|  |  |
| --- | --- |
| 产品名称Product name | 密级Confidentiality level |
| 设计说明书 | 内部公开 |
| 产品版本Product version | Total pages 共 74页 |
| 1.1 |

**Koi-鞋城Online产品软件设计说明书**

|  |  |  |  |
| --- | --- | --- | --- |
| 拟制人： | 袁顺鑫 | 时间： | 2024/5/18 |
| 评审人： |  | 时间： |  |
| 批准人： |  | 时间： |  |

**修订记录**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **修订内容** | **修订人** |
| 1.0 | 2024/5/18 | 初稿，模板制定 | 袁顺鑫 |
| 1.1 | 2024/5/28 | 完成编写 | 全体组员 |
|  |  |  |  |
|  |  |  |  |

**目录**

[1 研发背景 6](#_Toc168081694)

[2 阅读对象 6](#_Toc168081695)

[3 参考资料 6](#_Toc168081696)

[4 术语、缩略语 6](#_Toc168081697)

[5 概要设计 7](#_Toc168081698)

[5.1 需求概述 7](#_Toc168081699)

[5.2 系统功能整体流程 13](#_Toc168081700)

[5.3 系统数据流图 18](#_Toc168081701)

[6 详细设计 19](#_Toc168081702)

[6.1 用户模块详细设计 19](#_Toc168081703)

[6.1.1 模块界面 19](#_Toc168081704)

[6.1.2 模块内处理流程 19](#_Toc168081705)

[6.1.3 模块类图 21](#_Toc168081706)

[6.1.4 模块数据流 22](#_Toc168081707)

[6.2 企宣模块详细设计 22](#_Toc168081708)

[6.2.1 企宣界面 22](#_Toc168081709)

[6.2.2 模块内处理流程 22](#_Toc168081710)

[6.2.3 模块类图 24](#_Toc168081711)

[6.2.4 模块数据流 24](#_Toc168081712)

[6.3 商品模块详细设计 25](#_Toc168081713)

[6.3.1 模块界面 25](#_Toc168081714)

[6.3.2 模块内处理流程 25](#_Toc168081715)

[6.3.3 模块类图 28](#_Toc168081716)

[6.3.4 模块数据流 29](#_Toc168081717)

[6.4 交易模块详细设计 35](#_Toc168081718)

[6.4.1 模块界面 35](#_Toc168081719)

[6.4.2 模块内处理流程 35](#_Toc168081720)

[6.4.3 模块类图 43](#_Toc168081721)

[6.4.4 模块数据流 45](#_Toc168081722)

[6.5 优惠券模块详细设计 51](#_Toc168081723)

[6.5.1 模块界面 51](#_Toc168081724)

[6.5.2 模块内处理流程 52](#_Toc168081725)

[6.5.3 模块类图 52](#_Toc168081726)

[6.5.4 模块数据流 53](#_Toc168081727)

[6.6 评论模块详细设计 54](#_Toc168081728)

[6.6.1 模块界面 54](#_Toc168081729)

[6.6.2 模块内处理流程 54](#_Toc168081730)

[6.6.3 模块类图 55](#_Toc168081731)

[6.6.4 模块数据流 56](#_Toc168081732)

[6.7 媒资模块详细设计 56](#_Toc168081733)

[6.7.1 模块界面 56](#_Toc168081734)

[6.7.2 模块内处理流程 57](#_Toc168081735)

[6.7.3 模块类图 58](#_Toc168081736)

[6.7.4 模块数据流 58](#_Toc168081737)

[6.8 退款模块详细设计 59](#_Toc168081738)

[6.8.1 模块界面 59](#_Toc168081739)

[6.8.2 模块内处理流程 59](#_Toc168081740)

[6.8.3 模块类图 60](#_Toc168081741)

[6.8.4 模块数据流 60](#_Toc168081742)

[7 数据库设计 61](#_Toc168081743)

[7.1 数据库设计综述 61](#_Toc168081744)

[7.2 数据库逻辑结构设计 62](#_Toc168081745)

[7.3 数据库物理结构设计 63](#_Toc168081746)

[7.4 数据库安全设计 63](#_Toc168081747)

[7.5 数据字典 64](#_Toc168081748)

[8 系统可靠性设计 72](#_Toc168081749)

# 研发背景

系统名称为Koi-鞋城Online，类型为在线电子商务平台，Koi-鞋城Online 是一个集成了多种现代技术和服务的在线鞋类购物平台。在整个项目组合中，它位于核心地位，是公司拓展市场、提升品牌影响力和增加销售额的主要工具。Koi-鞋城Online 不仅承载了公司大量的线上销售业务，还通过数据分析、用户行为研究等功能，支持其他相关业务的发展和优化。

Koi-鞋城Online 的研发和运营旨在打造一个高效、便捷、安全和个性化的在线购物平台。Koi-鞋城Online 作为核心项目，发挥着重要作用，不仅直接带动销售增长，还通过数据和技术支持其他相关业务的发展。通过不断的技术创新和服务优化，Koi-鞋城Online 将持续提升用户满意度和品牌竞争力，实现长期发展目标。

# 阅读对象

内部管理人员：以便他们了解项目的整体情况和发展方向。

开发团队成员：作为开发任务的参考和指导。

市场营销团队：以便他们了解产品特点和市场定位，从而制定营销策略。

合作伙伴和投资人：以便他们了解产品和发展前景，从而做出合适的合作或投资决策。

# 参考资料

需求说明书

# 术语、缩略语

|  |  |
| --- | --- |
| 术语、缩略语 | 说明 |
| 分布式 | 分布式是指多个系统协同合作完成一个特定任务的系统，  它是不同的系统部署在不同的服务器上，服务器之间相互调用。 |
| 负载均衡 | 将用户访问请求，通过某种算法，分发到集群中的节点。 |

# 概要设计

## 需求概述

5.1.1． 用户模块：

  用户注册：允许用户创建新账户并提供必要的个人信息。

  用户登录：允许已注册用户通过用户名和密码登录系统。

  用户管理：允许管理员对用户账户进行管理，包括禁用、删除或修改用户信息。

  用户权限：实施不同的用户权限，例如管理员和普通用户，以控制其对系统功能和数据的访问级别。

5.1.2 商品模块：

  新增类别：

    1.允许管理员添加新的商品类别。

    2.管理员应该能够输入类别名称和其他相关信息。

  删除类别：

    1.允许管理员删除现有的商品类别。

    2.管理员应该能够选择要删除的类别，并确认删除操作。

    3.删除类别时，相关的商品和其他关联数据也应该被相应地处理。

  修改类别：

    1.允许管理员修改已存在的商品类别的属性。

    2.管理员应该能够选择要修改的类别，并提供新的类别名称和其他相关信息。

  停用类别：

    1.允许管理员停用某个商品类别，使其在系统中不再可见。

    2.停用的类别不会在商品列表和其他相关功能中显示。

  启用禁用类别：

    1.允许管理员启用或禁用某个商品类别。

    2.禁用的类别在系统中可见，但不可操作或使用。

  新增商品：

    1.允许管理员添加新的商品。

    2.管理员应该能够输入商品名称、价格、描述、图片等相关信息。

    3.新增的商品应该与所属的类别进行关联。

  删除商品：

    1.允许管理员删除现有的商品。

    2.管理员应该能够选择要删除的商品，并确认删除操作。

    3.删除商品时，相关的订单、评论和其他关联数据也应该被相应地处理。

  启用禁用商品：

    1.允许管理员启用或禁用某个商品。

    2.禁用的商品应该在系统中可见，但不可购买或显示在商品列表中。

  查询商品：

    1.允许用户通过关键词、类别或其他筛选条件来查询商品。

    2.查询结果应该根据用户的筛选条件进行过滤，并以可视化的方式展示给用户。

  修改商品：

    1.允许管理员修改已存在的商品的属性。

    2.管理员应该能够选择要修改的商品，并提供新的商品名称、价格、描述、图片等相关信息。

5.1.3 订单模块：

  获取订单明细详细信息：

    1.允许管理员查看特定订单的详细信息，包括订单号、购买商品、购买者信息、支付状态、配送信息等。

    2.显示订单的支付金额、下单时间、发货状态、交付状态等相关信息。

  审批退款申请：

    1.允许管理员审核用户提交的退款申请。

    2.管理员能够查看退款申请的详细信息，包括退款金额、申请理由和订单信息等。

    3.管理员可以选择同意或拒绝退款申请，并提供相应的反馈给用户。

  查看退款详情：

    1.允许管理员查看已审批的退款申请的详细信息。

    2.显示已退款的订单号、退款金额、退款时间和退款方式等相关信息。

  同意退货：

    1.允许管理员处理用户提交的退货申请。

    2.管理员应该能够查看退货申请的详细信息，包括退货商品、退货原因和订单信息等。

    3.管理员可以选择同意用户的退货请求，并提供进一步的指示和退货流程。

5.1.4 搜索模块：

  商品搜索：

    1.允许用户通过关键词搜索商品。

    2.搜索功能能够在商品名称、描述、类别、价格范围等关键属性中进行搜索。

  订单搜索：

    1.允许管理员或用户通过订单号、购买者信息、下单日期等关键属性搜索订单。

    2.搜索功能能够根据指定的搜索条件进行过滤和查找订单。

5.1.5 促销模块（优惠券）：

  新增优惠券：

    1.允许管理员创建新的优惠券。

    2.管理员能够设置优惠券的名称、折扣金额或折扣比例、有效期、适用范围等相关属性。

  分页查询优惠券：

    1.允许管理员按照分页的方式查询系统中的优惠券列表。

    2.查询结果包括优惠券的基本信息，如名称、折扣金额或折扣比例、有效期等。

  发放优惠券：

    1.允许管理员将优惠券发放给指定的用户。

    3.发放优惠券后，相关的用户能够在其账户中看到优惠券的详细信息。

  获取优惠券信息：

    1.允许用户查询自己账户中的优惠券信息。

    2.用户能够查看优惠券的名称、折扣金额或折扣比例、有效期等相关信息。

  暂停发放优惠券：

    1.允许管理员暂停某个优惠券的发放。

    2.暂停发放后，新用户将无法获得该优惠券，但已发放的优惠券仍然有效。

  删除优惠券：

    1.允许管理员删除不再需要的优惠券。

    2.删除优惠券后，相关的用户将无法再使用该优惠券。

  查看发放中的优惠券：

    1.允许管理员查看当前正在发放中的优惠券列表。

    2.查看结果包括优惠券的基本信息和发放数量等。

  查询兑换码：

    1.允许管理员或用户通过兑换码查询优惠券的可用性和相关信息。

5.1.6 日志模块：

  新增日志：

    1.允许系统或管理员向日志系统中添加新的日志记录。

    2.日志包括相关的事件、操作、时间戳、用户信息等。

  分页询日志：

    1.允许管理员按照分页的方式查询系统中的日志记录。

  查看日志具体信息：

    1.允许管理员查看特定日志记录的详细信息。

    2.管理员能够选择要查看的日志记录，并查看该日志的完整内容和相关信息。

5.1.7 评论模块：

  分页查询评论：

    1.允许用户或管理员按照分页的方式查询系统中的评论列表。

    2.查询结果根据时间顺序进行排序，并显示评论的摘要信息。

  查询指定评论：

    1.允许用户或管理员通过指定评论的ID或其他唯一标识符来查询特定的评论。

    2.查询结果应包括评论的详细信息。

  查询指定评论点赞状态：

    1.允许用户查询自己对于特定评论的点赞状态。

    2.查询结果可以显示用户是否已经点赞该评论。

5.1.8 企宣与广告模块：

  查询广告信息：

    1.允许管理员查询系统中的广告信息列表。

    2.查询结果包括广告的标题、内容、链接、展示时间等相关信息。

  新增广告：

    1.允许管理员创建新的广告。

    2.管理员能够设置广告的标题、内容、链接、展示时间等属性。

  禁用和启用广告：

    1.允许管理员禁用或启用特定的广告。

    2.禁用广告后，该广告将不再显示在系统中。

  删除广告：

    1.允许管理员删除不再需要的广告。

    2.删除广告后，该广告的相关信息将被永久删除。

  企业合作分页查询：

    1.允许管理员查询系统中的企业合作信息列表。

    2.查询结果根据时间顺序进行排序，并显示企业合作的摘要信息，如合作标题、时间等。

  新增企业合作：

    1.允许管理员创建新的企业合作。

    2.管理员能够设置合作的标题、内容、时间等属性。

  删除企业合作：

    1.允许管理员删除不再需要的企业合作。

    2.删除企业合作后，该合作的相关信息将被永久删除。

  查询合作具体信息：

    1.允许管理员查询特定企业合作的详细信息。

    2.管理员能够选择要查询的企业合作，并查看该合作的完整内容和相关信息。

  生成报表：

    1.允许管理员生成与企业宣传和广告相关的报表。

    2.报表可以包括广告展示次数、点击率、合作效果等数据。

  生成图片：

    1.允许管理员生成与企业宣传和广告相关的图片。

    2.图片可以用于广告展示、宣传资料等。

## 系统功能整体流程

5.2.1. 用户模块

功能：

用户注册：允许用户创建新账户并提供必要的个人信息。

用户登录：允许已注册用户通过用户名和密码登录系统。

用户管理：允许管理员对用户账户进行管理，包括禁用、删除或修改用户信息。

用户权限：实施不同的用户权限，例如管理员和普通用户，以控制其对系统功能和数据的访问级别。

运行流程：

用户访问注册/登录页面，填写必要信息进行注册或登录，系统验证用户信息并允许访问个人中心。管理员在后台管理界面对用户账户进行管理，包括禁用、删除或修改用户信息，根据用户角色分配不同的权限，控制其对系统功能和数据的访问。

界面：

注册/登录界面：输入用户名、密码、邮箱等信息。

个人中心界面：显示用户信息、订单历史、收藏夹等。

后台用户管理界面：管理员进行用户账户管理操作。

5.2.2. 商品模块

功能：

新增类别：允许管理员添加新的商品类别，输入类别名称和相关信息。

删除类别：允许管理员删除现有的商品类别，确认删除操作。

修改类别：允许管理员修改已存在的商品类别的属性。

停用类别：允许管理员停用某个商品类别，使其在系统中不再可见。

启用禁用类别：允许管理员启用或禁用某个商品类别。

新增商品：允许管理员添加新的商品，输入商品名称、价格、描述、图片等相关信息。

删除商品：允许管理员删除现有的商品，确认删除操作。

启用禁用商品：允许管理员启用或禁用某个商品。

查询商品：允许用户通过关键词、类别或其他筛选条件来查询商品。

修改商品：允许管理员修改已存在的商品的属性。

运行流程：

管理员在后台管理界面添加、删除、修改或停用商品类别，管理员在后台管理界面添加、删除、启用或禁用商品，并关联到相应类别。用户可以在前台通过关键词、类别或其他筛选条件查询商品。

界面：

商品管理后台界面：显示商品列表、类别管理、标签管理、上架/下架操作。

商品详情界面：显示商品名称、价格、描述、图片等信息，用户可以进行购买操作。

5.2.3. 订单模块

功能：

获取订单明细详细信息：允许管理员查看特定订单的详细信息，包括订单号、购买商品、购买者信息、支付状态、配送信息等。

审批退款申请：允许管理员审核用户提交的退款申请，并提供反馈。

查看退款详情：允许管理员查看已审批的退款申请的详细信息。

同意退货：允许管理员处理用户提交的退货申请，并提供进一步的指示和退货流程。

运行流程：

用户在前台生成订单并选择支付方式，系统记录订单信息。管理员在后台查看订单详情、审核退款申请、处理退货请求。

界面：

订单详情界面：显示订单号、购买商品、购买者信息、支付状态、配送信息等。

退款管理界面：显示退款申请的详细信息和审核结果。

5.2.4. 搜索模块

功能：

商品搜索：允许用户通过关键词搜索商品，在商品名称、描述、类别、价格范围等关键属性中进行搜索。

订单搜索：允许管理员或用户通过订单号、购买者信息、下单日期等关键属性搜索订单。

运行流程：

用户在前台输入关键词或筛选条件进行商品搜索，系统返回符合条件的商品列表。管理员或用户在前台或后台输入订单号或其他属性进行订单搜索，系统返回符合条件的订单列表。

界面：

商品搜索界面：用户输入关键词或筛选条件，系统返回符合条件的商品列表。

订单搜索界面：管理员或用户输入订单号或其他属性，系统返回符合条件的订单列表。

5.2.5. 促销模块（优惠券）

功能：

新增优惠券：允许管理员创建新的优惠券，设置优惠券的名称、折扣金额或折扣比例、有效期、适用范围等。

分页查询优惠券：允许管理员按照分页的方式查询系统中的优惠券列表。

发放优惠券：允许管理员将优惠券发放给指定的用户，用户可以在其账户中查看。

获取优惠券信息：允许用户查询自己账户中的优惠券信息。

暂停发放优惠券：允许管理员暂停某个优惠券的发放。

删除优惠券：允许管理员删除不再需要的优惠券。

查看发放中的优惠券：允许管理员查看当前正在发放中的优惠券列表。

查询兑换码：允许管理员或用户通过兑换码查询优惠券的可用性和相关信息。

运行流程：

管理员在后台创建、查询、发放、暂停发放或删除优惠券。

用户在前台查询和使用自己账户中的优惠券。

界面：

优惠券管理界面：管理员创建、查询、发放、暂停发放或删除优惠券。

用户优惠券界面：用户查看和使用自己的优惠券。

5.2.6. 日志模块

功能：

新增日志：允许系统或管理员向日志系统中添加新的日志记录。

分页查询日志：允许管理员按照分页的方式查询系统中的日志记录。

查看日志具体信息：允许管理员查看特定日志记录的详细信息。

运行流程：

系统自动记录关键操作日志，管理员在后台查看和管理日志记录。

界面：

日志管理界面：管理员查看和管理系统日志记录。

5.2.7 评论模块

功能：

分页查询评论：允许用户或管理员按照分页的方式查询系统中的评论列表。

查询指定评论：允许用户或管理员通过指定评论的ID或其他唯一标识符来查询特定的评论。

查询指定评论点赞状态：允许用户查询自己对于特定评论的点赞状态。

运行流程：

用户在前台提交评论，系统记录评论信息，管理员在后台分页查询和管理评论记录。

界面：

评论管理界面：管理员查看和管理用户评论。

评论详情界面：用户查看特定评论的详细信息和点赞状态。

5.2.8 企宣与广告模块

功能：

查询广告信息：允许管理员查询系统中的广告信息列表。

新增广告：允许管理员创建新的广告，设置广告的标题、内容、链接、展示时间等属性。

禁用和启用广告：允许管理员禁用或启用特定的广告。

删除广告：允许管理员删除不再需要的广告。

企业合作分页查询：允许管理员查询系统中的企业合作信息列表。

新增企业合作：允许管理员创建新的企业合作，设置合作的标题、内容、时间等属性。

删除企业合作：允许管理员删除不再需要的企业合作。

查询合作具体信息：允许管理员查询特定企业合作的详细信息。

生成报表：允许管理员生成与企业宣传和广告相关的报表。

生成图片：允许管理员生成与企业宣传和广告相关的图片。

运行流程：

管理员在后台创建、查询、禁用、启用或删除广告和企业合作信息，管理员生成相关报表和图片，用于宣传和广告展示。

界面：

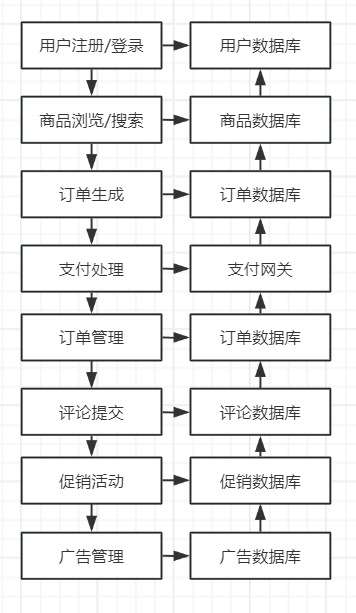
广告管理界面：管理员创建、查询、禁用、启用或删除广告。

企业合作管理界面：管理员创建、查询、删除企业合作信息。

报表生成界面：管理员生成与企业宣传和广告相关的报表。

图片生成界面：管理员生成与企业宣传和广告相关的图片。

## 系统数据流图



用户注册/登录：用户在前台注册或登录，数据流向用户数据库。

商品浏览/搜索：用户浏览或搜索商品，数据流向商品数据库，返回商品信息。

订单生成：用户下单，数据流向订单数据库，记录订单信息。

支付处理：用户支付订单，数据流向支付网关，确认支付信息后返回订单数据库更新状态。

订单管理：管理员在后台管理订单，包括查看、修改订单状态，数据流向订单数据库。

评论提交：用户提交评论，数据流向评论数据库。

促销活动：用户参与促销活动，数据流向促销数据库，返回优惠信息。

日志记录：系统记录操作日志，数据流向日志数据库。

广告管理：管理员管理广告，数据流向广告数据库，返回广告信息。

# 详细设计

## 用户模块详细设计

### 模块界面

1.登录页面：

提供账号，密码和验证的输入框，以及登录按钮。

如果登录失败，显示错误提示信息，并提供重新输入的选项。

2.注册页面：

包括填写账号、密码等基本信息的输入框。

必须包含勾选使用协议的复选框。

注册失败时，显示相应的错误信息，并提供重新注册的选项

3.用户列表页面：

显示所有用户的基本信息，如用户名、注册时间等。

提供搜索和筛选功能，方便管理员查找特定用户。

每个用户条目旁边有一些操作按钮，如修改状态、删除等，以便管理员对用户进行相应操作

### 模块内处理流程

1、登录功能

1. 页面渲染：前端页面加载时，渲染账号、密码输入框以及验证码输入框和登录按钮。同时，准备错误提示区域但默认不显示。
2. 表单提交：用户输入账号、密码和验证码后，点击登录按钮，前端通过AJAX异步提交表单数据到后端。
3. 后端验证：
   1. 验证账号是否存在。
   2. 验证密码是否正确。
   3. 验证验证码是否正确（可选，根据实现方式可能在前端完成或后端配合）。
4. 响应处理：
   1. 成功：后端返回成功消息，前端跳转到首页或指定页面。
   2. 失败：后端返回错误代码和错误信息，前端根据错误信息显示在错误提示区域，并保持当前页面供用户重新输入。

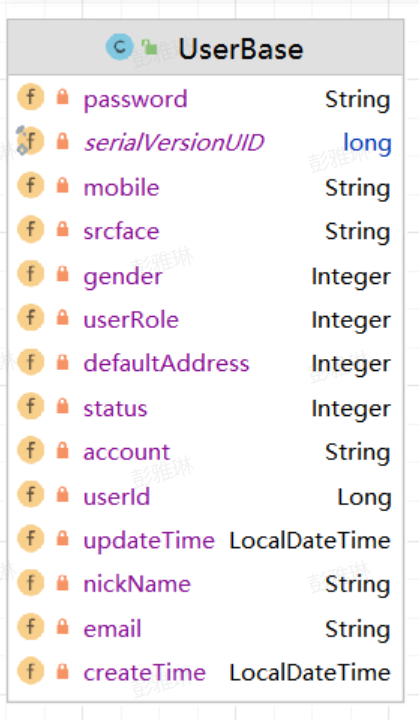
2、注册功能

1. 页面渲染：展示账号、密码等输入框，以及使用协议的复选框。提供注册按钮。
2. 用户输入：用户填写账号、密码信息，并勾选使用协议复选框。
3. 表单验证：前端初步验证输入格式（如账号长度、密码强度），然后提交表单。
4. 后端处理：
   1. 验证账号是否已存在。
   2. 检查必填项是否齐全。
   3. 确认使用协议是否被勾选。
5. 响应处理：
   1. 成功：后端处理注册逻辑，创建用户账号，返回成功信息，前端跳转到登录页面或提示注册成功。
   2. 失败：后端返回错误代码和信息（如账号重复、信息缺失），前端展示错误信息并允许用户修改后重新尝试注册

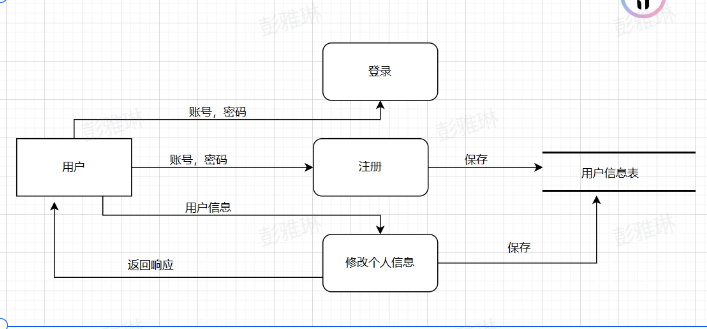
3、用户管理

1. 数据加载：页面加载时，前端发送请求到后端获取所有用户信息。
2. 后端响应：后端查询数据库，整理用户数据（如用户名、注册时间），并返回给前端。
3. 列表渲染：前端接收数据，动态生成用户列表，展示每个用户的基本信息，并为每个条目附加操作按钮（修改状态、删除）。
4. 搜索与筛选：
   1. 前端提供搜索框和筛选条件（如注册日期范围）。
   2. 用户输入关键字或选择筛选条件后，前端发送请求携带参数到后端。
   3. 后端根据参数过滤用户数据，返回过滤后的结果。
   4. 前端更新列表显示。
5. 操作处理：
   1. 用户点击操作按钮（如修改状态、删除），前端发送请求到后端，并附带用户ID和操作类型。
   2. 后端根据请求执行相应操作（修改数据库记录或删除用户）。
   3. 后端返回操作结果，前端根据结果提示用户（操作成功/失败）并刷新列表（如需要）

### 模块类图



### 模块数据流



## 企宣模块详细设计

### 企宣界面

1、广告管理界面：

显示当前所有的广告内容，包括标题、内容、图片/视频预览以及发布状态。

提供添加新广告或编辑现有广告的表单，包括标题、内容、图片/视频上传等字段。

允许管理员发布新广告或将现有广告下架，以便根据需要控制广告的显示状态。

2、企宣内容管理界面：

列出当前所有的企宣内容，包括标题、内容摘要、创建日期等信息。

提供添加新内容或编辑现有内容的表单，包括标题、内容、图片/视频上传等字段。

允许管理员发布新内容或将现有内容撤回，以便根据需要控制内容的展示状态。

### 模块内处理流程

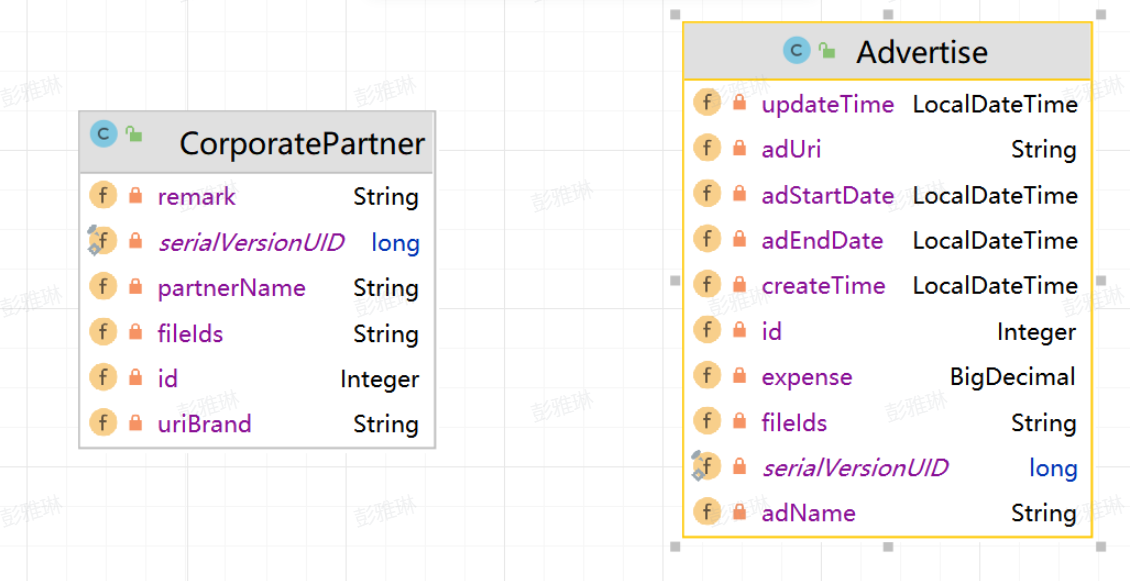
广告管理界面处理流程

1. 页面初始化：
   1. 加载时，前端向后端发送请求获取所有广告信息。
   2. 后端查询数据库，获取广告列表（标题、内容、图片/视频链接、发布状态），并返回给前端。
   3. 前端根据返回的数据渲染广告列表，每条广告包含预览、状态标记及操作按钮（编辑、上下架）。
2. 添加/编辑广告：
   1. 用户点击“添加新广告”或某广告的“编辑”按钮，弹出表单。
   2. 表单包含标题、内容输入框，以及图片/视频上传组件。
   3. 用户填写完毕后，提交表单至后端。
   4. 后端验证数据完整性及合法性，存储或更新数据库中的广告记录，并处理图片/视频上传（保存至服务器或云存储）。
3. 发布/下架广告：
   1. 用户点击广告条目旁的“发布”或“下架”按钮，前端发送请求到后端，包含广告ID和操作指令。
   2. 后端根据指令更新广告的发布状态，并反馈操作结果。
   3. 前端根据反馈更新列表中该广告的状态显示。

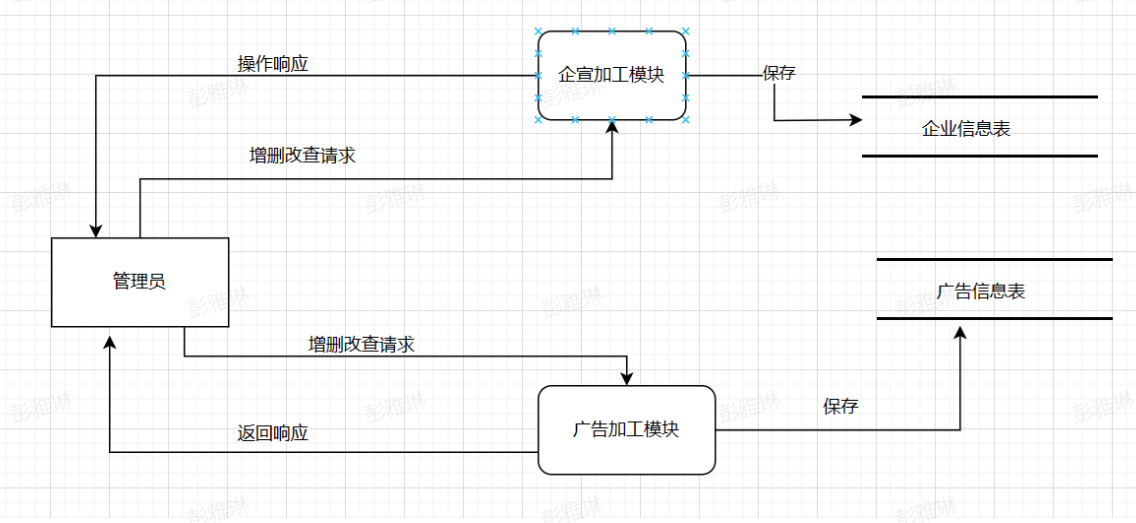
2、企宣内容管理界面处理流程

1. 内容列表加载：
   1. 页面加载时，前端请求后端获取企宣内容列表。
   2. 后端查询并返回内容列表（标题、内容摘要、创建日期等）。
   3. 前端渲染列表，每条内容显示关键信息及操作选项（编辑、发布/撤回）。
2. 添加/编辑内容：
   1. 用户选择“添加新内容”或点击某内容的“编辑”按钮，弹出编辑表单。
   2. 表单包括标题、内容输入区域及媒体上传选项。
   3. 用户提交表单，后端验证数据，保存或更新数据库记录，处理上传的图片/视频。
3. 发布/撤回内容：
   1. 用户选择某内容的“发布”或“撤回”操作。
   2. 前端发送请求，包含内容ID和操作类型。
   3. 后端根据请求更新内容的展示状态。
   4. 前端更新列表中内容的展示状态标志，反映最新状态

### 模块类图



### 模块数据流



## 商品模块详细设计

### 模块界面

### 模块内处理流程

#### 品牌相关界面

1. 更新品牌信息

根据传入的 BrandVO 对象更新品牌信息,包括品牌名称和图片ID。

更新完成后清理品牌缓存。

1. 创建新品牌

根据传入的 BrandVO 对象创建新的品牌。

创建完成后清理品牌缓存。

1. 删除品牌

根据传入的品牌ID删除品牌。

在删除前先检查该品牌下是否存在商品,如果存在则不允许删除。

删除成功后清理品牌缓存。

1. 分页查询品牌列表

首先尝试从缓存中获取分页数据,如果缓存中没有则从数据库查询。

查询数据库时,同时查询关联的用户和图片信息,并将结果封装为 BrandPageVO 对象。

查询结果缓存至 Redis。

1. 缓存管理

在更新、创建和删除品牌时清理整个品牌缓存。

在分页查询时,根据页码缓存对应页的数据。

#### 目录（分类模块）

1. 创建分类:

根据传入的 CategoryReqVO 对象创建一个新的 Category 实体,并保存到数据库。

如果传入了父级分类 ID,则将其设置为新分类的父级分类。

1. 查询分类路径:

通过分类 ID 递归查询分类的父级路径,并将分类名称添加到一个 LinkedList 中返回。

1. 删除分类:

检查是否有产品属于当前分类,如果有则抛出异常。

检查是否有子分类,如果有则抛出异常。

检查是否有优惠券与当前分类关联,如果有则抛出异常。

如果以上条件都满足,则删除该分类。

1. 排序分类:

根据传入的 OrderCategoryVO 列表,更新各个分类的序列号。

1. 更新分类:

检查 ID 是否存在,如果不存在则抛出异常。

检查父级分类 ID 是否存在,如果不存在则抛出异常。

将 CategoryReqVO 对象转换为 Category 实体并更新到数据库。

1. 查询分类列表:

首先查询出所有的顶级分类,并转换为 CategoryResVO 对象列表。

对每个顶级分类,递归查询其子分类,并将子分类添加到对应的 CategoryResVO 对象中。

最后,获取所有分类的创建人 ID,并通过 UserClient 查询获取创建人的用户名,设置到各个 CategoryResVO 对象中。

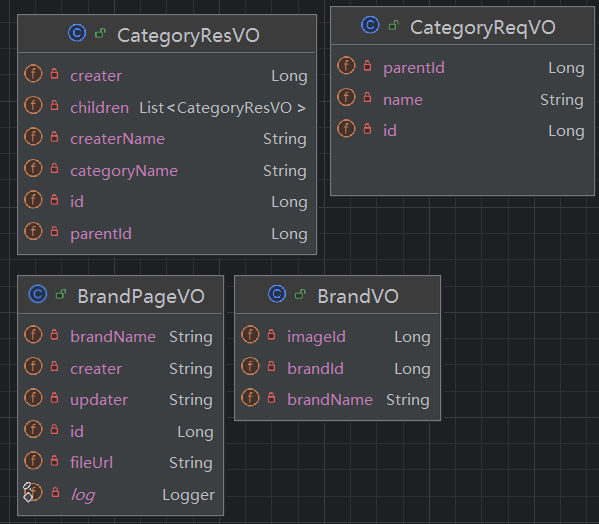
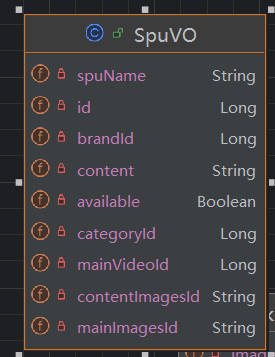
#### 标准商品（SPU）

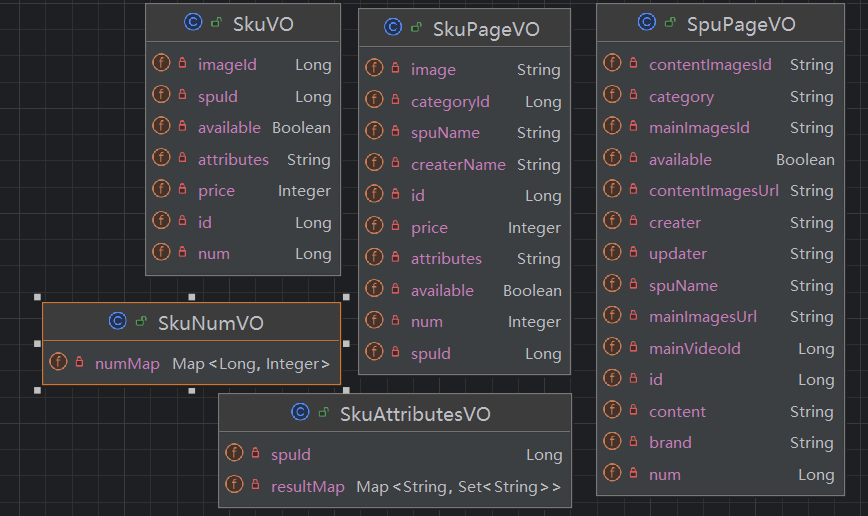
1. 创建SPU：创建SPU（Standard Product Unit）。将传入的VO对象进行处理，去除ID字段，然后保存到数据库。
2. 删除SPU方法：删除指定ID的SPU。首先查询与该SPU关联的SKU（Stock Keeping Unit）记录，并根据SKU的ID批量删除对应的记录。然后删除指定ID的SPU记录。
3. 更新SPU：更新SPU信息。如果传入的VO对象的ID为空或为0，则抛出异常。对传入的VO对象进行处理，去除多余字段后，根据ID更新对应的SPU记录。
4. 根据名字查询spu 方法：根据名称查询SPU列表。根据名称模糊查询符合条件的SPU记录，并进一步处理关联的图片和用户信息。根据图片ID查询图片URL，将处理后的SPU记录封装为VO对象列表返回。
5. 根据多个id查询SPU：根据ID列表查询SPU列表。根据ID批量查询符合条件的SPU记录，并封装为VO对象列表返回。
6. 分页查询SPU：分页查询SPU列表。根据查询条件进行查询，并根据查询结果处理关联的用户信息。将查询结果封装为分页对象返回。
7. 处理SPU的VO对象。将传入的VO对象中的图片ID进行处理，去除无效的ID，并判断类别和品牌是否存在。根据处理后的结果返回处理后的VO对象或null。

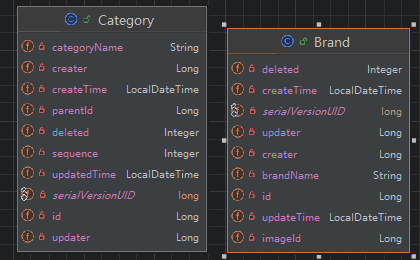
#### 商品库存（SKU）

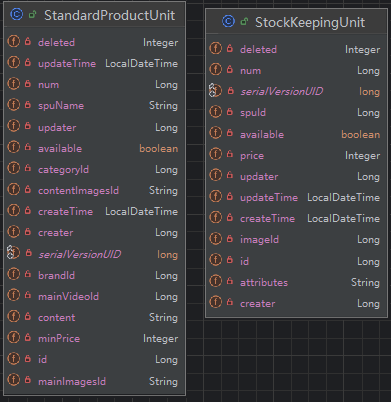
1. 创建SKU（库存单元）：在创建之前进行一些验证，如验证imageId的合法性和attributes是否为JSON格式。然后查询spuId对应的标准产品单元数量，如果为0则抛出异常。接着将vo转换为StockKeepingUnit对象并保存，然后根据条件更新相关数据。
2. 删除SKU：首先根据id查询对应的SKU，如果存在则进行一系列更新操作，包括更新spu的数量和最低价格，并删除该SKU。
3. 更新SKU：对传入的vo进行验证，包括判断id是否为空、imageId的合法性和attributes是否为JSON格式。然后根据id查询对应的SKU，如果存在则根据条件进行更新操作。
4. 根据SPU ID分页查询SKU：根据查询条件进行查询，并获取相关信息，如图片URL、创建者名称和SPU名称，封装成SkuPageVO对象并返回。
5. 获取SPU对应的属性集合：根据spuId查询相应的SKU，然后将每个SKU的属性解析为Map，并整合成一个属性集合。
6. 根据属性和SPU ID获取对应的SKU：根据传入的属性和spuId查询SKU，并将查询到的结果封装成SkuPageVO对象返回。
7. 根据SKU ID列表获取SKU信息：根据传入的SKU ID列表查询对应的SKU，并获取相关信息，如图片URL和SPU名称，封装成SkuPageVO对象列表返回。
8. 更新SKU库存数量：根据传入的SKU ID和数量的映射关系进行更新操作，先查询对应的SKU，然后根据条件更新相关数据。

### 模块类图







### 模块数据流

#### 品牌

更新品牌：

输入数据流向：接收一个 BrandVO 对象作为输入参数。

内部数据流向：从输入的 BrandVO 对象中提取属性。

输出数据流向：将提取的属性用于更新 Brand 对象，并返回更新结果。

删除品牌：

输入数据流向：接收一个 BrandVO 对象作为输入参数。

内部数据流向：从输入的 BrandVO 对象中提取属性。

输出数据流向：将提取的属性用于创建 Brand 对象，并返回创建结果。

删除品牌方法：

输入数据流向：接收一个 brandId 的 Long 值作为输入参数。

内部数据流向：根据 brandId 进行条件判断。

输出数据流向：根据条件判断的结果进行处理，并返回处理结果。

分页查询：

输入数据流向：接收一个 BrandQuery 对象作为输入参数。

内部数据流向：根据输入参数进行查询操作，获取符合条件的 Brand 对象列表。

输出数据流向：对查询结果进行处理，包括从缓存获取部分数据、获取相关联的用户信息和图像信息，并返回一个包含查询结果的分页数据对象。

#### 分类

创建分类：

输入数据流向：接收一个 CategoryReqVO 对象作为输入参数。

内部数据流向：从输入的 CategoryReqVO 对象中提取属性。

输出数据流向：将提取的属性用于创建 Category 对象，并返回创建结果。

查询：

输入数据流向：接收一个 id 的 Long 值和一个 LinkedList<String> 对象作为输入参数。

内部数据流向：根据 id 查询相应的 Category 对象，并将该对象的 categoryName 属性添加到 names 链表中。如果该 Category 对象有父级（parent），则递归调用该方法。

输出数据流向：返回填充了 categoryName 的 names 链表。

删除分类

输入数据流向：接收一个 id 的 Long 值作为输入参数。

内部数据流向：根据 id 进行条件判断，包括查询是否有子分类、是否有关联的SPU和优惠券。

输出数据流向：根据条件判断的结果进行处理，并返回处理结果。

排序分类：

输入数据流向：接收一个 OrderCategoryVO 对象列表作为输入参数。

内部数据流向：根据输入参数的 id 查询相应的 Category 对象，并根据 OrderCategoryVO 对象的 sequence 属性更新 Category 对象的 sequence 属性。

输出数据流向：批量更新 Category 对象的 sequence 属性，并返回更新结果。

更新：

输入数据流向：接收一个 CategoryReqVO 对象作为输入参数。

内部数据流向：根据输入参数的 id 进行条件判断，包括验证 id 是否存在以及验证 parentId 是否存在。

输出数据流向：根据条件判断的结果进行处理，并返回处理结果。

查询所有：

输入数据流向：无输入参数。

内部数据流向：查询所有的 Category 对象，并获取关联的用户信息。

输出数据流向：对查询结果进行处理，包括设置用户名称，并返回一个包含查询结果的 CategoryResVO 对象列表。

#### 标准商品

createSpu(SpuVO vo) 方法：

输入数据流向：接收一个 SpuVO 对象作为输入参数。

内部数据流向：对输入参数进行处理，将其 id 设置为 null，并对 SpuVO 对象进行修剪（trimSpuVO）。

输出数据流向：将修剪后的 SpuVO 对象转换为 StandardProductUnit 对象，并调用 save 方法进行保存，返回保存结果。

removeSpu(Long id) 方法：

输入数据流向：接收一个 id 的 Long 值作为输入参数。

内部数据流向：根据 id 查询与之关联的 StockKeepingUnit 对象，并将其 id 收集为一个列表，然后批量删除这些 StockKeepingUnit 对象。

输出数据流向：根据 id 删除与之关联的 StandardProductUnit 对象，并返回删除结果。

updateSpu(SpuVO vo) 方法：

输入数据流向：接收一个 SpuVO 对象作为输入参数。

内部数据流向：对输入参数进行处理，包括验证 id 是否为空以及修剪（trimSpuVO）SpuVO 对象。

输出数据流向：将修剪后的 SpuVO 对象转换为 StandardProductUnit 对象，并调用 updateById 方法进行更新，返回更新结果。

queryByName(String name) 方法：

输入数据流向：接收一个 name 的字符串作为输入参数。

内部数据流向：根据 name 进行模糊查询，获取与之匹配的 StandardProductUnit 对象列表。根据查询结果获取关联的用户信息、图片信息，并进行处理和转换，生成 SpuPageVO 对象列表。

输出数据流向：返回处理后的 SpuPageVO 对象列表。

queryById(List<Long> ids) 方法：

输入数据流向：接收一个 id 的 Long 列表作为输入参数。

内部数据流向：根据 id 列表批量查询与之匹配的 StandardProductUnit 对象列表，并将其转换为 SpuPageVO 对象列表。

输出数据流向：返回 SpuPageVO 对象列表。

queryByPage(SpuQuery query) 方法：

输入数据流向：接收一个 SpuQuery 对象作为输入参数。

内部数据流向：根据 SpuQuery 对象的条件构建查询条件，执行查询操作，获取分页结果。同时，获取关联的用户信息，并进行处理和转换，生成 SpuPageVO 对象列表。

输出数据流向：返回分页结果 PageDTO<SpuPageVO> 对象。

#### 库存单位

public boolean createSku(SkuVO vo)

输入数据：SkuVO对象（包含创建SKU所需的信息）

输出数据：布尔值，表示创建SKU的成功与否

内部数据流向：根据输入的SkuVO对象，进行一系列验证和处理操作，可能包括验证属性、生成SKU ID、设置默认值等。最后将相关数据保存到数据库中。

public boolean removeSku(Long id)

输入数据：Long类型的SKU ID

输出数据：布尔值，表示删除SKU的成功与否

内部数据流向：根据输入的SKU ID查询数据库中对应的SKU数据，并执行一系列删除和更新操作，可能包括删除关联数据、更新其他相关数据等。最后从数据库中删除该SKU数据。

public boolean updateSku(SkuVO vo)

输入数据：SkuVO对象（包含更新SKU所需的信息）

输出数据：布尔值，表示更新SKU的成功与否

内部数据流向：根据输入的SKU ID查询数据库中对应的SKU数据，并根据SkuVO对象中的参数进行更新操作，可能包括验证、更新数量、更新价格等。最后保存更新后的数据到数据库中。

public PageDTO<SkuPageVO> queryPageBySpuId(SkuQuery query)

输入数据：SkuQuery对象（包含查询条件）

输出数据：PageDTO<SkuPageVO>对象，表示根据查询条件获取的分页SKU数据

内部数据流向：根据SkuQuery对象中的查询条件，从数据库中获取满足条件的SKU数据。然后进行一些处理和转换操作，可能包括数据排序、分页处理等。最后将处理后的数据封装到PageDTO<SkuPageVO>对象中并返回。

public Map<String, Set<String>> getAttributes(Long spuId)

输入数据：Long类型的spuId

输出数据：Map<String, Set<String>>对象，表示根据spuId获取到的SKU属性信息

内部数据流向：根据spuId查询数据库中对应的SKU数据，然后将属性信息进行整理并存储到Map<String, Set<String>>对象中，并返回该对象。

public SkuPageVO getSkuByAttributes(String attributes, Long spuId)

输入数据：String类型的属性信息和Long类型的spuId

输出数据：SkuPageVO对象，表示根据属性信息和spuId获取到的符合条件的SKU数据

内部数据流向：根据传入的属性信息和spuId查询数据库中对应的SKU数据，并进行一些处理和转换操作，可能包括解析属性信息、匹配属性值等。最后将符合条件的SKU数据封装到SkuPageVO对象中并返回。

public List<SkuPageVO> getSkuById(List<Long> skuID)

输入数据：List<Long>类型的SKU ID列表

输出数据：List<SkuPageVO>对象，表示根据SKU ID列表获取到的SKU数据列表

内部数据流向：根据传入的SKU ID列表查询数据库中对应的SKU数据，并进行一些处理和转换操作，最后将获取到的SKU数据列表返回。

public void updateSkuNum(Map<Long,Integer> numMap)

输入数据：Map<Long,Integer>类型的SKU ID和数量信息

输出数据：无返回值，但会批量更新SKU的数量

内部数据流向：根据传入的SKU ID和数量信息，查询数据库中对应的SKU数据并进行更新操作，最后保存更新后的数据到数据库中。

## 交易模块详细设计

### 模块界面

### 模块内处理流程

#### 购物车

create(ShoppingChartVO vo)方法：

根据传入的vo对象，设置其id为null。

使用lambdaQuery方法查询符合条件的ShoppingChart对象。

如果查询结果不为空，则将vo对象的数量与查询结果的数量相加，并将查询结果的id赋值给vo对象的id，然后调用update方法进行更新。

如果查询结果为空，则调用save方法保存vo对象。

update(ShoppingChartVO vo)方法：

检查vo对象的id是否为空，如果为空则抛出异常。

调用updateById方法根据vo对象的id更新对应的ShoppingChart对象。

getShoppingChart(ShoppingChartVO vo)方法：

调用productClient的getSkuById方法，通过传入的skuId获取SkuPageVO对象列表。

如果获取到的列表为空，则抛出异常，表示sku id无效。

从列表中获取第一个SkuPageVO对象，并检查其数量是否小于传入的vo对象的数量，如果小于则抛出异常，表示数量无效。

使用BeanUtils.copyBean方法将vo对象转换为ShoppingChart对象，并设置userId、spuId和price等属性。

remove(List<Long> ids)方法：

使用lambdaQuery方法查询符合条件的ShoppingChart对象列表。

将查询结果的id列表与传入的ids列表取交集，保留在ids列表中存在的id。

调用removeByIds方法根据保留的id列表批量删除ShoppingChart对象。

pageQuery(ShoppingChartQuery query)方法：

使用lambdaQuery方法查询符合条件的ShoppingChart对象分页结果。

获取查询结果的记录列表。

如果记录列表不为空，则进行以下操作：

提取记录列表中的skuId集合。

调用productClient的getSkuById方法，通过skuId集合获取对应的SkuPageVO对象列表，并转换为以id为键的Map对象。

遍历记录列表，将每个ShoppingChart对象转换为ShoppingChartDTO对象，并设置其相关属性，如图片、商品名称、属性、是否可用等。

使用PageDTO.of方法将查询结果封装为PageDTO对象并返回。

#### 地址

create(AddressVO vo)方法：

获取当前用户的userId。

将vo对象的id设置为null。

使用isMobileNum方法验证vo对象的phone字段是否为有效的手机号码。

使用BeanUtils.copyBean方法将vo对象转换为Address对象，并设置userId属性。

调用save方法保存Address对象。

update(AddressVO vo)方法：

检查vo对象的id是否为空，如果为空则抛出异常。

使用isMobileNum方法验证vo对象的phone字段是否为有效的手机号码。

获取当前用户的userId。

使用BeanUtils.copyBean方法将vo对象转换为Address对象，并设置userId属性。

调用updateById方法根据vo对象的id更新对应的Address对象。

remove(List<Long> ids)方法：

使用baseMapper的selectBatchIds方法根据ids列表查询对应的Address对象列表。

遍历Address对象列表，如果地址对象不为空且其userId不等于当前用户的userId，则抛出异常，表示无权限操作该地址。

调用removeByIds方法根据ids列表批量删除Address对象。

pageQueryAddress(AddressQuery query)方法：

使用lambdaQuery方法查询当前用户的地址对象的分页结果。

获取查询结果的记录列表。

如果记录列表不为空，则进行以下操作：

将每个Address对象转换为AddressVO对象，并添加到AddressVO列表中。

使用PageDTO.of方法将查询结果封装为PageDTO对象并返回。

#### 订单

预览订单：  
preViewFromChart(List<Long> shoppingCharts)方法：

检查shoppingCharts列表是否为空，如果为空则抛出异常。

创建一个PreviewOrderDTO对象作为返回结果。

使用chartMapper的selectList方法查询当前用户在购物车中指定id的ShoppingChart对象列表。

如果查询结果列表不为空，则进行以下操作：

调用productClient的getSkuById方法，通过查询结果中的skuId集合获取对应的SkuPageVO对象列表。

调用validateSkuPageVO方法验证SkuPageVO列表的有效性。

将shoppingCharts列表中的元素转换为字符串，并使用逗号连接生成chartsKey。

使用redisTemplate将skuById对象转换为JSON格式字符串，并存储在Redis中，设置有效期为RedisConstants.DURATION\_MINUTES分钟。

如果skuById列表为空，则抛出异常。

将skuById列表中的每个元素转换为OrderProductDTO对象，并设置相关属性，如id、price、cateId等，并添加到orderProducts列表中。

调用promotionClient的findDiscountSolution方法，通过orderProducts列表获取对应的优惠券信息列表。

对discounts列表进行处理，去除优惠券的具体折扣详情。

计算skuById列表中商品总价的和，并设置res对象的finalPrice和rawPrice属性为该总价。

如果discounts列表不为空，则进行以下操作：

遍历discounts列表，找到折扣金额最大的优惠券，记录其折扣金额和对应的规则，并将其id添加到res对象的coupons属性中。

将discounts列表中的每个元素的折扣金额置为空。

设置res对象的discounts属性为discounts列表，finalPrice属性为总价减去最大折扣金额，discountAmount属性为最大折扣金额，rules属性为对应规则。

设置res对象的shoppingCharts属性为传入的shoppingCharts列表。

返回res对象作为预览订单的结果。

下单：  
使用addressMapper根据vo中的addressId查询对应的Address对象。

如果查询结果为空，则抛出异常。

将address对象的id设置为null，创建一个Orders对象。

将address对象转换为JSON格式字符串，并设置到orders对象的address属性中。

将当前用户的ID设置为orders对象的userId属性。

使用chartMapper查询当前用户在购物车中指定id的ShoppingChart对象列表。

如果查询结果为空，则抛出异常。

将vo中的shoppingCharts列表中的元素转换为字符串，并使用逗号连接生成chartsKey。

从redis中根据用户ID和chartsKey获取skuList（购物车中商品的信息列表）。

如果skuList不为空，则将其转换为List<SkuPageVO>类型的对象列表。

否则，调用productClient的getSkuById方法，通过shoppingList中的skuId集合获取对应的SkuPageVO对象列表。

调用validateSkuPageVO方法验证SkuPageVO列表的有效性。

将skuVOs列表中的每个元素转换为OrderProductDTO对象，并设置相关属性，如id、price、cateId等，并添加到orderProducts列表中。

计算shoppingList中每个购物车商品的数量和价格的乘积的和，并设置orders对象的rawPrice和finalPrice属性为该和。

如果vo中的coupons列表不为空，则进行以下操作：

调用promotionClient的queryDiscountDetailByOrder方法，通过OrderCouponDTO对象（包含coupons和orderProducts）获取优惠券的折扣信息。

如果discountDTO不为空，则将orders对象的finalPrice属性设置为原总价减去优惠券的折扣金额，并将优惠券的折扣详情存储在map中。

如果保存orders对象失败，则抛出异常。

对skuVOs列表并行处理，将每个元素转换为OrderDetails对象，并设置相关属性，如id、orderId、skuId等。

如果finalMap不为空，则根据finalMap更新orderDetails对象的finalPrice属性。

将处理后的orderDetails对象列表保存到数据库中。

如果保存orderDetails对象失败，则抛出异常。

根据orderDetails列表生成一个包含商品ID和数量的列表。

如果vo中的coupons列表不为空，则调用promotionClient的writeOffCoupon方法核销优惠券。

删除购物车中对应的购物车项。

根据商品ID和数量列表更新商品的库存数量。

返回true，表示订单创建成功。

发货：

delivery(Long orderId)方法：

调用orderMapper的deliveryOrder方法，将orderId作为参数传入。

如果返回结果为1，则返回true，表示订单发货成功；否则返回false。

收货：

finishOrder(Long orderId)方法：

调用orderMapper的updateOrderStatusByUser方法，将订单状态设置为成功（OrdersStatus.SUCCESS.getValue()），orderId和当前用户的ID作为参数传入。

如果返回结果为1，则返回true，表示订单状态更新成功；否则返回false。

删除订单：

deleteOrder(Long orderId)方法：

使用orderDetailsMapper根据orderId查询对应的OrderDetails对象列表，并提取其id组成一个列表。

调用orderDetailsService的removeByIds方法，根据detailsIDs列表批量删除OrderDetails对象。

调用baseMapper的delete方法，根据条件删除Orders对象，条件为orderId等于指定值且订单状态为关闭或成功。

如果返回结果为1，则返回true，表示订单删除成功；否则返回false。

取消订单：

canceledOrder(Long orderId)方法：

使用lambdaQuery根据orderId和deliveryStatus等于0查询Orders对象。

如果查询结果为空，则抛出异常。

使用orderDetailsMapper根据orderId和refundStatus为空的条件查询对应的OrderDetails对象列表。

计算所有OrderDetails对象的数量和最终价格的乘积之和，并将结果存储在reduce变量中。

使用orderDetailsService的updateBatchById方法批量更新OrderDetails对象的canceled属性为true。

调用orderMapper的updateOrderStatusByUser方法，将订单状态设置为关闭（OrdersStatus.CLOSED.getValue()），orderId和当前用户的ID作为参数传入。

根据OrderDetails对象列表生成包含商品ID和数量的列表。

调用productClient的updateSkuNum方法，更新商品的库存数量。

返回true，表示订单取消成功。

分页查询订单：

pageQuery(OrderQuery query)方法：

创建LambdaQueryChainWrapper对象wrapper，设置查询条件为userId等于当前用户的ID。

如果query中的spuName不为空，则调用orderDetailsMapper的selectOrdersBySpuName方法，根据spuName查询对应的订单ID列表，然后在wrapper中设置条件为订单ID在该列表中。

使用wrapper.page方法，根据query中的分页信息执行查询操作，获取Page对象page。

从page中获取查询结果列表records。

如果records不为空，则执行以下操作：

提取records中每个Orders对象的ID，生成一个ID列表idList。

使用orderDetailsMapper根据orderId在idList中的值进行查询，获取OrderDetails对象列表orderDetails。

创建一个HashMap对象dtoMap，用于存储订单ID和对应的OrderDetailsDTO列表。

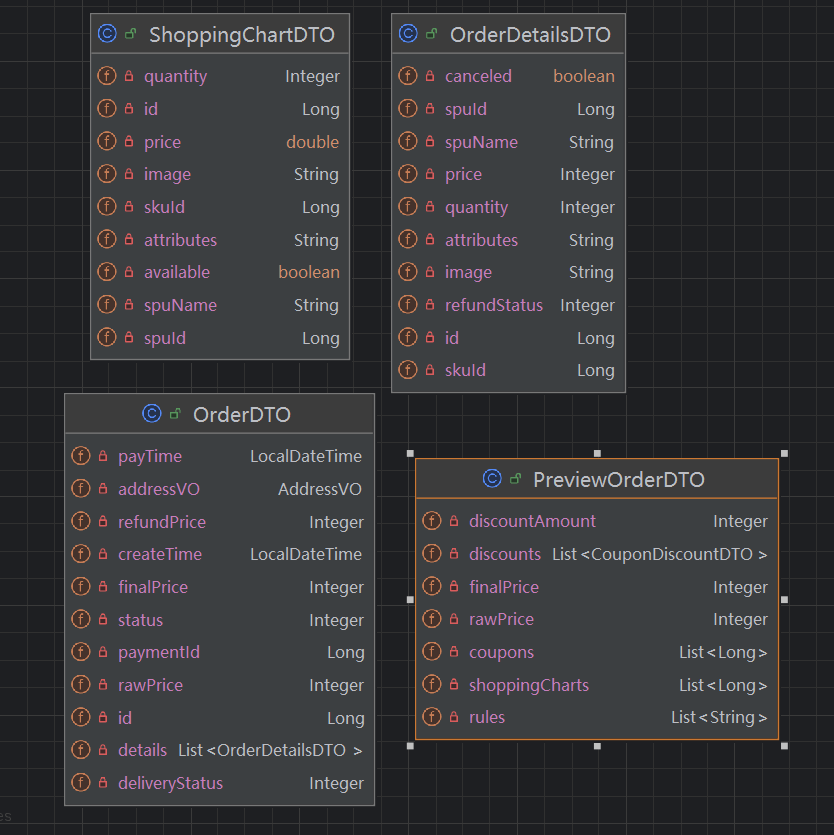
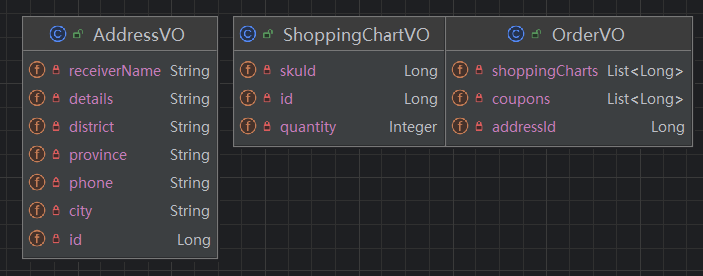
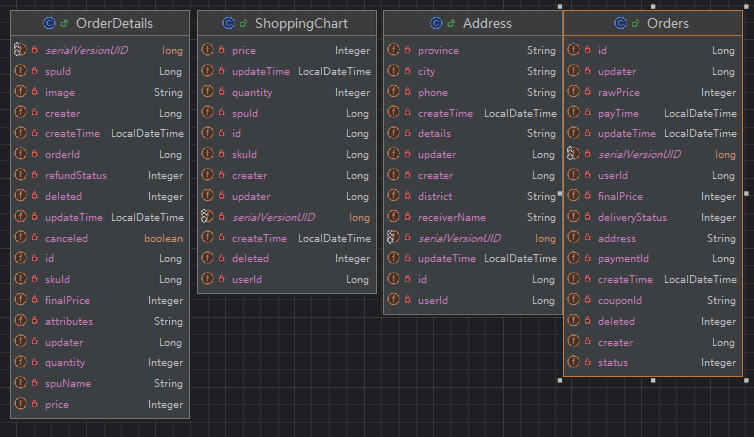
遍历orderDetails列表，根据订单ID将OrderDetails对象转换为OrderDetailsDTO对象，并将其添加到dtoMap中对应的列表中。

使用records列表中的每个Orders对象和相应的OrderDetailsDTO列表，以及Address字段的解析结果构造OrderDTO对象。

将构造的OrderDTO对象列表collect和Page对象page作为参数，创建一个PageDTO对象res。

返回res，表示订单查询结果。

### 模块类图



### 模块数据流

#### 购物车

创建地址：

输入数据：AddressVO对象（包含创建地址所需的信息）。

输出数据：布尔值，表示创建地址的成功与否。

内部数据流向：获取当前用户的ID（userId）；将AddressVO对象的ID设置为null；验证手机号是否合法；将AddressVO对象转换为Address对象；将userId设置为Address对象的userId；保存Address对象到数据库中。

更新地址：

输入数据：AddressVO对象（包含更新地址所需的信息）。

输出数据：布尔值，表示更新地址的成功与否。

内部数据流向：验证AddressVO对象的ID是否为空；验证手机号是否合法；获取当前用户的ID（userId）；将AddressVO对象转换为Address对象；将userId设置为Address对象的userId；根据ID更新数据库中对应的Address记录。

删除地址：

输入数据：List<Long>类型的地址ID列表。

输出数据：布尔值，表示删除地址的成功与否。

内部数据流向：根据地址ID列表从数据库中查询对应的Address记录；遍历查询结果，检查每个Address记录是否存在且是否属于当前用户；如果存在不属于当前用户的地址，则抛出异常；根据ID列表删除数据库中的Address记录。

分页查询地址：

输入数据：AddressQuery对象（包含地址查询条件）。

输出数据：PageDTO<AddressVO>对象，表示分页查询的结果。

内部数据流向：根据当前用户的ID（userId）和查询条件，在数据库中执行分页查询操作；将查询结果转换为AddressVO对象列表；将查询结果和分页信息封装到PageDTO<AddressVO>对象中，并返回。

验证手机号是否合法：

输入数据：手机号码（String类型）。

输出数据：布尔值，表示手机号的合法性。

内部数据流向：使用正则表达式对手机号进行匹配验证，返回验证结果。

#### 购物车

创建购物车项：

输入数据：ShoppingChartVO对象（包含创建购物车项所需的信息）。

输出数据：布尔值，表示创建购物车项的成功与否。

内部数据流向：将ShoppingChartVO对象的ID设置为null；根据当前用户ID和商品SKU ID，在数据库中执行查询操作，获取购物车项；如果购物车项已存在，则更新购物车项的数量和ID，并调用更新方法；否则，将ShoppingChartVO对象转换为ShoppingChart对象，并保存到数据库中。

更新购物车项：

输入数据：ShoppingChartVO对象（包含更新购物车项所需的信息）。

输出数据：布尔值，表示更新购物车项的成功与否。

内部数据流向：验证ShoppingChartVO对象的ID是否为空；根据ShoppingChartVO对象获取对应的ShoppingChart对象，并调用更新方法。

获取购物车项：

输入数据：ShoppingChartVO对象（包含获取购物车项所需的信息）。

输出数据：ShoppingChart对象，表示购物车项。

内部数据流向：根据ShoppingChartVO对象的SKU ID，调用productClient的getSkuById方法获取商品信息；如果商品信息为空，则抛出异常；否则，根据ShoppingChartVO对象的信息创建ShoppingChart对象，并设置用户ID、SPU ID和价格，然后返回。

移除购物车项：

输入数据：List<Long>类型的购物车项ID列表。

输出数据：布尔值，表示移除购物车项的成功与否。

内部数据流向：根据当前用户ID，在数据库中查询购物车项列表；筛选出与购物车项ID列表相匹配的购物车项；调用removeByIds方法从数据库中删除购物车项。

分页查询购物车项：

输入数据：ShoppingChartQuery对象（包含购物车项查询条件）。

输出数据：PageDTO<ShoppingChartDTO>对象，表示分页查询的结果。

内部数据流向：根据当前用户ID和查询条件，在数据库中执行分页查询操作；获取查询结果的购物车项列表；根据购物车项列表中的SKU ID集合，调用productClient的getSkuById方法获取商品信息并转换为Map结构；将购物车项列表和商品信息进行关联，构建ShoppingChartDTO对象列表；返回分页查询结果。

#### 订单模块

预览：  
输入数据：shoppingCharts（列表）

内部操作：

判断shoppingCharts是否为空

创建res对象作为结果

使用chartMapper根据用户ID和shoppingCharts查询数据库，获取list

通过productClient调用getSkuById方法，传入list中的SkuId集合，获取skuById

对skuById进行验证

将shoppingCharts转换为字符串形式，并拼接为chartsKey

使用redisTemplate将skuById转换为JSON字符串并存入Redis中

对skuById进行判空操作

将skuById转换为orderProducts列表，进行一系列属性设置

通过promotionClient调用findDiscountSolution方法，传入orderProducts，获取discounts

计算skuById中商品的总价值，赋值给sum

对discounts进行一系列操作，包括筛选出最大折扣金额的方案，设置res中的相关属性

设置res中的最终价格、原始价格、折扣金额、规则和购物车信息

输出数据：res（PreviewOrderDTO对象）

下单：  
输入数据：OrderVO对象（vo）

输出数据：布尔值（true表示订单创建成功）

内部数据流向概述：

查询并获取地址对象（address）

判断地址对象是否为空

对地址对象进行处理

创建订单对象（orders）

设置订单对象的地址和用户ID

查询购物车列表（shoppingList）

判断购物车列表是否为空

处理购物车列表相关操作

获取Redis中的数据

解析获取的数据为SKUVO对象列表（skuVOs）

验证SKUVO对象列表

处理SKUVO对象列表相关操作

计算价格和设置订单对象

查询折扣详情并设置订单对象

保存订单对象

处理订单详情列表（details）

保存订单详情列表

处理其他相关操作，如更新库存、核销优惠券等

方法名：delivery(Long orderId)

输入数据：订单ID (orderId)

输出数据：boolean值，表示订单是否成功交付

数据流向概述：

调用orderMapper的deliveryOrder(orderId)方法，传入订单ID。

返回deliveryOrder方法的返回值作为输出。

方法名：finishOrder(Long orderId)

输入数据：订单ID (orderId)

输出数据：boolean值，表示订单是否成功完成

数据流向概述：

调用orderMapper的updateOrderStatusByUser()方法，传入订单状态为已完成、订单ID和用户信息。

返回updateOrderStatusByUser方法的返回值作为输出。

方法名：deleteOrder(Long orderId)

输入数据：订单ID (orderId)

输出数据：boolean值，表示订单是否成功删除

数据流向概述

查询并获取订单详情ID列表。

调用orderDetailsService的removeByIds()方法，删除订单详情。

调用baseMapper的delete()方法，删除订单。

返回delete方法的返回值作为输出。

方法名：canceledOrder(Long orderId)

输入数据：订单ID (orderId)

输出数据：boolean值，表示订单是否成功取消

数据流向概述：

通过lambdaQuery查询订单信息。

如果订单信息为空，抛出异常。

查询并获取订单详情列表。

计算退款金额。

更新订单详情为已取消状态。

更新订单状态为已关闭状态。

创建IdAndNumDTO列表，并更新商品库存。

返回true作为输出。

方法名：pageQuery(OrderQuery query)

输入数据：订单查询参数 (OrderQuery对象)

输出数据：PageDTO<OrderDTO>对象

数据流向概述：

根据用户ID从lambdaQuery查询订单列表。

如果查询参数中包含商品名称，通过orderDetailsMapper查询满足条件的订单ID列表。

分页查询订单列表。

获取查询结果的记录列表。

查询订单详情列表，并将其转换为订单详情DTO。

将订单信息和订单详情DTO组合成OrderDTO对象。

将OrderDTO对象添加到列表中。

返回包含分页信息和订单DTO列表的PageDTO对象作为输出

## 优惠券模块详细设计

### 模块界面

 **优惠券管理界面**：

* 列表展示所有优惠券
* 添加新优惠券按钮
* 编辑、删除、启用/禁用优惠券的操作按钮

 **新增/编辑优惠券界面**：

* 输入优惠券名称
* 设置折扣金额或折扣比例
* 选择有效期
* 选择适用范围（如全场、某类商品）

 **用户优惠券查看界面**：

* 列表展示用户账户中的优惠券
* 优惠券的名称、折扣信息、有效期等详情

 **优惠券查询界面**：

* 输入关键词、折扣范围等筛选条件
* 查询结果展示

### 模块内处理流程

 **新增优惠券流程**：

1. 管理员点击“新增优惠券”按钮
2. 填写优惠券信息
3. 提交表单，后端验证数据
4. 数据库插入新的优惠券记录
5. 返回优惠券管理界面并刷新列表

 **编辑优惠券流程**：

1. 管理员在优惠券列表中选择一个优惠券并点击“编辑”
2. 修改优惠券信息
3. 提交表单，后端验证数据
4. 数据库更新相应优惠券记录
5. 返回优惠券管理界面并刷新列表

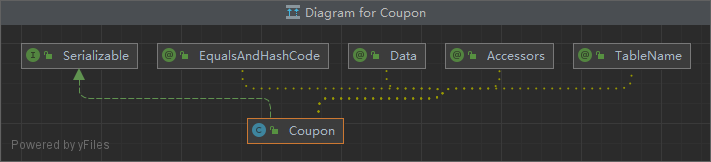
 **删除优惠券流程**：

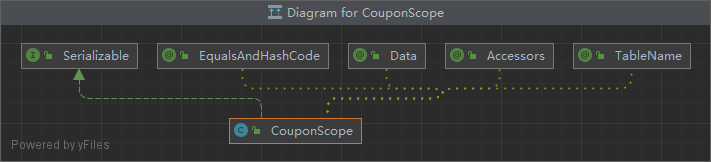
1. 管理员在优惠券列表中选择一个优惠券并点击“删除”
2. 确认删除操作
3. 数据库删除相应优惠券记录
4. 返回优惠券管理界面并刷新列表

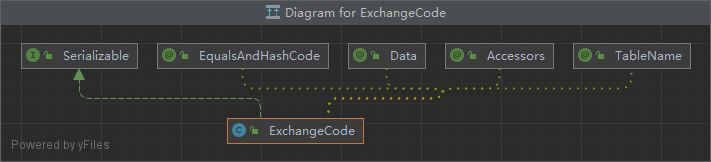
 **用户查看优惠券流程**：

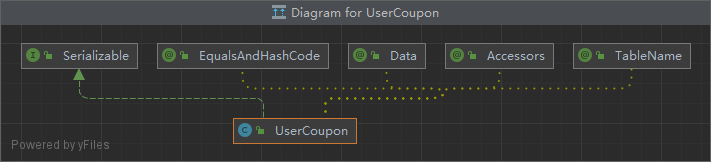
1. 用户登录并访问“我的优惠券”界面
2. 前端请求后端获取用户的优惠券列表
3. 数据库查询用户的优惠券记录
4. 返回优惠券数据并在界面展示

### 模块类图









### 模块数据流

 **新增优惠券**：

* 用户输入优惠券信息并提交
* 前端发送POST请求到后端
* 后端验证数据并插入数据库
* 返回成功/失败响应

 **编辑优惠券**：

* 用户修改优惠券信息并提交
* 前端发送PUT请求到后端
* 后端验证数据并更新数据库
* 返回成功/失败响应

 **删除优惠券**：

* 用户点击删除按钮并确认
* 前端发送DELETE请求到后端
* 后端删除数据库记录
* 返回成功/失败响应

 **查看用户优惠券**：

* 用户访问“我的优惠券”页面
* 前端发送GET请求到后端
* 后端查询数据库并返回用户的优惠券列表
* 前端展示优惠券信息

## 评论模块详细设计

### 模块界面

 **商品详情页面**：

* 显示商品的详细信息
* 显示用户评论列表
* 提交评论的输入框和提交按钮

 **评论管理界面（后台）**：

* 显示所有商品的评论列表
* 筛选和搜索评论
* 审核、删除评论的操作按钮

 **用户个人中心评论界面**：

* 显示用户发表的所有评论
* 编辑、删除评论的操作按钮

### 模块内处理流程

 **提交评论流程**：

1. 用户在商品详情页面输入评论内容
2. 点击“提交评论”按钮
3. 前端发送评论数据到后端
4. 后端验证数据并插入数据库
5. 返回成功/失败响应并刷新评论列表

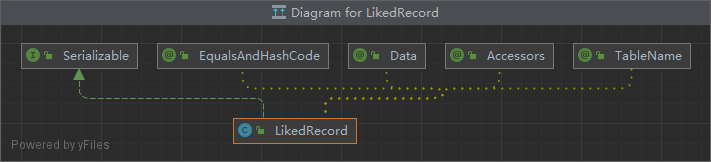
 **审核评论流程（后台）**：

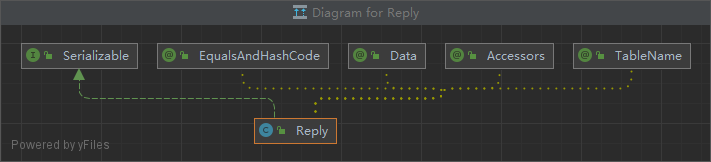
1. 管理员在评论管理界面选择待审核评论
2. 点击“审核通过”或“审核不通过”按钮
3. 后端更新评论状态
4. 返回成功/失败响应并刷新评论列表

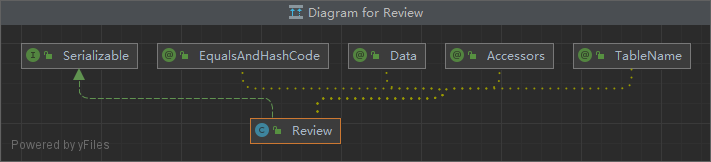
 **删除评论流程**：

1. 用户或管理员在评论列表中选择评论并点击“删除”
2. 确认删除操作
3. 后端删除数据库记录
4. 返回成功/失败响应并刷新评论列表

### 模块类图







### 模块数据流

 **提交评论**：

* 用户输入评论并提交
* 前端发送POST请求到后端
* 后端验证数据并插入数据库
* 返回成功/失败响应并更新前端界面

 **审核评论**：

* 管理员选择待审核评论并操作
* 前端发送PUT请求到后端
* 后端更新评论状态
* 返回成功/失败响应并更新前端界面

 **删除评论**：

* 用户或管理员选择评论并操作
* 前端发送DELETE请求到后端
* 后端删除数据库记录
* 返回成功/失败响应并更新前端界面

## 媒资模块详细设计

### 模块界面

 **商品详情页面**：

* 显示商品的详细信息，包括图片、视频等媒资内容
* 提供媒资的浏览和放大查看功能

 **媒资管理界面（后台）**：

* 显示所有商品的媒资列表
* 提供上传、删除、编辑媒资的操作按钮

 **用户上传媒资界面**：

* 用户上传商品评论图片或视频的界面
* 提供上传文件的输入框和提交按钮

### 模块内处理流程

 **上传媒资流程**：

1. 用户或管理员在上传媒资界面选择文件
2. 点击“上传”按钮
3. 前端将文件发送到后端
4. 后端保存文件并记录媒资信息到数据库
5. 返回成功/失败响应并刷新媒资列表

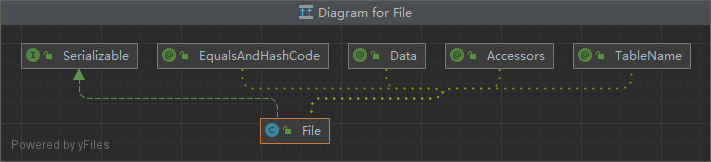
 **删除媒资流程**：

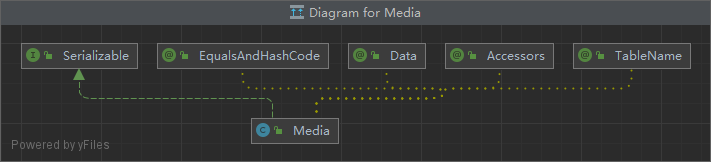
1. 管理员在媒资管理界面选择媒资并点击“删除”
2. 确认删除操作
3. 后端删除文件和数据库记录
4. 返回成功/失败响应并刷新媒资列表

 **编辑媒资信息流程**：

1. 管理员在媒资管理界面选择媒资并点击“编辑”
2. 修改媒资信息
3. 前端将修改后的数据发送到后端
4. 后端更新数据库记录
5. 返回成功/失败响应并刷新媒资列表

### 模块类图





### 模块数据流

 **上传媒资**：

* 用户或管理员选择文件并上传
* 前端发送POST请求到后端
* 后端保存文件并插入媒资记录到数据库
* 返回成功/失败响应并更新前端界面

 **删除媒资**：

* 管理员选择媒资并删除
* 前端发送DELETE请求到后端
* 后端删除文件和数据库记录
* 返回成功/失败响应并更新前端界面

 **编辑媒资信息**：

* 管理员选择媒资并编辑信息
* 前端发送PUT请求到后端
* 后端更新数据库记录
* 返回成功/失败响应并更新前端界面

## 退款模块详细设计

### 模块界面

 **用户退款申请页面**：

* 用户填写退款申请表单，包括订单号、退款理由、退款金额等信息
* 提交按钮

 **管理员退款管理界面**：

* 显示所有退款申请的列表
* 每个申请显示订单号、用户信息、申请日期、退款金额、申请状态等
* 管理员可以选择审批或拒绝退款申请
* 详细信息查看按钮，显示具体的退款申请详情

 **退款详情页面**：

* 显示选定退款申请的详细信息，包括订单号、用户信息、申请日期、退款理由、退款金额、支付状态等
* 审批按钮和拒绝按钮

### 模块内处理流程

 **用户提交退款申请流程**：

1. 用户在退款申请页面填写退款申请表单
2. 点击“提交”按钮
3. 前端将退款申请数据发送到后端
4. 后端保存退款申请记录到数据库
5. 返回成功/失败响应，用户界面显示提交成功/失败信息

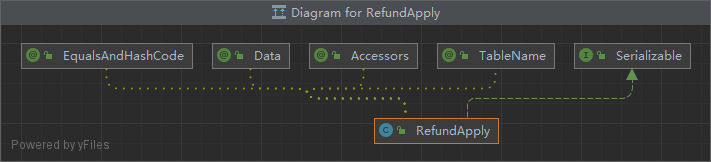
 **管理员审批退款申请流程**：

1. 管理员在退款管理界面选择退款申请并点击“查看详情”
2. 在退款详情页面，管理员查看退款申请的详细信息
3. 点击“审批”或“拒绝”按钮
4. 前端将操作请求发送到后端
5. 后端更新退款申请状态并处理相关业务逻辑
6. 返回成功/失败响应，刷新管理员界面

 **管理员查看退款详情流程**：

1. 管理员在退款管理界面选择退款申请并点击“查看详情”
2. 前端请求退款详情数据
3. 后端返回退款详情数据
4. 前端显示退款申请的详细信息

### 模块类图



### 模块数据流

 **用户提交退款申请**：

* 用户填写退款申请表单
* 前端发送POST请求到后端
* 后端保存退款申请记录到数据库
* 返回成功/失败响应

 **管理员审批退款申请**：

* 管理员选择退款申请并查看详情
* 前端发送GET请求获取退款详情数据
* 后端返回退款详情数据
* 管理员审批或拒绝退款申请
* 前端发送PUT请求更新退款申请状态
* 后端更新数据库记录
* 返回成功/失败响应

 **管理员查看退款详情**：

* 管理员选择退款申请并查看详情
* 前端发送GET请求获取退款详情数据
* 后端返回退款详情数据
* 前端显示退款申请的详细信息

# 数据库设计

## 数据库设计综述

**1. MySQL数据库选型原因**

稳定性和可靠性：MySQL 是一个成熟的开源关系型数据库管理系统，具有很高的稳定性和可靠性，广泛应用于各种规模的互联网项目。

性能优越：MySQL在读操作方面性能卓越，适合电商平台这种读多写少的业务场景。同时，通过合理的配置和优化，MySQL能够满足高并发环境下的数据处理需求。

扩展性强：MySQL 支持主从复制和分布式架构，能够通过读写分离和分库分表等手段实现水平扩展，适应业务量的增长。

丰富的生态和工具：MySQL 拥有丰富的生态系统和管理工具，如MySQL Workbench、phpMyAdmin等，方便开发和运维人员进行数据库管理和优化。

社区支持：MySQL 作为开源软件，拥有庞大的用户社区，提供了丰富的文档和资源，遇到问题时能够迅速找到解决方案。

**2. 数据表规模**

Koi-鞋城Online系统中涉及多个模块，总共包含22张数据表。

**3. 业务能力**

高并发处理能力：通过使用MySQL的主从复制和读写分离技术，提升数据库的读写性能，支持高并发的用户访问和操作。

数据一致性保障：通过事务管理和锁机制，确保数据库操作的原子性、一致性、隔离性和持久性（ACID），保障数据的一致性和完整性。

快速数据查询：通过建立合适的索引（如B树索引、哈希索引等）和优化查询语句，提高数据查询的速度和效率，满足用户快速获取商品和订单信息的需求。

灵活的数据扩展：支持数据库的水平拆分和垂直拆分，通过分库分表技术将数据分布在多个物理节点上，提升系统的扩展性和容灾能力。

备份与恢复能力：定期进行数据备份，通过全量备份和增量备份相结合的方式，保障数据的安全性和可恢复性，在数据丢失或系统故障时能够迅速恢复业务。

## 数据库逻辑结构设计

**1. 数据库逻辑模式（概念模式）**

逻辑模式是数据库的核心模式，描述了数据库中所有数据的结构、数据类型及其关系。它通常通过E-R图（实体-关系图）或关系模式（表结构）来表示。在Koi-鞋城Online系统中，逻辑模式主要包括以下几个主要实体和关系：

用户（Users）

商品（Products）

订单（Orders）

评论（Comments）

优惠券（Coupons）

**2. 外模式（视图模式）**

外模式是逻辑模式的子集，为不同的用户或应用程序提供特定的视图。外模式可以限制用户只能访问和操作他们所需的数据，提供安全性和简化数据处理。在Koi-鞋城Online系统中，可能存在以下外模式：

用户视图：展示用户的基本信息和订单记录。

管理员视图：展示所有用户、商品、订单、评论和优惠券的管理操作。

客服视图：展示用户订单和评论的处理界面。

这些外模式是逻辑模式的不同视图，每个视图针对特定用户角色提供不同的数据访问权限和操作功能。

**3. 内模式（存储模式）**

内模式描述了数据在物理存储设备上的具体存储方式，包括文件结构、索引、数据块和页等。内模式的设计直接影响数据库系统的性能和效率。在Koi-鞋城Online系统中，内模式涉及以下方面：

表和索引的物理存储结构：如B树索引、哈希索引等。

数据文件和日志文件的存储管理：如InnoDB存储引擎的表空间和事务日志。

数据分区和分片：根据业务需求对大表进行水平或垂直分区，提升查询和写入性能。

备份和恢复策略：如全量备份、增量备份和日志备份等。

## 数据库物理结构设计

**1.数据的内存分配**

缓存和缓冲池：

缓存：使用缓存（如Redis）存储经常访问的数据，减少数据库的负载。

缓冲池：数据库管理系统（如MySQL的InnoDB引擎）使用缓冲池来缓存数据页和索引页，提高数据访问速度。

**2. 空间要求**

数据表的初始大小和增长策略：

数据表的大小取决于行数和每行数据的大小。应根据业务需求预估表的大小，并设置合理的增长策略。

索引空间：

索引可以显著提升查询性能，但也会占用额外的存储空间。应合理设计索引以平衡性能和空间。

## 数据库安全设计

**1. 身份验证**

强密码策略：

要求用户和管理员使用强密码，包括字母、数字和特殊字符，并设置最小长度。

定期强制用户更改密码。

多因素认证（MFA）：

对管理员和关键用户启用多因素认证，以增加登录安全性。

数据库用户权限管理：

为不同的数据库用户分配最小权限原则（Least Privilege），确保每个用户只能访问其工作所需的数据和功能。

**2. 访问控制**

角色与权限：

定义不同的角色（如管理员、普通用户）并为其分配相应的权限。

在应用程序层面控制用户的访问权限。

IP白名单：

仅允许特定IP地址访问数据库服务器。

网络防火墙和安全组：

使用防火墙和安全组限制数据库服务器的网络访问，仅开放必要的端口（如MySQL的3306端口）。

**3. 备份与恢复**

定期备份：

实施每日、每周和每月的数据库备份策略。

备份文件应存储在安全的位置，并定期测试恢复过程。

灾难恢复计划：

制定灾难恢复计划，确保在发生意外事件时能够快速恢复数据库。

## 数据字典

1. 用户表

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **默认值** | **注释** |
| user\_id | bigint | NOT NULL | 用户ID |
| user\_role | tinyint unsigned | 2 | 1 管理员 2用户 |
| status | tinyint | 1 | 状态 0禁用 1启用 |
| account | varchar(255) | NOT NULL | 账号 |
| nick\_name | varchar(32) | NULL | 昵称 |
| password | varchar(32) | '' | 密码 |
| mobile | varchar(16) | NULL | 手机号(唯一) |
| gender | tinyint unsigned | 2 | 性别 0-female 1-male 2-未知 |
| email | varchar(100) | NULL | 邮箱 |
| srcface | varchar(255) | '' | 头像 |
| default\_address | int | NULL | 默认地址id |
| create\_time | datetime | NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP | 创建时间 |
| update\_time | datetime | NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP | 修改时间 |

1. 企宣模块
   * 1. 企业合作表

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **默认值** | **注释** |
| user\_id | bigint | NOT NULL | 用户ID |
| user\_role | tinyint unsigned | 2 | 1 管理员 2用户 |
| status | tinyint | 1 | 状态 0禁用 1启用 |
| account | varchar(255) | NOT NULL | 账号 |
| nick\_name | varchar(32) | NULL | 昵称 |
| password | varchar(32) | '' | 密码 |
| mobile | varchar(16) | NULL | 手机号(唯一) |
| gender | tinyint unsigned | 2 | 性别 0-female 1-male 2-未知 |
| email | varchar(100) | NULL | 邮箱 |
| srcface | varchar(255) | '' | 头像 |
| default\_address | int | NULL | 默认地址id |
| create\_time | datetime | NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP | 创建时间 |
| update\_time | datetime | NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP | 修改时间 |

* + 1. 广告表

|  |  |  |  |
| --- | --- | --- | --- |
| id | int | AUTO\_INCREMENT | ID |
| partner\_name | varchar(255) | NOT NULL | 合作商名称 |
| uri\_brand | varchar(255) | NOT NULL | 品牌链接地址 |
| remark | text | NULL | 备注 |
| file\_ids | varchar(255) | NULL | 关联图片ids |

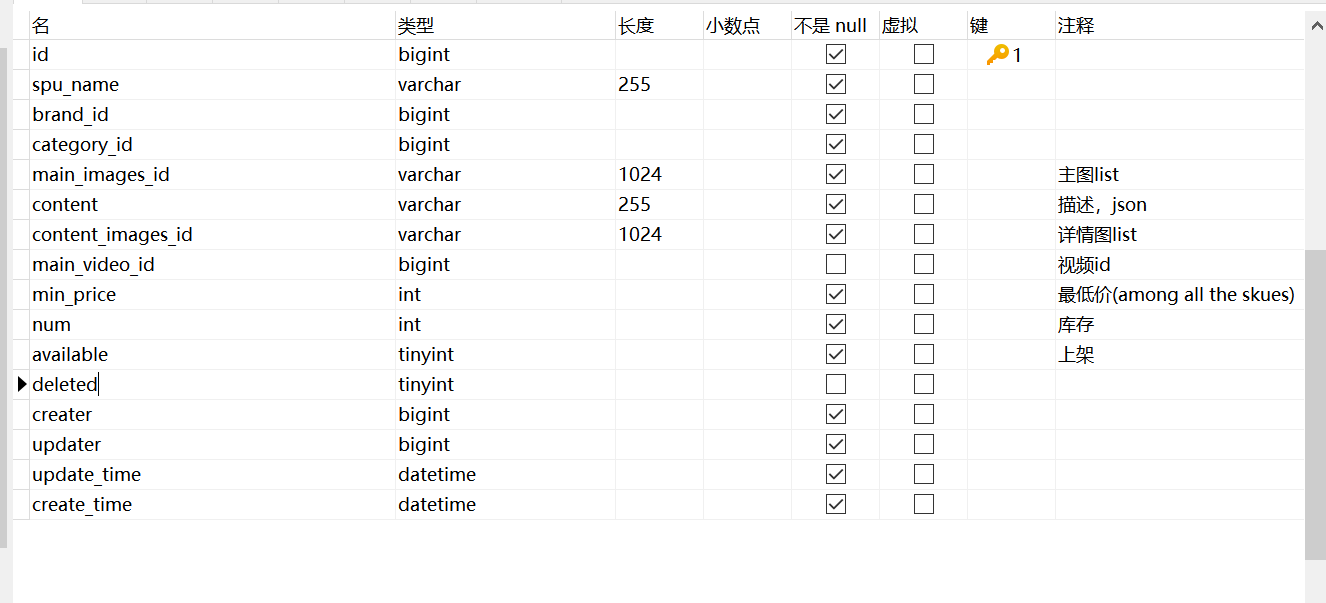
1. 品牌表



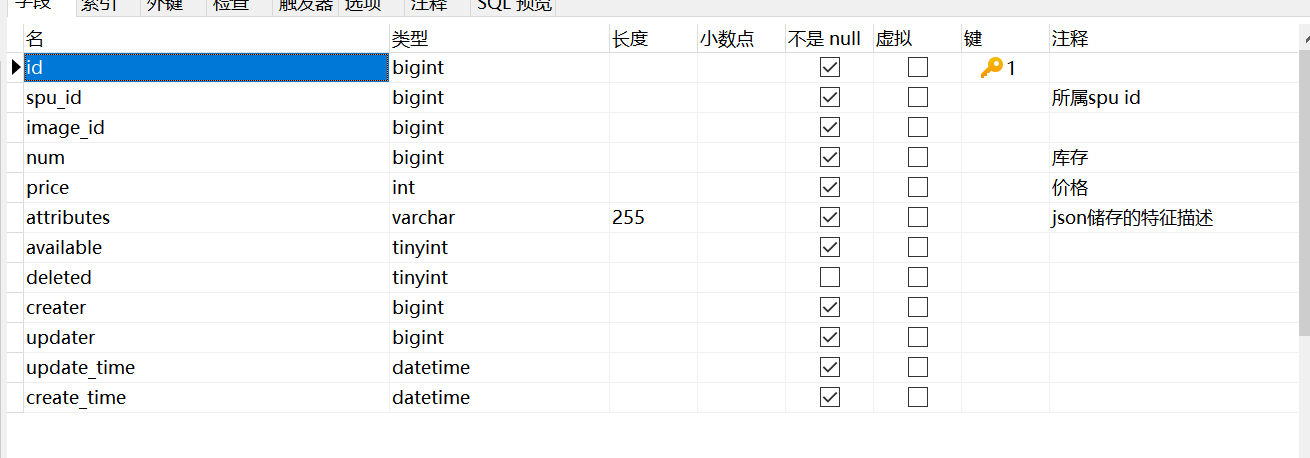
1. 分类表



1. 商品表



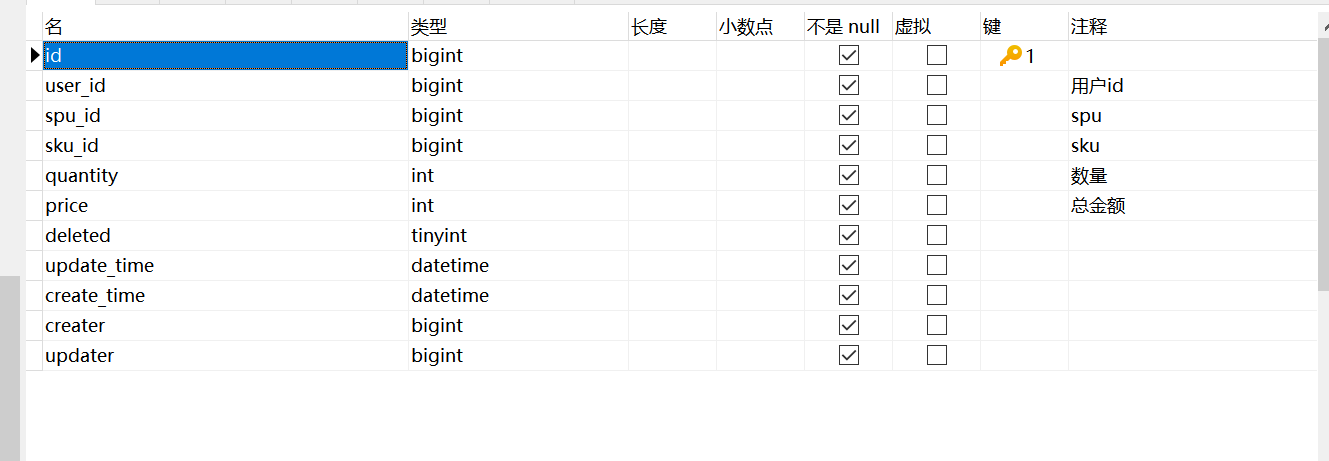
1. 库存表



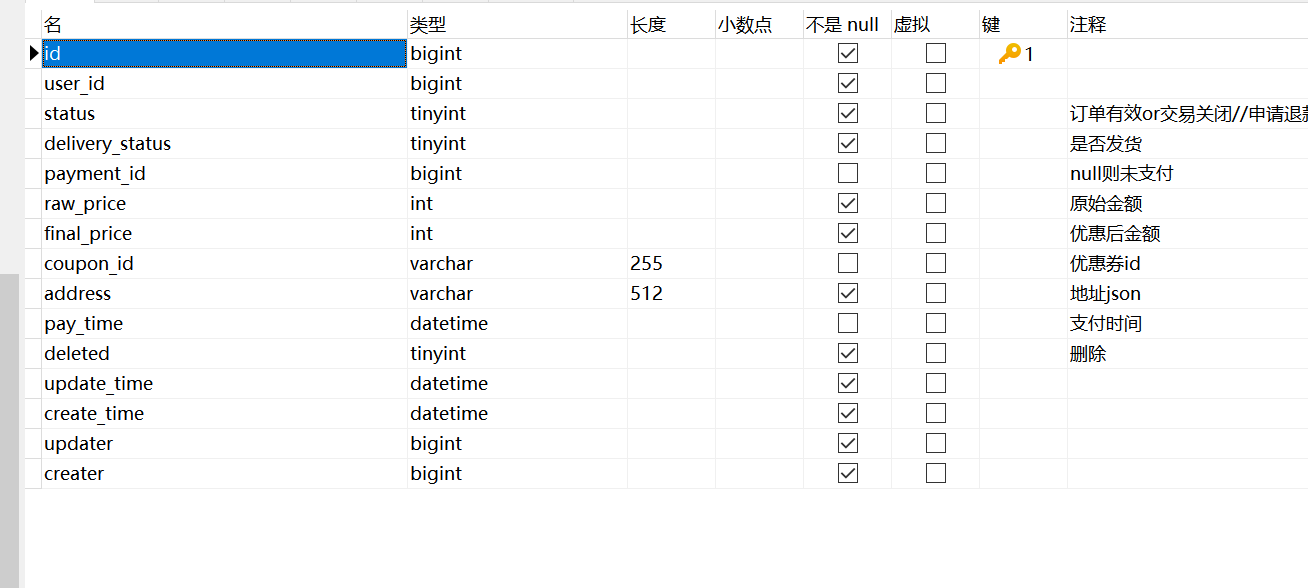
1. 地址表



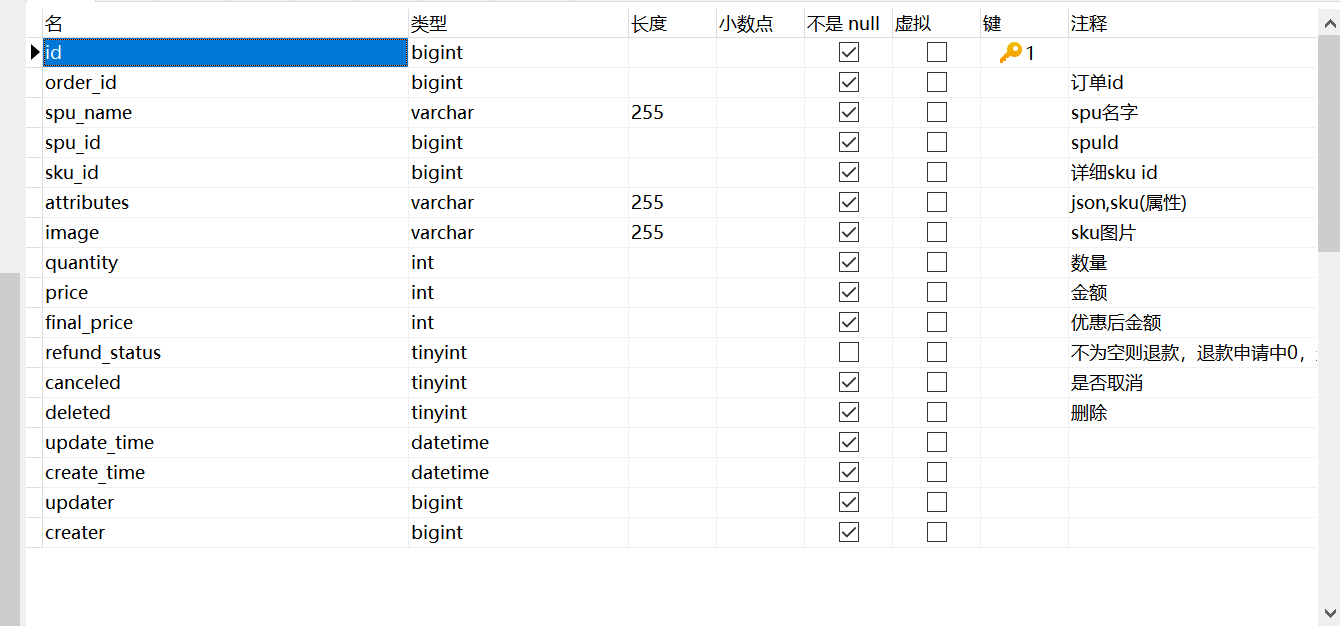
1. 购物车表



1. 订单表



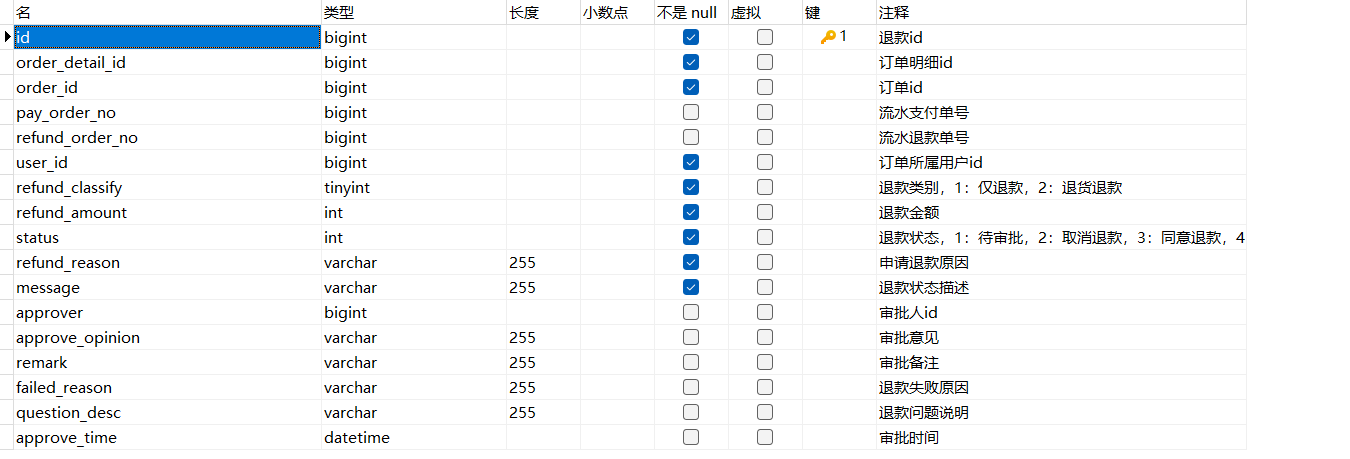
1. 订单详情表



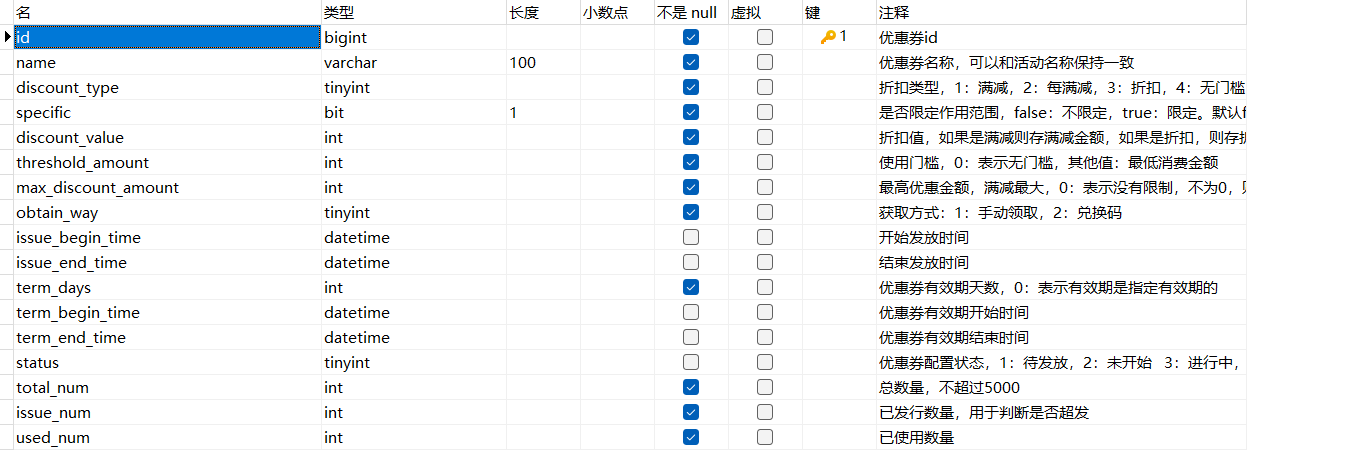
1. 支付表



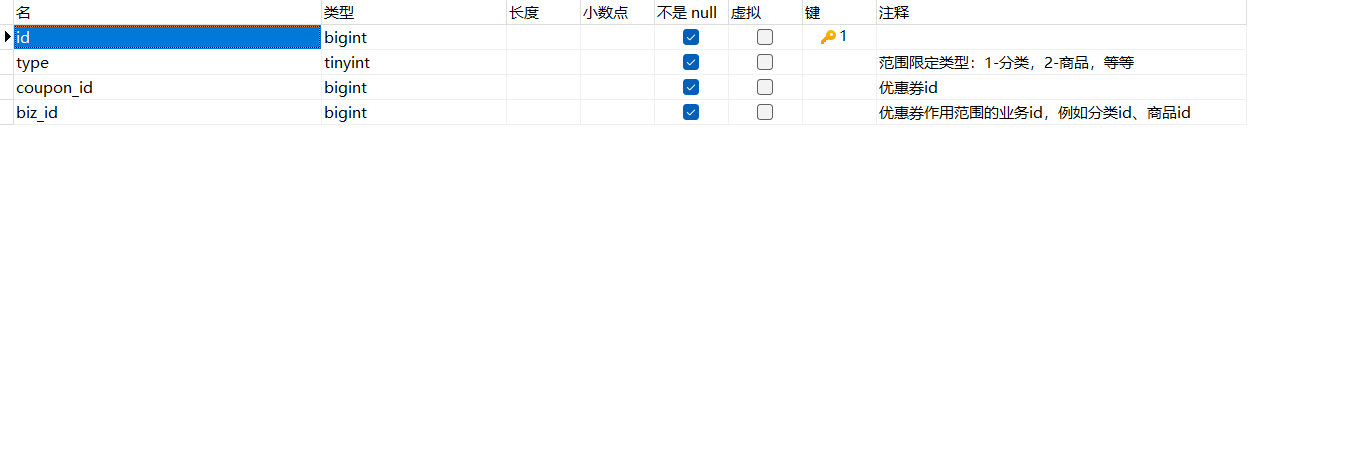
1. 退款表



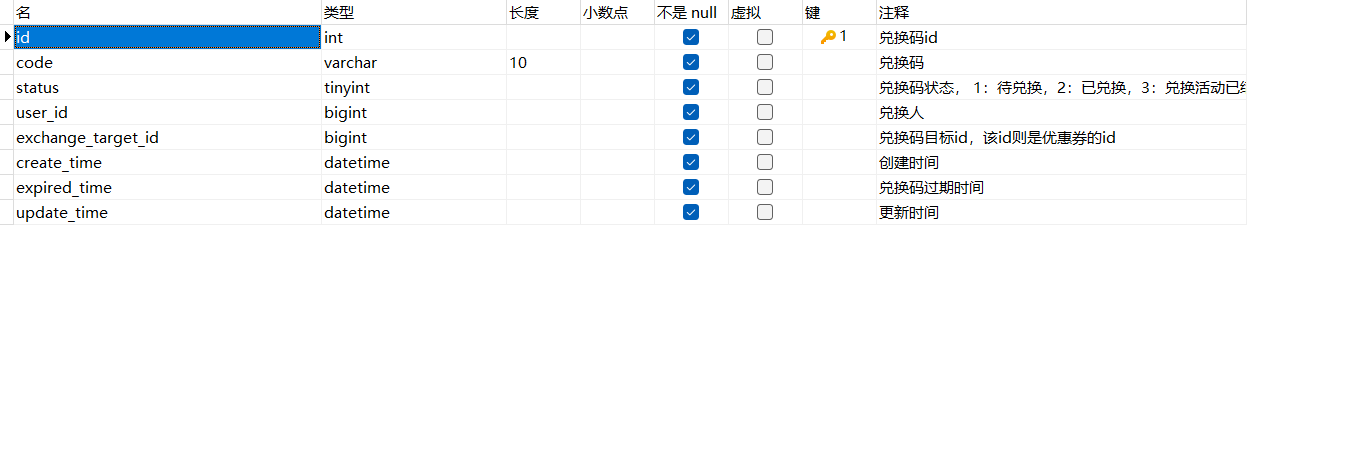
1. 优惠券表



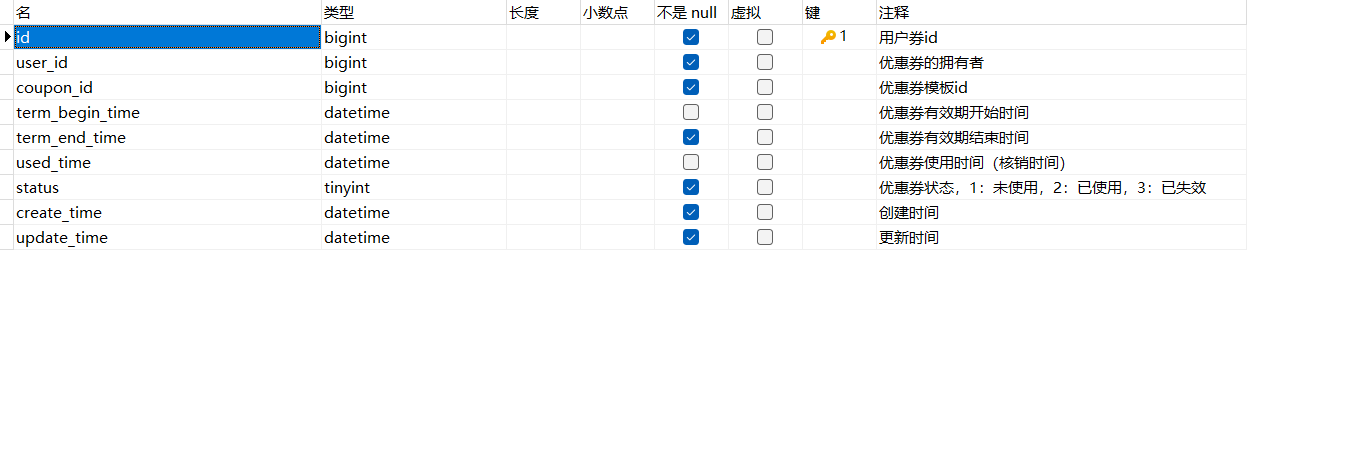
1. 优惠券使用范围表



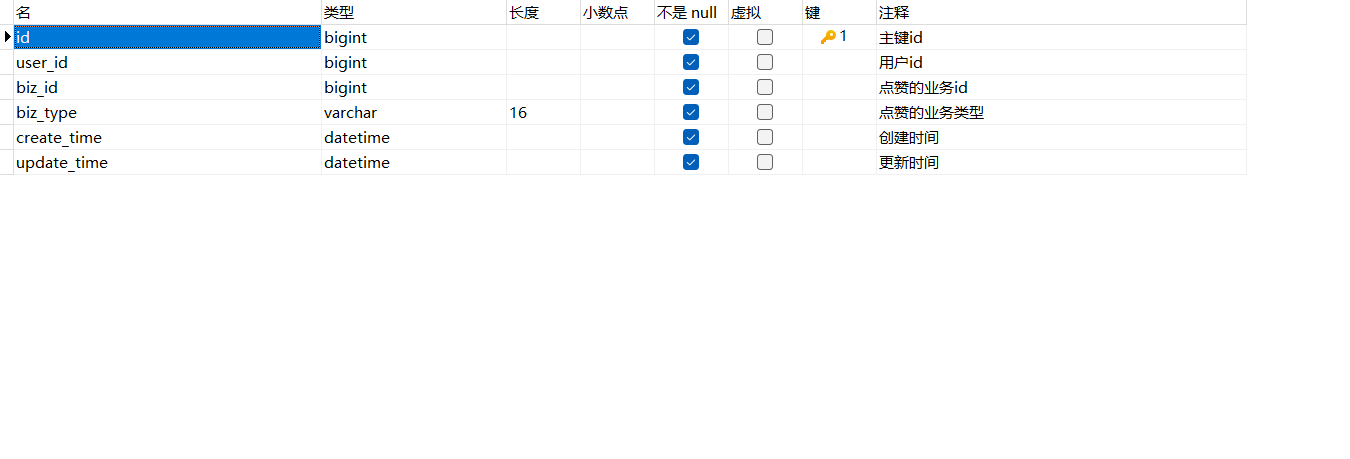
1. 兑换码表



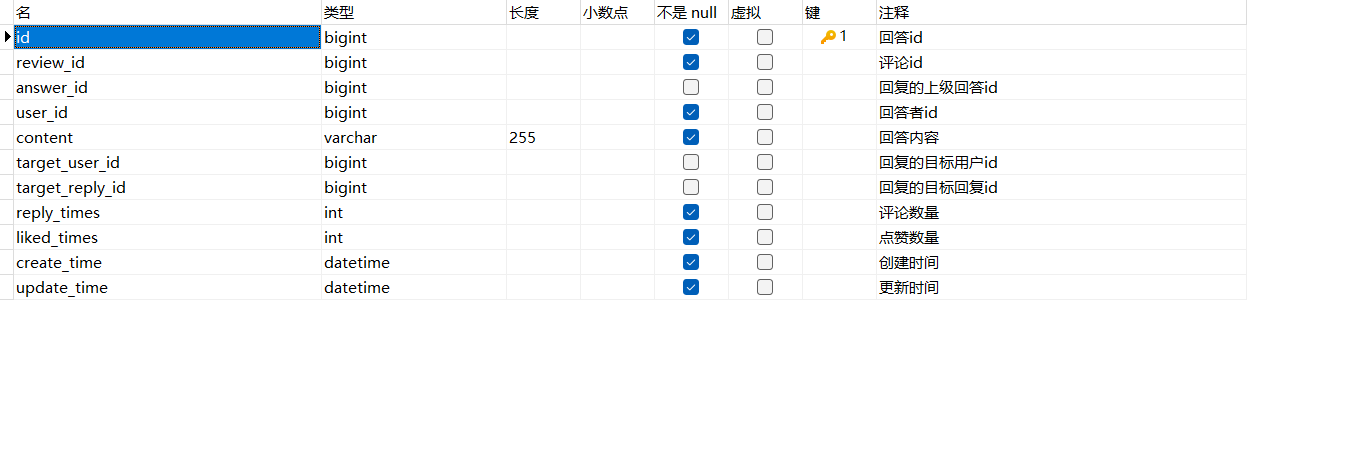
1. 用户优惠券表



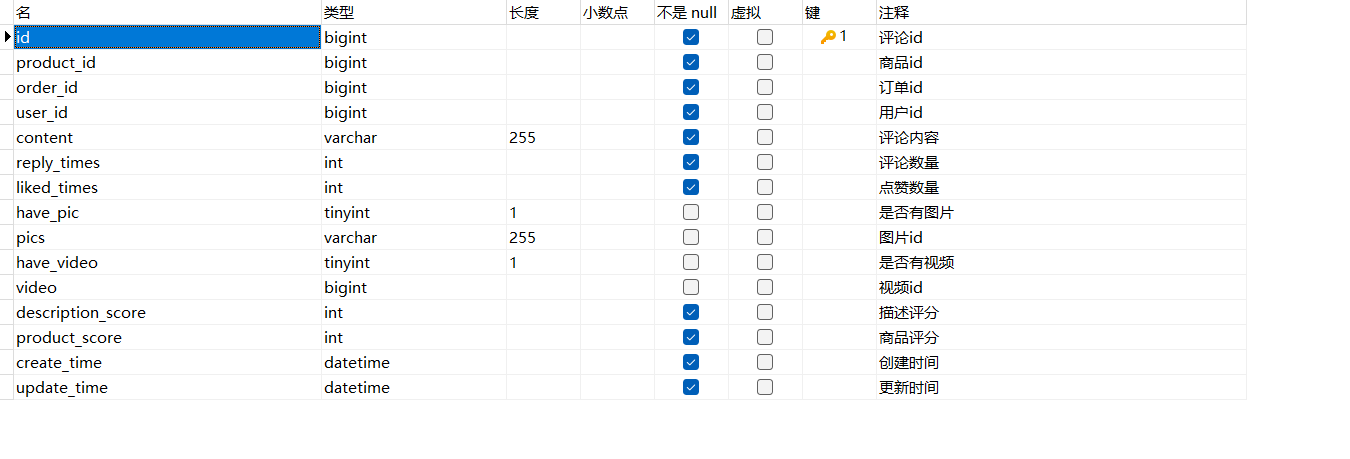
1. 点赞信息表



1. 评论回复表



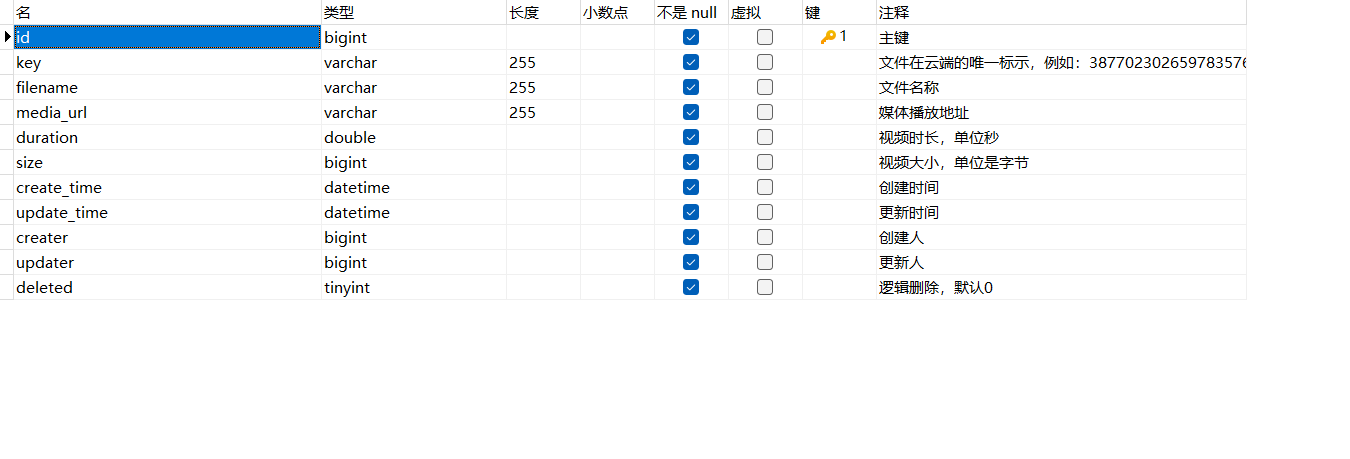
1. 评论表



1. 图片文件表



1. 视频文件表



# 系统可靠性设计

1. 安全性设计

1.1 用户身份验证与授权

用户注册与登录：使用安全的加密算法（MD5）对用户密码进行加密存储，防止密码泄露。

多因素身份验证：支持短信验证码进行身份验证，增强用户账户的安全性。

权限管理：基于角色的访问控制，根据用户角色（管理员、普通用户等）授予不同的操作权限，限制敏感操作的访问。

1.2 数据传输与存储安全

数据加密传输：使用HTTPS协议加密传输数据，防止数据在传输过程中被截获和篡改。

数据库加密：对存储在数据库中的敏感数据进行加密，确保数据在存储过程中安全。

备份与恢复：定期进行数据库备份，确保在数据丢失或损坏时能够迅速恢复。

1.3 安全审计与监控

日志记录：详细记录用户操作日志、系统错误日志等，方便后期审计和问题排查。

入侵检测：部署入侵检测系统（IDS），实时监控和分析系统中的可疑活动，及时发现和处理安全威胁。

漏洞扫描与修复：定期进行安全漏洞扫描，及时修复系统漏洞，保持系统的安全性。

2. 性能设计

2.1 系统架构与扩展性

分布式架构：采用分布式架构设计，支持水平扩展，确保系统能够处理大规模的并发请求。

缓存机制：使用缓存技术（如Redis）缓存热点数据，减少数据库访问，提高系统响应速度。

负载均衡：部署负载均衡器（如Nginx），均衡分配用户请求到多个服务器，防止单点故障。

2.2 数据库优化

读写分离：采用主从数据库架构，分离读写操作，提升数据库的并发处理能力。

数据库分片：将大数据量的表进行水平分片，分散存储，减少单表的负载。

2.3 性能监控与优化

压力测试：定期进行压力测试，评估系统在高负载下的表现，并进行性能优化。

3. 可靠性设计

3.1 高可用设计

服务器冗余：通过部署多台服务器，提高系统的容灾能力，确保在某台服务器故障时，系统仍能正常运行。

自动故障切换：配置自动故障切换机制（如使用Keepalived），在服务器出现故障时，自动切换到备用服务器，保证系统的连续性。

3.2 数据可靠性

数据备份：定期进行全量和增量备份，确保在数据丢失或损坏时能够迅速恢复。

数据恢复演练：定期进行数据恢复演练，验证备份数据的可用性和恢复流程的可操作性。

3.3 容错与恢复

服务降级：在系统负载过高或某些服务不可用时，进行服务降级处理，保证核心功能的可用性。

重试机制：对于关键操作，采用重试机制，确保操作在出现临时故障时能够成功执行。