物车	操作成功通知
	{
	"errno": 0,
	"errmsg": "成功"
	}	

买家修改自己的购物车内某个商品的数量或规格	操作成功通知
	{
	"errno": 0,
	"errmsg": "成功"
	}	

买家删除购物车中商品	操作成功通知
	{
	"errno": 0,
	"errmsg": "成功"
	}	

买家新增收货地址	操作成功通知，以及添加的地址信息
	{
	"errno": 0,
	"errmsg": "成功",
	"data": {
	"id": 0,
	"region": {
	"id": 0,
	"name": "string"
	},
	"detail": "string",
	"consignee": "string",
	"mobile": "string",
	"beDefault": true
	}
	}
		601——达到地址簿上限

买家查询自己的所有收货地址	操作成功通知，以及买家的收货地址列表
	{
	"errno": 0,
	"errmsg": "string",
	"data": {
	"page": 0,
	"pageSize": 0,
	"total": 0,
	"pages": 0,
	"list": [
	{
	"id": 0,
	"region": {
	"id": 0,
	"name": "string"
	},
	"detail": "string",
	"consignee": "string",
	"mobile": "string",
	"beDefault": true
	}	

	] } }	

买家将自己的某个收货地址设为默认收货地址	操作成功通知
	{
	"errno": 0,
	"errmsg": "成功"
	}	

买家修改自己的某个收货地址	操作成功通知
	{
	"errno": 0,
	"errmsg": "成功"
	}	

买家删除自己的某个收货地址	操作成功通知
	{
	"errno": 0,
	"errmsg": "成功"
	}	

买家查看自己的优惠券列表	操作成功通知，以及买家的优惠券列表
	{
	"errno": 0,
	"errmsg": "成功",
	"data": {
	"page": 0,
	"pageSize": 0,
	"total": 0,
	"pages": 0,
	"list": [
	{
	"id": 0,
	"activity": {
	"id": 0,
	"name": "string",
	"imageUrl": "string",
	"beginTime": "string",
	"endTime": "string",
	"quantity": 0,
	"couponTime": "string"
	},
	"name": "string",
	"couponSn": "string"
	}
	]	

	} }	

买家领取优惠券	操作成功通知，以及领取的优惠券的详情信息
	{
	"errno": 0,
	"errmsg": "成功",
	"data": [
	{
	"id": 0,
	"activity": {
	"id": 0,
	"name": "string",
	"imageUrl": "string",
	"beginTime": "string",
	"endTime": "string",
	"quantity": 0,
	"couponTime": "string"
	},
	"customerId": 0,
	"name": "string",
	"couponSn": "string",
	"state": 0,
	"beginTime": "string",
	"endTime": "string",
	"gmtCreate": "string",
	"gmtModified": "string",
	"creator": {
	"id": 0,
	"name": "string"
	},
	"modifier": {
	"id": 0,
	"name": "string"
	}
	}
	]
	}
		630——未到领优惠券时间
		631——优惠券领罄
		632——优惠活动终止
		633——不可重复领优惠券

3.1.5 算法

用户模块涉及到算法或使用较复杂技术点的有以下功能：

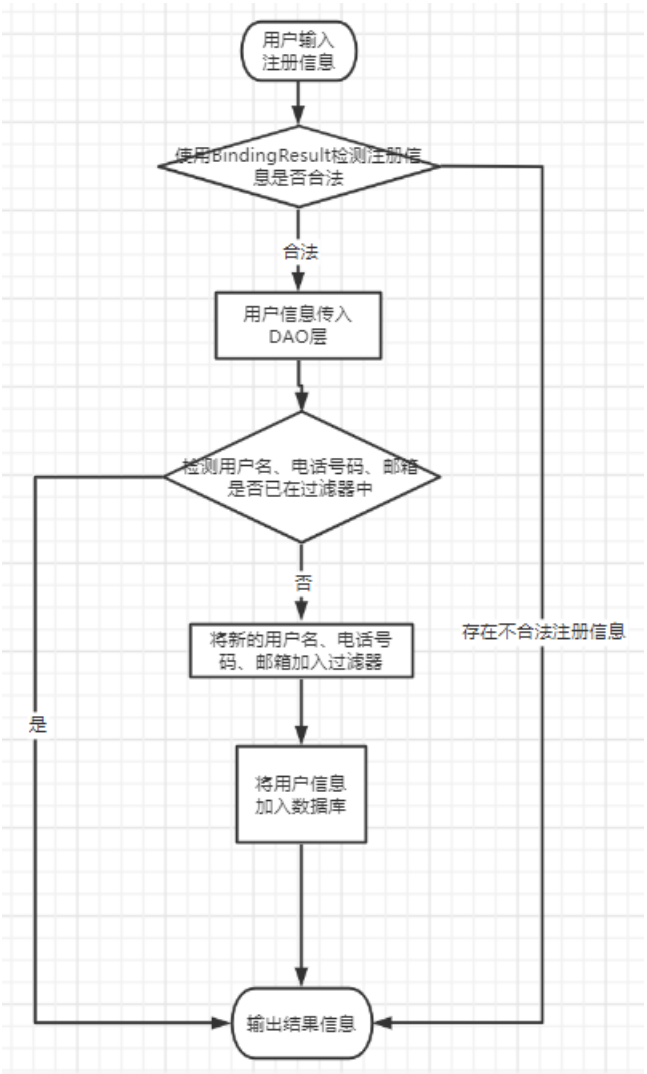
1. 用户注册：

需使用 redis 和布隆过滤器，在数据库没有 UNIQUE 索引的前提下，排查新注册的用户的用户名、邮箱、电话是否与已注册的用户重复

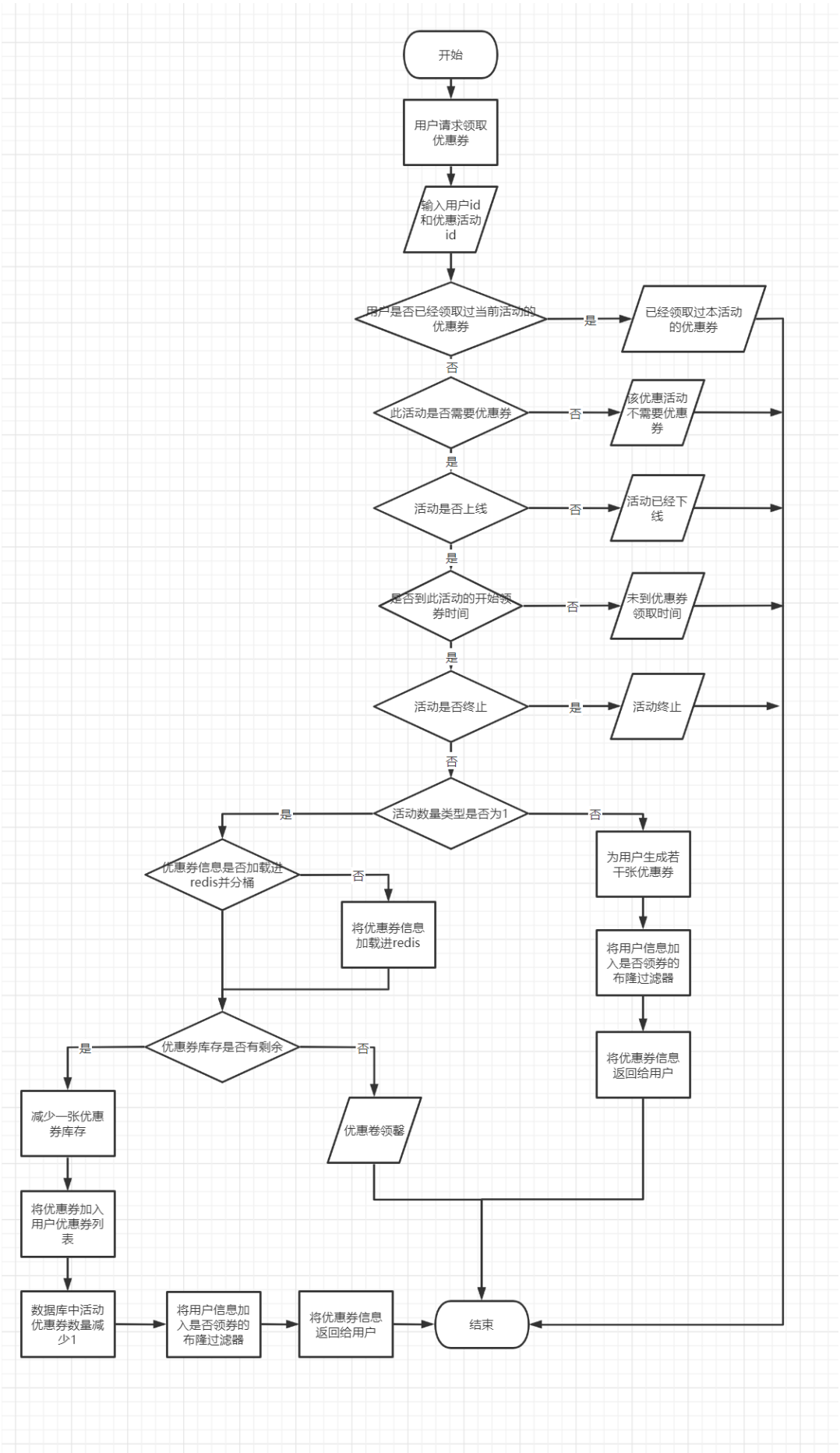
2. 领取优惠券：
   需要使用布隆过滤器来快速判断当前用户是否领取过该优惠券、使用 redis 分桶来提升读写优惠券数量的速度、使用 lua 脚本来将判断是否还有优惠券库存和扣优惠券库存写成一原子操作、使用 rocketmq 来发送异步消息提升系统的响应速度。

3.1.6 程序逻辑

用户注册算法逻辑：



领取优惠券算法：


开始
用户请求领取优惠券
输入用户id和优惠活动id
用户是否已经领取过当前活动的优惠券
是
已经领取过本活动的优惠券
否
此活动是否需要优惠券
否
该优惠活动不需要优惠券
是
活动是否上线
否
活动已经下线
是
是否到此活动的开始领券时间
否
未到优惠券领取时间
是
活动是否终止
是
活动终止
否
活动数量类型是否为1
是
否
优惠券信息是否加载进redis并分桶
否
将优惠券信息加载进redis
为用户生成若干张优惠券
将用户信息加入是否领券的布隆过滤器
将优惠券信息返回给用户
优惠券库存是否有剩余
是
否
减少一张优惠券库存
优惠卷领罄
将优惠券加入用户优惠券列表
数据库中活动优惠券数量减少1
将用户信息加入是否领券的布隆过滤器
将优惠券信息返回给用户
结束

3.1.7 接口

本模块所有功能通过 RESTful API 调用，请求类型 +API 可唯一确定一个接口，所有 API 如下：

功能	类型	API 接口	简单描述
管理员获得所有买家的状态	GET	/customers/states	
用户注册	POST	/customers	无需登录
买家查看自己的信息	GET	/self	
买家修改自己的信息	PUT	/self	
用户重置密码	PUT	/customers/password/reset	向用户邮箱发送验证码 用户在一段时间内能发送一次验证码
用户修改密码	PUT	/customers/password	密码长度为 6，且至少包含大写字母，小写字母，数字和特殊符号 验证码只能使用一次，验证码有效期 5 分钟
平台管理员获得所有用户列表	GET	/shops/{id}/customers/all	
用户登录	POST	/login	
用户登出	GET	/logout	
平台管理员查看任意买家信息	GET	/shops/{shopId}/customers/{id}	
平台管理员封禁买家	PUT	/shops/{did}/customers/{id}/ban	
平台管理员解禁买家	PUT	/shops/{did}/customers/{id}/release	
买家查看购物车列表	GET	/carts	
买家将商品加入购物车	POST	/carts	
买家清空购物车	DELETE	/carts	
买家修改购物车内商品数量与规格	PUT	/carts/{id}	
买家删除购物车内某个商品	DELETE	/carts/{id}	
买家新增地址	POST	/addresses	限制每个买家最多有 20 个地址
买家查询自己的所有收货地址	GET	/addresses	
买家将自己的某个收货地址设为默认收货地址	PUT	/addresses/{id}/default	需将原有的默认地址改为普通地址
买家修改自己的某个	PUT	/addresses/{id}	

收货地址			
买家删除自己的某个收货地址	DELETE	/addresses/{id}	
买家查看自己的优惠券列表	GET	/coupons	买家查看自己的优惠券列表并获取简要信息
买家领取优惠券	POST	/couponactivities/{id}/coupons	判断优惠活动的 quantityType 为 0，且用户无此优惠卷，生成优惠卷的数目为 quantity 判断优惠活动的 quantityType 为 1，去从优惠卷中领一张优惠卷 若优惠活动 quantity 为-1，表示该优惠活动不需要优惠卷，返回 504 错误

3.1.8 存储分配

用户模块对于硬盘空间无特殊要求，对于内存空间有以下要求：

要求项	内存要求	说明
JVM 虚拟机	256MB——512MB	可支持 jar 包运行即可
Redis	1GB——2GB	高速缓存，基于内存实现，用于支持高并发功能 在用户模块中主要用于支持布隆过滤器、用户注册、优惠券领取
RocketMQ	≥2GB	消息队列，用于减缓数据库读写压力，包含 nameserver 和 broker，对内存要求较高 在用户模块中主要用于支持优惠券领取
MySQL	1GB	不单独为 Customer 模块设立数据库，可与其他模块共用数据库 若有要求，后续可继续添加 MySQL 服务器

3.1.9 限制条件

1. 服务器限制

最多拥有 10 台服务器，其中只有两台服务器内存为 4G，其余服务器内存均为 2G。需

要在服务器数量与内存限制下，合理分配模块部署

2. 接口限制

本系统无前端界面，所有请求只可通过手动输入调用接口，只可通过控制台或终端查看输出结果

3. 运行环境限制

系统的部分功能必须借助 Linux 系统运行

4. 网络限制

系统所有模块仅可接收内网（172.16）的请求

3.1.10 测试要点

用户模块主要进行功能测试与性能测试

用户模块的功能测试要点如下：

1. 需求中所有功能测试正常
2. 需满足除 vo 和 po 类以外的所有类的白盒测试代码覆盖率均不小于 70%
3. 测试用例错误不超过 30 个

用户模块性能测试要点如下：

1. 测试 API：

   POST /coupon/couponactivities/{id}/coupons
2. 测试说明：

   用普通客户登录,选择一个优惠活动，集中领优惠卷, 要求 90%请求在 50ms 内完成，不能出现超领。
3. 测试及格指标：

   系统每秒可正确处理 500 次请求，且系统不崩溃

3.2 售后模块

3.2.1 功能

售后 aftersales 模块的所有功能如下：

1. 获得售后单的所有状态
2. 买家提交售后单
3. 买家查询自己的所有的售后单信息：按时间倒序，可以根据售后状态选择
4. 管理员查看所有售后单：管理员可以通过 id 查看所有售后单
5. 买家通过 id 查看售后单：只能查自己的售后单
6. 买家修改售后信息：在店家发货前买家可以修改信息
7. 买家取消售后单和逻辑删除售后单：售后单完成之前，买家取消售后单；售后单完成之后，买家逻辑删除售后单
8. 买家填写售后的运单信息
9. 买家确认售后单结束

10. 管理员根据售后单 id 查询售后单信息
11. 管理员同意/不同意（退款，换货，维修）
12. 店家验收收到买家的退（换）货：
    a) 如果是退款，则退款给用户，退款成功写入 price
    b) 如果是维修，根据维修价格新建售后支付对象，支付成功写入 price
    c) 如果是换货，新建售后订单，price 写 0
13. 店家发货：（维修：填写寄回的运单单号）
14. 获得售后单的的支付信息

3.2.2 性能

无

3.2.3 输入项目

因本系统无前端界面，所有的请求仅可通过 HttpRequest 的方式进行输入，输入可包括四个部分：
method：http 请求方法类型
header：http 请求头部信息
path：http 请求 url 路径中的参数信息
body：http 请求 body 中的信息，采用 json 字符串格式
query：httl 请求 url 路径后附加参数信息

类型	功能	header	path	body	query
GET	获得售后单的所有状态				
GET	获得售后单的的支付信息	用户 token	售后 id		
GET	买家查询所有的售后单信息（可根据售后状态选择）	用户 token			开始时间 售后状态 结束时间 页码 每页数目
GET	管理员查看所有售后单（可根据售后类型和状态选择）	用户 token			售后类型 售后状态 开始时间 结束时间 页码 每页数目
GET	买家根据售后单 id 查询售后单信息	用户 token	售后单 id		
PUT	买家修改售后单信息（店家	用户 token	售后单 id		买家可修改的信息：地址，售

	生成售后单前）				后商品的数量，申请售后的原因，联系人以及联系电话
PUT	买家填写售后的运单信息	用户 token	售后单 id		运单号
PUT	买家确认售后单结束	用户 token	售后单 id		
PUT	管理员同意/不同意（退款，换货，维修）	用户 token	店铺 id 售后单 id		处理意见
PUT	店家验收收到买家的退（换）货	用户 token	店铺 id 售后单 id		处理意见
PUT	店家发货	用户 token	店铺 id 售后单 id		运单号
DELETE	买家取消售后单和逻辑删除售后单	用户 token	售后单 id		
POST	买家提交售后单	用户 token	订单明细 id	售后服务信息	

3.2.4 输出项目

因本系统不设置前端界面，只可通过控制台或终端查看输出结果，结果以 HttpResponse 返回，有效内容包括以下三部分：
errno：操作状态码，操作成功为 0，操作失败为对应错误码
errmsg：操作状态信息，操作成功为“成功”，操作失败为对应错误信息
data：操作成功的返回值，具体返回值由功能定义

功能	正常输出	异常输出（错误码）

获得售后单的所有状态	{
	"errno": 0,
	"errmsg": "string",
	"data": [
	{
	"code": 0,
	"name": "string"
	}
	]
	}	

买家提交售后	{	

单	"errno": 0,
	"errmsg": "成功",
	"data": {
	"page": 0,
	"pageSize": 0,
	"total": 0,
	"pages": 0,
	"list": [
	{
	"id": 0,
	"serviceSn": "string",
	"type": 0,
	"reason": "string",
	"price": 0,
	"quantity": 0,
	"customerLogSn": "string",
	"shopLogSn": "string",
	"state": 0
	}
	]
	}
	}	

管理员查看所有售后单（可根据售后类型和状态选择）	{
	"errno": 0,
	"errmsg": "string",
	"data": {
	"page": 0,
	"pageSize": 0,
	"total": 0,
	"pages": 0,
	"list": [
	{
	"id": 0,
	"serviceSn": "string",
	"type": 0,
	"reason": "string",
	"price": 0,
	"quantity": 0,
	"customerLogSn": "string",
	"shopLogSn": "string",
	"state": 0
	}
	]
	}	

	}	

买家根据售后单 id 查询售后单信息	{
	"errno": {},
	"errmsg": "string",
	"data": {
	"id": 0,
	"orderId": 0,
	"orderItemId": 0,
	"customer": {
	"id": 0,
	"name": "string"
	},
	"shopId": "string",
	"serviceSn": "string",
	"type": 0,
	"reason": "string",
	"price": 0,
	"quantity": 0,
	"region": {
	"id": 0,
	"name": "string"
	},
	"detail": "string",
	"consignee": "string",
	"mobile": "string",
	"customerLogSn": "string",
	"shopLogSn": "string",
	"state": 0
	}
	}	

买家修改售后单信息（店家生成售后单前）	{
	"errno": 0,
	"errmsg": "成功"
	}	

买家取消售后单和逻辑删除售后单	{
	"errno": 0,
	"errmsg": "成功"
	}	

买家填写售后的运单信息	{
	"errno": 0,
	"errmsg": "成功"
	}	

买家确认售后单结束	{ "errno": 0,	

	"errmsg": "成功" }	

管理员根据售后单 id 查询售后单信息	{
	"errno": {},
	"errmsg": "string",
	"data": {
	"id": 0,
	"orderId": 0,
	"orderItemId": 0,
	"customer": {
	"id": 0,
	"name": "string"
	},
	"shopId": "string",
	"serviceSn": "string",
	"type": 0,
	"reason": "string",
	"price": 0,
	"quantity": 0,
	"region": {
	"id": 0,
	"name": "string"
	},
	"detail": "string",
	"consignee": "string",
	"mobile": "string",
	"customerLogSn": "string",
	"shopLogSn": "string",
	"state": 0,
	"creator": {
	"id": 0,
	"name": "string"
	},
	"gmtCreate": "string",
	"gmtModified": "string",
	"modifier": {
	"id": 0,
	"name": "string"
	}
	}
	}	

管理员同意/不同意（退款，换货，维修）	{ "errno": 0, "errmsg": "成功"	

	}	

店家验收收到买家的退（换）货	{
	"errno": 0,
	"errmsg": "成功"
	}	

店家发货	{
	"errno": 0,
	"errmsg": "成功"
	}	

获得售后单的的支付信息	{
	"errno": 0,
	"errmsg": "string",
	"data": {
	"id": 0,
	"tradeSn": "string",
	"patternId": 0,
	"documentId": "string",
	"documentType": 0,
	"descr": "string",
	"amount": 0,
	"actualAmount": 0,
	"payTime": "string",
	"state": 0,
	"beginTime": "string",
	"endTime": "string"
	}
	}	

3.2.5 算法

售后模块未使用复杂算法

3.2.6 程序逻辑

由于售后模块未使用复杂算法，故无特殊程序逻辑

3.2.7 接口

本模块所有功能通过 RESTful API 调用，请求类型 +API 可唯一确定一个接口，所有 API 如下：

功能	类型	API 接口	简单描述

获得售后单的所有状态	GET	/aftersales/states	无需登录
买家提交售后单	POST	/orderitems/{id}/aftersales	
买家查询所有的售后单信息（可根据售后状态选择）	GET	/aftersales	按照时间倒序,（买家通过这个 API 只能查询到自己的售后单）
管理员查看所有售后单（可根据售后类型和状态选择）	GET	/shops/{id}/aftersales	管理员可通过售后单 ID 查看所有售后单
买家根据售后单 id 查询售后单信息	GET	/aftersales/{id}	买家通过这个 API 只能查询到自己的售后单
买家修改售后单信息（店家生成售后单前）	PUT	/aftersales/{id}	在店家发货之前买家可以修改信息
买家取消售后单和逻辑删除售后单	DELETE	/aftersales/{id}	售后单完成之前，买家取消售后单；售后单完成之后，买家逻辑删除售后单
买家填写售后的运单信息	PUT	/aftersales/{id}/sendback	修改售后单状态
买家确认售后单结束	PUT	/aftersales/{id}/confirm	修改售后单状态
管理员根据售后单 id 查询售后单信息	GET	/shops/{shopId}/aftersales/{id}	
管理员同意/不同意（退款，换货，维修）	PUT	/shops/{shopId}/aftersales/{id}/confirm	可根据修改售后单的类型
店家验收收到买家的退（换）货	PUT	/shops/{shopId}/aftersales/{id}/receive	如果是退款，则退款给用户，退款成功写入 price 如果是维修，根据维修价格新建售后支付对象，支付成功写入 price 如果是换货，新建售后订单，price 写 0
店家发货	PUT	/shops/{shopId}/aftersales/{id}/deliver	维修：填写寄回的运单单号
获得售后单的的支付信息	GET	/aftersales/{id}/payments	

3.2.8 存储分配

售后模块对于硬盘空间无特殊要求，对于内存空间有以下要求：

要求项	内存要求	说明
JVM 虚拟机	256MB——512MB	可支持 jar 包运行即可
MySQL	1GB	不单独为 Aftersales 模块设立数据

		库，可与其他模块共用数据库 若有要求，后续可继续添加 MySQL 服务器

3.2.9 限制条件

1. 服务器限制

最多拥有 10 台服务器，其中只有两台服务器内存为 4G，其余服务器内存均为 2G。需要在服务器数量与内存限制下，合理分配模块部署

2. 接口限制

本系统无前端界面，所有请求只可通过手动输入调用接口，只可通过控制台或终端查看输出结果

5. 运行环境限制

系统的部分功能必须借助 Linux 系统运行

6. 网络限制

系统所有模块仅可接收内网（172.16）的请求

3.2.10 测试要点

售后模块的功能测试要点如下：

1. 需求中所有功能测试正常
2. 需满足除 vo 和 po 类以外的所有类的白盒测试代码覆盖率均不小于 70%
3. 测试用例错误不超过 30 个

3.3 清算模块

3.3.1 功能

清算模块的所有功能如下：

1. 获得清算单的所有状态
2. 平台管理员或商家获取符合条件的清算单
3. 商家或平台管理员根据清算单 id 查询清算单详细信息
4. 管理员按条件查某笔的进账
5. 管理员按条件查对应清算单的出账
6. 管理员按 id 查出账对应的进账
7. 开始清算
8. 用户获取自己因分享得到收入返点的记录

3.3.2 性能

无

3.3.3 输入项目

因本系统无前端界面，所有的请求仅可通过 HttpRequest 的方式进行输入，输入分为几种类型，每个类型可包括四个部分：
method：http 请求方法类型
header：http 请求头部信息
path：http 请求 url 路径中的参数信息
body：http 请求 body 中的信息，采用 json 字符串格式
query：httl 请求 url 路径后附加参数信息

类型	功能	header	path	body	query
GET	获取清算单所有状态	无	无	无	无
GET	平台管理员或商家获取符合条件的清算单简单信息	用户 token	店铺 Id	无	开始日期、结束日期、是否已打款、页码、每页数目
GET	查询指定清算单详情	用户 token	店铺 Id 清算单 Id	无	无
GET	管理员按条件查某笔的进账	用户 token	店铺 Id	无	订单 id、货品 id、清算单 id、页码、每页数目
GET	管理员按条件查对应清算单的出账	用户 token	店铺 id	无	订单 id、货品 id、清算单 id、页码、每页数目
GET	管理员按 id 查出账对应的进账	用户 token	店铺 id 出账单 id	无	无
PUT	开始清算	用户 token	店铺 id	开始清算所需信息 startInfo（开始时间和结束时间）	无
GET	用户获取自己因分享得到收入返点的记录	用户 token	无	无	开始日期（创建时间）、结束日期（创建时间）、页码、每页数目
GET	用户获取自己因分享得到收入返点的记录	用户 token	无	无	开始日期（创建时间）、结束日期（创建时间）、页码、每页数目

3.3.4 输出项目

因本系统不设置前端界面，只可通过控制台或终端查看输出结果，结果以 HttpResponse 返回，有效内容包括以下三部分：
errno：操作状态码，操作成功为 0，操作失败为对应错误码
errmsg：操作状态信息，操作成功为“成功”，操作失败为对应错误信息
data：操作成功的返回值，具体返回值由功能定义

功能	正常输出	异常输出（错误码）

获取清算单的所有状态	{
	"errno": 0,
	"errmsg": "成功",
	"data": {
	"code": 0,
	"name": "string"
	}
	}	

平台管理员或商家获取符合条件的清算单简单信息	{
	"errno": 0,
	"errmsg": "成功",
	"data": {
	"page": 0,
	"pageSize": 0,
	"total": 0,
	"pages": 0,
	"list": [
	{
	"id": 0,
	"shop": {
	"id": 0,
	"name": "string"
	},
	"liquidDate": "string",
	"expressFee": 0,
	"commission": 0,
	"shopRevenue": 0,
	"point": 0,
	"state": 0
	}
	]
	}
	}	

查询指定清算单详情	{ "errno": 0,	

	"errmsg": "成功",
	"data": {
	"id": 0,
	"shop": {
	"id": 0,
	"name": "string"
	},
	"liquidDate": "string",
	"expressFee": 0,
	"commission": 0,
	"shopRevenue": 0,
	"point": 0,
	"state": 0,
	"creator": {
	"id": 0,
	"name": "string"
	},
	"gmtCreate": "string",
	"gmtModified": "string",
	"modifier": {
	"id": 0,
	"name": "string"
	}
	}
	}	

管理员按条件查某的进账	{
	"errno": 0,
	"errmsg": "成功",
	"data": {
	"page": 0,
	"pageSize": 0,
	"total": 0,
	"pages": 0,
	"list": [
	{
	"id": 0,
	"shop": {
	"id": 0,
	"name": "string"
	},
	"product": {
	"id": 0,
	"name": "string"
	},	

	"amount": 0,
	"quantity": 0,
	"commission": 0,
	"point": 0,
	"shopRevenue": 0,
	"expressFee": 0,
	"creator": {
	"id": 0,
	"name": "string"
	},
	"gmtCreate": "string",
	"gmtModified": "string",
	"modifier": {
	"id": 0,
	"name": "string"
	}
	}
	]
	}
	}	

管理员按条件查对应清算单的出账	{
	"errno": 0,
	"errmsg": "成功",
	"data": {
	"page": 0,
	"pageSize": 0,
	"total": 0,
	"pages": 0,
	"list": [
	{
	"id": 0,
	"shop": {
	"id": 0,
	"name": "string"
	},
	"product": {
	"id": 0,
	"name": "string"
	},
	"amount": 0,
	"quantity": 0,
	"commission": 0,
	"point": 0,
	"shopRevenue": 0,	

	"expressFee": 0,
	"creator": {
	"id": 0,
	"name": "string"
	},
	"gmtCreate": "string",
	"gmtModified": "string",
	"modifier": {
	"id": 0,
	"name": "string"
	}
	}
	]
	}
	}	

管理员按 id 查出账对应的进账	{
	"errno": 0,
	"errmsg": "成功",
	"data": {
	"id": 0,
	"shop": {
	"id": 0,
	"name": "string"
	},
	"product": {
	"id": 0,
	"name": "string"
	},
	"amount": 0,
	"quantity": 0,
	"commission": 0,
	"point": 0,
	"shopRevenue": 0,
	"expressFee": 0,
	"creator": {
	"id": 0,
	"name": "string"
	},
	"gmtCreate": "string",
	"gmtModified": "string",
	"modifier": {
	"id": 0,
	"name": "string"
	}	

	} }	

开始清算	{
	"errno": 0,
	"errmsg": "成功"
	}	

用户获取自己因分享得到收入返点的记录	{
	"errno": 0,
	"errmsg": "成功",
	"data": {
	"page": 0,
	"pageSize": 0,
	"total": 0,
	"pages": 0,
	"list": [
	{
	"id": 0,
	"shop": {
	"id": 0,
	"name": "string"
	},
	"product": {
	"id": 0,
	"name": "string"
	},
	"amount": 0,
	"quantity": 0,
	"commission": 0,
	"point": 0,
	"shopRevenue": 0,
	"expressFee": 0,
	"creator": {
	"id": 0,
	"name": "string"
	},
	"gmtCreate": "string",
	"gmtModified": "string",
	"modifier": {
	"id": 0,
	"name": "string"
	}
	}
	]
	}	

	}	

用户获取自己因分享得到收入返点的记录	{
	"errno": 0,
	"errmsg": "成功",
	"data": {
	"page": 0,
	"pageSize": 0,
	"total": 0,
	"pages": 0,
	"list": [
	{
	"id": 0,
	"shop": {
	"id": 0,
	"name": "string"
	},
	"product": {
	"id": 0,
	"name": "string"
	},
	"amount": 0,
	"quantity": 0,
	"commission": 0,
	"point": 0,
	"shopRevenue": 0,
	"expressFee": 0,
	"creator": {
	"id": 0,
	"name": "string"
	},
	"gmtCreate": "string",
	"gmtModified": "string",
	"modifier": {
	"id": 0,
	"name": "string"
	}
	}
	]
	}
	}	

3.3.5 算法

清算模块未使用较为复杂的算法，仅仅是业务逻辑上的困难

3.3.6 程序逻辑

因清算模块未使用较为复杂的算法，其程序逻辑即为业务逻辑

3.3.7 接口

本模块所有功能通过 RESTful API 调用，请求类型 +API 可唯一确定一个接口，所有 API 如下：

功能	类型	API 接口	简单描述
获取清算单的所有状态	GET	/liquidation/states	
平台管理员或商家获取符合条件的清算单简信息	GET	/shops/{shopId}/liquidation	查询自己的符合条件的清算汇总，可根据以下条件搜索： • 开始时间 • 结束时间 • 是否已打款
查询指定清算单详情	GET	/shops/{shopId}/liquidation/{id}	商家或平台管理员根据清算单 id 查询清算单详细信息
管理员按条件查某笔的进账	GET	/shops/{shopId|/revenue	商铺管理员只能查看本商铺的进账信息，查询非本商铺信息返回 403 错误 平台管理员可以查看每个商铺的进账信息
管理员按条件查对应清算单的出账	GET	/shops/{shopId}/expenditure	商铺管理员只能查看本商铺的出账信息，查询非本商铺信息返回 403 错误 平台管理员可以查看每个商铺的出账信息
管理员按 id 查出账对应的进账	GET	/shops/{shopId}/expenditure/{id}/revenue	商铺管理员只能查看本商铺的出账信息，查询非本商铺信息返回 403 错误 平台管理员可以查看每个商铺的出账信息
开始清算	PUT	/shops/{shopId}/liquidation/start	

用户获取因自己分享得到收入返点的记录	GET	/pointrecords/revenue	按照创建时间倒序
用户获取自己因分享得到收入返点的记录	GET	/pointrecords/expenditure	按照创建时间倒序

3.3.8 存储分配

清算模块对于硬盘空间无特殊要求，对于内存空间有以下要求：

要求项	内存要求	说明
JVM 虚拟机	256MB——512MB	可支持 jar 包运行即可
MySQL	1GB	不单独为 liquidation 模块设立数据库，可与其他模块共用数据库若有要求，后续可继续添加 MySQL 服务器

3.3.9 限制条件

1. 服务器限制

最多拥有 10 台服务器，其中只有两台服务器内存为 4G，其余服务器内存均为 2G。需要在服务器数量与内存限制下，合理分配模块部署

2. 接口限制

本系统无前端界面，所有请求只可通过手动输入调用接口，只可通过控制台或终端查看输出结果

3. 运行环境限制

系统的部分功能必须借助 Linux 系统运行

4. 网络限制

系统所有模块仅可接收内网（172.16）的请求

3.3.10 测试要点

1. 对每一个 get 请求根据其提供的参数（条件），要求从数据库中查找到符合条件的正确记录并返回，对不合法的请求应返回相应的错误码。

2. 对开始清算的功能，要求做到清算出所给的时间范围内所有平台支付佣金和物流费用（在实际中应为一天一次清算），所有商铺的订单收款、订单退款、售后退款和售后支付。正确计算佣金、返点和店铺收入等字段，正确判别商铺 id，分享者 id，货品 id 等标识信息，将清算好的清算单和相应的入账单和出账单存入数据库。

3.4 分享模块

3.4.1 功能

分享 Share 模块的所有功能如下：

1. 分享者生成分享链接：根据买家 id 和商品 id 生成唯一的分享链接。
2. 顾客查询所有自己的分享记录：买家查询所有分享记录，可设置货品 id、开始时间、结束时间、页码、页大小等可选参数。
3. 查看商品的详细信息（需登录，从分享模式查看商品）：根据分享 id 和货品 id 查看分享的商品
4. 管理员查询商品分享记录：商铺管理员只能查询本店商铺的商品，可进行分页查看。
5. 分享者查询所有分享成功记录：可设置货品 id、开始时间、结束时间、页码、页大小等可选参数。
6. 管理员查询所有分享成功记录：只能查询本店商铺的商品，可设置开始时间、结束时间、页码、页大小等可选参数。

3.4.2 性能

分享模块的功能对性能的要求不大。

3.4.3 输入项目

因本系统无前端界面，所有的请求仅可通过 HttpRequest 的方式进行输入，输入可包括四个部分：

method：http 请求方法类型

header：http 请求头部信息

path：http 请求 url 路径中的参数信息

body：http 请求 body 中的信息，采用 json 字符串格式

query：httl 请求 url 路径后附加参数信息

类型	功能	header	path	body	query
POST	分享者生成分享链接	用户 token	货品销售 id		
GET	顾客查询所有自己的分享记录	用户 token			货品 Id、分享创建时间区间的开始时间、分享创建时间区间的结束时间、页数、页大小
GET	查看商品的详细信息（需登录，从分享模式查看商品）	用户 token	分享 ID 货品 ID		
GET	管理员查询商品分享记录	用户 token	店铺 Id 货品 Id		页数、页大小

GET	分享者查询所有分享成功记录	用户 token			货品 Id、分享创建时间区间的开始时间、分享创建时间区间的结束时间、页数、页大小
GET	管理员查询所有分享成功记录	用户 token	店铺 Id 货品 Id		分享创建时间区间的开始时间、分享创建时间区间的结束时间、页数、页大小

3.4.4 输出项目

因本系统不设置前端界面，只可通过控制台或终端查看输出结果，结果以 HttpResponse 返回，有效内容包括以下三部分：
errno：操作状态码，操作成功为 0，操作失败为对应错误码
errmsg：操作状态信息，操作成功为“成功”，操作失败为对应错误信息
data：操作成功的返回值，具体返回值由功能定义

功能	正常输出	异常输出（错误码）

分享者生成分享链接	操作成功通知以及分享的信息
	{
	"errno": 0,
	"errmsg": "成功",
	"data": {
	"id": "string",
	"sharer": {
	"id": 0,
	"name": "string"
	},
	"onsale": {
	"id": 0,
	"shop": {
	"id": 0,
	"name": "string"
	},
	"product": {
	"id": 0,
	"name": "string",
	"imageUrl": "string"
	},
	"price": 0,
	"beginTime": "string",
	"endTime": "string",
	"quantity": 0,
	"type": 0,
	"activityId": 0,	

	"shareActId": 0,
	"numKey": 0,
	"maxQuantity": 0,
	"creator": {
	"id": 0,
	"name": "string"
	},
	"gmtCreate": "string",
	"gmtModified": "string",
	"modifier": {
	"id": 0,
	"name": "string"
	}
	},
	"quantity": 0,
	"creator": {
	"id": 0,
	"name": "string"
	},
	"gmtCreate": "string",
	"gmtModified": "string",
	"modifier": {
	"id": 0,
	"name": "string"
	}
	}
	}	

顾客查询所有自己的分享记录	{
	"errno": 0,
	"errmsg": "成功",
	"data": {
	"page": 0,
	"pageSize": 0,
	"total": 0,
	"pages": 0,
	"list": [
	{
	"id": "string",
	"sharer": {
	"id": 0,
	"name": "string"
	},
	"onsale": {
	"id": 0,	

	"shop": {
	"id": 0,
	"name": "string"
	},
	"product": {
	"id": 0,
	"name": "string",
	"imageUrl": "string"
	},
	"price": 0,
	"beginTime": "string",
	"endTime": "string",
	"quantity": 0,
	"type": 0,
	"activityId": 0,
	"shareActId": 0,
	"numKey": 0,
	"maxQuantity": 0,
	"creator": {
	"id": 0,
	"name": "string"
	},
	"gmtCreate": "string",
	"gmtModified": "string",
	"modifier": {
	"id": 0,
	"name": "string"
	}
	},
	"quantity": 0,
	"creator": {
	"id": 0,
	"name": "string"
	},
	"gmtCreate": "string",
	"gmtModified": "string",
	"modifier": {
	"id": 0,
	"name": "string"
	}
	}
	]
	}
	}	

查看商品的详细信息（需登录，从分享模式查看商品）	{
	"errno": 0,
	"errmsg": "成功",
	"data": {
	"id": 0,
	"shop": {
	"id": 0,
	"name": "string"
	},
	"goodsId": 0,
	"onSaleId": 0,
	"name": "string",
	"skuSn": "string",
	"imageUrl": "string",
	"originalPrice": 0,
	"weight": 0,
	"price": 0,
	"quantity": 0,
	"state": 0,
	"unit": "string",
	"barCode": "string",
	"originPlace": "string",
	"category": {
	"id": 0,
	"name": "string"
	},
	"shareable": false
	}
	}	

管理员查询商品分享记录	{
	"errno": 0,
	"errmsg": "成功",
	"data": {
	"page": 0,
	"pageSize": 0,
	"total": 0,
	"pages": 0,
	"list": [
	{
	"id": "string",
	"sharer": {
	"id": 0,
	"name": "string"
	},	

	"onsale": {
	"id": 0,
	"shop": {
	"id": 0,
	"name": "string"
	},
	"product": {
	"id": 0,
	"name": "string",
	"imageUrl": "string"
	},
	"price": 0,
	"beginTime": "string",
	"endTime": "string",
	"quantity": 0,
	"type": 0,
	"activityId": 0,
	"shareActId": 0,
	"numKey": 0,
	"maxQuantity": 0,
	"creator": {
	"id": 0,
	"name": "string"
	},
	"gmtCreate": "string",
	"gmtModified": "string",
	"modifier": {
	"id": 0,
	"name": "string"
	}
	},
	"quantity": 0,
	"creator": {
	"id": 0,
	"name": "string"
	},
	"gmtCreate": "string",
	"gmtModified": "string",
	"modifier": {
	"id": 0,
	"name": "string"
	}
	}
	]	

	} }	

分享者查询所有分享成功记录	{
	"errno": 0,
	"errmsg": "成功",
	"data": {
	"page": 0,
	"pageSize": 0,
	"total": 0,
	"pages": 0,
	"list": [
	{
	"id": 0,
	"onsale": {
	"id": 0,
	"shop": {
	"id": 0,
	"name": "string"
	},
	"product": {
	"id": 0,
	"name": "string",
	"imageUrl": "string"
	},
	"price": 0,
	"beginTime": "string",
	"endTime": "string",
	"quantity": 0,
	"type": 0,
	"activityId": 0,
	"shareActId": 0,
	"numKey": 0,
	"maxQuantity": 0,
	"creator": {
	"id": 0,
	"name": "string"
	},
	"gmtCreate": "string",
	"gmtModified": "string",
	"modifier": {
	"id": 0,
	"name": "string"
	}
	},	

	"sharerId": 0,
	"customerId": 0,
	"orderItemId": 0,
	"rebate": 0,
	"creator": {
	"id": 0,
	"name": "string"
	},
	"gmtCreate": "string",
	"gmtModified": "string",
	"modifier": {
	"id": 0,
	"name": "string"
	}
	}
	]
	}
	}	

管理员查询所有分享成功记录	{
	"errno": 0,
	"errmsg": "成功",
	"data": {
	"page": 0,
	"pageSize": 0,
	"total": 0,
	"pages": 0,
	"list": [
	{
	"id": 0,
	"onsale": {
	"id": 0,
	"shop": {
	"id": 0,
	"name": "string"
	},
	"product": {
	"id": 0,
	"name": "string",
	"imageUrl": "string"
	},
	"price": 0,
	"beginTime": "string",
	"endTime": "string",
	"quantity": 0,	

	"type": 0,
	"activityId": 0,
	"shareActId": 0,
	"numKey": 0,
	"maxQuantity": 0,
	"creator": {
	"id": 0,
	"name": "string"
	},
	"gmtCreate": "string",
	"gmtModified": "string",
	"modifier": {
	"id": 0,
	"name": "string"
	}
	},
	"sharerId": 0,
	"customerId": 0,
	"orderItemId": 0,
	"rebate": 0,
	"creator": {
	"id": 0,
	"name": "string"
	},
	"gmtCreate": "string",
	"gmtModified": "string",
	"modifier": {
	"id": 0,
	"name": "string"
	}
	}
	]
	}
	}	

3.4.5 算法

分享模块未使用较为复杂的算法，所有操作仅为简单的增删改查操作

3.4.6 程序逻辑

分享模块未使用较为复杂的算法，因此程序逻辑略

3.4.7 接口

本模块所有功能通过 RESTful API 调用，请求类型 +API 可唯一确定一个接口，所有 API 如下：

功能	类型	API 接口	简单描述
分享者生成分享链接	POST	/onsale/{id}/shares	根据买家 id 和商品 id 生成唯一的分享链接
顾客查询所有自己的分享记录	GET	/shares	
查看商品的详细信息（需登录，从分享模式查看商品）	GET	/shares/{sid}/products/{id}	
管理员查询商品分享记录	GET	/shops/{did}/products/{id}/shares	商铺管理员只能查询本店商铺的商品
分享者查询所有分享成功记录	GET	/beshared	
管理员查询所有分享成功记录	GET	/shops/{did}/products/{id}/beshared	

3.4.8 存储分配

分享模块对于硬盘空间无特殊要求，对于内存空间有以下要求：

要求项	内存要求	说明
JVM 虚拟机	256MB——512MB	可支持 jar 包运行即可
MySQL	1GB	不单独为 Share 模块设立数据库，可与其他模块共用数据库 若有要求，后续可继续添加 MySQL 服务器

3.4.9 限制条件

1. 服务器限制

最多拥有 10 台服务器，其中只有两台服务器内存为 4G，其余服务器内存均为 2G。需要在服务器数量与内存限制下，合理分配模块部署

2. 接口限制

本系统无前端界面，所有请求只可通过手动输入调用接口，只可通过控制台或终端查看输出结果

3. 运行环境限制

系统的部分功能必须借助 Linux 系统运行

4. 网络限制

系统所有模块仅可接收内网（172.16）的请求

3.4.10 测试要点

分享模块的功能测试要点如下：

1. 需求中所有功能测试正常
2. 需满足除 vo 和 po 类以外的所有类的白盒测试代码覆盖率均不小于 70%
3. 测试用例错误不超过 30 个