厦門大學

软件工程导论

详细设计说明书

组　　别　3-4 组

组　　员　黄子安 崔方博 范周喆 侯妤欣 徐森彬

2023　　年　12　月　16　日

1. 引言

1.1 编写目的

OOMALL 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顾客模块和订单模块中的详细设计。总目标为开发一个电子商城系统 OOMALL 的顾客模块和订单模块。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。本小组负责其中订单模块的开发。项目选用 Java 语言编写代码，用到了 Java 项目开发中的常用的技术和框架如，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。

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：3-4 小组
c. 客户端用户：有电商系统使用需求的用户，包括顾客、商户和服务商
d. 后台管理用户：后台管理人员

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
商铺：Shop，是系统的第二主要用户，提供商品的销售等
管理员：Admin，电商平台的管理员，负责日常维护和监控
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
分账：由于中国人民银行 281 号文的要求，支付系统需要基于分账模式完成用户的支付。微信支付基于服务商模式实现支付分账，支付宝系统基于互联网平台直付通实现支付分账

1.4 参考资料

1.需求规格说明书 3-4 小组
2.概要设计说明书 3-4 小组
3.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
4.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

5.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
6.郑志成. 京东到家支付平台的高可用性架构设
计. https://www.zhihu.com/question/527868488/answer/2438919186
7.微信支付. https://pay.weixin.qq.com/wiki/doc/apiv3/apis/chapter8_1_1.shtml
8.支付宝互联网平台直付通产品. https://opendocs.alipay.com/open/02e7gm?ref=api

2. 总体设计

2.1 需求概述

2.1.1 顾客模块需求概述

顾客模块旨在提供用户友好的注册、个人信息管理和优惠券领取等功能，同时为平台管理员提供用户管理工具。

- 用户注册：
  - 新用户注册：提供简便的注册流程，包括基本信息的填写和账户安全设置。
- 顾客管理个人信息：
  - 查看个人信息：用户可随时查看个人信息，包括购物记录、积分等。
  - 修改个人信息：提供修改个人信息的功能，包括联系方式、收货地址等。
  - 修改重置密码：用户可以通过合法途径修改密码，同时提供密码找回功能。
- 顾客与优惠券：
  - 查看领取优惠券：用户可以浏览可用的优惠券，选择领取并在购物时使用。
- 购物车管理：
  - 增删改查购物车：用户可自由管理购物车，包括添加新商品、删除不需要的商品等。
  - 批量下单：用户可以方便地批量下单购物车中的商品，提高购物效率。
- 平台管理员修改查看和管理用户：
  - 查看用户信息：管理员可以查看用户基本信息、交易记录等，以便提供更好的客户服务。
  - 修改用户信息：在必要时，管理员可以协助用户修改个人信息，确保数据准确性。

2.2.2 订单模块需求概述

订单模块负责管理订单的创建、支付、取消、修改和查询等全流程，同时提供商户查看和管理店铺订单的功能。

- 创建订单：
  - 选择商品创建订单：用户可从商品列表中选择所需商品，创建订单并选择支付方式。
  - 订单备注：用户可以添加订单备注，如特殊需求、送货时间等信息。

- 支付订单：
  - 顾客支付：系统应当支持多种支付方式，确保支付的安全和便捷。
  - 订单支付状态：实时更新支付状态，及时通知用户支付结果。
- 顾客取消订单：
  - 顾客取消：提供用户友好的取消订单流程，同时及时更新库存信息。
  - 取消原因反馈：用户取消订单时，系统应当提供反馈渠道，了解取消原因。
- 修改订单：
  - 修改订单信息：用户可以在订单未发货状态下修改收货地址等信息。
  - 订单状态变更：系统应当及时反馈订单状态的变更，确保用户和商户都能了解订单的实时状态。
- 查询订单：
  - 查询订单信息：提供用户方便的查询订单功能，包括按时间范围、订单状态等条件检索。
  - 订单详情：用户可以查看订单详细信息，包括商品清单、配送信息等。
- 商户查看店里所有订单：
  - 订单概览：商户可以查看店铺内所有订单的概览，了解销售情况。
  - 订单明细：商户可以查看每个订单的详细信息，包括顾客信息、商品清单等。
- 商户取消订单：
  - 商户取消：商户可在订单未发货状态下取消订单，提供操作权限和相应反馈。
- 商家接受订单：
  - 接受订单操作：商户可以接受顾客提交的订单，确保订单正常处理流程。
  - 提供反馈：系统应当提供商户对订单接受情况的反馈，确保订单信息同步。

2.2 软件结构

OOMALL 各模块之间的调用关系如图所示：

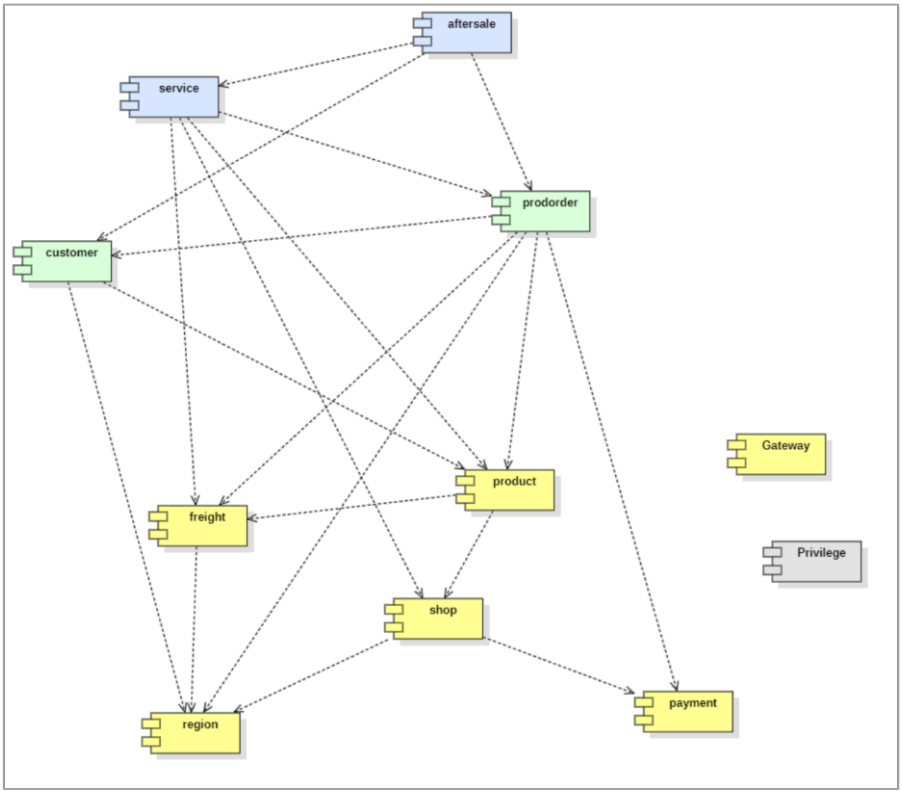


图 1 OOMALL 各模块的调用关系

为了**减少模块之间的耦合关系**，系统按照各个模块之间的因果关系建立了如下构建图，规定依赖箭头尾部的模块可以直接调用箭头头部的模块，但是不可以在反过来的情况下进行直接调用，从而避免模块之间出现双向依赖，降低模块与模块之间的耦合性

对于模块之间可以直接调用的方向，**调用模块**使用 Spring Cloud 提供的 OpenFeign 技术通过 HTTP 请求向**被调用模块**发起请求，完成相关的业务功能。

对于模块之间无法直接调用的方向，项目使用了**消息驱动的软件体系结构**，各个模块通过发送和接收消息进行异步通信，在具体技术上使用了 RocketMQ 服务器实现消息的发送和订阅接收。

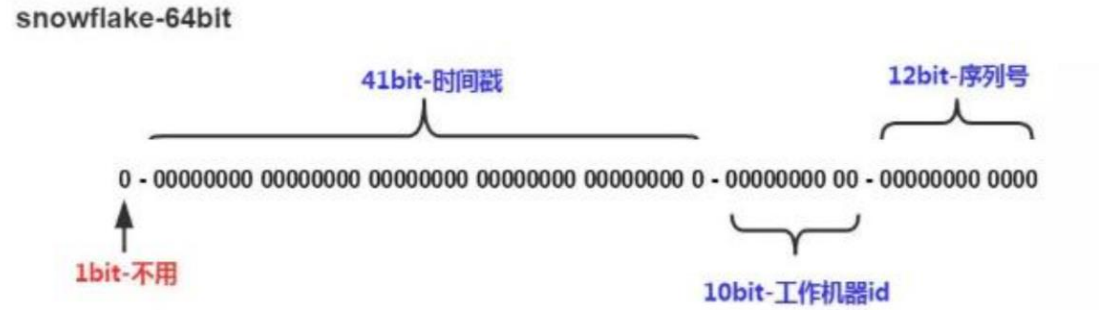
对于我们大作业所需要的订单模块可以调用物流模块、产品模块、地区模块和支付模块；而顾客模块可以直接调用产品模块和地区模块，即通过 openfeign 的发送 HTTP 请求来获取对应的信息和业务处理；而对于支付模块无法直接调用订单模块，因此在支付单完成后通知订单模块修改状态的时候支付模块无法直接调用订单模块，而是需要通过消息队列的方式发送支付单消息，由订单模块订阅该消息并进行处理。

2.3 工具类

2.3.1 雪花算法

程序中多个模块会使用到雪花算法，包括生成订单的序列号、优惠券的序列号等，因此在这里先提前说明程序中所使用的雪花算法：

SnowFlake 是 Twitter 公司开发的一种算法，目的是在分布式系统中产生全局唯一且趋势递增的 ID。



雪花算法生成的序列号组成部分（64bit）如下

1.第一位 占用 1bit，其值始终是 0，没有实际作用。

2.时间戳 占用 41bit，精确到毫秒，总共可以容纳约 69 年的时间。

3.工作机器 id 占用 10bit，其中高位 5bit 是数据中心 ID，低位 5bit 是工作节点 ID，做多可以容纳 1024 个节点。

4.序列号 占用 12bit，每个节点每毫秒 0 开始不断累加，最多可以累加到 4095，一共可以产生 4096 个 ID。

SnowFlake 算法在同一毫秒内最多可以生成 **1024 X 4096 = 4194304** 个全局唯一 ID

但是 twitter 公司原版的雪花算法存在时钟回拨的问题，即当系统时钟因某些因素回滚时，产生的 id 就有可能重复。原版对于这个问题的处理是直接抛异常，这样会导致 id 生成失败。其实这个问题可以通过对时间戳生成器加锁来解决。我们使用的是改进后的雪花算法，修复了时钟回拨的问题。

3．程序描述

3.1 顾客模块

3.1.1 功能

1. 注册：提供注册表单，包括用户名、密码、手机号等信息。执行输入验证，确保用户名的唯一性，密码的强度，手机号的有效性等。存储用户信息（用户名、密码、手机号等）到数据库。
2. 登录：提供登录表单，要求用户输入用户名和密码。验证用户提供的凭据是否匹配数据库中的记录。如果验证成功，创建会话并将用户信息存储在会话中。
3. 登出：销毁用户会话，清除相应的会话信息。重定向用户到登出成功页面或登录页面。
4. 顾客查看自己的信息：提供用户信息页面，显示用户的个人信息如用户名、手机号、地址等。
5. 顾客修改信息：提供用户信息编辑页面，允许用户修改个人信息。执行输入验证，确保修改后的信息的有效性。更新数据库中的用户信息。
6. 顾客修改密码：提供修改密码表单，要求用户输入当前密码和新密码。验证当前密码是否正确，新密码的强度是否符合规定。更新数据库中的密码信息。
7. 重置密码：提供重置密码功能，通过电子邮件或手机验证码验证用户身份。允许用户输入新密码，并进行密码重置。
8. 平台管理员获取用户列表：提供管理员页面，显示所有注册用户的列表。提供分页和搜索功能，以便管理员能够轻松找到特定用户。
9. 平台管理员查看顾客信息：提供管理员页面，允许管理员通过用户名或其他唯一标识查看特定顾客的详细信息。
10. 平台管理员删除顾客：在管理员界面提供删除用户的选项。执行删除操作前，确认管理员的身份验证。从数据库中删除用户的相关信息。
11. 平台管理员封禁买家：提供管理员界面，允许管理员选择要封禁的用户。封禁操作前，确认管理员的身份验证。更新数据库中用户的状态，使其被封禁。
12. 平台管理员解禁买家：提供管理员页面，显示被封禁用户列表。允许管理员选择解禁的用户，并进行解禁操作。更新数据库中用户的状态，使其解禁。
13. 顾客获取购物车列表：提供购物车页面，显示用户已选择的商品列表。包括商品名称、价格、数量等信息。
14. 顾客将商品加入购物车：在商品页面提供“加入购物车”按钮。允许用户选择商品数量和规格。将商品添加到购物车，并更新购物车数量。
15. 顾客清空购物车：在购物车页面提供“清空购物车”按钮。确认用户的操作，清除购物车中所有商品。
16. 顾客删除购物车中的商品：在购物车页面提供删除按钮，允许用户删除特定商品。更新购物车中的商品列表。
17. 顾客修改购物车中单个商品的数量或规格：在购物车页面提供编辑按钮，允许用户修改商品数量和规格。执行输入验证，确保修改后的数量和规格的有效性。更新购物车中的商品信息。

18. 顾客新增收货地址：在用户地址页面提供“新增地址”选项。提供表单，允许用户输入新的收货地址信息。执行输入验证，确保地址信息的有效性。将新地址添加到用户的地址列表中。
19. 顾客查询所有已有的地址信息：在用户地址页面显示所有已有的收货地址列表。包括地址名称、收货人姓名、联系电话等信息。
20. 顾客设置默认地址：允许用户在地址列表中选择默认地址。更新数据库中用户的默认地址信息。
21. 顾客修改自己的地址信息：在用户地址页面提供编辑按钮，允许用户修改地址信息。执行输入验证，确保修改后的地址信息的有效性。更新数据库中的地址信息。
22. 顾客删除地址：在用户地址页面提供删除按钮，允许用户删除特定的收货地址。确认用户的操作，从数据库中删除地址信息。
23. 顾客查看优惠券列表：在用户页面提供“优惠券”选项，显示用户可用的优惠券列表。包括优惠券名称、折扣金额、使用条件等信息。

3.1.2 性能

系统要能满足万人同时在线，且 2000 人同时访问一个功能时没有明显卡顿，即：

在并发量不超过 2000 transaction/seconds 的情况下，顾客模块的每个功能的平均响应时间应在 100ms 左右，响应时间最高不超过 300ms。

3.1.3 输入项目

功能	用户输入信息
用户注册	用户名、密码、姓名、用户手机号
买家查看自己信息	无
买家修改自己的信息	用户昵称、手机号
用户修改密码	验证码、新密码
用户重置密码	用户名、手机号
平台管理员获取所有用户列表	用户名、姓名、手机号
管理员查看任意买家信息	顾客 id
管理员逻辑删除顾客	顾客 id
平台管理员封禁买家	顾客 id
平台管理员解禁买家	顾客 id
用户名密码登录	用户名、密码
用户登出	无
买家获得购物车列表	无
买家将商品加入购物车	商品 id、数量
买家清空购物车	无
买家修改购物车单个商品的数量或规格	商品 id、数量
买家删除购物车中商品	购物车 id
买家新增地址	地区 id、地址、收件人姓名、手机号
买家查询所有已有的地址信息	无

买家设置默认地址	地址 id
买家修改自己的地址信息	原地区 id、新地区 id、地址、收货人姓名、手机号
买家删除地址	地区 id
顾客查看优惠券列表	待查看的优惠券类型
买家领取活动优惠券	活动 id

3.1.4 输出项目

功能	输出信息	异常输出
用户注册	用户名、用户姓名、手机号	611：用户名已被注册 616：电话号码已被注册
买家查看自己信息	用户名、姓名、手机号	
买家修改自己的信息	系统提示操作成功	613：电话号码已被注册
用户修改密码	系统提示操作成功	612：不能与旧密码相同
用户重置密码	系统提示操作成功	614：与系统预留的电话不一致
平台管理员获取所有用户列表	顾客 id 和姓名	
管理员查看任意买家信息	顾客 id、用户名、姓名、是否有效	
管理员逻辑删除顾客	系统提示操作成功	
平台管理员封禁买家	系统提示操作成功	
平台管理员解禁买家	系统提示操作成功	
用户名密码登录	Token	609：用户名不存在或者密码错误 610：用户被禁止登录
用户登出	系统提示操作成功	
买家获得购物车列表	商品 id、商品名、商品价格、收藏数量、商品状态、购物车总数、总价格、活动 id、活动类型、活动名称	
买家将商品加入购物车	商品 id、商品名、商品价格、收藏数量、商品状态、购物车总数、总价格、活动 id、活动类型、活动名称	
买家清空购物车	系统提示操作成功	
买家修改购物车单个商品的数量或规格	系统提示操作成功	
买家删除购物车中商品	系统提示操作成功	
买家新增地址	地区 id、地区名、地址、收件人名字、收件人手机	601：达到地址簿上限

	号、是否是默认地址	
买家查询所有已有的地址信息	地区 id、地区名、地址、收件人名字、收件人手机号、是否是默认地址	
买家设置默认地址	系统提示操作成功	
买家修改自己的地址信息	系统提示操作成功	
买家删除地址	系统提示操作成功	
顾客查看优惠券列表	优惠券 id、优惠券活动、优惠券数量、优惠券有效时间	
买家领取活动优惠券	系统提示操作成功	696：未到优惠卷领取时间 697：优惠卷领罄 698：优惠卷活动终止 699：不可重复领优惠卷

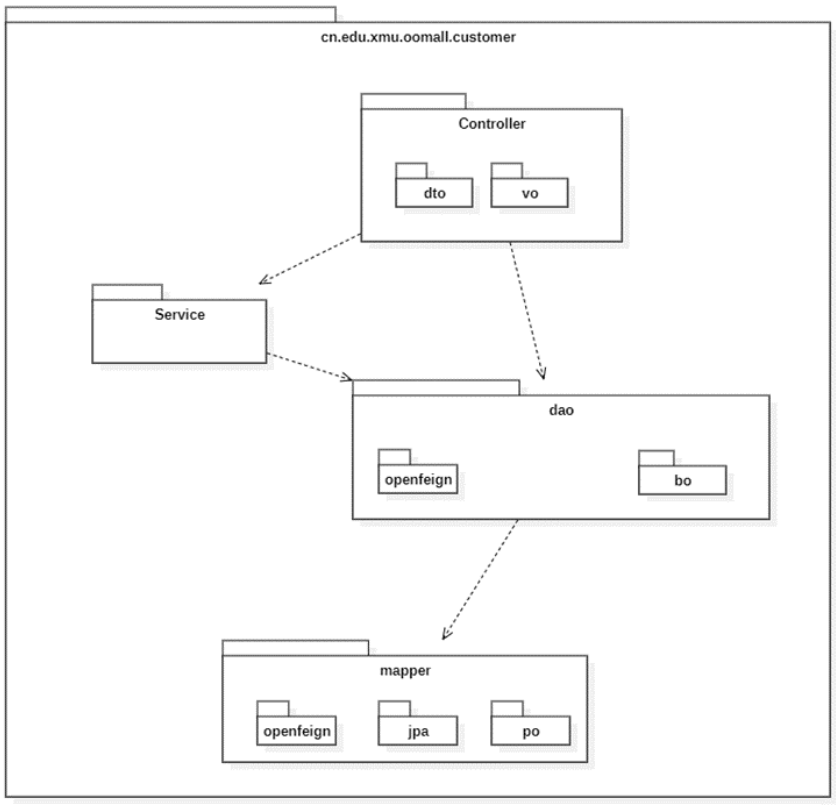
3.1.5 算法

1. 领取优惠券算法：在本商务系统中优惠券可以分为两类，一种是总数限量，一种是个人限量，因此在顾客领取优惠券时，需要先查看顾客的优惠券列表，查看顾客已经领取该优惠券。如果未领取则领取优惠券，同时判断活动类型，如果优惠券是定量的，需扣除平台该优惠券的总量。

2. 生成顾客号和优惠券序列号需要用到雪花算法。雪花算法详情见 2.3 工具类。

3.1.6 程序逻辑

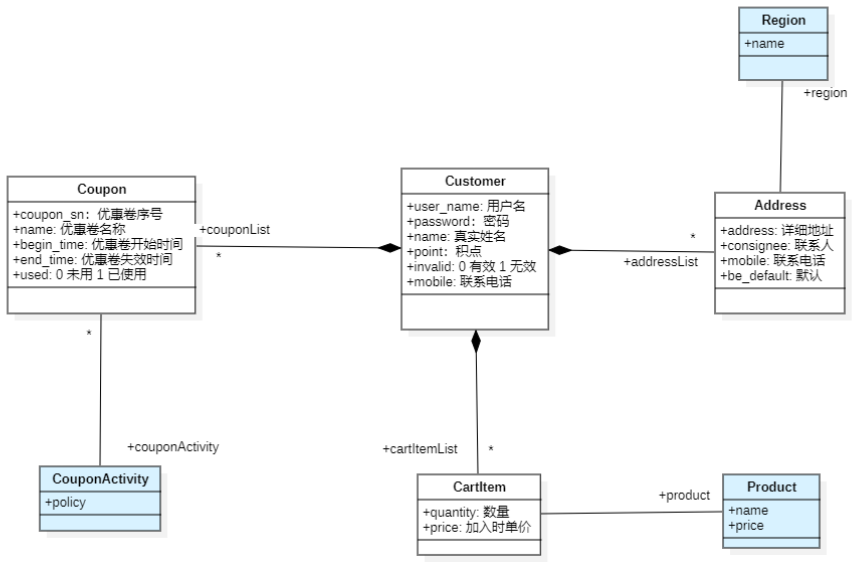
3.1.6.1 包图



项目采用了六边形体系结构和层次体系结构

- Mapper 和 Controller 在六边形体系结构的最外层，用来和系统外界交互。Controller 层使用 vo 对象接收来自前端的数据，实现将 json 转换成 vo 对象，之后将其转换成 bo 传给 Service 层；Mapper 层使用 openfeign 和其他模块交互，使用 jpa 对数据库进行修改，po 对象用来接收数据库返回的信息，与关系数据库中的对象建立映射关系。
- Service 层完成业务的实现，通过协调业务对象 bo 和调用数据访问对象 Dao 的方式完成业务处理
- Dao层主要职责是完成数据的增删改查，实现bo和po的转换，此外在使用openfeigh技术和其他模块交互时也会将 bo 转换为对应模块请求所需的格式；Dao 包中还有 bo 作为业务对象，根据领域驱动设计中的满血模型概念会执行职责，实现数据本身和对应操作在同个对象中，提高了代码的内聚程度。

3.1.6.2 对象模型



对象	含义
Customer	顾客信息
Coupon	优惠券信息
Address	顾客的信息
CartItem	购物车商品信息
CouponActivity	外部商品模块优惠券活动信息
Region	外部地区模块的地区信息
Product	外部商品模块商品信息

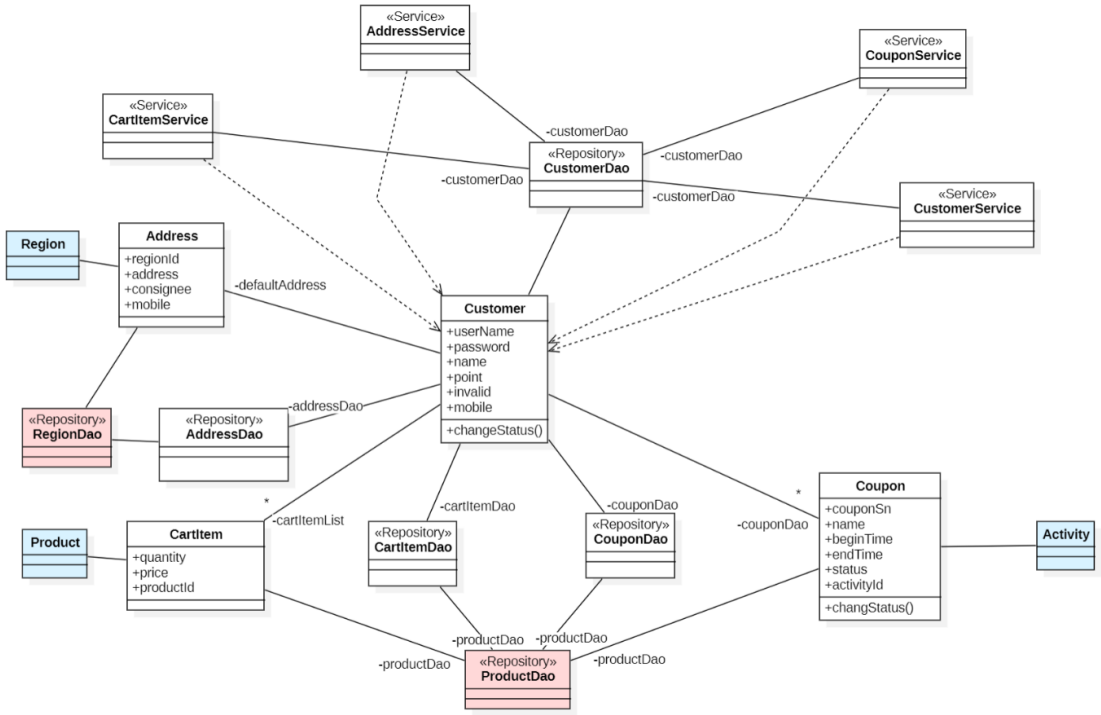
1、 API 以顾客为中心，在调用的时候通过顾客类获得其余类的信息，因此对象模型中体现类与类之间关联关系的箭头方向指向顾客

2、 顾客和地址是一对多的关系，因为一个顾客可以同时管理多个收货地址

3、 优惠券指的是被顾客领取的优惠券，如果指活动发放的优惠券，会导致出现大量的存储浪费，所以将其设计成只有优惠券被领取了才会成为实体类，同理优惠券的状态机是从待使用开始而不是从待领取开始

4、 顾客和优惠券、地址、购物车明细都是组合关系，它们符合组合关系的要求，是构成了整体和局部的关系，局部对象（优惠券、购物车明细、地址）必须属于且只属于一个母体对象（顾客），母体对象负责创建和销毁局部对象

3.1.6.3 类图

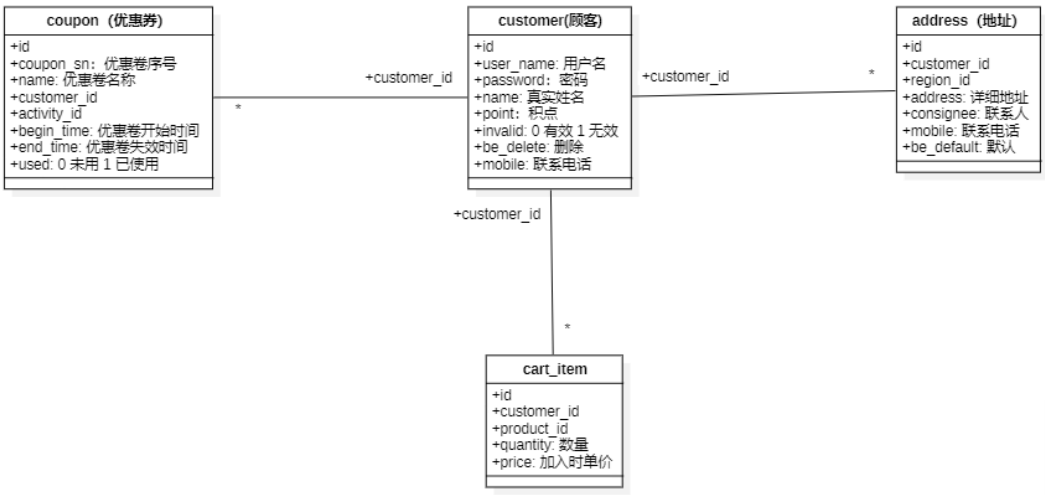


红色的 Dao 类主要是用于调用其他模块，在其中完成请求数据的组装和响应数据的解析，因此为了区分将其设置成红色。

蓝色的类表示这些对象位于外部模块，在本模块不会对其进行存储操作，由外部模块维护这些对象，本模块使用这些对象的时候会通过 openfeign 请求的方式获取对应的信息

因为项目基于 DDD（领域驱动设计）的充血模型思想，即对于业务对象 Bo 会根据职责分配的原则为其提供的业务处理能力，实现系统中对象数据和操作的内聚，因此需要建立 Bo 对象和 Dao 对象的关联关系，这里因为在对象模型上出现大量的组合关系，因此会频繁使用 GRASP（职责分配原则）中的创建者原则由顾客对象创建对应的购物车明细、优惠券和收货地址对象。

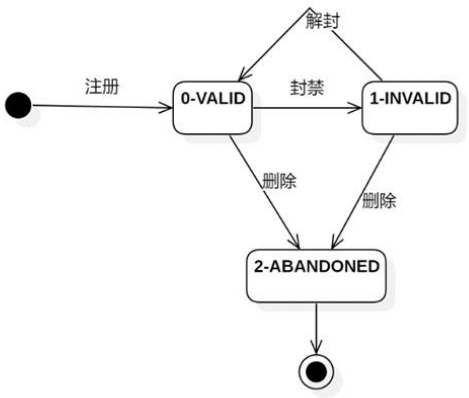
3.1.6.4 数据库



数据库模型和对象模型中顾客和其余部分方向出现了相反，没有从 customer 指向边上的三个类，因为如果 customer 对于其他对象是一对多的关系，在数据库中需要第三张表进行存储连接，为了节约存储的开销，将它们转换为多对一的关系，将 customer_id 作为字段放在边上的三个数据表里，从而不用额外增加一张表，数据库模型和对象模型中这里出现的反向问题需要在代码中进行处理

3.1.6.5 状态机图

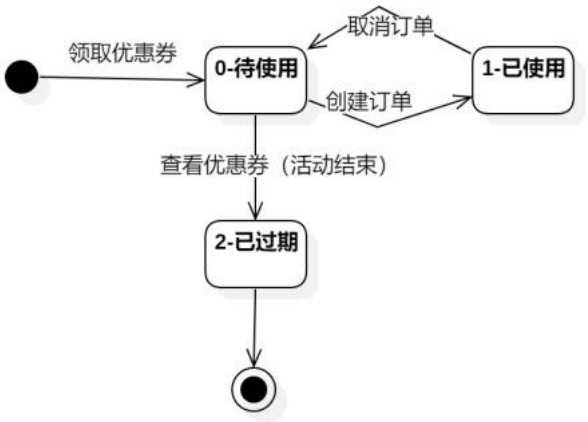
顾客状态机图



顾客有三种状态，有效（VALID)、已封禁（INVALID)、已删除（ABANDONED）
顾客状态类型的设定主要是考虑到这三个状态将会影响顾客的登录操作：

- 顾客的状态决定了顾客是否可以进行登录操作，处于有效状态（VALID）的顾客可以正常登录并且进行登录后的操作。
- 处于已封禁的顾客（INVALID）无法进行登陆操作，只有管理员将顾客解封后才能回到有效状态进行登录。
- 处于已删除状态的顾客将被永久删除，无法登录。

优惠券状态机图



优惠券有待使用、已使用、已过期三种状态。

- 优惠券的起始状态是待使用状态

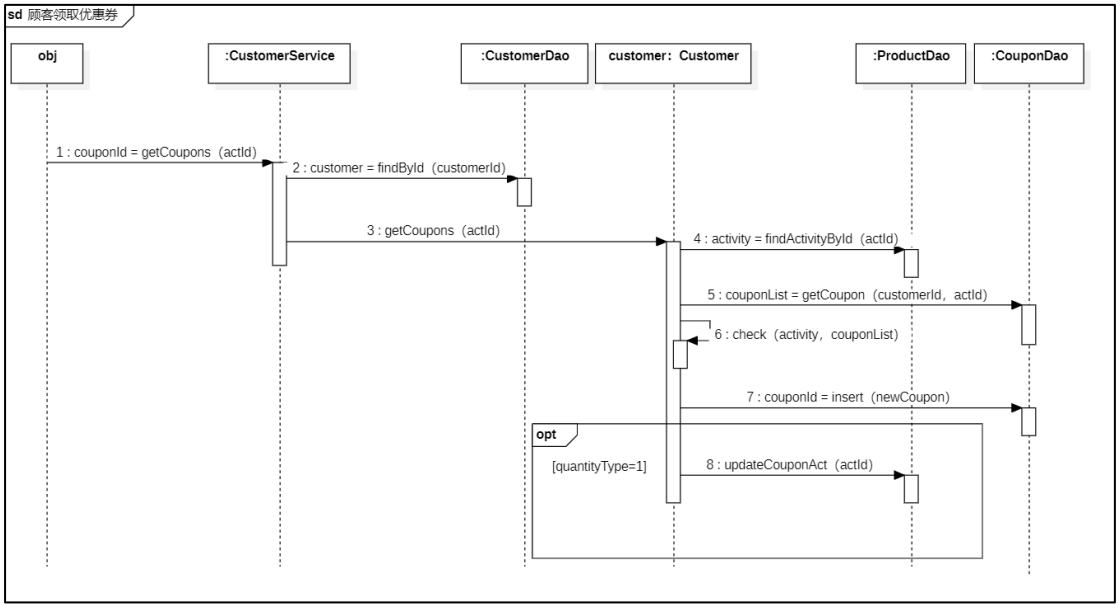
原因：如果从优惠券创建（即活动的创建）开始，系统将要保存巨量的优惠券数据。而事实上，真正被用户领取的优惠券数据远小于此，将初始状态设成用户已经领取的、待使用的状态，可以减小系统负载。

- 取消订单有两个 API，因为顾客和商户都能进行取消订单的操作。
- 不论优惠券是否被使用，其最终状态均是已过期。因为已经使用的优惠券有可能因为取消订单而回到待使用状态，于是，已使用不能作为最终状态。
- 优惠券是否过期的相关状态转移 API 是查看优惠券，因为在平时不会去关心优惠券是否还有效，只有在使用或者想使用的时候会关心是否过期，因此在查看优惠券 API 中检验是否过期即可，可以减小系统的负担

3.1.6.6 时序图

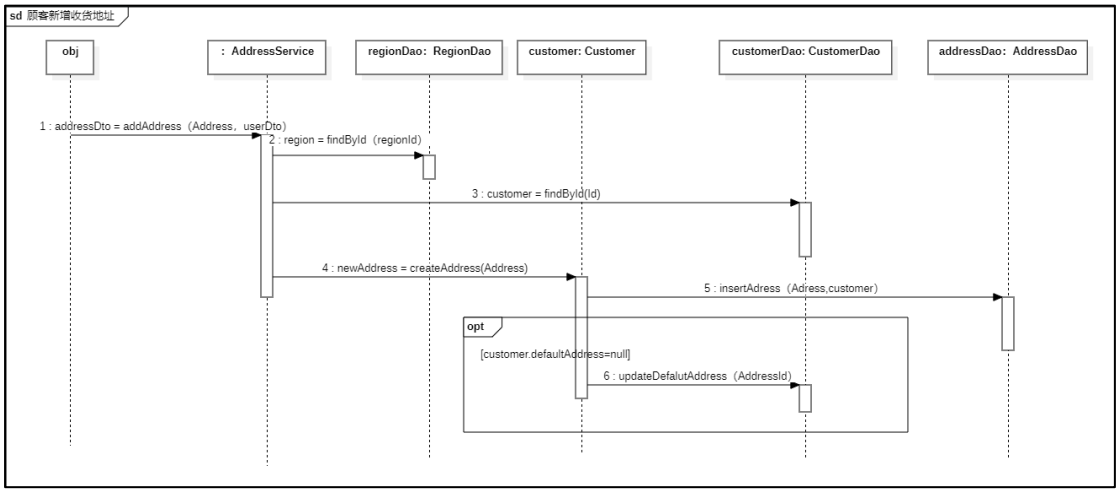
顾客模块我们选取了一部分的 API 进行了时序图的设计与分析：

顾客领取优惠券



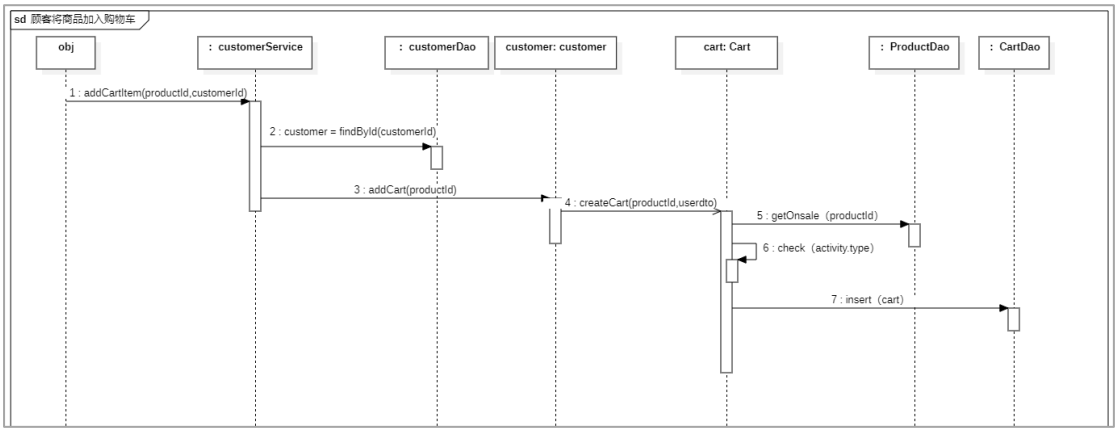
对于顾客领取优惠券，首先调用 customerService 的 getCoupons 方法，CustomerService 层调用 customerDao 层的 findById 方法，解析 token 中的用户 id，将 id 变对象 customer，接着 CustomerService 层调用 bo 对象（customer）的 getCoupons 方法，将职责分配给 customer 对象。Customer 对象首先通过传进来的 actid 找到对应的活动，再查找自己拥有的优惠券，判断是否已经领取。如果未领取则将优惠券插入数据库，表示已领取。同时判断活动类型是否是定量优惠券活动，如果是则扣掉平台的优惠券数量。

顾客新增收货地址



AddressService 通过传进来的 regionid 创造 region 对象，再通过 customerid 创造 customer 对象。将创建新收货地址的职责分给 customer 对象，即 AddressService 调用 customer 的 createAddress 方法。Customer 调用 AddressDao 执行插入新地区的操作。再判断当前 customer 是否有默认地址，如果没有则修改 customer 对象的默认地址属性，将该新地址设置为默认地址。

顾客将商品加入购物车



CustomerService 调用 customerDao 方法将 customerid 变成 customer 对象，接着将加入购物车的职责分给 customer 对象。Customer 对象调用 cart 对象的 createcar 方法，将职责继续分给 cart 对象。Cart 对象调用 productDao，将 productid 变成 onsale 对象，以此查看该商品目前所属的优惠活动。调用 carDao 将该商品加入购物车。

3.1.7 接口

OOMALL | 1.3.2 | mingqcn | SwaggerHub
顾客模块接口描述、请求方式、请求路径如下：


POST /customers 注册用户
GET /customers 买家查看自己信息
PUT /customers 买家修改自己的信息
PUT /customers/password 用户修改密码
PUT /customers/password/reset 用户重置密码
GET /shops/{shopId}/customers 平台管理员获取所有用户列表
POST /customers/login 用户名密码登录
GET /customers/logout 用户登出
GET /shops/{shopId}/customers/{id} 管理员查看任意买家信息
DELETE /shops/{shopId}/customers/{id} 管理员查看逻辑删除顾客
PUT /shops/{did}/customers/{id}/ban 平台管理员封禁买家
PUT /shops/{did}/customers/{id}/release 平台管理员解禁买家
GET /carts 买家获得购物车列表

3.1.8 存储分配

使用 MySQL 数据库进行存储，存储的表格信息如下：
customer_address：存储顾客添加的所有的地址信息
customer_cart：存储购物车里的商品信息
customer_coupon：存储优惠活动的信息
customer_customer：存储顾客的基本信息
在我们的 docker swarm 集群中，mysql 只部署到一个服务节点上面

3.1.9 限制条件

需要事先部署运行项目的服务器群上建立 docker swarm 集群，要按照项目的部署说明在华为云的 7 台云耀云服务器上部署好相应数量和配置的 nacos，mongoDB，mysql，redis，rocketmq 的 docker 容器。其中 mysql 服务器部署了一台，mongo 服务器部署了三台用来支持事务(主要用到其中一台)，nacos 部署了一台，rocketMQ 部署了 nameserver 和 broker，因受到物理机内存、CPU、IO 读写速率、网络传输效率的限制，系统的性能存在一定的瓶颈。有一些 API 需要用户登录之后才能完成相应的操作，在顾客模块中，完成具体的功能需要有对应的权限来完成。

3.1.10 测试要点

1. 登录状态检查
测试要点：确保顾客在进行任何非浏览商品的操作之前都需要先检查登录状态。
验证方法：尝试执行需要登录状态的操作，如修改个人信息、添加到购物车等，确保系统正确地要求用户登录。

2. 注册信息合法性检查
测试要点：验证顾客注册时输入的用户名、密码、手机号等信息的合法性。
验证方法：尝试使用合法和非法的用户名、密码、手机号进行注册，确保系统对于不合法输入给出正确的反馈，并对合法输入进行成功注册。

3. 输入参数合法性检查
测试要点：确保各个功能接口对输入参数进行了正确的合法性检查。
验证方法：提供不同的输入参数，包括边界值测试、非法字符测试等，确保系统在接收到不合法输入时有适当的处理和反馈。

4. 查询操作结果验证
测试要点：查询顾客信息、购物车信息等，确保返回的结果正确。
验证方法：执行查询操作，检查返回结果与期望结果是否一致，包括顾客信息的准确性和购物车商品的正确展示。

5. 写入操作结果验证
测试要点：修改顾客信息、地址、新增购物车、领取优惠券等写入操作是否成功。
验证方法：执行写入操作，检查数据库中的数据是否被正确修改，确保写入操作的结果符合预期。

6. 优惠券领取时间检查
测试要点：验证优惠券是否在上线后才能被顾客领取。
验证方法：尝试在上线前和上线后领取优惠券，确保系统在规定时间内限制了优惠券的领取

对于**单元测试**使用白盒测试方法，基本路径测试和测试代码的覆盖率可以确保在代码级别对以上功能进行测试。
对于**集成测试**使用黑盒测试方法，基于等价类划分、边界值还有异常值推断来验证系统整体功能的正确性。
对于**性能测试**使用 Jmeter 在短时间内大量生成优惠券的请求，从而验证在高并发领取优惠券场景下系统是否可以处理海量请求，且保证优惠券不会超领。

3.2 订单模块

3.2.1 功能

订单模块需要实现的功能有：

1. 获得订单的所有状态：提供一个接口，允许用户通过订单号或其他标识符获得订单的当前状态。状态包括已下单、已付款、已发货、已完成。
2. 买家查询名下订单：创建一个查询接口，允许买家通过其用户 ID 或其他身份标识符检索与其账户相关的所有订单。返回订单的基本信息，如订单号、状态、商品信息等。
3. 买家申请建立订：提供一个下单接口，允许买家选择订单类型（普通、团购、预售），并提供相应的商品信息、数量等。确保订单信息的完整性和准确性。
4. 买家查询订单完整信息：提供一个详细查询接口，允许买家获取订单的所有相关信息，包括商品详情、价格、数量、支付状态等。
5. 买家修改本人名下订单：提供修改订单的接口，允许买家修改订单中的商品数量、地址等信息。确保在订单尚未处理或发货时才能进行修改。
6. 买家取消本人名下订单：提供取消订单的接口，允许买家取消尚未处理或发货的订单。发货后可能需要提供退货流程。
7. 买家支付订单：提供支付接口，支持不同支付方式（支付宝、微信支付）。确保支付流程安全可靠，提供支付成功与失败的反馈信息。
8. 店家查询商户所有订单：创建一个接口，允许店家获取其商户下的所有订单摘要信息，包括订单号、总金额、状态等。
9. 店家修改订单：提供修改订单的接口，通常限于留言、备注等非关键信息的修改。可以用于与买家进行沟通或记录特殊处理。
10. 店家查询店内订单完整信息：提供详细查询接口，允许店家获取与其店铺相关的订单的所有详细信息。
11. 管理员取消本店铺订单：提供管理员级别的接口，用于取消特定店铺的订单，可能需要特殊权限验证。
12. 店家接受订单：提供接受订单的接口，表示商家已经确认并接受了订单。
13. 店家发出包裹：提供发货接口，用于商家标记订单为已发货，并提供相关的物流信息。

3.2.2 性能

系统要能满足万人同时在线，且 2000 人同时访问一个功能时没有明显卡顿，即：

在并发量不超过 2000 transaction/seconds 的情况下，订单模块的每个功能的平均响应时间应在 100ms 左右，响应时间最高不超过 300ms。

3.2.3 输入项目

功能	用户输入信息
获得订单的所有状态	无
买家查询名下订单	订单编号、订单状态、开始时间、结束时间

买家申请建立订单	订单子项 id、订单子项数量、使用的优惠券 id、使用的积点、订单活动 id、收货人姓名、收货人手机号、收货地址 id、收货地址、备注
买家查询订单完整信息	订单 id
买家修改本人名下订单	订单 id、收货人姓名、地区 id、收货地址、收货人手机号
买家取消本人名下订单	订单 id
买家支付订单	订单 id、支付渠道
店家查询商户所有订单	商户 id、查询的购买者用户 id、订单 Sn、开始时间、结束时间
店家修改订单	商户 id、订单号、留言
店家查询店内订单完整信息	商户 id、订单 id
管理员取消本店铺订单	商户 id、订单 id
店家接受订单	商户 id、订单 id
店家发出包裹	商户 id、订单 id、物流号

3.2.4 输出项目

功能	正常输出	异常输出
获得订单的所有状态	订单状态	
买家查询名下订单	订单 id、订单状态、创建人、原价、折扣价、物流单号	
买家申请建立订单	订单子项 id、订单子项数量、使用的优惠券 id、使用的积点、订单活动 id、收货人姓名、收货人手机号、收货地址 id、收货地址、备注	296：货品（id = %d）库存不足 802：销售对象(id=%d)的数量(%d)超过单次可购买数量(%d)
买家查询订单完整信息	订单 id、订单号、顾客 id、顾客名、商户 id、商户名、订单状态、原价、折扣价、收货信息、订单子项信息（商品 id、订单 id、商品名、商品数量、折扣价、活动 id、活动类型）、优惠券（优惠券 id、优惠活动 id、优惠活动名、优惠券数量、优惠活动时间）	
买家修改本人名下订单	系统提示操作成功	
买家取消本人名下订单	系统提示操作成功	
买家支付订单	系统提示操作成功	
店家查询商户所有订单	订单 id、订单状态、创建人、原价、折扣价、物流单号	
店家修改订单	系统提示操作成功	

店家查询店内订单完整信息	订单 id、订单号、顾客 id、顾客名、商户 id、商户名、订单状态、原价、折扣价、收货信息、订单子项信息（商品 id、订单 id、商品名、商品数量、折扣价、活动 id、活动类型）、优惠券（优惠券 id、优惠活动 id、优惠活动名、优惠券数量、优惠活动时间）	
管理员取消本店铺订单	系统提示操作成功	
店家接受订单	系统提示操作成功	
店家发出包裹	系统提示操作成功	

3.2.5 算法

1. 拆分订单算法

在划分订单时，前端将根据商铺 id 分类传给后端，分包算法先根据订单各个订单子项的活动类型进行分类。在已经按照活动类型分类后，还需要继续根据不同活动类型执行不同操作，具体的进一步分包逻辑如下：

- 对于预售订单子项，每个订单子项分到一个新订单中
- 对于团购订单子项，相同活动的团购活动分到一个新订单中
- 对于普通订单子项和非预售团购活动订单子项，无需进行更进一步的细分

在分完订单之后，调用计算运费算法计算运费。

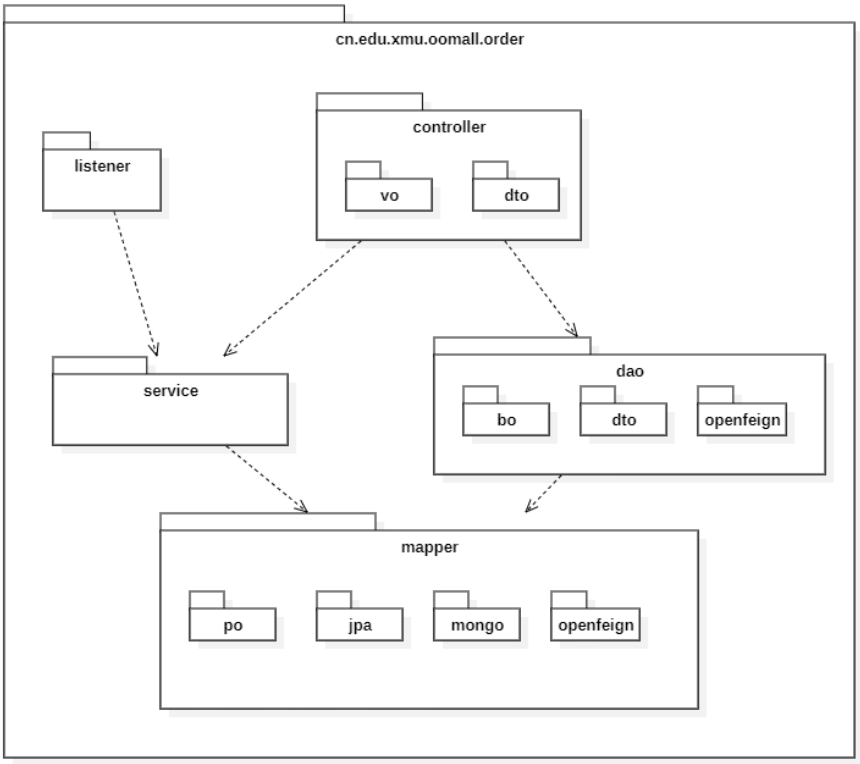
2. 分包算法

在订单已经根据商铺 id 和活动类型分类完后，需要对订单进行拆分包裹的算法，因为一个订单可以有多个包裹，每个包裹可以有多个订单子项，一个订单子项也可以由于数量过多而被分到不同的包裹中。分包需要根据商品的数量、商品的物流渠道 id 等进行进一步的分包。本项目用到的分包算法有背包算法、贪心算法等多种算法，以保证分包后的运费最少，减轻平台交易中的物流成本。

2. 生成订单序列号需要用到雪花算法，雪花算法详情见 2.3 工具类。

3.2.6 程序逻辑

3.2.6.1 包图

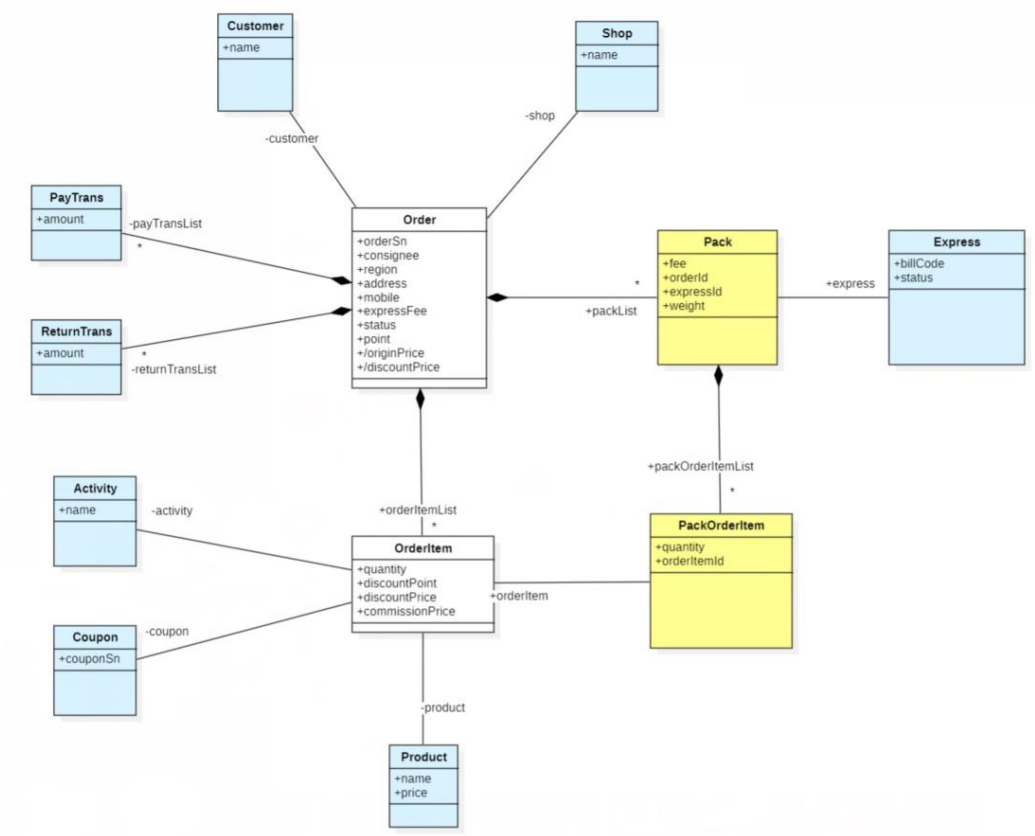


包图其余部分和顾客模块相同，这里对顾客模块没有的部分进行说明

Listerner：支付订单需要使用 RocketMQ，所以需要对应的包来存放相关类，另外 listerner 位于控制器层所在包的同一高度，因为根据六边形体系结构与外部系统交互的消息队列服务器应该位于系统的最外层，这里不是 Mapper 层因为不能向上传递订阅的消息，因此只能放在控制器；另外 RocketMQ 实际上调用了 Service 的方法，从功能业务看也应该位于控制器层包对应的高度上

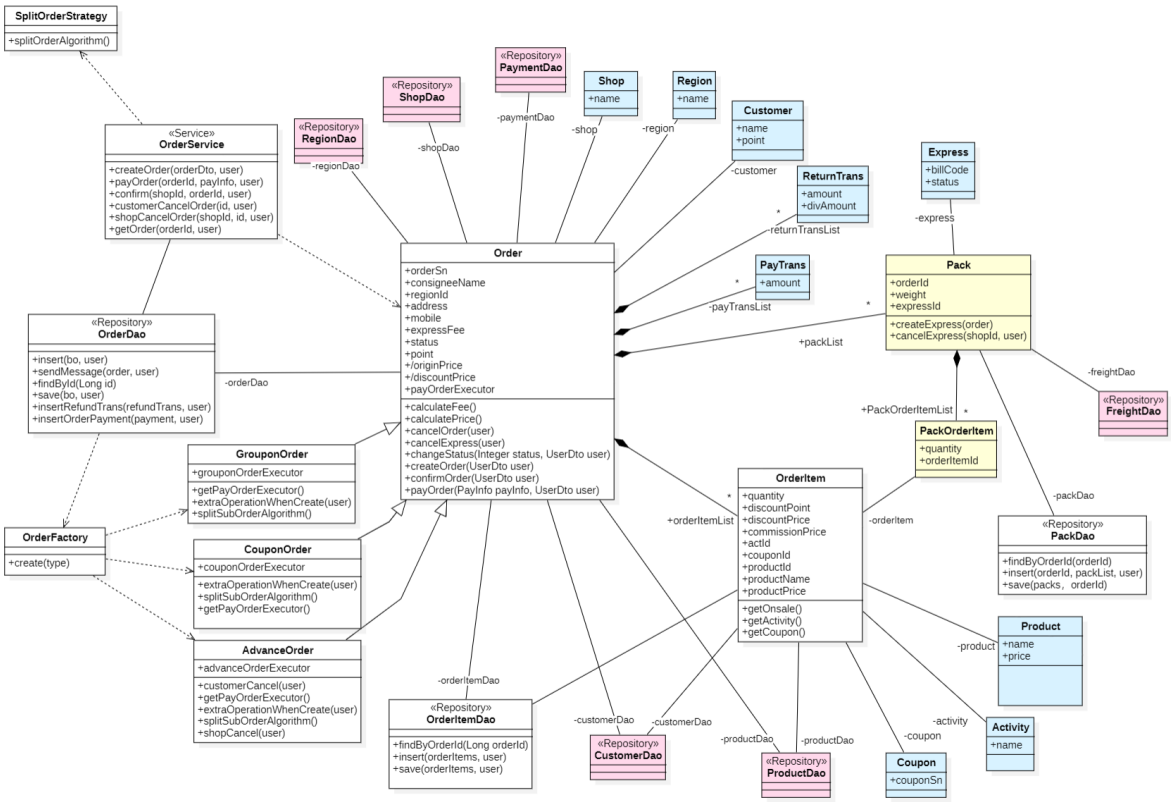
Config：该模块使用了 RocketMQ 和 Mongo，所以需要存放对应的配置类

3.2.6.2 对象模型



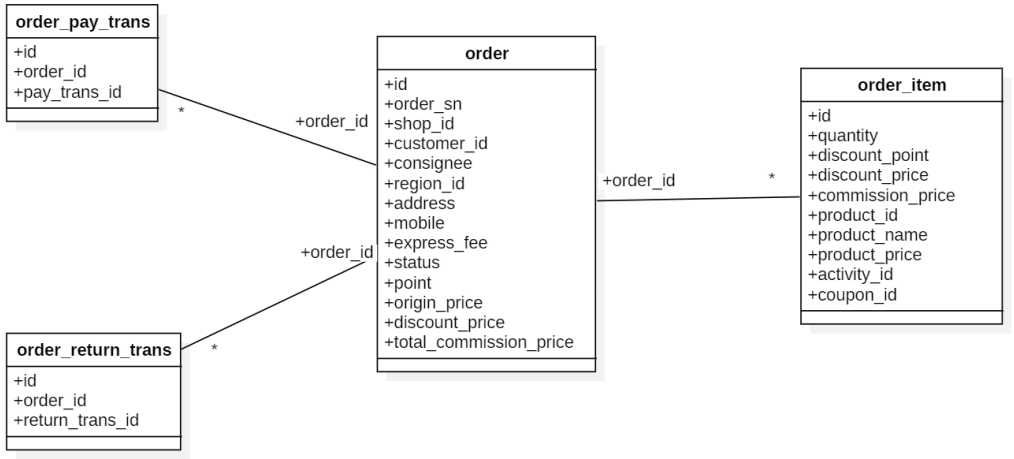
类	含义
Order	订单及其相关信息
OrderItem	订单明细信息
Pack	包裹信息
PackOrderItem	与包裹相关联的订单明细信息
Customer	外部顾客模块的顾客信息
Shop	外部模块商铺的商铺信息
PayTrans	与订单相关联的支付单信息，来源于外部支付模块
ReturnTrans	与订单相关联的退款单信息，来源于外部支付模块
Activity	外部产品模块的活动信息
Coupon	外部顾客模块的优惠券信息
Product	外部产品模块的产品信息
Express	外部物流模块的运单信息

3.2.6.3 类图



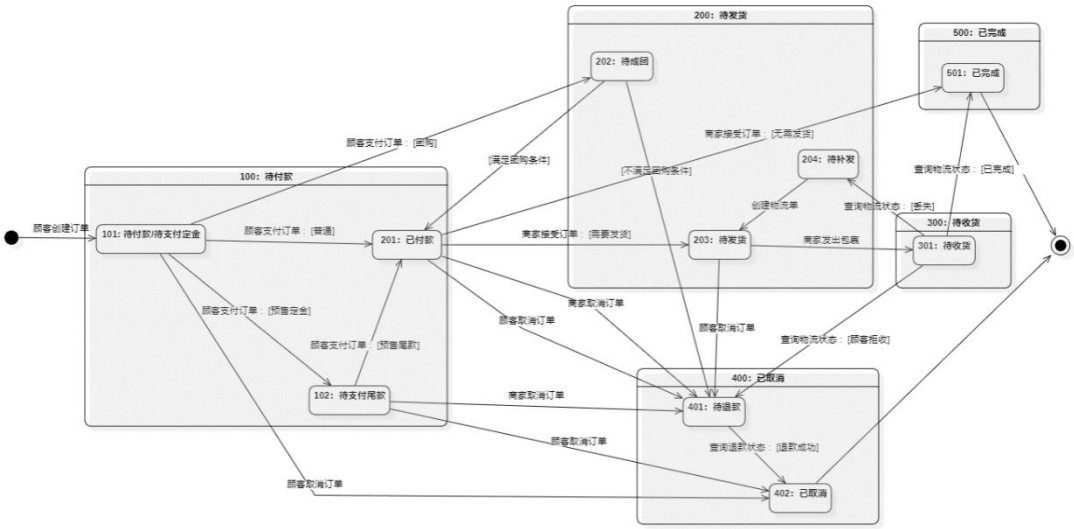
这里为了实现订单种类的多态特性，使用了 Java 的多态创建了三个订单类型的子类，在其中通过重写方法和模版模式的方法实现多态和满足开闭原则，即代码对于扩展是开放的，并且对于修改是封闭的。

3.2.6.4 数据库图



表名	存储数据意义
order	订单的整体信息
order_item	订单中每个明细的信息
order_pay_trans	订单与每一笔支付单的对应关系表
order_return_trans	订单与每一笔退款单的对应关系表

3.2.6.5 状态机图



订单的状态我们区分为顾客视角下的状态和商户视角下的状态，其中顾客视角下的状态属于大的订单状态（即图中大的框），是虚化的概念，方便顾客的理解和查看，这五个状态包括：

100 待付款,200 待发货,300 待收货,400 已取消,500 已完成

商户视角的订单信息可以用大状态下包含的若干小状态表示，是实际的订单状态，其中虽然 300 待收货和 500 已完成只包括一个状态，但为了方便区分顾客和商户的视角差异，仍保留了各自的小状态。

订单状态的变化可以描述如下：

- 首先订单由顾客创建订单产生，此时为 101 待付款/待支付定金状态；
- 对于不同种类的待付款状态的订单，可以分为：
  - 团购订单：

  首先顾客要全款支付团购订单，订单转为 202 待成团状态；

  定时器每天会自动调用查询团购结果的接口来更新订单的状态，如果团购时间结束，且满足团购条件（团购达到一定的人数），则订单会转为 201 已付款状态，并自动产生退款退还给顾客；如果不满足团购条件（团购没有达到一定的人数），则会取消订单，进入 401 待退款状态，将顾客支付的全部费用退还给顾客，团购失败。

  - 预售订单：

  顾客首先要支付定金，订单进入 102 待支付尾款状态；

  如果在 102 待支付尾款状态下，商家取消订单，则需要将顾客所付定金退还给顾客，进入 401 待退款状态；如果是顾客取消订单，则定金不会退还，订单会进入 402 已取消状态；

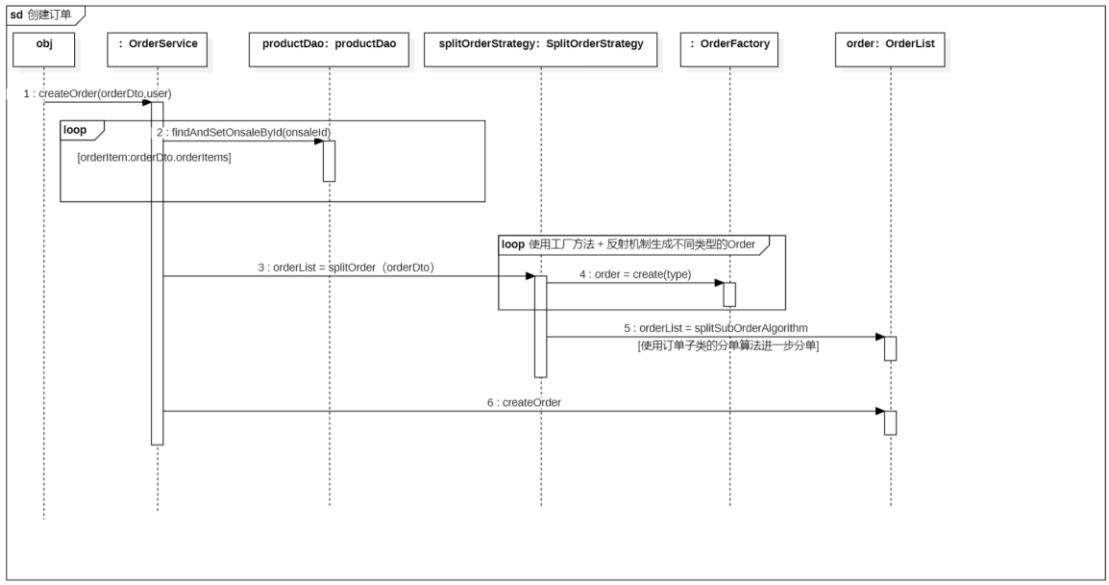
  - 普通订单：

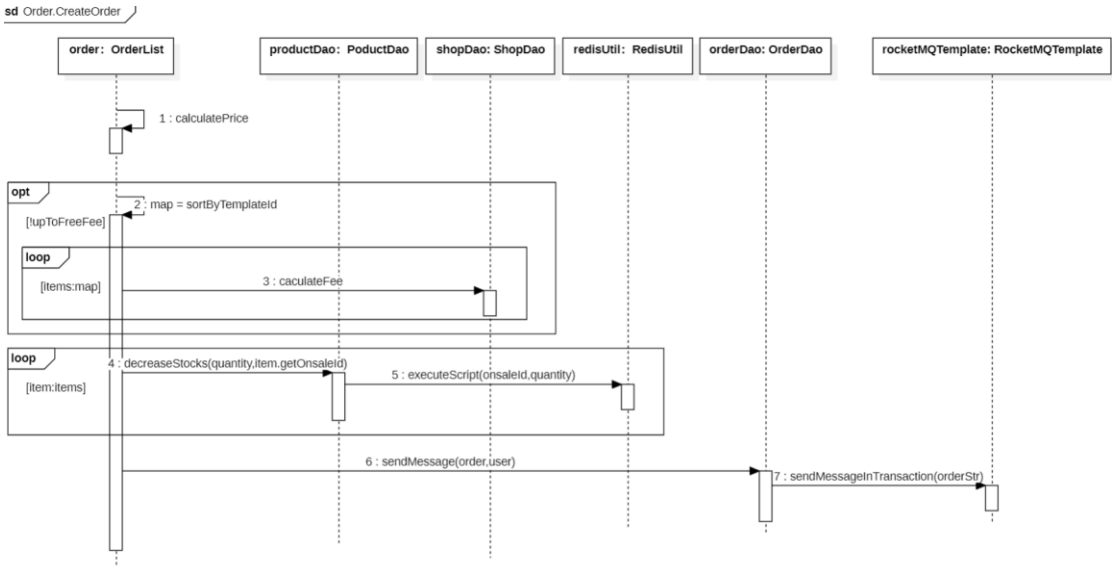
  顾客可以直接支付订单，进入 201 已付款状态；
- 在已付款状态下，需要商家先进行接受订单，如果商品是不需要发货的电子商品，则可以直接完成，进入 501 已完成状态；如果是需要发货的订单，商家接受订单时系统会自动同步创建物流单，如果是需要拆单发货的商品，系统会自动将一个订单拆分成多个物流单并记录到数据库中，订单进入 203 待发货状态。
- 待发货状态下的订单，商户发货后订单状态更新为 301 待收货状态；
- 在待收货下的订单，是处于物流运输状态下的订单，在此过程中可以通过物流模块的接口查询物流状态来更新订单状态，查询物流状态又会出现以下几种情况：
  - 物流状态已收货：说明订单已完成，进入 501 已完成状态
  - 物流状态丢失：说明在途包裹丢失，需要商户进行补发，订单状态转到 204 待补发状态，调用创建物流单进行重新进入 203 待发货状态。
  - 物流状态顾客拒接签收：需要进行退款，进入 401 待退款状态
- 在 100 待付款和 200 待发货这两个大状态下的订单用户可以直接取消使订单进入 400 已取消大状态，其他状态则不可以直接取消；

3.2.6.6 时序图

顾客创建订单

创建订单的业务非常复杂，因此将时序图拆分成三个部分进行说明，首先对于前端传输过来的订单需要根据商品的活动种类执行拆单算法，对应的拆单要求和算法见 3.2.5





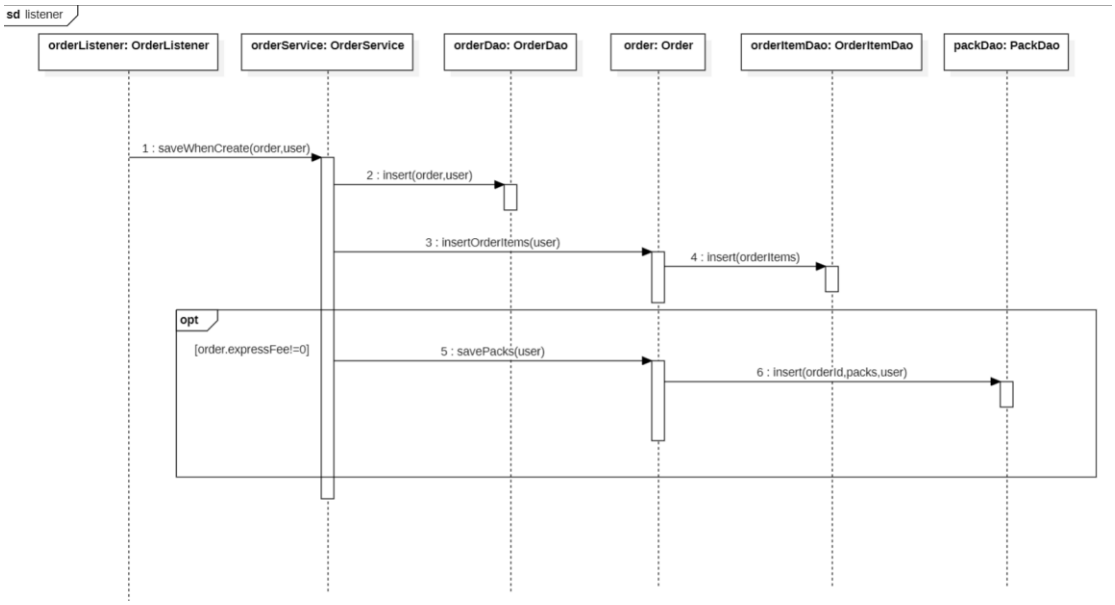
之后根据具体的活动类型执行进一步的拆单和创建订单业务，这里会计算是否到达免邮门槛，如果未达到则需要计算运费，达到免邮门槛则可以在商户接受订单的时候进行计算；

此外这里存在很多非常复杂的技术细节，**需要考虑创建订单在高并发场景下服务器的负载和数据一致性问题**，具体的解决措施如下：

1、 使用消息队列 RocketMQ 实现削峰填谷，在大量订单涌入的时候使用消息队列将对应的订单作为消息发到消息队列服务器，而不是对每一个进来的订单都执行数据库读写操作，数据库读写的开销比较大，如果对每个订单都进行数据库读写会导致线程的阻塞和最后服务器的崩溃。

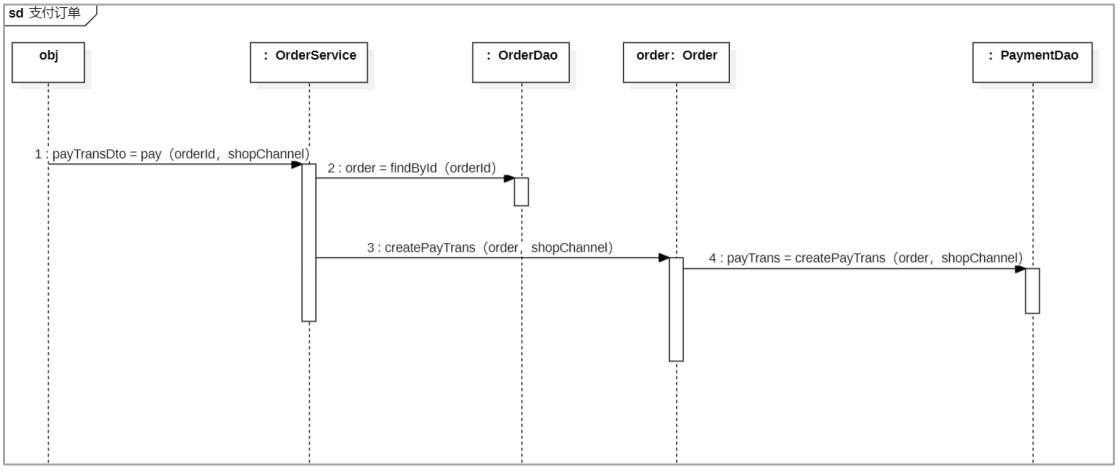
2、 对于高并发数据一致性的情形主要是要避免出现超卖的情况，这里采用了 Redis 技术将库存读取到 Redis 中，当订单进来的时候会先扣除 Redis 中的库存信息，Redis 通过 lua 脚本的方式进行判断当前库存是否充足并且返回扣除后的结果，可以保证一致性，即这里判断和扣除库存操作时一气呵成不会被打断的，从而避免出现数据的不一致和超卖的情形。

3、 最后这里还要保证 RocketMQ 消息的事务性，即如果在订单存储到数据库的时候如果消息队列崩溃会没有执行扣除产品模块的库存操作，最终会导致超卖，因此需要保证每次创建订单消息队列都发出消息，所以需要使用 RocketMQ 的事务消息，要实现这个功能需要将创建预订单和存储订单分离，上述步骤都为创建预订单，之后往 RocketMQ 上发送一个半消息，RocketMQ 收到这条半消息后将会回调订单模块，这个时候订单模块将完成订单数据的存储，如下时序图所示

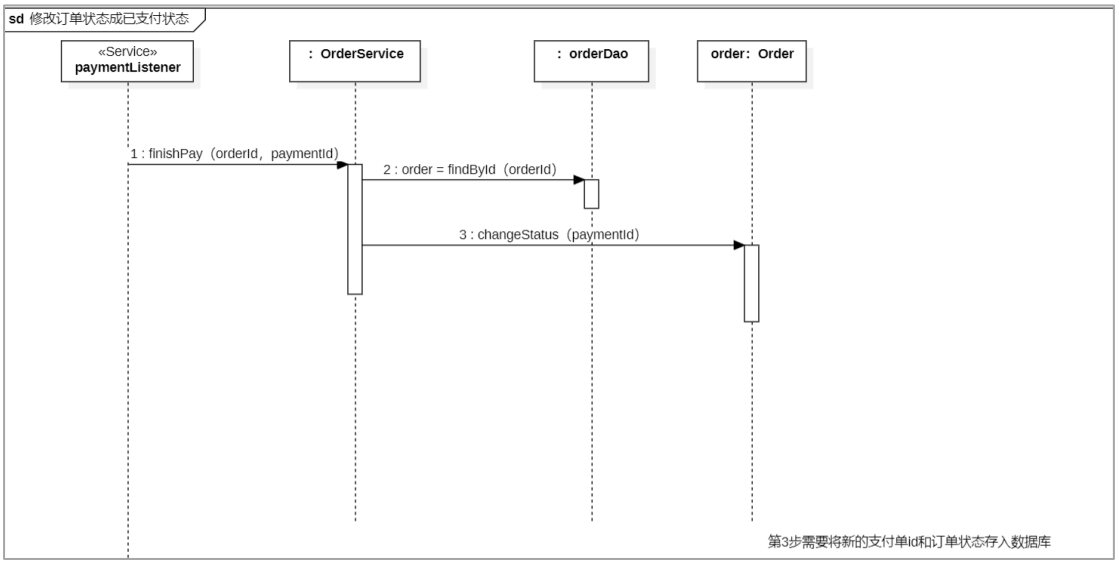


这里需要判断订单是否达到免邮门槛来存储包裹，如果达到了免邮门槛则还没有计算对应的包裹，因此需要无需存储包裹，否则需要存储生成的订单；最后完成订单存储后提交事务，消息队列中的消息被发出，产品模块订阅该条消息之后实现扣库存操作，从而完成一个订单的创建。

顾客支付订单

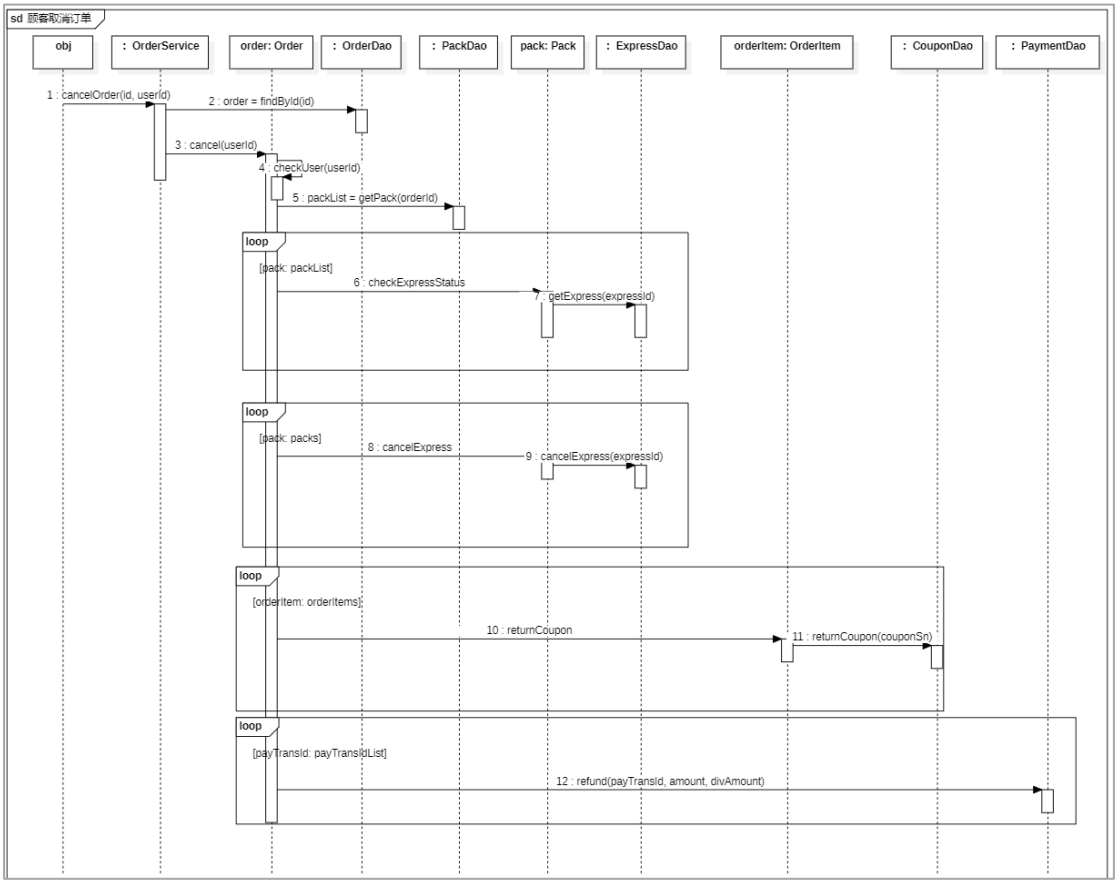


支付订单调用 OrderService 的 pay 方法，OrderService 调用 OrderDao 的 findById 方法将 orderId 变成 order 对象，OrderService 调用 order 对象的 createPayTrans 方法，将职责分给 order。Order 调用 PaymentDao 的 createPayTrans 方法将职责分给支付模块。



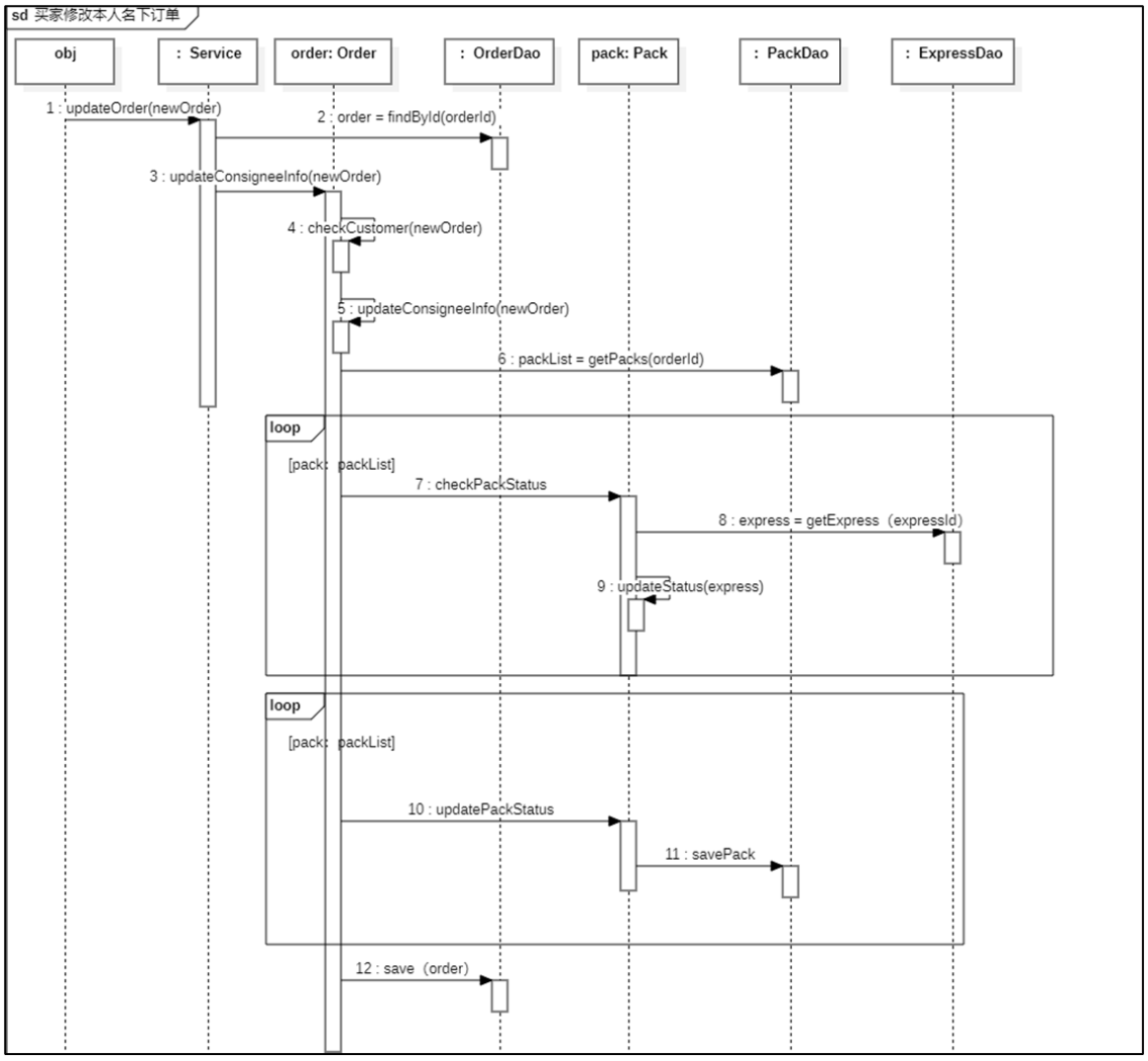
支付模块的 paymentListener 调用 OrderService 修改订单状态，OrderService 调用 OrderDao 将 id 变成 order 对象，进而调用 order 对象的 changeStatus 方法。

顾客取消订单



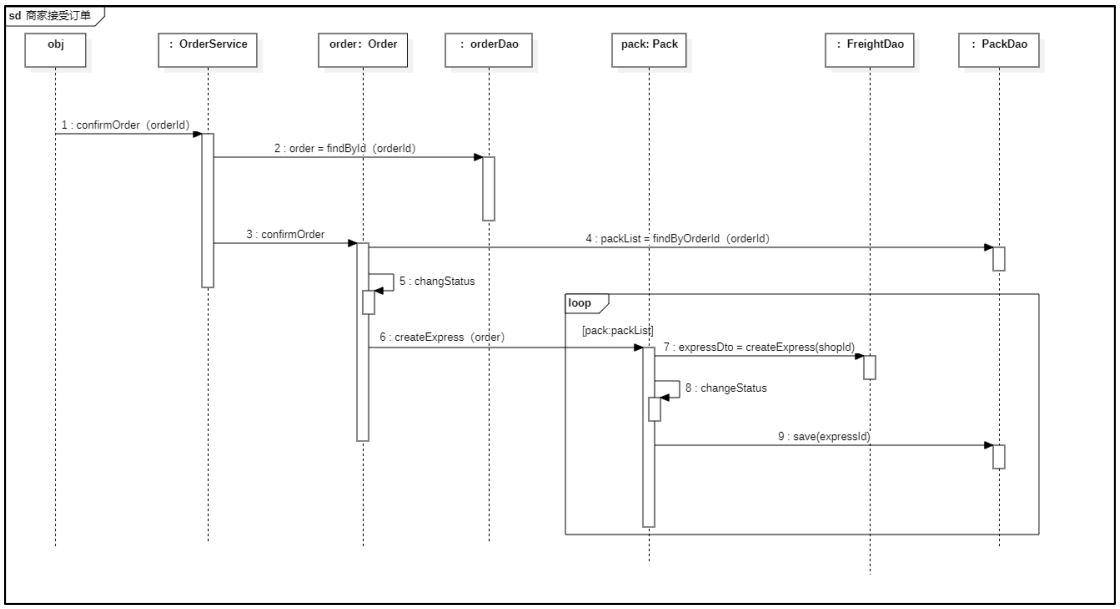
- 1-4 校验 userId 是否合法，因为顾客只能修改取消自己的订单
- 5-6 根据订单查找到订单对应的包裹（一个或多个），检查是否有包裹已发货
- 8-9 根据订单里的包裹来取消所有的物流订单，
- 10 取消订单后需要退回原有的优惠券
- 11 退款

顾客修改订单



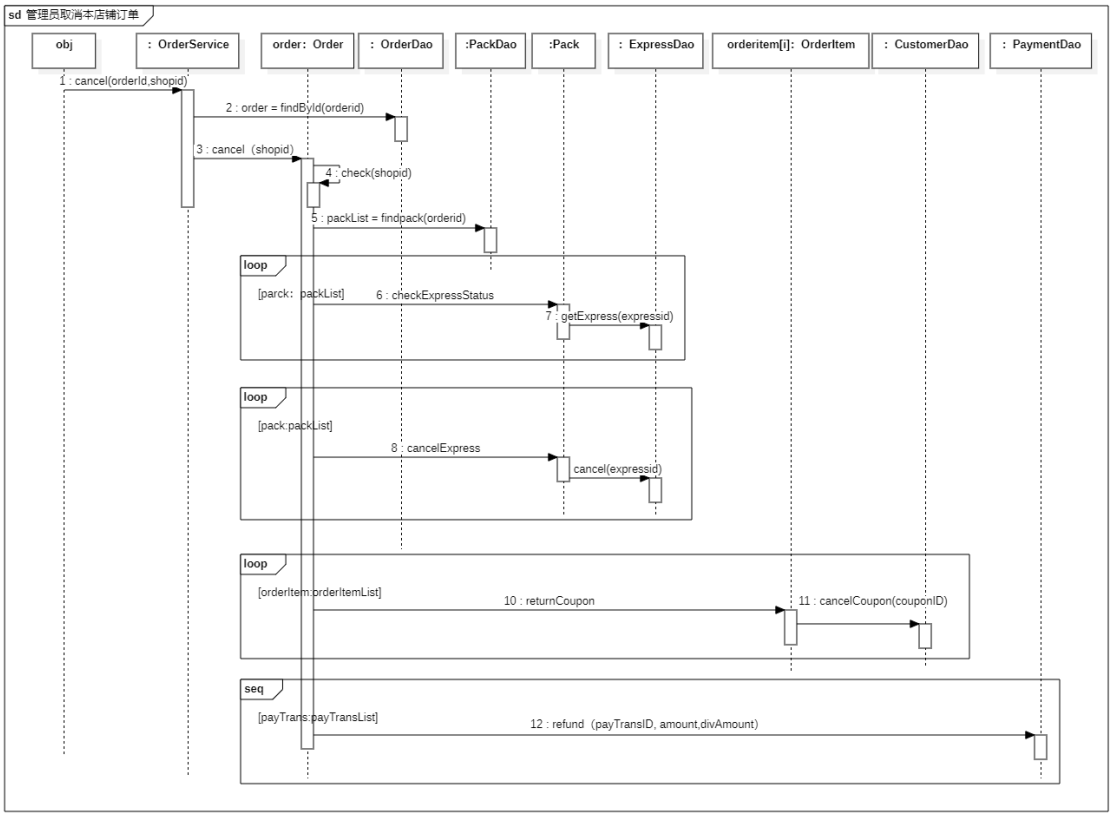
- 传入修改后的新的订单信息(只能修改收件人收货地址信息)
- service 层根据 orderId 查到 oldOrder 的 bo 对象，然后将修改订单的职责分配给 bo 对象
- bo 先检查顾客信息是否一致，合法则通过 orderId 查到所有的 pack，把查询订单状态和修改运单的职责分配给每个 pack，如果查询的所有 express 都还没有发货，能成功修改 express 的收件人信息，则重新修改运单信息，最后更新 order 中的收件人信息保存到数据库中。
- 如果在某个 pack 发生错误则进行回滚，为了保持事务的一致性，所以将查询订单状态与修改订单信息分为两次循环

商家接受订单



商家接受订单首先要根据 orderID 查到顾客创建的订单，然后根据订单的类型（普通、团购、预售）修改订单的状态，然后根据订单里的包裹的分包信息对每一个包裹分别创建物流单，这里采用信息专家和创建者的原则用 order 对象去处理创建订单里所有物流单的请求，order 再把创建物流单的职责分配给每一个包裹。

管理员取消订单



- 通过 orderid 找到 order，校验 order 的 shopid 和传进来的 shopid 是否一致。
- 通过 orderid 找到订单对应的包裹，判断这些包裹是否已经发货。只要有一个已发货就无法取消订单。
- 如果都未发货，则循环遍历包裹，分别用 pack 对象调用 expressDao 完成取消物流单。
- 由于一个 orderItem 对应一个优惠券（或没有优惠活动），遍历每一个 orderItem，调用 CustomerDao 将优惠券退回给顾客。
- 调用支付模块，把订单保存的 amount 和 divAmount 和支付 id 传给 PaymentDao 完成退款业务。通过 orderid 找到订单对应的包裹，判断这些包裹是否已经发货。只要有一个已发货就无法取消订单。

3.2.7 接口

OOMALL | 1.3.2 | mingqcn | SwaggerHub

订单模块接口描述、请求方式、请求路径如下：

3.2.8 存储分配

使用 MySQL 数据库和 MongoDB 数据库
MySQL 数据库建立以下表格：
Customer_address 顾客地址表
Customer_cart 顾客购物车表
Customer_coupon 顾客优惠券表
Customer_customer 顾客基本信息表
Order_item 订单子项信息表
Order_order 订单基本信息表
Order_payment 订单支付记录表
Order_refund 订单退款记录表

MongoDB 数据库建立 Order_pack 集合，用于存放订单中的分包 pack 对象，Pack 对象的属性包括 orderId 和 packs，packs 中要多个 pack，其中每一个 pack 里面存储的有物流费用 fee、运单号 id(expressId)，以及包裹的重量 weight，pack 中包含的 orderItem 的数量和信息(packItems)等
存储格式为：

```
{
  orderId: "123456789",
  [
    {
    fee: 10,
    expressId: "123123123",
    weight: 10,
      [
        {
        quantity: 5,
        orderItemId: "987654321"
        }
      ]
    }
  ]
}
```

3.2.9 限制条件

1. 在运行项目的服务器上需要安装 docker。为了使项目能在不同环境下使用，我们采用配置 docker 容器，就可以跨平台，跨服务器，实现应用程序跨平台

间的无缝衔接。因此，我们需要将开发完成的应用模块打包成一个 docker 镜像。
2. 用户需要登录之后才能完成相应的操作，在顾客和订单模块中，查询的相关功能是需要管理员或者商户权限才可以使用，修改的相关功能只有相关的商户才可以使用。

需要事先部署运行项目的服务器群上建立 docker swarm 集群，要按照项目的部署说明在华为云的 7 台云耀云服务器上部署好相应数量和配置的 nacos，mongoDB，mysql，redis，rocketmq 的 docker 容器。其中 mysql 服务器部署了一台，mongo 服务器部署了三台用来支持事务(主要用到其中一台)，nacos 部署了一台，rocketMQ 部署了 nameserver 和 broker，因受到物理机内存、CPU、IO 读写速率、网络传输效率的限制，系统的性能存在一定的瓶颈。
有一些 API 需要用户登录之后才能完成相应的操作，在顾客模块中，完成具体的功能需要有对应的权限来完成。

3.2.10 测试要点

1、顾客创建订单的优惠价格计算、积点计算和运费计算是否正确：
- 优惠价格计算：可以编写单元测试来验证订单中是否正确应用了优惠折扣。例如，测试一个特定商品购买时是否正确应用了折扣，或者测试购物车中多个商品的折扣计算是否准确。
- 积点计算：类似于优惠价格计算，可以编写单元测试来验证订单中的积点计算是否正确。测试可以涵盖不同商品的积点计算、积点与订单总额的关系等。
- 运费计算：同样，可以编写单元测试来验证订单中的运费计算是否准确。测试可以包括不同地区的运费计算、不同商品组合的运费计算等。

2、商户接受订单时的分包是否合理，是否成功调用物流的 API 发货：
- 分包合理性：可以编写集成测试来模拟商户接受订单并进行分包操作，然后验证分包结果是否符合预期。测试可以涵盖不同商品组合、不同包裹尺寸等情况下的分包结果。
- 物流 API 调用：可以编写集成测试来模拟商户使用物流 API 发货，并验证 API 是否成功调用、物流状态是否更新等。测试可以模拟不同物流服务提供商的 API 调用，以确保系统能够与各种物流服务进行正常交互。

3、修改和删除订单是否生效：
- 修改订单：可以编写单元测试来验证修改订单功能是否正确实现。测试可以包括修改订单中的商品数量、修改收货地址等操作，然后验证修改后的订单信息是否正确。
- 删除订单：同样，可以编写单元测试来验证删除订单功能是否生效。测试可以模拟删除订单并验证订单是否从系统中完全删除。

4、查询订单时是否具有对应的权限和资源：
- 查询订单时请求的资源是否为该用户所拥有，订单模块需要对非法的请求资源的情况进行拦截。
- 查询订单时请求的资源是否合法，资源如果不存在说明也是非法的请求，需要拒绝访问。

对于**单元测试**使用白盒测试方法，基本路径测试和测试代码的覆盖率可以确保在代码级别对以上功能进行测试。
对于**集成测试**使用黑盒测试方法，基于等价类划分、边界值还有异常值推断来验证系统整体功能的正确性。
对于**性能测试**使用 Jmeter 在短时间内发送大量创建订单的请求，从而验证在秒杀的场景下系统是否可以处理海量请求而不会发生服务器的崩溃