# 目录

### [绪论 1](#_bookmark0)

* 1. [研究的背景及意义 1](#_bookmark1)
     1. [选题的背景 1](#_bookmark2)
     2. [国内外研究现状 1](#_bookmark3)
     3. [研究的意义 1](#_bookmark4)
  2. [系统目标 2](#_bookmark5)

### [需求分析 3](#_bookmark6)

[2.1 业务需求 3](#_bookmark7)

[2.1.1 主要业务流程 3](#_bookmark8)

[2.2 功能需求 7](#_bookmark9)

[2.2.1 角色分析 7](#_bookmark10)

[2.2.2 业务功能 7](#_bookmark11)

[2.3 非功能需求 29](#_bookmark12)

[2.3.1 环境需求 29](#_bookmark13)

[2.3.2 性能需求 29](#_bookmark14)

[2.3.3 安全需求 30](#_bookmark15)

### [总体设计 31](#_bookmark16)

* 1. [系统设计的原则 31](#_bookmark17)
  2. [系统体系结构设计 31](#_bookmark18)
  3. [系统功能结构设计 32](#_bookmark19)
     1. [商品管理子系统模块构建设计 33](#_bookmark20)
     2. [用户管理子系统模块构建设计 34](#_bookmark21)
     3. [系统管理子系统模块构建设计 34](#_bookmark22)

### [数据库设计 36](#_bookmark23)

* 1. [概念结构设计 36](#_bookmark24)

[4.1.1 设计思路 36](#_bookmark25)

[4.1.2 E-R 图 36](#_bookmark26)

* 1. [逻辑结构设计 41](#_bookmark27)

[4.2.1 设计思路 41](#_bookmark28)

[4.2.2 逻辑模型 41](#_bookmark29)

* 1. [物理结构设计 44](#_bookmark30)

[4.3.1 存取方式 44](#_bookmark31)

[4.3.2 存储结构 44](#_bookmark32)

### [界面设计 45](#_bookmark33)

* 1. [界面关系图或工作流图 45](#_bookmark34)
  2. [界面设计成果 45](#_bookmark35)

[5.2.1 主界面 45](#_bookmark36)

[5.2.2 子界面 46](#_bookmark37)

### [详细设计 55](#_bookmark38)

* 1. [系统主要功能模块介绍 55](#_bookmark39)
  2. [功能模块设计 55](#_bookmark40)
     1. [商品信息模块 55](#_bookmark41)
     2. [收藏夹模块 58](#_bookmark42)
     3. [用户信息模块 62](#_bookmark43)

[6.2.4 订单模块 65](#_bookmark44)

* + 1. [商品信息管理模块 70](#_bookmark45)
    2. [商品分类管理模块 74](#_bookmark46)
    3. [用户信息管理模块 76](#_bookmark47)

### [7 编码 81](#_bookmark48)

* 1. [代码实现与核心算法 81](#_bookmark49)
  2. [代码优化分析 85](#_bookmark50)

### [8 测试 86](#_bookmark51)

* 1. [测试方案设计 86](#_bookmark52)

[8.1.1 测试策略 86](#_bookmark53)

[8.1.2 测试进度安排 87](#_bookmark54)

[8.1.3 测试资源 87](#_bookmark55)

[8.1.4 关键测试点 88](#_bookmark56)

* 1. [测试用例构建 88](#_bookmark57)
     1. [测试用例编写约定 88](#_bookmark58)
     2. [测试用例设计 88](#_bookmark59)
     3. [关键测试用例 90](#_bookmark60)
     4. [测试用例维护 91](#_bookmark61)

### [总结与展望 92](#_bookmark62)

* 1. [设计工作总结 92](#_bookmark63)
  2. [未来工作展望 92](#_bookmark64)

### [谢辞 93](#_bookmark65)

[**引用文献 94**](#_bookmark66)

### [附录 A 外文翻译—原文部分 96](#_bookmark67)

[**附录 B 外文翻译—译文部分 103**](#_bookmark68)

### [附录 C 软件使用说明书 108](#_bookmark69)

[**附录 D 主要源代码 112**](#_bookmark70)

# 1 绪论

## 研究的背景及意义

* + 1. 选题的背景

随着人们生活水平的不断提高，大学生的经济意识和购买能力不断增强。在互联网背景下，大学生的消费观念在悄然改变，大学校园二手交易市场应运而生， 二手交易也成为当代大学生生活不可缺少的一部分。建立一个校园二手交易平台， 不仅可以方便在校的学生，同时也营造了勤俭节约的校园文化氛围，更加锻炼了学生自主创业、实践创新的能力。

当前许多大学生都会通过网购来购买自己所需的物品，但因为有时买卖方距离较远，从付款到收货，中间会耽搁五六天，甚至十几天，而校园交易就省去了这样的麻烦，一旦买卖方取得联系，达成共识，在几个小时之内就可以完成交易。而且我们在购物网站上经常会买到实物与描述不符的商品，而校园交易则可以当面检查商品，来判断是否符合购买者自身要求，且在校大学生大多具有良好的品质，可很好的避免买到不符合自身需求的商品，很好的增加了交易的可靠性。所以说校园市场开发是很具有潜力的。

随着移动互联网的快速发展，校园二手交易市场以微电商为运营模式，采用区域划分策略，精准、快速、便利地为高校大学生提供高性价比的二手物品，促进资源的循环使用，减少浪费。

* + 1. 国内外研究现状

校园二手交易平台从其功能实现角度类似于现在的电子商务网站，如淘宝， 京东，都是图片结合文字描述商品信息的。而买家和卖家都是在校学生，出售与购买规模较小。但是不同于那些购物网站。校园二手交易平台只提供一个公布信息的渠道，而不会对信息进行监控。校园市场开发是很具有潜力的。学生只需在网站上发布售卖的商品，便可等待顾客购买，消费者也只需在网页上浏览，点击下单便可获得心仪的商品。所以说，校园二手交易平台一定会在各大高校流行起来。

* + 1. 研究的意义

随着计算机网络技术的飞速发展，Web 应用在生活中地位日益突出。而在这其中，网络购物成为大学生日常生活的主要采购途径。校园二手交易平台就为广

大学生提供了这样一个途径，学生可以将自己闲置的资源放在网上出售，也可以从中寻找自己需要的物件，将资源价值最大化。由于均是在校学生陈列的商品， 与学生生活息息相关，如功能手机，播放器，二手自行车等，有的学生有需求， 而有的学生有闲置。这样做不但实现了物品的循环利用，也为自己创造了一定的经济价值。大学是一个购物水平相对较高的地方，每年都有毕业生离开校园，这些东西作为废品卖掉太可惜，每年也有新生入学，购买全新的又过于昂贵。所以说，校园二手交易平台贴合学生需求，也能降低消费成本。

## 系统目标

本校园二手交易平台需要实现三个子系统，分别是商品管理，用户管理，系统管理。主要功能介绍如下：

用户可以浏览商品、发布商品信息、搜索商品、修改已发布商品信息、下架商品。用户也可以将喜欢的商品增加到收藏夹、删除收藏夹商品、查询收藏夹商品。对于交易完成的商品，用户可以查询已售商品、已购商品、以及完成评价功能。用户可以对个人信息进行管理，包括登录注册、修改个人信息和查看个人信息的功能。

管理员可以对商品信息、商品分类、用户信息进行管理。商品信息管理包括了商品审核、商品删除、商品查询功能。用户信息管理包括了用户查询、用户删除、用户信用度修改功能。商品分类管理包含增加商品分类、删除商品分类和查询商品分类三个功能。

# 需求分析

## 业务需求

1. 该系统能够提供分类浏览商品服务，使网站条目清晰，方便浏览。
2. 该系统能够提供用户发布商品，对自己发布的商品进行修改，下架自己发布的商品以及搜索商品服务。
3. 该系统能够提供收藏夹服务，方便用户对喜欢物品的操作。
4. 该系统能够提供订单服务，使得每一笔支出收入都明确清晰。
5. 该系统能够提供用户信息服务，完善用户个人信息，增加系统安全性。
6. 管理员能够对商品的信息进行管理。商品信息管理指：管理员可以审核用户发布的商品，若审核不通过可以删除用户发布的商品，还可以查询商品。
7. 管理员能够对商品分类信息进行管理。商品分类管理指：管理员可以手动增加商品分类，修改商品分类，查询商品分类以及删除商品分类。
8. 管理员能够对用户信息进行管理。用户信息管理指：管理员可以查询用户，删除用户以及修改用户信用度。

### 主要业务流程

* + - 1. 查询收藏夹商品业务流程图

如下图 2-1 所示：

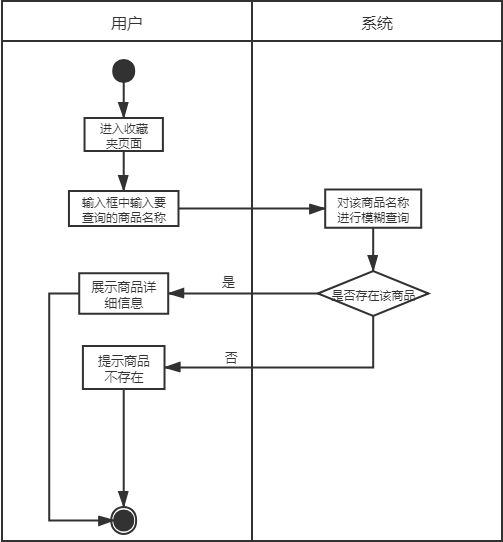


图 2-1 查询收藏夹商品业务流图

* + - 1. 商品信息发布业务流程图

如下图 2-2 所示：

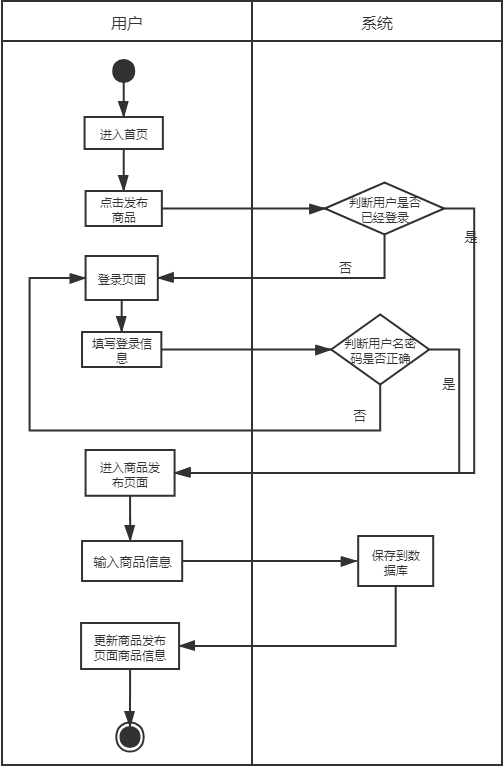


图 2-2 商品信息发布业务流程图

* + - 1. 查询已购商品业务流图

如下图 2-3 所示：

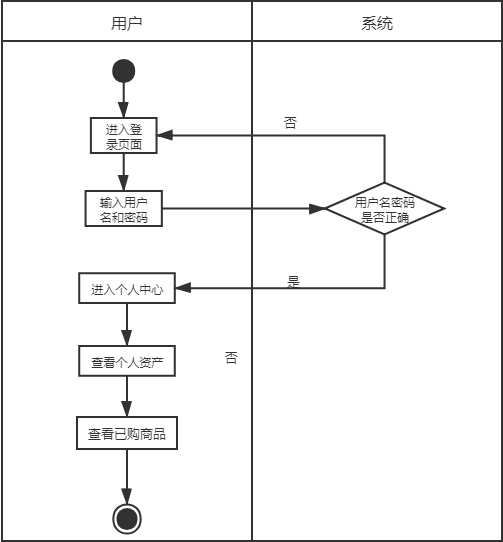


图 2-3 查询已购商品业务流图

* + - 1. 用户登陆注册业务流图

如下图 2-4 所示：

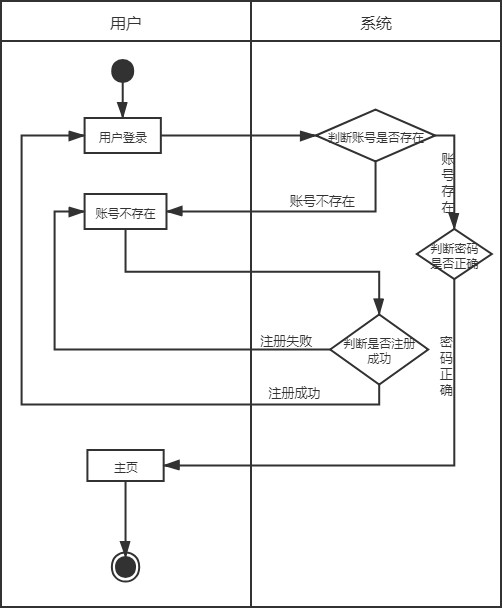


图 2-4 用户登陆注册业务流图

* + - 1. 管理员查询商品业务流图

如下图 2-5 所示：

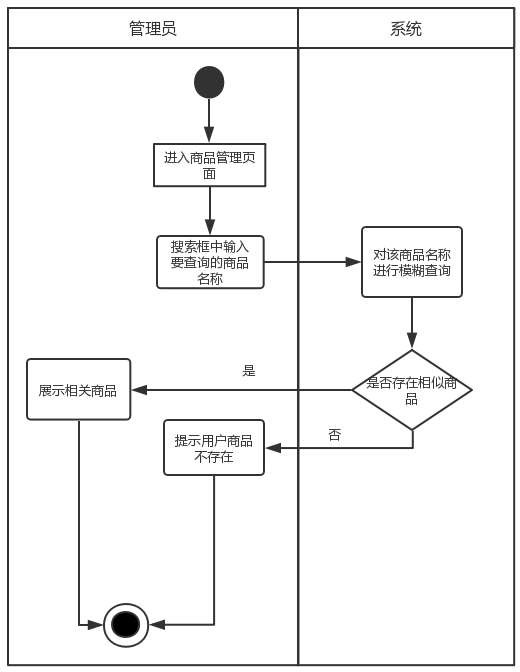


图 2-5 管理员查询商品业务流图

* + - 1. 管理员添加商品分类业务流图

如下图 2-6 所示：

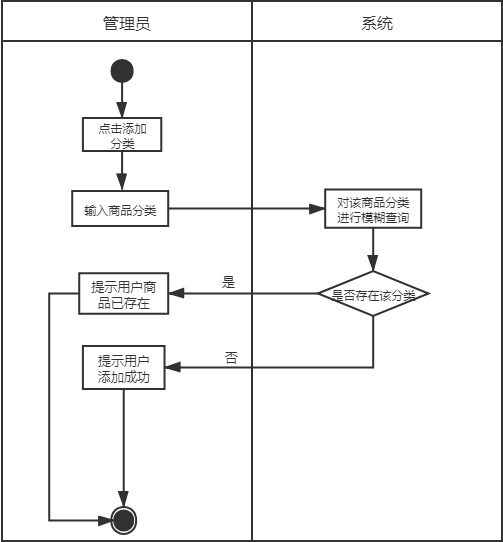


图 2-6 管理员添加商品分类业务流图

* + - 1. 管理员查询用户业务流图

如下图 2-7 所示：

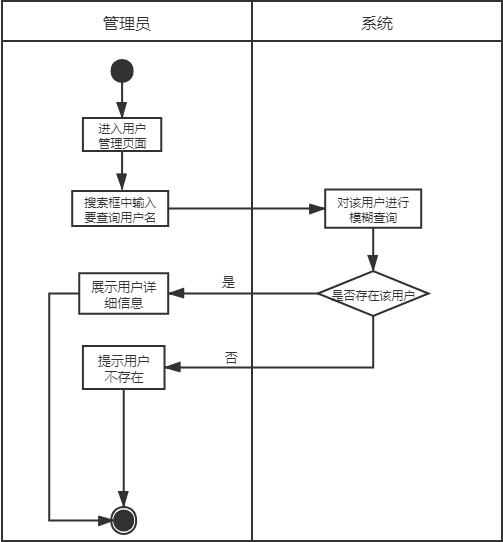


图 2-7 管理员查询用户业务流图

## 功能需求

* + 1. 角色分析

角色分析如表 2-1 所示：

表 2-1 系统角色表

|  |  |
| --- | --- |
| 角色 | 职责或功能 |
| 用户 | 用户可以在该网站购买自己心仪的商品，也可以在该网站售卖自  己的商品 |
| 系统管理员 | 管理和维护整个系统的用户组织结构，负责对用户、角色、用户  级别的增、删、改、查等管理。 |

* + 1. 业务功能

系统用例图如图 2-8 所示：

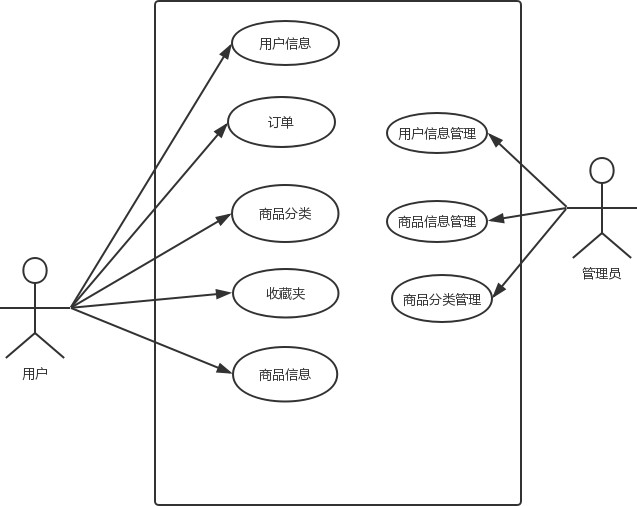


图 2-8 系统用例图

* + - 1. 商品信息模块

商品信息模块如图 2-9 所示：

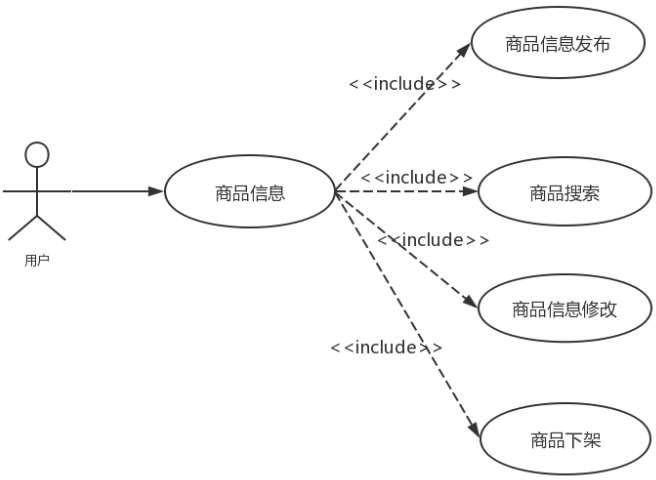


图 2-9 商品信息模块

该模块的商品信息发布用例分析表如下表 3-2 所示：

表 3-2 商品信息发布用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0001 | 用例名称 | 商品信息发布 |
| 用例描述 | 用户编写商品基本信息，上传商品相关图片，点击上传按  钮，系统将商品保存到数据库中并返回上传成功。 | | |
| 执行者 | 用户 | | |
| 前置条件 | 用户必须已经登录 | | |
| 后置条件 | 将用户上传的商品保存到数据库中 | | |
| 涉众利益 | 用户希望商品上传后可以返回上传成功或失败 | | |
| 用例场景 | 商品信息发布界面 | | |
| 基本事件流 | 基本流程（Base Flow）   1. 用户点击发布商品按钮； 2. 系统显示发布商品页面； 3. 用户填写商品详细信息； 4. 用户点击上传按钮； 5. 系统保存用户上传的商品信息； 6. 系统返回上传成功页面； 扩展流程（Extend Flow）   （3）没有上传图片：  ① 系统提示没有上传图片  ② 用例结束 | | |

续表 3-2 商品信息发布用例描述

|  |  |
| --- | --- |
|  | 字段列表（Filed List）  商品信息＝商品编号+商品名称＋商品标题+价格+新旧程度+分类+日期+图片  业务规则（Business role）   1. 上传商品名称不能为空； 2. 上传商品价格不能为空； 3. 上传商品新旧程度不能为空； 4. 必须上传商品的相关图片； |
| 特殊需求  （Special Requireme nt） | 1. 用户所有操作应在 3s 内得到响应； 2. 页面简洁，用户能在 1min 内完成上传业务； |
| 编写人 |  |

该模块的商品搜索用例分析表如下表 3-3 所示：

表 3-3 商品搜索用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0002 | 用例名称 | 商品搜索 |
| 用例描述 | 用户在搜索框中输入要查询的商品，系统返回商品的信  息并显示。 | | |
| 执行者 | 用户 | | |
| 前置条件 | 用户必须已经登录 | | |
| 后置条件 | 将用户搜索的商品显示在页面上 | | |
| 涉众利益 | 用户希望可以返回查询成功或失败 | | |
| 用例场景 | 商品信息展示界面 | | |
| 基本事件流 | 基本流程（Base Flow）   1. 用户输入要查询的商品； 2. 系统模糊查询相匹配的项； 3. 系统返回匹配的商品详细信息； 4. 若没有相匹配的商品返回空； 字段列表（Filed List）   商品信息＝商品编号+商品名称＋商品标题+价格+新旧  程度+分类+日期+图片 | | |
| 特殊需求 | 1. 用户所有操作应在 3s 内得到响应； 2. 页面简洁，用户能在 1min 内完成查询业务； | | |
| 编写人 |  | | |

该模块的商品信息修改用例分析表如下表 3-4 所示：

续表 3-4 商品信息修改用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0003 | 用例名称 | 商品信息修改 |
| 用例描述 | 用户点击修改按钮，在用户修改商品信息界面修改商品  信息。点击确认后，系统保存修改信息并刷新页面。 | | |
| 执行者 | 用户 | | |
| 前置条件 | 用户必须已经登录 | | |
| 后置条件 | 将用户修改后的商品显示在页面上 | | |
| 涉众利益 | 用户希望可以返回修改成功或失败 | | |
| 用例场景 | 商品信息相修改界面 | | |
| 基本事件流 | 基本流程（Base Flow）   1. 用户修改商品信息； 2. 用户点击确认； 3. 系统将修改后的信息保存到数据库； 4. 系统返回修改结果 5. 系统刷新页面；   扩展流程（Extend Flow）  （3）没有上传图片：  ① 系统提示没有上传图片  ② 用例结束  字段列表（Filed List）  商品信息＝商品编号+商品名称＋商品标题+价格+新旧程度+分类+日期+图片 | | |
| 特殊需求  （Special Requireme nt） | 1. 用户所有操作应在 3s 内得到响应； 2. 页面简洁，用户能在 1min 内完成修改业务； | | |
| 编写人 |  | | |

该模块的商品下架用例分析表如下表 3-5 所示：

表 3-5 商品下架用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0004 | 用例名称 | 商品下架 |
| 用例描述 | 用户点击下架按钮，系统修改商品状态并刷新页面。 | | |
| 执行者 | 用户 | | |
| 前置条件 | 用户必须已经登录 | | |
| 后置条件 | 修改商品状态改为下架 | | |
| 涉众利益 | 用户希望商品下架后可以返回下架成功或失败 | | |

续表 3-5 商品下架用例描述

|  |  |
| --- | --- |
| 用例场景 | 商品信息展示界面 |
| 基本事件流 | 基本流程（Base Flow）   1. 用户点击下架； 2. 系统返回是否确认下架； 3. 用户点击确认； 4. 系统将在数据库中修改商品状态； 5. 系统返回修改结果 6. 系统刷新页面；   字段列表（Filed List）  商品信息＝商品编号+商品名称＋商品标题+价格+新旧程度+分类+日期+图片 |
| 特殊需求  （Special Requireme nt） | 1. 用户所有操作应在 3s 内得到响应； 2. 页面简洁，用户能在 1min 内完成修改业务； |
| 编写人 |  |

* + - 1. 收藏夹模块

收藏夹模块如图 2-10 所示：

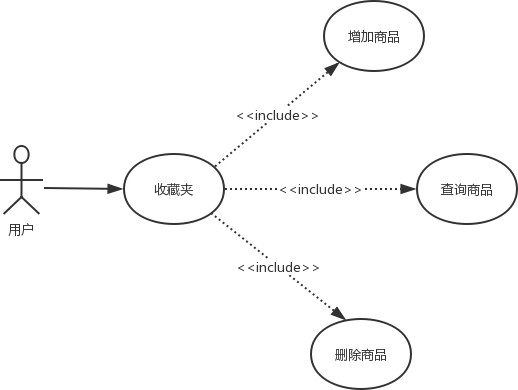


图 2-10：收藏夹模块

该模块的增加商品到收藏夹用例分析表如下表 3-6 所示：

表 3-6 增加商品到收藏夹用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0005 | 用例名称 | 增加商品到收藏夹 |
| 用例描述 | 用户在商品详情页面点击加入收藏夹，系统将相关信息  保存到数据库。 | | |
| 执行者 | 用户 | | |

续表 3-6 增加商品到收藏夹用例描述

|  |  |
| --- | --- |
| 前置条件 | 用户必须已经登录 |
| 后置条件 | 将收藏夹信息保存到数据库中 |
| 涉众利益 | 用户希望可以返回收藏成功或失败 |
| 用例场景 | 商品详情界面 |
| 基本事件流 | 基本流程（Base Flow）   1. 用户点击收藏商品按钮； 2. 系统检查用户是否已经登录； 3. 若没有登陆，系统提醒用户登录并跳转到登录页面； 4. 若已经登录，系统保存用户收藏信息； 5. 若收藏成功，系统返回收藏成功并在页面显示； 6. 若收藏失败，系统返回收藏失败并在页面显示； 字段列表（Filed List）   收藏夹＝收藏夹编号+用户编号＋商品编号  业务规则（Business role）  （1）用户必须已经登录 |
| 特殊需求  （Special Requireme nt） | 1. 用户所有操作应在 3s 内得到响应； 2. 页面简洁，用户能在 1min 内完成收藏业务； |
| 编写人 |  |

该模块的从收藏夹删除商品用例分析表如下表 3-7 所示：

表 3-7 从收藏夹删除商品用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0006 | 用例名称 | 从收藏夹删除商品 |
| 用例描述 | 用户选择要删除的商品，点击删除按钮，系统从数据库  中删除商品。 | | |
| 执行者 | 用户 | | |
| 前置条件 | 用户必须已经登录 | | |
| 后置条件 | 将商品从数据库中删除 | | |
| 涉众利益 | 用户希望可以返回删除成功或失败 | | |
| 用例场景 | 收藏夹界面 | | |
| 基本事件流 | 基本流程（Base Flow）   1. 用户点击删除按钮； 2. 系统弹出确认框； 3. 管理员点击确认按钮； 4. 系统从数据库中删除该条商品信息； | | |

续表 3-7 从收藏夹删除商品用例描述

|  |  |
| --- | --- |
|  | 1. 系统返回删除成功； 2. 系统刷新返回删除成功后的页面； 字段列表（Filed List）   收藏夹＝收藏夹编号+用户编号＋商品编号  业务规则（Business role）  （1）用户必须已经登录 |
| 特殊需求  （Special Requireme nt） | 1. 用户所有操作应在 3s 内得到响应； 2. 页面简洁，用户能在 1min 内完成删除业务； |
| 编写人 |  |

该模块的从收藏夹查询用例分析表如下表 3-8 所示：

表 3-8 从收藏夹查询商品用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0007 | 用例名称 | 从收藏夹查询商品 |
| 用例描述 | 用户在搜索框中输入要查询的商品，系统返回商品的信息  并显示。 | | |
| 执行者 | 用户 | | |
| 前置条件 | 用户必须已经登录 | | |
| 后置条件 | 将用户搜索的商品显示在页面上 | | |
| 涉众利益 | 用户希望可以返回查询成功或失败 | | |
| 用例场景 | 商品信息展示界面 | | |
| 基本事件流 | 基本流程（Base Flow）   1. 用户输入要查询的商品； 2. 系统模糊查询相匹配的项； 3. 系统返回匹配的商品详细信息； 4. 若没有相匹配的商品返回空； 字段列表（Filed List） 5. 商品信息＝商品编号+商品名称＋商品标题+价格+ 新旧程度+分类+日期+图片 6. 收藏夹＝收藏夹编号+用户编号＋商品编号 | | |
| 特殊需求  （Special Requireme nt） | 1. 用户所有操作应在 3s 内得到响应； 2. 页面简洁，用户能在 1min 内完成查询业务； | | |
| 编写人 |  | | |

* + - 1. 订单模块

订单模块如图 2-11 所示：

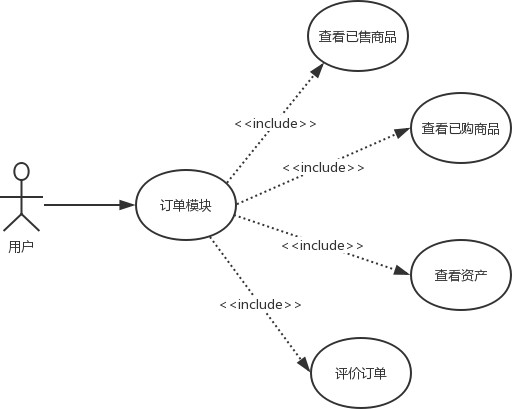


图 2-11 订单模块

该模块的查看已售商品用例分析表如下表 3-9 所示：

表 3-9 查看已售商品用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0008 | 用例名称 | 查看已售商品 |
| 用例描述 | 用户点击查看已售商品，系统跳转到已售商品展示页面，  展示已售商品。 | | |
| 执行者 | 用户 | | |
| 前置条件 | 用户必须已经登录 | | |
| 后置条件 | 展示已售商品 | | |
| 涉众利益 | 用户希望能按方格形式清晰展示商品，方便观看。 | | |
| 用例场景 | 已售商品展示界面 | | |
| 基本事件流 | 基本流程（Base Flow）   1. 用户点击查看商品按钮； 2. 系统检查用户是否登录； 3. 系统查询用户已售商品； 4. 系统将商品展示在页面； 扩展流程（Extend Flow）   （3）没有已售商品：  ① 系统提示没有已售商品  ② 用例结束  字段列表（Filed List）  商品信息＝商品编号+商品名称＋商品标题+价格+新旧程 | | |

续表 3-9 查看已售商品用例描述

|  |  |
| --- | --- |
|  | 度+分类+日期+图片  业务规则（Business role）  （1）用户必须已经登录； |
| 特殊需求  （Special Requireme nt） | 1. 户所有操作应在 3s 内得到响应； 2. 页面简洁，用户能在 1min 内完成查询业务； |
| 编写人 |  |

该模块的查看已售商品用例分析表如下表 3-10 所示：

表 3-10 查看已购商品用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0009 | 用例名称 | 查看已购商品 |
| 用例描述 | 用户点击查看已购商品，系统跳转到已购商品展示页面，  展示已购商品。 | | |
| 执行者 | 用户 | | |
| 前置条件 | 用户必须已经登录 | | |
| 后置条件 | 展示已购商品 | | |
| 涉众利益 | 用户希望能按方格形式清晰展示商品，方便观看。 | | |
| 用例场景 | 已购商品展示界面 | | |
| 基本事件流 | 基本流程（Base Flow）   1. 用户点击查看商品按钮； 2. 系统检查用户是否登录； 3. 系统查询用户已售商品； 4. 系统将商品展示在页面； 扩展流程（Extend Flow）   （3）没有已购商品：  ① 系统提示没有已购商品  ② 用例结束  字段列表（Filed List）  商品信息＝商品编号+商品名称＋商品标题+价格+新旧程度+分类+日期+图片  业务规则（Business role）  （1）用户必须已经登录； | | |
| 特殊需求 | 1. 用户所有操作应在 3s 内得到响应； 2. 页面简洁，用户能在 1min 内完成查询业务； | | |
| 编写人 |  | | |

该模块的查看资产用例分析表如下表 3-11 所示：

表 3-11 查看资产用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0010 | 用例名称 | 查看资产 |
| 用例描述 | 用户点击查看资产，系统跳转到用户资产展示页面，展示  用户资产。 | | |
| 执行者 | 用户 | | |
| 前置条件 | 用户必须已经登录 | | |
| 后置条件 | 展示资产 | | |
| 涉众利益 | 用户希望能清晰准确地展示已购，已售资产 | | |
| 用例场景 | 资产展示界面 | | |
| 基本事件流 | 基本流程（Base Flow）   1. 用户点击查看资产按钮； 2. 系统检查用户是否登录； 3. 系统查询用户资产； 4. 系统将商品展示在页面； 字段列表（Filed List）   资产信息＝资产编号+已购资产+已售资产+用户编号业务规则（Business role）  （1）用户必须已经登录； | | |
| 特殊需求  （Special Requireme nt） | 1. 用户所有操作应在 3s 内得到响应； 2. 页面简洁，用户能在 1min 内完成查询业务； | | |
| 编写人 |  | | |

该模块的评价订单用例分析表如下表 3-12 所示：

表 3-12 评价订单用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0011 | 用例名称 | 评价订单 |
| 用例描述 | 用户点击评价按钮后输入评价内容，点击确认完成评价。 | | |
| 执行者 | 用户 | | |
| 前置条件 | 卖家与买家交易成功 | | |
| 后置条件 | 将用户评价保存到数据库中 | | |
| 涉众利益 | 用户希望评价后可以返回评价成功或失败 | | |
| 用例场景 | 商品评价界面 | | |

续表 3-12 评价订单用例描述

|  |  |
| --- | --- |
| 基本事件流 | 基本流程（Base Flow）   1. 用户点击评价按钮； 2. 系统显示输入框； 3. 用户填写评价信息； 4. 用户点击确认按钮； 5. 系统保存用户评价信息； 6. 系统返回评价成功或评价失败； 7. 若评价成功将评价信息显示在商品下方。扩展流程（Extend Flow）   （3）没有填写评价信息：  ① 系统提示没有填写评价信息  ② 用例结束  字段列表（Filed List）  评价＝评价编号＋评价内容+评价日期+用户编号＋商品编号+订单编号  业务规则（Business role）  （1）上传评价内容不能为空； |
| 特殊需求 | 1. 用户所有操作应在 3s 内得到响应； 2. 页面简洁，用户能在 1min 内完成评价业务； |
| 编写人 |  |

* + - 1. 用户信息模块

用户信息模块如图 2-12 所示：

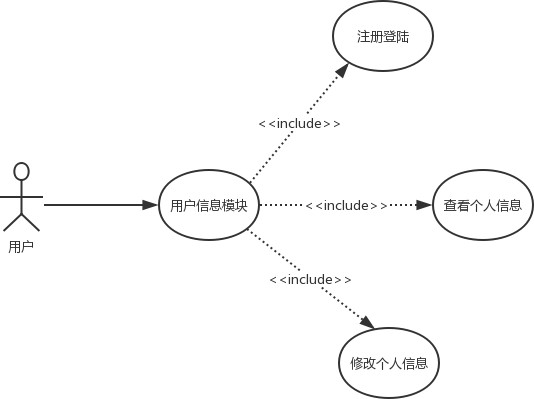


图 2-12 用户信息模块

该模块的注册登录用例分析表如下表 3-13 所示：

表 3-13 注册登录用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0012 | 用例名称 | 注册登录 |
| 用例描述 | 用户输入用户名密码登录，系统验证用户名和密码，若用户名不存在则跳转到注册页面，若密码错误则让用户重新  输入密码。 | | |
| 执行者 | 用户 | | |
| 前置条件 | 用户已经该打开网站 | | |
| 后置条件 | 展示首页 | | |
| 涉众利益 | 用户希望能返回注册登录成功或失败 | | |
| 用例场景 | 资产展示界面 | | |
| 基本事件流 | 基本流程（Base Flow）   1. 用户点击登录按钮； 2. 系统返回登录页面； 3. 用户输入用户名和密码； 4. 系统检查用户名； 5. 系统检查密码； 6. 系统返回首页；   扩展流程（Extend Flow）   1. 用户名不存在：   ① 系统返回注册页面  ② 用例结束   1. 密码错误：   ① 系统提示密码错误  ② 用例结束  字段列表（Filed List）  用户信息=用户编号+用户昵称+密码+真实姓名+性别+地址+头像  业务规则（Business role）   1. 用户昵称不能为空； 2. 用户密码不能为空； 3. 用户头像若不上传则设为默认头像； | | |
| 特殊需求 | 1. 用户所有操作应在 3s 内得到响应； 2. 页面简洁，用户能在 1min 内完成注册登录业务； | | |
| 编写人 |  | | |

该模块的查看个人信息用例分析表如下表 3-14 所示：

表 3-14 查看个人信息用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0013 | 用例名称 | 查看个人信息 |
| 用例描述 | 用户点击查看个人信息，系统跳转到用户个人信息展示页  面，展示用户个人信息。 | | |
| 执行者 | 用户 | | |
| 前置条件 | 用户必须已经登录 | | |
| 后置条件 | 展示用户个人信息 | | |
| 涉众利益 | 用户希望个人信息展示页面能隐藏密码 | | |
| 用例场景 | 个人信息展示界面 | | |
| 基本事件流 | 基本流程（Base Flow）   1. 用户点击查看个人信息； 2. 系统检查用户是否登录； 3. 系统查询用户个人信息； 4. 系统将个人信息展示在页面； 字段列表（Filed List）   用户信息=用户编号+用户昵称+密码+真实姓名+性别+地址+头像  业务规则（Business role）  （1）用户必须已经登录； | | |
| 特殊需求  （Special Requireme nt） | 1. 用户所有操作应在 3s 内得到响应； 2. 页面简洁，用户能在 1min 内完成查询业务； | | |
| 编写人 |  | | |

该模块的修改个人信息用例分析表如下表 3-15 所示：

表 3-15 修改个人信息用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0014 | 用例名称 | 修改个人信息 |
| 用例描述 | 用户点击修改按钮，在用户修改个人信息界面修改自己的  个人信息。点击确认后，系统保存修改信息并刷新页面。 | | |
| 执行者 | 用户 | | |
| 前置条件 | 用户必须已经登录 | | |
| 后置条件 | 将用户修改的信息保存到数据库中 | | |
| 涉众利益 | 用户希望修改后可以返回修改成功或失败 | | |
| 用例场景 | 用户修改个人信息界面 | | |

续表 3-15 修改个人信息用例描述

|  |  |
| --- | --- |
| 基本事件流 | 基本流程（Base Flow）   1. 用户修改个人信息； 2. 用户点击确认； 3. 系统将修改后的信息保存到数据库； 4. 系统返回修改结果 5. 系统刷新页面；   扩展流程（Extend Flow）  （3）用户昵称已存在：  ① 系统返回用户昵称已存在  ② 用例结束  字段列表（Filed List）  用户信息=用户编号+用户昵称+密码+真实姓名+性别+地址+头像  业务规则（Business role）  （1）用户昵称不能重复 |
| 特殊需求  （Special Requireme nt） | 1. 用户所有操作应在 3s 内得到响应； 2. 页面简洁，用户能在 1min 内完成修改业务； |
| 编写人 |  |

* + - 1. 商品信息管理模块

商品信息管理模块如图 2-13 所示：

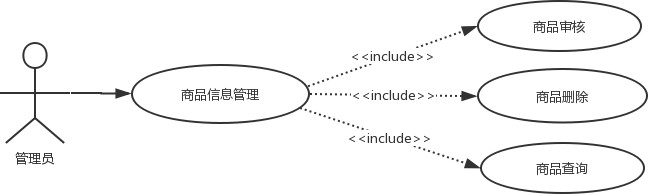


图 2-13 商品信息管理模块

该模块的商品审核用例分析表如下表 3-16 所示：

表 3-16 商品审核用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0015 | 用例名称 | 商品审核 |
| 用例描述 | 管理员对于用户上传的物品进行审核，若审核通过，则系  统添加一条商品记录，若审核不通过，删除该商品。 | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员必须已经登录 | | |
| 后置条件 | 返回商品添加结果 | | |
| 涉众利益 | 用户希望能返回商品添加成功或失败 | | |
| 用例场景 | 商品审核界面 | | |
| 基本事件流 | 基本流程（Base Flow）   1. 管理员对商品审核； 2. 管理员点击通过或不通过； 3. 若管理员点击通过，系统修改商品状态为已审核； 4. 若管理员点击不通过，系统删除商品信息； 字段列表（Filed List）   商品信息＝商品编号+商品名称＋商品标题+价格+新旧程度+分类+日期+图片  业务规则（Business role）  （1）管理员必须已经登录； | | |
| 特殊需求  （Special Requireme nt） | 1. 管理员所有操作应在 3s 内得到响应； 2. 页面简洁，用户能在 1min 内完成审核业务； | | |
| 编写人 |  | | |

该模块的商品删除用例分析表如下表 3-17 所示：

表 3-17 商品删除用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0016 | 用例名称 | 商品删除 |
| 用例描述 | 管理员选择要删除的商品，点击删除按钮，系统从数据库  中删除商品。 | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员必须已经登录 | | |
| 后置条件 | 将商品从数据库中删除 | | |
| 涉众利益 | 管理员希望可以返回删除成功或失败 | | |
| 用例场景 | 商品管理界面 | | |

续表 3-17 商品删除用例描述

|  |  |
| --- | --- |
| 基本事件流 | 基本流程（Base Flow）   1. 管理员点击删除按钮； 2. 系统弹出确认框； 3. 管理员点击确认按钮； 4. 系统从数据库中删除该条商品信息； 5. 系统返回删除成功； 6. 系统刷新返回删除成功后的页面； 字段列表（Filed List）   商品信息＝商品编号+商品名称＋商品标题+价格+新旧程度+分类+日期+图片  业务规则（Business role）  （1）管理员必须已经注册登录； |
| 特殊需求  （Special Requireme nt） | 1. 管理员所有操作应在 3s 内得到响应； 2. 页面简洁，管理员能在 1min 内完成删除业务； |
| 编写人 |  |

该模块的商品查询用例分析表如下表 3-18 所示：

表 3-18 商品查询用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0017 | 用例名称 | 商品查询 |
| 用例描述 | 管理员点击查看商品信息，系统跳转到商品展示页面，展  示商品详细信息。 | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员必须已经登录 | | |
| 后置条件 | 展示商品详细信息 | | |
| 涉众利益 | 管理员希望能清晰地展示展示所有匹配的商品 | | |
| 用例场景 | 商品信息展示界面 | | |
| 基本事件流 | 基本流程（Base Flow）   1. 管理员输入要查询的商品； 2. 系统模糊查询相匹配的项； 3. 系统返回匹配的商品详细信息； 4. 若没有相匹配的商品返回空； 字段列表（Filed List）   商品信息＝商品编号+商品名称＋商品标题+价格+新旧程  度+分类+日期+图片 | | |

续表 3-18 商品查询用例描述

|  |  |
| --- | --- |
|  | 业务规则（Business role）  （1）管理员必须已经登录； |
| 特殊需求  （Special Requireme nt） | 1. 管理员所有操作应在 3s 内得到响应； 2. 页面简洁，用户能在 1min 内完成查询业务； |
| 编写人 |  |

* + - 1. 商品分类管理模块

商品分类管理模块如图 2-14 所示：

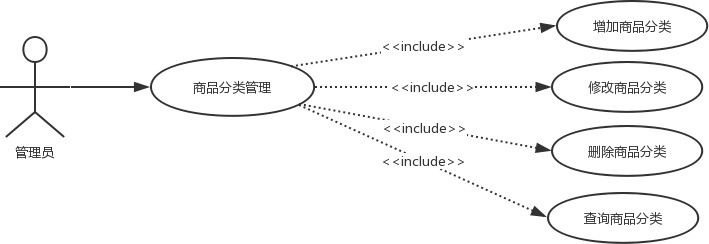


图 2-14 商品分类管理模块

该模块的增加商品分类用例分析表如下表 3-19 所示：

表 3-19 增加商品分类用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0018 | 用例名称 | 增加商品分类 |
| 用例描述 | 管理员点击增加分类按钮，跳转到增加分类页面，管理员  填写要增加的分类，确认后系统保存相应分类。 | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员必须已经登录 | | |
| 后置条件 | 将管理员上传的分类信息保存到数据库中 | | |
| 涉众利益 | 管理员希望可以返回增加成功或失败 | | |
| 用例场景 | 增加商品分类界面 | | |

续表 3-19 增加商品分类用例描述

|  |  |
| --- | --- |
| 基本事件流 | 基本流程（Base Flow）   1. 管理员点击增加商品分类按钮； 2. 系统显示增加商品分类页面； 3. 管理员填写商品分类信息； 4. 管理员点击保存按钮； 5. 系统保存管理员上传的商品分类信息； 6. 系统返回保存成功； 扩展流程（Extend Flow）   （2）该分类已经存在：  ① 系统提示该分类已经存在  ② 用例结束  字段列表（Filed List）  商品分类信息＝商品分类编号+商品分类名称+图片业务规则（Business role）  （1）上传商品分类名称不能已经存在； |
| 特殊需求 | 1. 用户所有操作应在 3s 内得到响应； 2. 页面简洁，用户能在 1min 内完成上传业务； |
| 编写人 |  |

该模块的修改商品分类用例分析表如下表 3-20 所示：

表 3-20 修改商品分类用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0019 | 用例名称 | 修改商品分类 |
| 用例描述 | 管理员点击修改按钮，在管理员修改商品分类信息界面修  改商品的分类信息。点击确认后，系统保存修改信息并刷新页面。 | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员必须已经登录 | | |
| 后置条件 | 将管理员修改的信息保存到数据库中 | | |
| 涉众利益 | 管理员希望修改后可以返回修改成功或失败 | | |
| 用例场景 | 管理员修改商品分类信息界面 | | |
| 基本事件流 | 基本流程（Base Flow）   1. 管理员修改商品分类信息； 2. 管理员点击确认； 3. 系统将修改后的信息保存到数据库； 4. 系统返回修改结果 | | |

续表 3-20 修改商品分类用例描述

|  |  |
| --- | --- |
|  | （5）系统刷新页面；  扩展流程（Extend Flow）  （3）商品分类已存在：  ① 系统返回商品分类已存在  ② 用例结束  字段列表（Filed List）  商品分类信息＝商品分类编号+商品分类名称+图片业务规则（Business role）  （1）商品分类信息不能重复 |
| 特殊需求  （Special Requireme nt） | 1. 用户所有操作应在 3s 内得到响应； 2. 页面简洁，用户能在 1min 内完成修改业务； |
| 编写人 |  |

该模块的商品分类删除用例分析表如下表 3-21 所示：

表 3-21 商品分类删除用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0020 | 用例名称 | 商品分类删除 |
| 用例描述 | 管理员选择要删除的商品分类，点击删除按钮，系统从数  据库中删除商品分类。 | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员必须已经登录 | | |
| 后置条件 | 将商品分类从数据库中删除 | | |
| 涉众利益 | 管理员希望可以返回删除成功或失败 | | |
| 用例场景 | 商品分类管理界面 | | |
| 基本事件流 | 基本流程（Base Flow）   1. 管理员点击删除按钮； 2. 系统弹出确认框； 3. 管理员点击确认按钮； 4. 系统从数据库中删除该条商品分类信息； 5. 系统返回删除成功； 6. 系统刷新返回删除成功后的页面； 字段列表（Filed List）   商品分类信息＝商品分类编号+商品分类名称+图片业务规则（Business role）  （1）管理员必须已经登录； | | |

续表 3-21 商品分类删除用例描述

|  |  |
| --- | --- |
| 特殊需求 | 1. 管理员所有操作应在 3s 内得到响应； 2. 页面简洁，管理员能在 1min 内完成删除业务； |
| 编写人 |  |

该模块的商品分类查询用例分析表如下表 3-22 所示：

表 3-22 商品分类查询用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0021 | 用例名称 | 商品分类查询 |
| 用例描述 | 管理员点击查看商品分类信息，系统跳转到商品分类展示  页面，展示商品分类信息。 | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员必须已经登录 | | |
| 后置条件 | 展示商品分类信息 | | |
| 涉众利益 | 管理员希望能清晰地展示展示所有匹配的商品分类 | | |
| 用例场景 | 商品分类信息展示界面 | | |
| 基本事件流 | 基本流程（Base Flow）   1. 管理员输入要查询的商品分类信息； 2. 系统模糊查询相匹配的项； 3. 系统返回匹配的商品分类信息； 4. 若没有相匹配的商品分类返回空； 字段列表（Filed List）   商品分类信息＝商品分类编号+商品分类名称+图片业务规则（Business role）  （1）管理员必须已经登录； | | |
| 特殊需求 | （1）管理员所有操作应在 3s 内得到响应； | | |
| 编写人 |  | | |

* + - 1. 用户信息管理模块

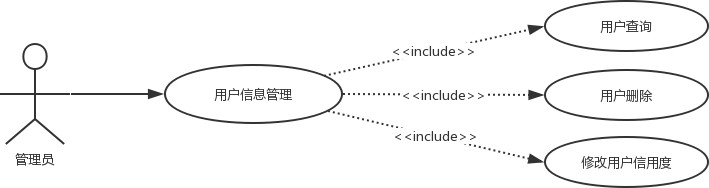
商品分类管理模块如图 2-15 所示：

图 2-15 用户信息管理

该模块的用户查询用例分析表如下表 3-23 所示：

表 3-23 用户查询用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0022 | 用例名称 | 用户查询 |
| 用例描述 | 管理员在搜索框中输入要查询的用户，系统返回用户的信  息并显示。 | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员必须已经登录 | | |
| 后置条件 | 将查询到的用户信息在页面上显示 | | |
| 涉众利益 | 管理员希望可以返回查询成功或失败 | | |
| 用例场景 | 用户查询界面 | | |
| 基本事件流 | 基本流程（Base Flow）   1. 管理员输入要查询的用户； 2. 系统模糊查询相匹配的项； 3. 系统返回匹配的用户详细信息； 4. 若没有相匹配的用户返回空； 字段列表（Filed List）   用户信息=用户编号+用户昵称+密码+真实姓名+性别+地址+头像 | | |
| 特殊需求 | 1. 用户所有操作应在 3s 内得到响应； 2. 页面简洁，用户能在 1min 内完成查询业务； | | |
| 编写人 |  | | |

该模块的用户删除用例分析表如下表 3-24 所示：

表 3-24 用户删除用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0023 | 用例名称 | 用户删除 |
| 用例描述 | 管理员选择要删除的用户，点击删除按钮，系统从数据库  中删除用户信息。 | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员必须已经登录 | | |
| 后置条件 | 将用户从数据库中删除 | | |
| 涉众利益 | 管理员希望可以返回删除成功或失败 | | |
| 用例场景 | 用户管理界面 | | |

续表 3-24 用户删除用例描述

|  |  |
| --- | --- |
| 基本事件流 | 基本流程（Base Flow）   1. 管理员点击删除按钮； 2. 系统弹出确认框； 3. 管理员点击确认按钮； 4. 系统从数据库中删除该条用户信息； 5. 系统返回删除成功； 6. 系统刷新返回删除成功后的页面； 字段列表（Filed List）   用户信息=用户编号+用户昵称+密码+真实姓名+性别+地址+头像  业务规则（Business role）  （1）管理员必须已经登录； |
| 特殊需求 | 1. 管理员所有操作应在 3s 内得到响应； 2. 页面简洁，管理员能在 1min 内完成删除业务； |
| 编写人 |  |

该模块的修改用户信用度用例分析表如下表 3-25 所示：

表 3-25 修改用户信用度用例描述

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0024 | 用例名称 | 修改用户信用度 |
| 用例描述 | 管理员点击违规按钮，系统按公式修改用户信用度并保存  到数据库。 | | |
| 执行者 | 管理员 | | |
| 前置条件 | 管理员必须已经登录 | | |
| 后置条件 | 将修改后的信用度信息保存到数据库中 | | |
| 涉众利益 | 管理员希望修改后可以返回修改成功或失败 | | |
| 用例场景 | 管理员修改用户信用度界面 | | |
| 基本事件流 | 基本流程（Base Flow）   1. 管理员点击违规按钮； 2. 系统将按公式修改用户信用度； 3. 系统将修改结果保存到数据库； 4. 系统刷新页面；   扩展流程（Extend Flow）  （3）用户信用度达到下界：  ① 系统禁止用户再发布商品  ② 用例结束 | | |

续表 3-25 修改用户信用度用例描述

|  |  |
| --- | --- |
|  | 字段列表（Filed List）  用户信息=用户编号+用户昵称+密码+真实姓名+性别+地址+头像  业务规则（Business role）  （1）用户信用度不能为负值 |
| 特殊需求  （Special Requireme nt） | 1. 用户所有操作应在 3s 内得到响应； 2. 页面简洁，用户能在 1min 内完成修改业务； |
| 编写人 |  |

## 非功能需求

* + 1. 环境需求

服务器端环境如表 2-2 所示：

表 2-2 服务器端环境

|  |  |
| --- | --- |
| 操作系统 | Ubuntu |
| Web 服务器 | Apache tomcat |
| 数据库 | mysql |

客户端环境如表 2-3 所示：

表 2-3 客户端环境

|  |  |
| --- | --- |
| 操作系统 | Ubuntu |
| 浏览器 | google |
| 分辨率 | 1024×768 像素 |

* + 1. 性能需求

1. 易操作性

对用户来说需要页面操作简单、界面简洁、新用户能够在一小时内快速学会使用该平台。

1. 可维护性

系统能做到使自身在数据损坏、丢失等情况下将备份数据倒回，实现数据的恢复。提供对系统自身的集中操作维护的功能。

1. 系统稳定性

系统有效工作时间要求≥99.5%。web 服务持续稳定工作时间≥3 天。

1. 响应速度

用户必须 0-5 秒在响应用户请求。

* + 1. 安全需求

1. 容错性

当用户输入或误操作导致非法数据产生时，系统应具有一定的容错机制。在这种情况下，系统应给出提示，提示用户重新输入或者进行自动的修复校正，而不应该使系统直接崩溃。

1. 系统内部安全

系统必须确保授权用户的合法使用，不同的角色有不同的权限。

1. 系统运行安全

系统支持定期自动和手动数据备份，能够在数据损坏或数据丢失的情况下找回数据，实现一定程度的数据恢复。

# 总体设计

## 系统设计的原则

本系统的设计将严格遵循以下的设计原则：

1. 实用性原则。在进行该系统的设计时，我将系统的实用性要求放在了首位。该系统应该切合大学生的实际需求，致力于为大学生带来方便，解决其所需。
2. 安全性原则。系统提供安全机制以防止非法授权操作，为各子系统提供严格灵活的用户和权限划分，对于数据库管理，建立起完备的数据备份恢复机制。
3. 可扩展性原则。对后续系统的升级，系统的所有功能模块的设计都预留了扩展接口。

## 系统体系结构设计

B/S 结构，也成为浏览器(Browser)/服务器(Server)结构，浏览器(Browser) 是一个软件，客户端的应用程序都要在浏览器上运行，向用户展示。服务器(Server)是指开发好的应用程序需要部署在服务器上运行，客户端通过浏览器， 输入指定的 IP 地址，即可访问应用程序。B/S 结构的特点有以下几点：

1. 客户端免安装

应用程序只需要部署在服务器即可。这样可以降低安装带来的麻烦和大量的人力成本。

1. 更新快

由于这种结构的应用程序只需要安装在服务器上，也就是说，只需要更新服务器的应用程序，客户端就可以使用更新后的应用程序。

1. 容易集中控制

如果需要停止使用应用程序，只需要将服务器端的应用程序停止运行或卸载即可。

1. 安全风险大

由于这种结构的应用程序，客户端是通过浏览器对应用程序访问，对于应用程序来讲，开放面比较宽，如果部署在互联网上，风险会更大，人们可以利用浏览器的漏洞对系统进行攻击。这种结构毕竟是应用程序与其他程序(浏览器)之间的通讯，而浏览器由于不是应用程序的组成，开发方无法控制浏览器，风险就必然增加。

校园二手交易平台结构图如图 3-1 所示：

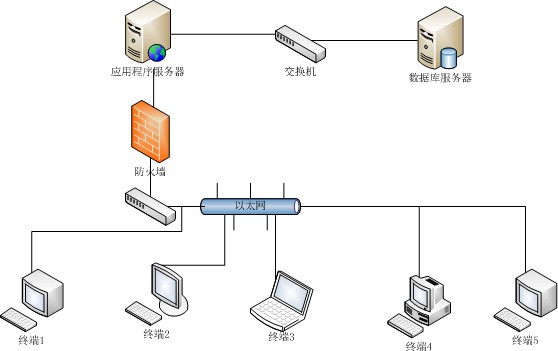


图 3-1 校园二手交易平台结构图

校园二手交易平台技术架构设计如图 3-2 所示：

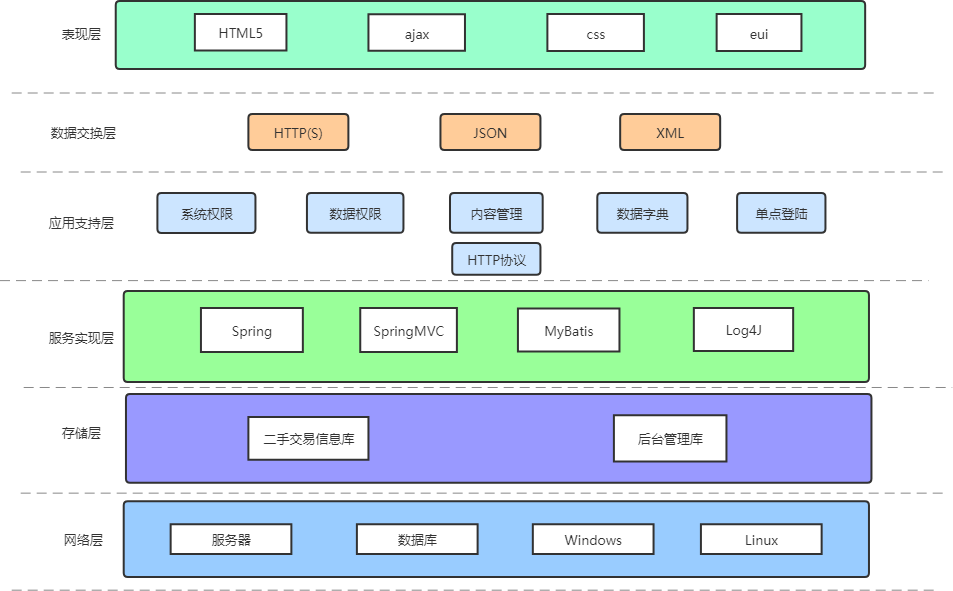


图 3-2 校园二手交易平台技术架构图

## 系统功能结构设计

子系统关系图如图 3-3 所示：

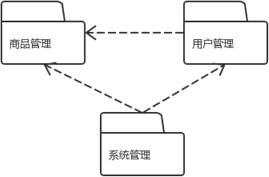


图 3-3 子系统关系图

* + 1. 商品管理子系统模块构建设计

商品管理子系统：包括商品信息发布、商品信息修改、商品搜索、商品下架、商品分类查看五个包模块。商品管理子系统关系图如图 3-4 所示：

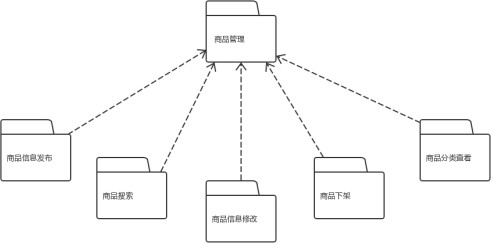


图 3-4 商品管理子系统关系图

1. 商品信息发布：用户上传商品的详细信息，提交发布请求。
2. 商品信息修改：用户修改商品的详细信息，提交修改请求。
3. 商品搜索：用于在搜索框输入关键字，系统在全部商品中进行模糊查询， 并显示查询结果。
4. 商品下架：卖家用户对正在出售的商品进行下架操作，下架后的商品仍可被查询到但无法被购买。
5. 商品分类查看：用户、游客查看全部商品分类。
   * 1. 用户管理子系统模块构建设计

用户管理子系统：包括用户收藏夹管理、订单管理、用户信息管理三个包模块，其中订单管理包包括已购商品订单、已售商品订单及评价。用户管理子系统关系图如图 3-5 所示：

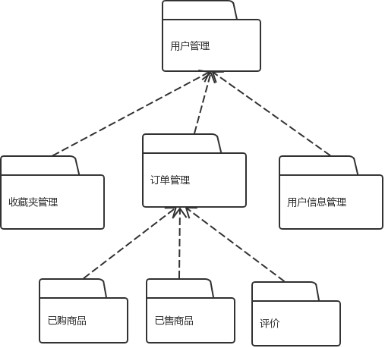


图 3-5 用户管理子系统关系图

1. 用户收藏夹管理：用户添加新商品到收藏夹，用户取消收藏收藏夹里的商品，用户查询收藏夹里的商品。
2. 已购商品：用户可以查看自己已购商品的所有订单的详细信息，以及订单的资产支出情况。
3. 已售商品：用户可以查看自己已售商品的所有订单的详细信息，以及订单的资产收入情况。
4. 评价：用户可以对自己完成的已购订单和已售订单进行评价。
5. 用户信息管理：游客可以进行注册，注册成功的用户进行登录后可以对自己的个人信息进行查看和修改。
   * 1. 系统管理子系统模块构建设计

系统管理子系统：包括对商品信息管理、对商品分类管理、对用户信息管理。系统管理子系统关系图如图 3-6 所示：

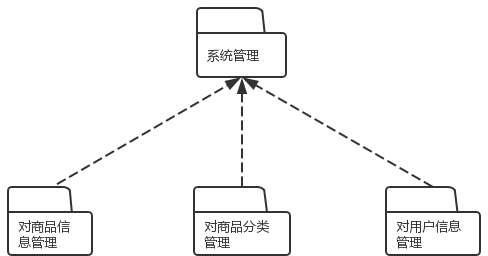


图 3-6 系统管理子系统关系图

1. 商品信息管理：管理员对用户发布的商品进行审核，若审核通过则发布成功，若审核失败则删除商品。管理员也可以对商品进行查询。
2. 商品分类管理:管理员在后台增加商品的分类，删除商品的分类，以及查询商品的分类来实现对商品分类的管理。
3. 用户信息管理：管理员可对平台内的所有用户进行查询，该权限仅限管理员。对于信用度极低且存在违规操作，发布含有违规内容的用户进行删除。若用户存在违规操作，管理员可通过降低该用户的信用度对用户进行管理。

# 数据库设计

## 概念结构设计

### 设计思路

采用 E-R 方法进行数据库概念设计，可以分成 3 步进行：首先设计局部 E-R 模式，然后把各局部 E-R 模式综合成一个全局的 E-R 模式，最后对全局 E-R 模式进行优化，得到最终的 E-R 模式，即概念模式。本系统概念结构设计：

* + - 1. 订单表（订单编号，卖家姓名，买家姓名，买家编号，商品编号，总价，单价，商品数量，订单状态，生成时间，发货时间，成交时间）；
      2. 用户表（用户编号，用户名称，用户密码，用户年龄，用户性别，用户联系电话，用户邮箱，用户收货地址）；
      3. 管理员表（管理员编号，管理员名称，管理员密码）；
      4. 商品信息表（商品编号，商品名称，商品价格，商品数量，用户编号， 商品分类编号，商品图片，商品简介，商品新旧程度，商品原价，商品距离下架日期，商品状态）；
      5. 商品类型表（商品分类编号，商品分类名称）；
      6. 收藏夹表（收藏夹编号，用户名称，添加商品的时间，商品编号）；
      7. 评论表（评价编号，进行评价的用户名，评价的内容，对订单的打分， 评价提交的时间，评价对应的订单号）；
      8. 收货地址表（收货地址编号，卖家姓名，买家姓名，买家收货地址， 买家联系方式，是否为默认的收货地址）。

### E-R 图

1. 本系统的实体关系图，如图 4-1 所示：

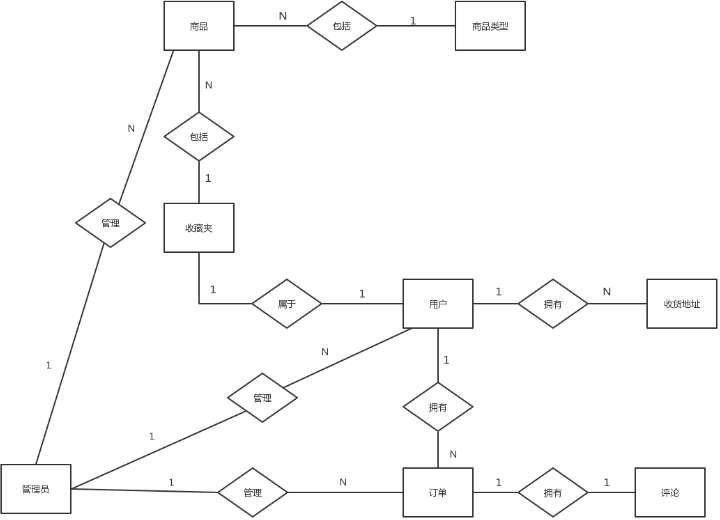


图 4-1 实体关系图

1. 订单实体图，如图 4-2 所示： 实体：订单

属性：订单编号，卖家姓名，买家姓名，买家编号，商品编号，总价，单价，商品数量，订单状态，生成时间，发货时间，成交时间

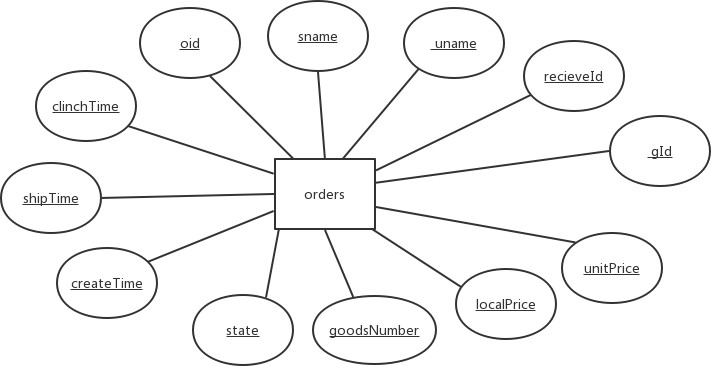


图 4-2 订单实体图

1. 用户实体图，如图 4-3 所示： 实体：用户

属性：用户编号，用户名称，用户密码，用户年龄，用户性别，用户联系电话， 用户邮箱，用户收货地址

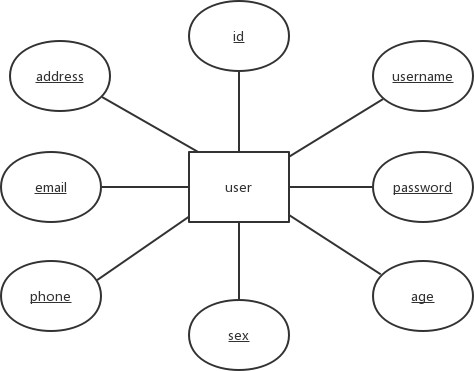


图 4-3 用户实体图

1. 管理员实体图，如图 4-4 所示： 实体：管理员

属性：管理员编号，管理员名称，管理员密码

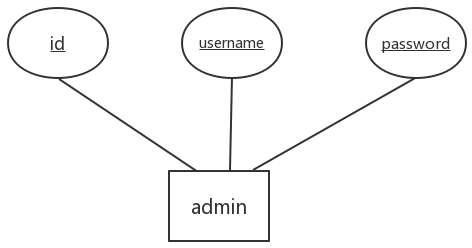


图 4-4 管理员实体图

1. 商品信息实体图，如图 4-5 所示： 实体：商品信息

属性：商品编号，商品名称，商品价格，商品数量，用户编号，商品分类编号，

商品图片，商品简介，商品新旧程度，商品原价，商品距离下架日期，商品状态

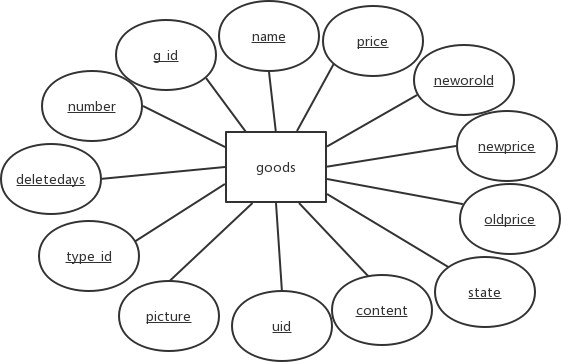


图 4-5 商品信息实体图

1. 商品类型实体图，如图 4-6 所示： 实体：商品类型

属性：商品分类编号，商品分类名称

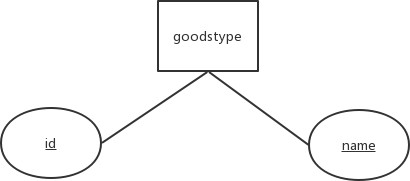


图 4-6 商品类型实体图

1. 收藏夹实体图，如图 4-7 所示： 实体：收藏夹

属性：收藏夹编号，用户名称，添加商品的时间，商品编号

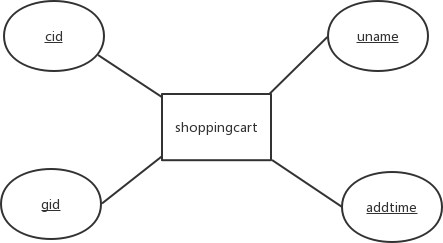


图 4-7 收藏夹实体图

1. 评论实体图，如图 4-8 所示： 实体：评论

属性:评价编号，进行评价的用户名，评价的内容，对订单的打分，评价提交的时间，评价对应的订单号

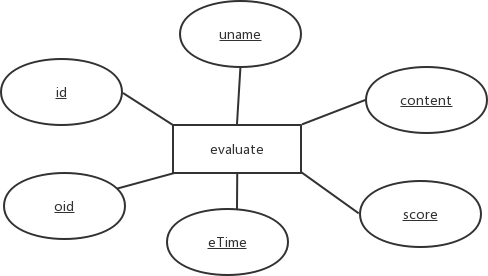


图 4-8 评论实体图

1. 收货地址实体图，如图4-9所示： 实体：收货地址

属性：收货地址编号，卖家姓名，买家姓名，买家收货地址，买家联系方式，是否为默认的收货地址

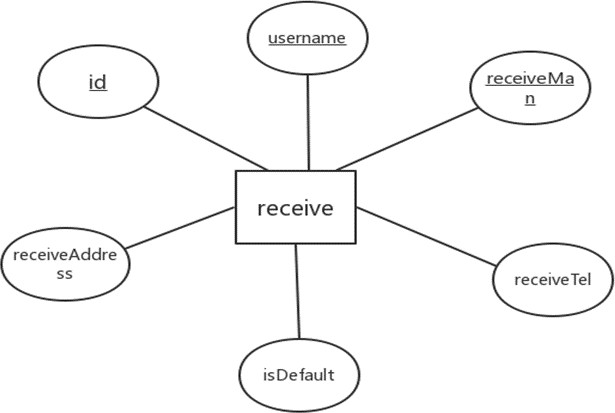


图4-9 收货地址实体图

## 逻辑结构设计

### 设计思路

把概念结构设计阶段设计好的基本 E-R 图转换为与选用 DBMS 产品所支持的数据模型相符合的逻辑结构。设计的步骤：将概念结构转化为一般的关系、网状、层次模型，然后将转换来的关系、网状、层次模型向特定 DBMS 支持下的数据模型转换，再对数据模型进行优化。

### 逻辑模型

1. 订单表如表 4-1 所示：

表 4-1 订单表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 数据类型 | 字段  长度 | 是否  主键 | 说明 |
| 1 | oId | 整型 | 11 | 是 | 订单编号 |
| 2 | sname | 字符串 | 255 | 否 | 卖家姓名 |
| 3 | uname | 字符串 | 255 | 否 | 买家姓名 |
| 4 | receiveId | 整型 | 11 | 否 | 买家编号 |
| 5 | gId | 整型 | 11 | 否 | 商品编号 |
| 6 | unitPrice | 浮点型 |  | 否 | 总价 |
| 7 | localPrice | 浮点型 |  | 否 | 单价 |
| 8 | goodsNumber | 整型 | 11 | 否 | 商品数量 |
| 9 | state | 整型 | 11 | 否 | 订单状态 |
| 10 | createTime | 日期 |  | 否 | 生成时间 |
| 11 | shipTime | 日期 |  | 否 | 发货时间 |
| 12 | clichTime | 日期 |  | 否 | 成交时间 |

1. 用户表如表 4-2 所示：

表 4-2 用户表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 数据类型 | 字段  长度 | 是否  主键 | 说明 |
| 1 | id | 整型 | 11 | 是 | 用户编号 |
| 2 | username | 字符串 | 255 | 否 | 用户名称 |
| 3 | password | 字符串 | 255 | 否 | 用户密码 |
| 4 | age | 整型 | 11 | 否 | 用户年龄 |
| 5 | sex | 字符串 | 255 | 否 | 用户性别 |
| 6 | phone | 字符串 | 255 | 否 | 用户联系电话 |
| 7 | email | 字符串 | 255 | 否 | 用户邮箱 |
| 8 | address | 字符串 | 255 | 否 | 用户收货地址 |

1. 管理员表如表 4-3 所示：

表 4-3 管理员表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 数据类型 | 字段  长度 | 是否  主键 | 说明 |
| 1 | id | 整型 | 11 | 是 | 管理员编号 |
| 2 | name | 字符串 | 255 | 否 | 管理员名称 |
| 3 | password | 字符串 | 255 | 否 | 管理员密码 |

1. 商品信息表如表 4-4 所示：

表 4-4 商品信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 数据类型 | 字段  长度 | 是否  主键 | 说明 |
| 1 | g\_id | 整型 | 11 | 是 | 商品编号 |
| 2 | name | 字符串 | 255 | 否 | 商品名称 |
| 3 | price | 浮点型 |  | 否 | 商品价格 |
| 4 | number | 字符串 | 255 | 否 | 商品数量 |
| 5 | uid | 整型 | 11 | 否 | 用户编号 |
| 6 | type\_id | 整型 | 11 | 否 | 商品分类编号 |
| 7 | picture | 字符串 | 255 | 否 | 商品图片 |
| 8 | content | 字符串 | 255 | 否 | 商品简介 |
| 9 | neworold | 字符串 | 255 | 否 | 商品新旧程度 |
| 10 | oldprice | 字符串 | 255 | 否 | 商品原价 |
| 11 | deletedays | 字符串 | 255 | 否 | 商品距离下架日期 |
| 12 | state | 字符串 | 255 | 否 | 商品状态 |

1. 商品类型表如表 4-5 所示：

表 4-5 商品类型表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 数据类型 | 字段  长度 | 是否  主键 | 说明 |
| 1 | id | 整型 | 11 | 是 | 商品分类编号 |
| 2 | name | 字符串 | 255 | 否 | 商品分类名称 |

1. 收藏夹表如表 4-6 所示：

表 4-6 收藏夹表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 数据类型 | 字段  长度 | 是否  主键 | 说明 |
| 1 | cid | 整型 | 11 | 是 | 收藏夹编号 |
| 2 | uname | 字符串 | 255 | 否 | 用户名称 |
| 3 | addtime | 日期 |  | 否 | 添加商品的时间 |
| 4 | gid | 整型 | 11 | 否 | 商品编号 |

1. 评论表如表 4-7 所示：

表 4-7 评论表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 数据类型 | 字段  长度 | 是否  主键 | 说明 |
| 1 | id | 整型 | 11 | 是 | 评价编号 |
| 2 | uname | 字符串 | 255 | 否 | 进行评价的用户名 |
| 3 | content | 字符串 | 255 | 否 | 评价的内容 |
| 4 | score | 整型 | 11 | 否 | 对订单的打分 |
| 5 | eTime | 日期 |  | 否 | 评价提交的时间 |
| 6 | oid | 整型 | 11 | 否 | 评价对应的订单号 |

1. 收货地址表如表 4-8 所示：

表 4-8 收获地址表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 数据类型 | 字段  长度 | 是否  主键 | 说明 |
| 1 | id | 整型 | 11 | 是 | 收货地址编号 |
| 2 | username | 字符串 | 255 | 否 | 卖家姓名 |
| 3 | receiveMan | 字符串 | 255 | 否 | 买家姓名 |
| 4 | receiveAddress | 字符串 | 255 | 否 | 买家收货地址 |
| 5 | receiveTel | 字符串 | 255 | 否 | 买家联系方式 |
| 6 | isDefault | 整型 | 11 | 否 | 是否为默认的收货地址 |

## 物理结构设计

### 存取方式

本平台数据库使用 MySQL，采用 B+树索引存取方法。为了满足数据库的实体完整性，本平台数据库为每个表都建立起了一个主键。在用户表中，为用户表中的用户名建立唯一索引，用于登录时使用；在管理员表中，为管理员表的用户名建立唯一索引。在商品类型表中，为类型名建立唯一索引。

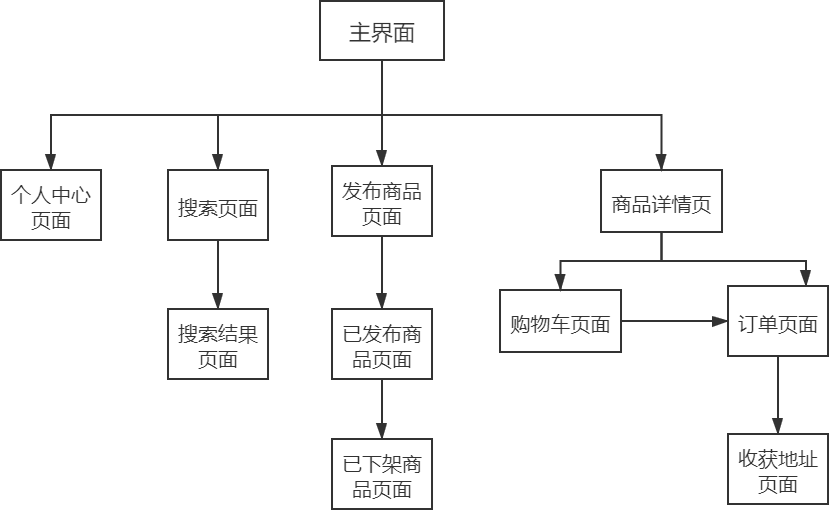
### 存储结构

确定数据库物理结构主要指确定数据的存放位置和存储结构，包括确定关系、索引、聚簇、日志、备份等的存储安排和存储结构，确定系统配置等。确定数据的存放位置和存储结构要综合考虑存取时间、存储空间利用率和维护代价三方面 的因素。这三个方面常常是相互矛盾的，因此需要进行权衡选择一个折中方案。

# 界面设计

## 界面关系图或工作流图

页面关系图如图 5-1 所示：



## 界面设计成果

图 5-1 页面关系图

* + 1. **主界面**
       1. 首页 1

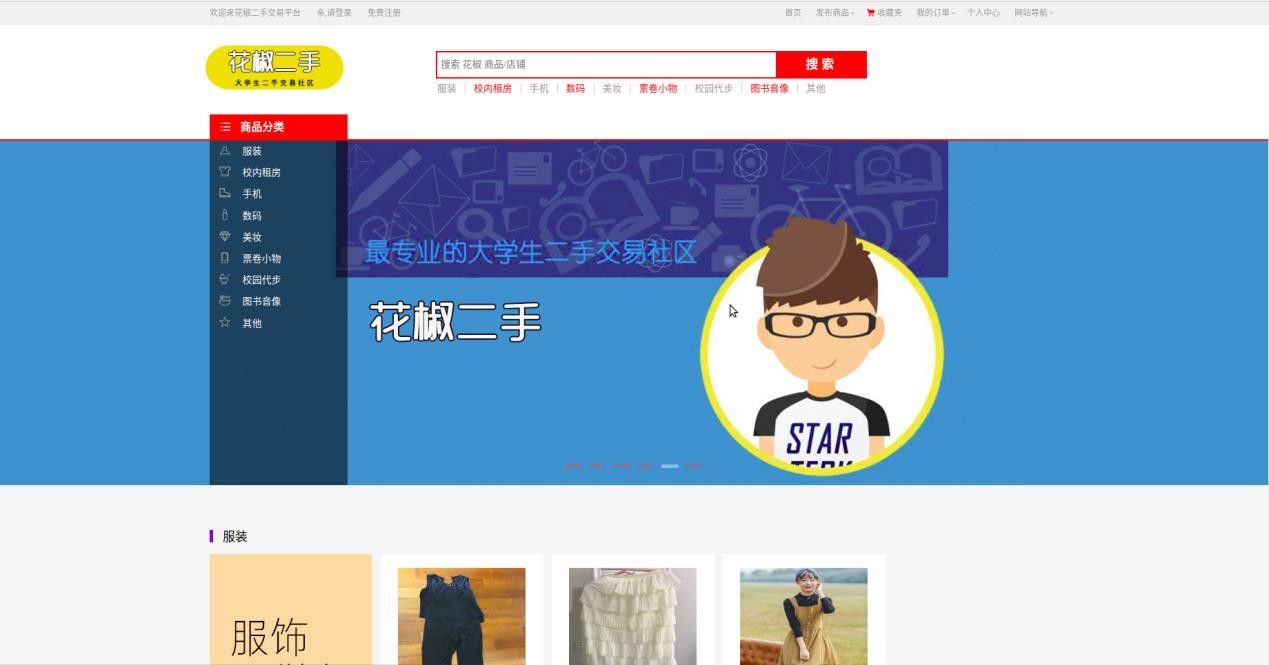
首页负责展示所有商品以及商品分类，如图 5-2 所示：

图 5-2 首页

* + - 1. 首页 2

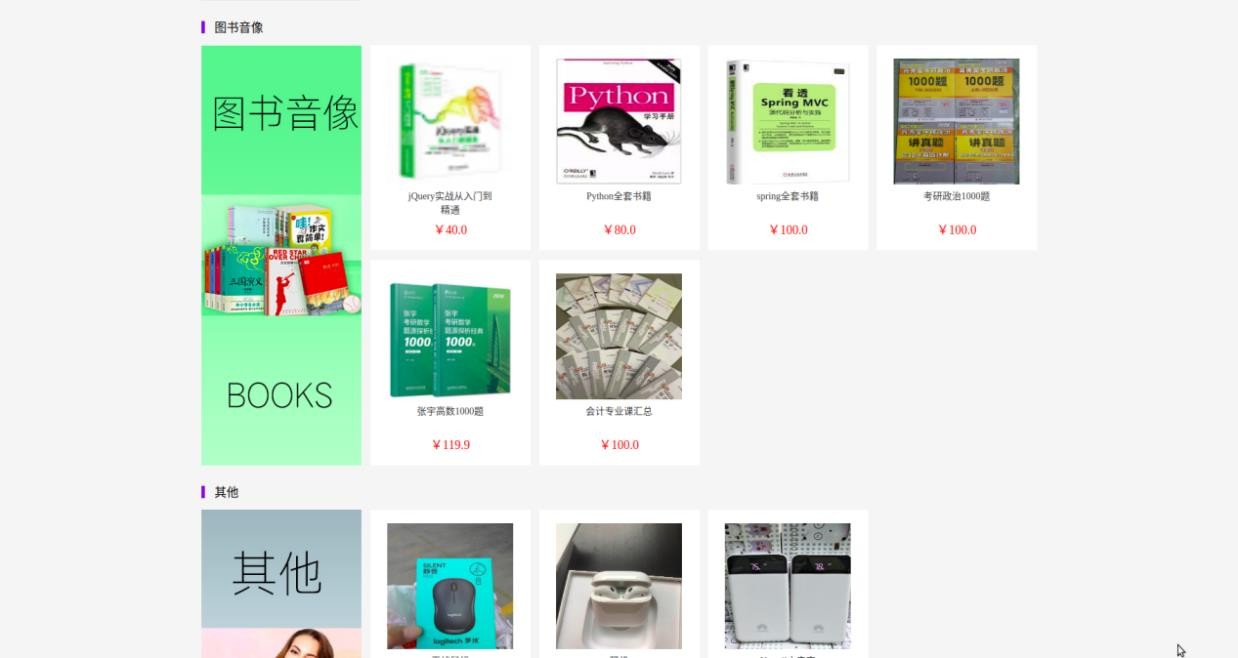
首页负责展示所有商品以及商品分类，如图 5-3 所示：

图 5-3 首页

### 子界面

* + - 1. 登录页面

登陆页面包括用户名和密码，用户必须先登录才能完成购买商品等操作。如图 5-4 所示：

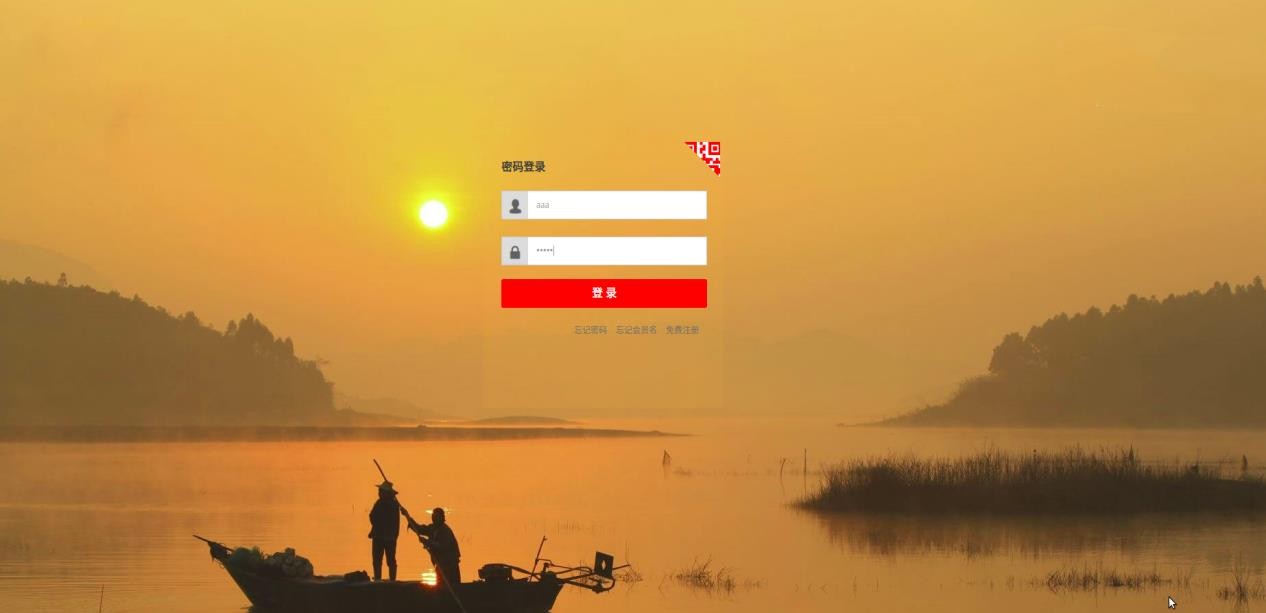
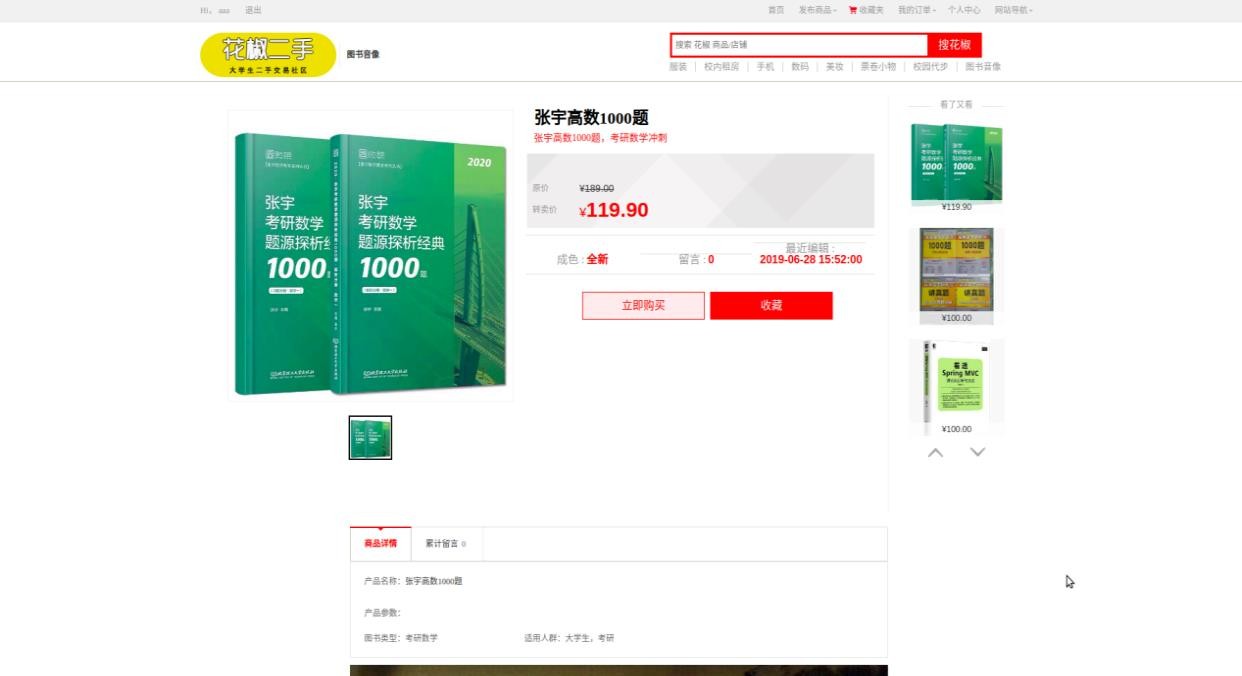


图 5-4 登录页面

* + - 1. 商品详情页面

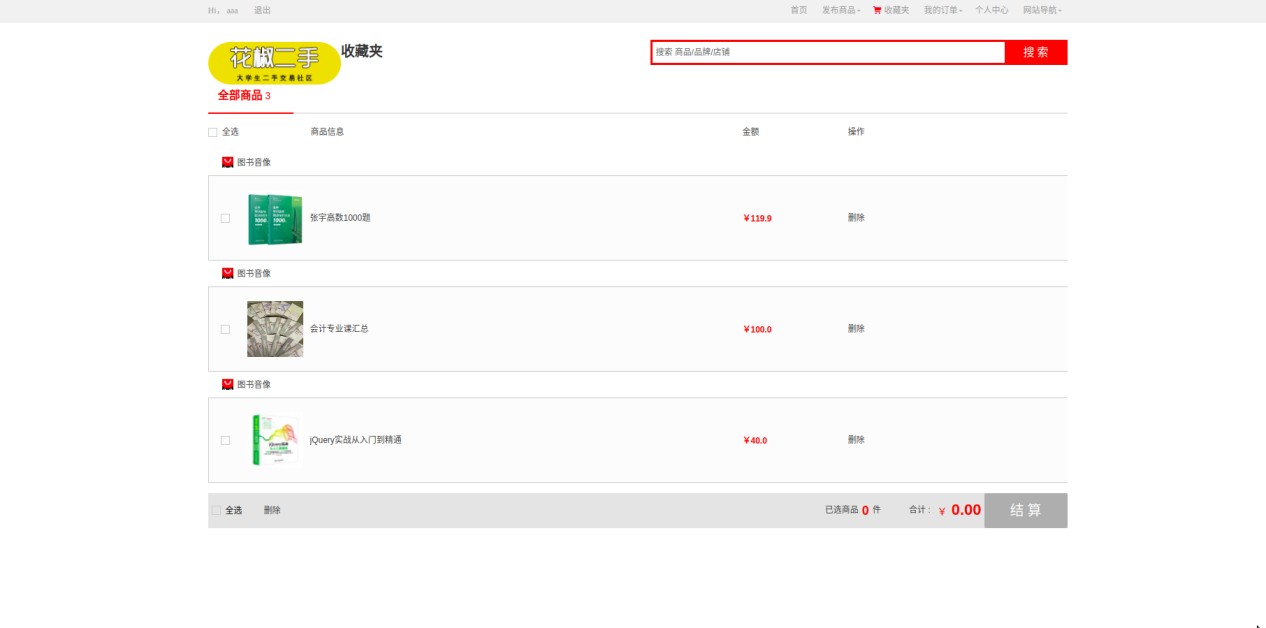
商品详情页展示商品的详细信息，包括价格，成色等。用户可以在商品详情页选择加入购物车或购买。如图 5-5 所示：



* + - 1. 收藏夹页面

图 5-5 商品详情页面

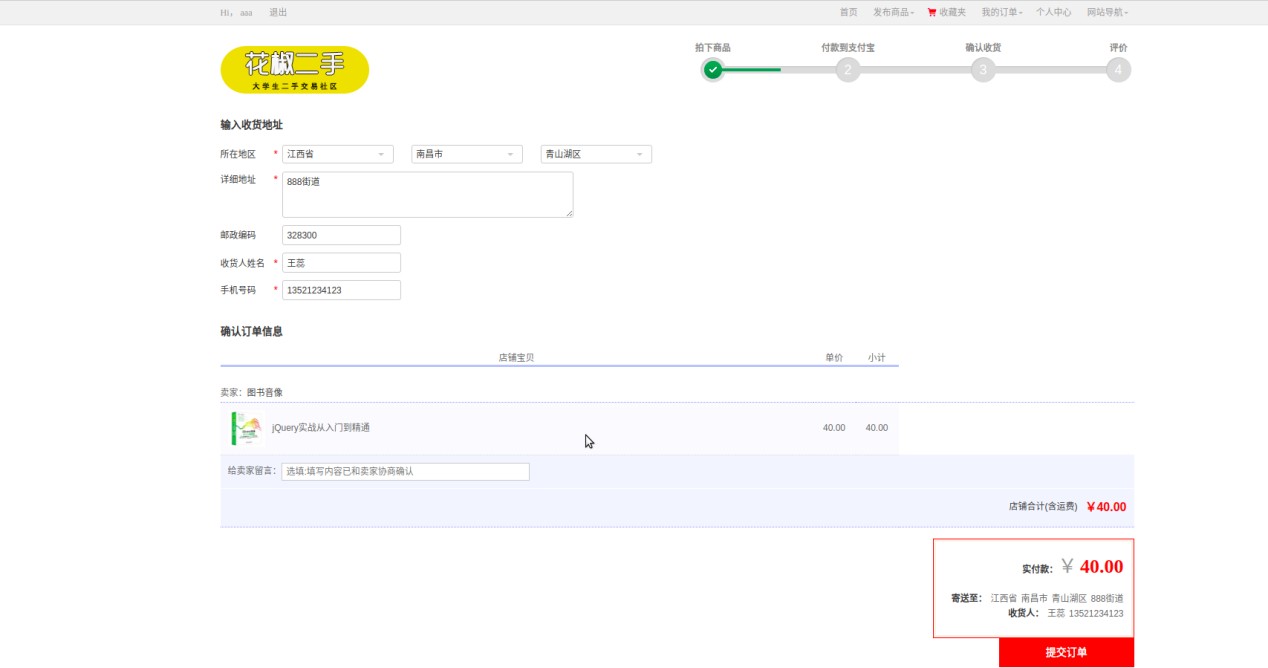
收藏夹页面展示个人收藏的所有商品。如图 5-6 所示：



* + - 1. 收获地址页面

图 5-6 收藏夹页面

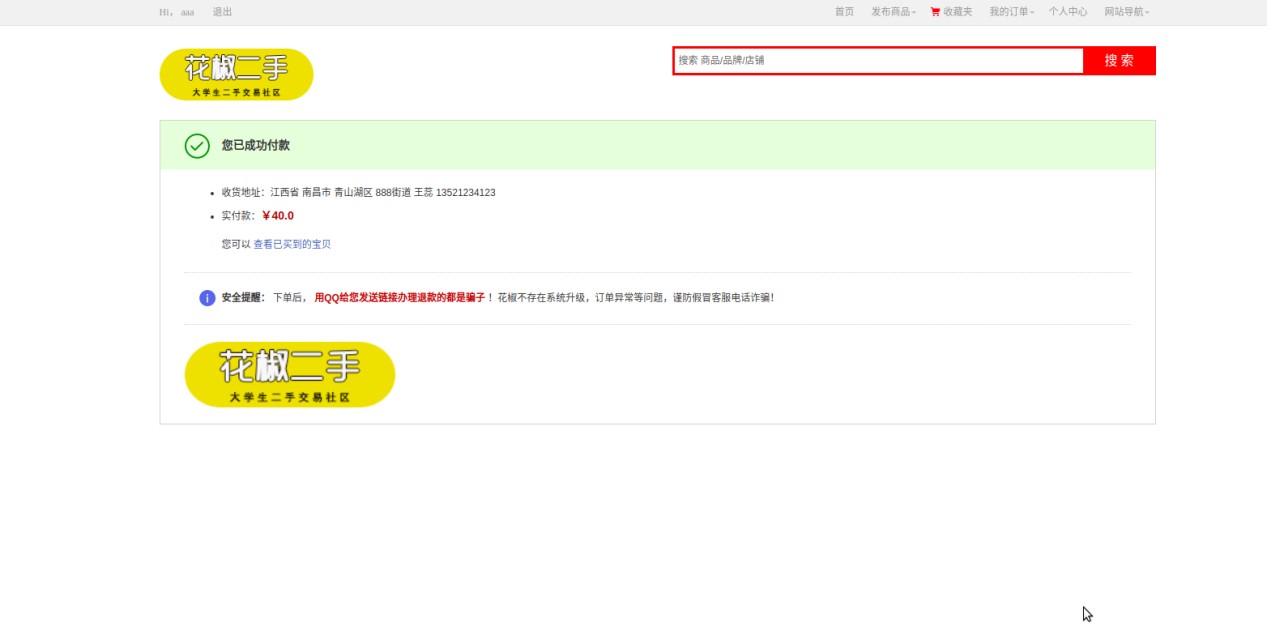
收获地址页面在用户拍下商品后用于用户填写个人的收货地址。如图 5-7 所示：



* + - 1. 付款成功页面

图 5-7 收获地址页面

用户提交订单成功后会跳转到付款成功页面，提示用户已经购买成功。如图

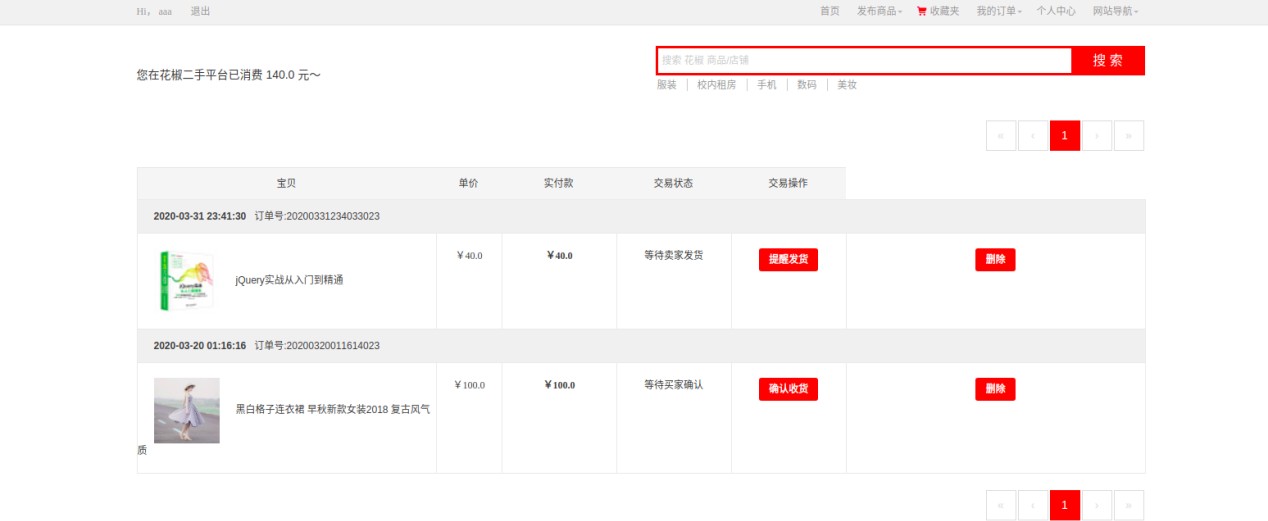
5-8 所示：

* + - 1. 订单页面

图 5-8 付款成功页面

订单页面展示用户已经购买的商品，以及各商品目前所处的状态。如图 5-9

所示：



* + - 1. 个人中心页面

图 5-9 订单页面

个人中心页面展示用户的个人信息。用户也可以在个人中心页面修改个人信息。如图 5-10 所示：

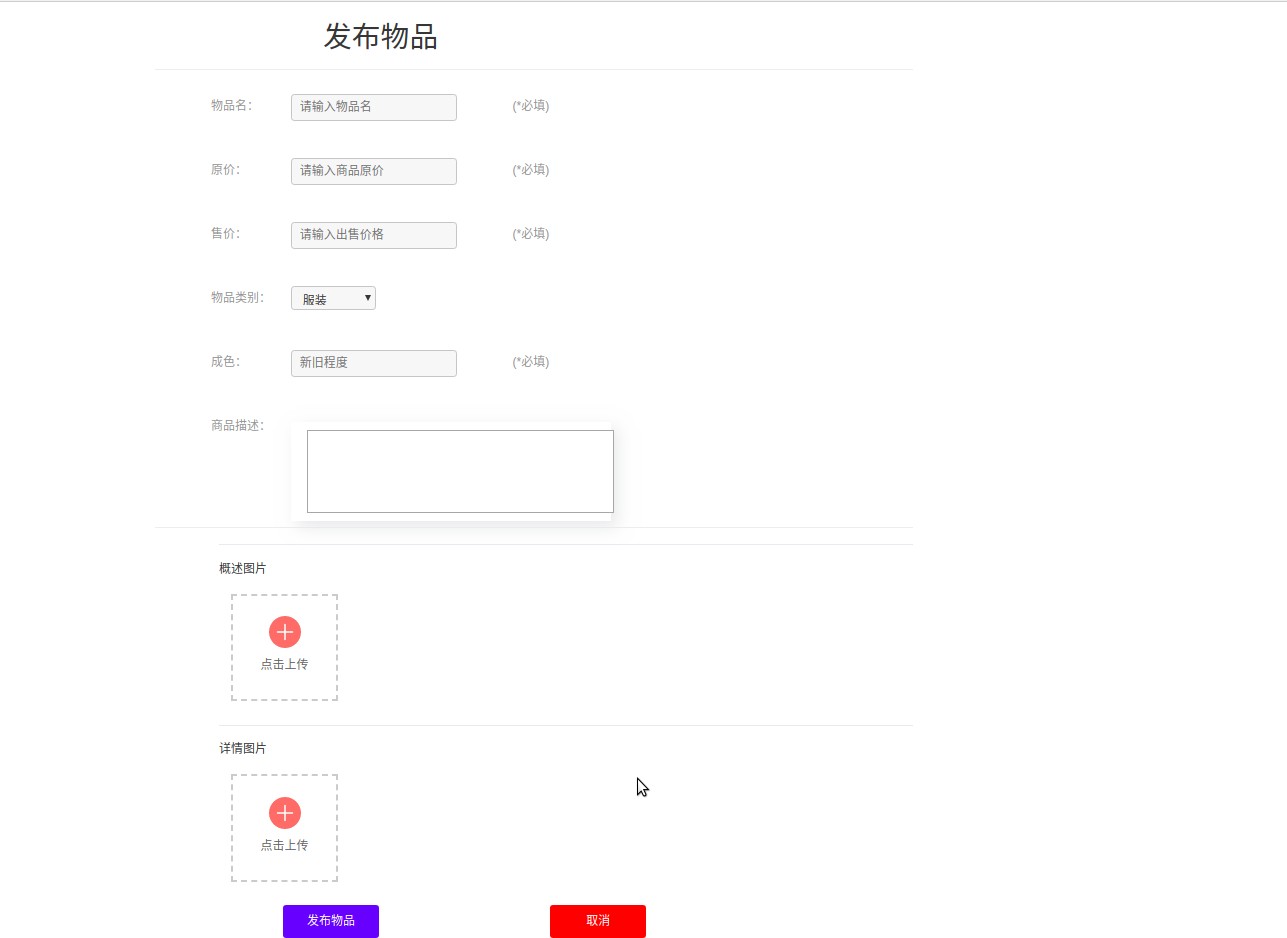


* + - 1. 商品发布页面

图 5-10 个人中心页面

商品发布页面用于用户发布商品，包括商品的名称，价格，图片等。如图 5-11

所示：



* + - 1. 已发布商品页面

图 5-11 商品发布页面

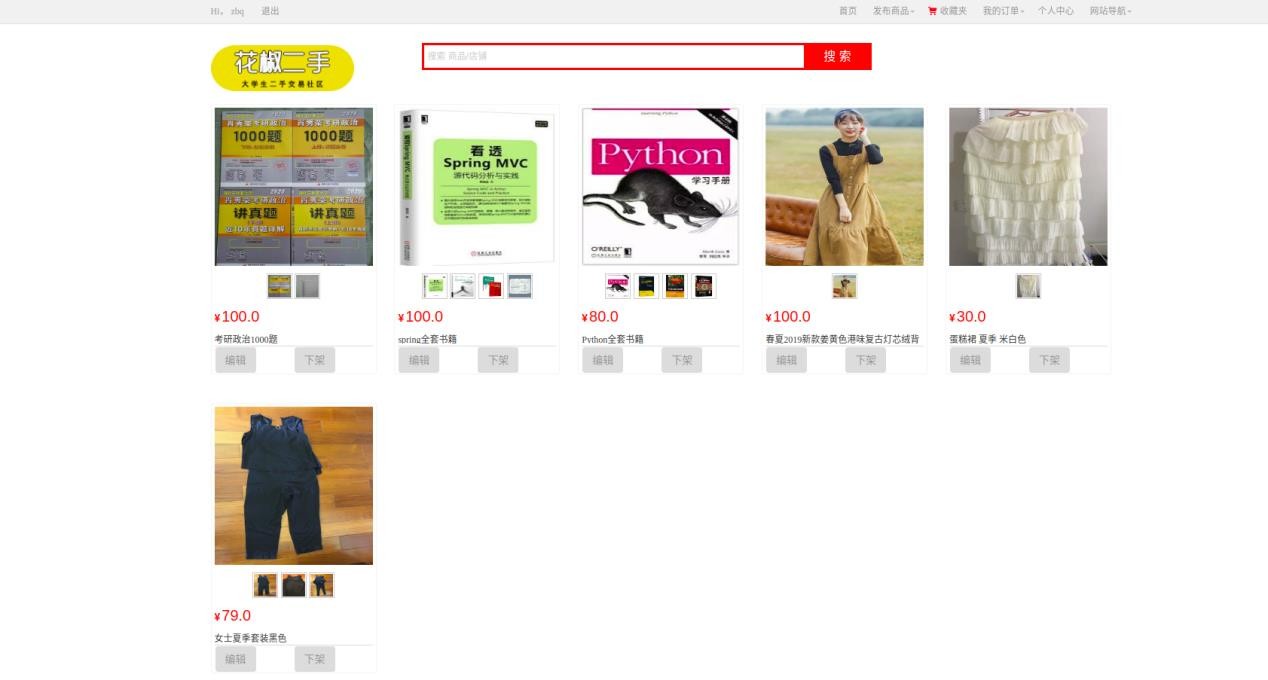
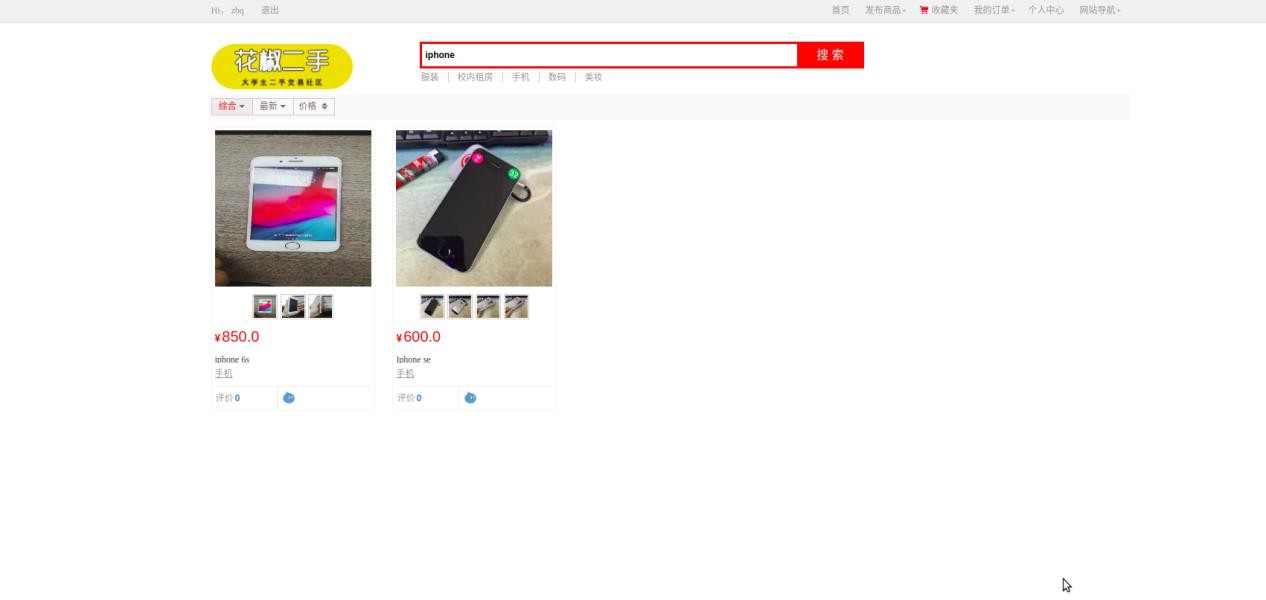
已发布商品页面展示用户发布的所有商品。如图 5-12 所示：

图 5-12 已发布商品页面

* + - 1. 商品查询页面

商品查询页面展示与用户查询关键词相匹配的所有商品。如图 5-13 所示：

* + - 1. 后台登录页面

图 5-13 商品查询页面

后台登陆页面用于管理员登录后台，包括用户名和密码。如图 5-14 所示：

* + - 1. 后台管理首页

图 5-14 后台登录页面

后台管理首页以图表的形式清晰的展示用户数，订单数等。如图 5-15 所示：



图 5-15 后台管理首页

* + - 1. 后台商品信息管理页面

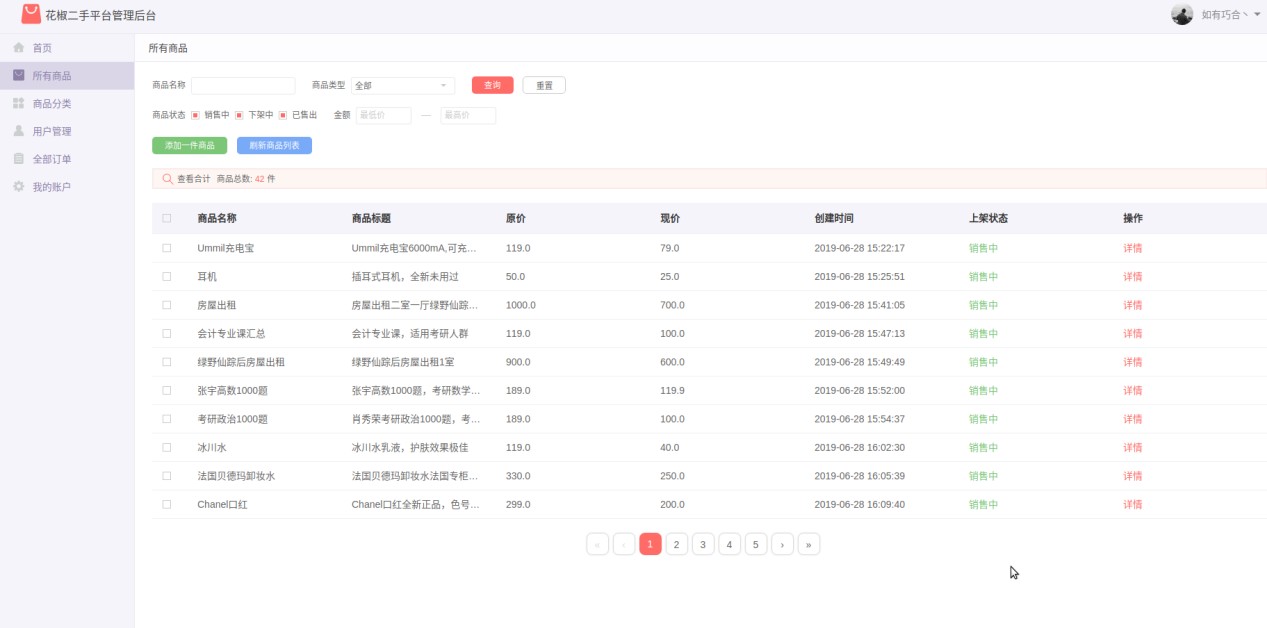
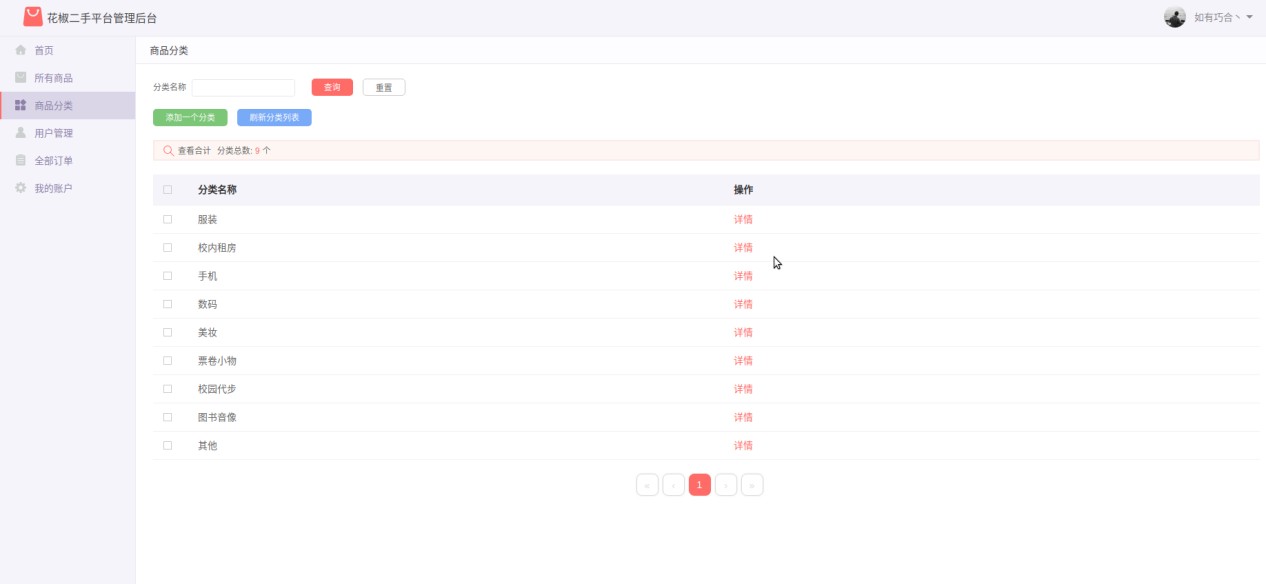
后台商品信息管理页面展示所有用户发布的商品，用户下架的商品，以及用户已售的商品。如图 5-16 所示：

图 5-16 后台商品信息管理页面

* + - 1. 后台商品分类管理页面

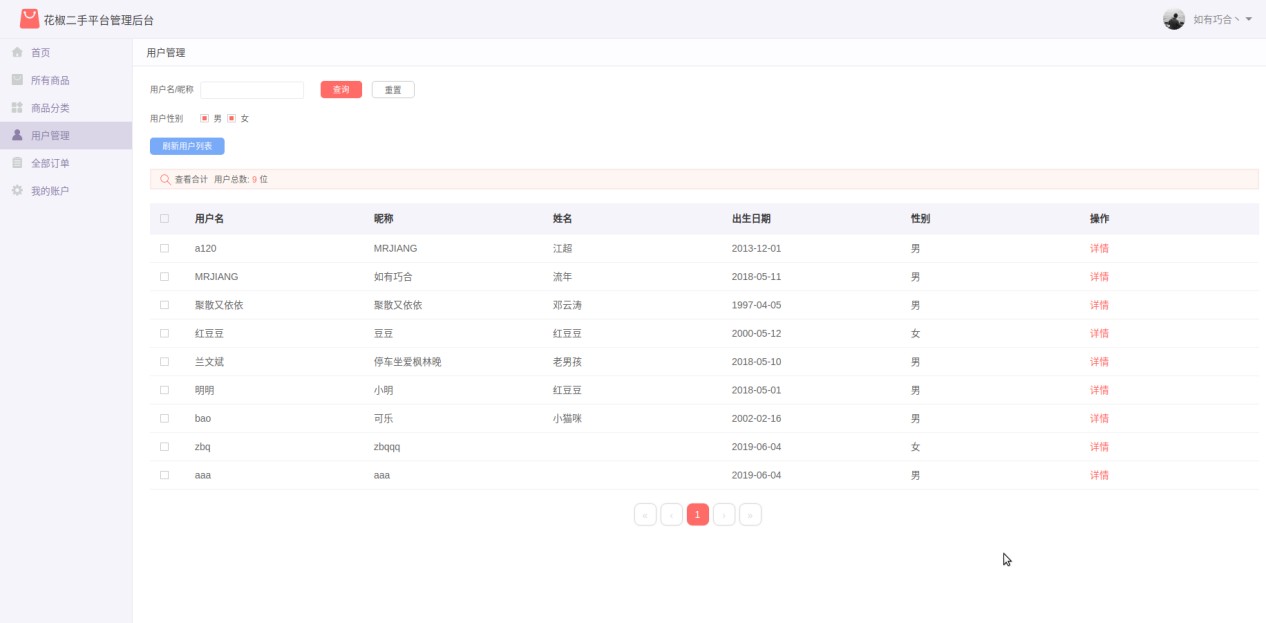
后台商品分类页面展示所有的商品分类。如图 5-17 所示：



* + - 1. 用户管理页面

图 5-17 后台商品分类管理页面

用户管理页面展示所有的用户信息。如图 5-18 所示：



* + - 1. 订单管理页面

图 5-18 用户管理页面

订单管理页面展示所有的订单信息，包括等待发货，交易成功等状态信息。如图 5-19 所示：

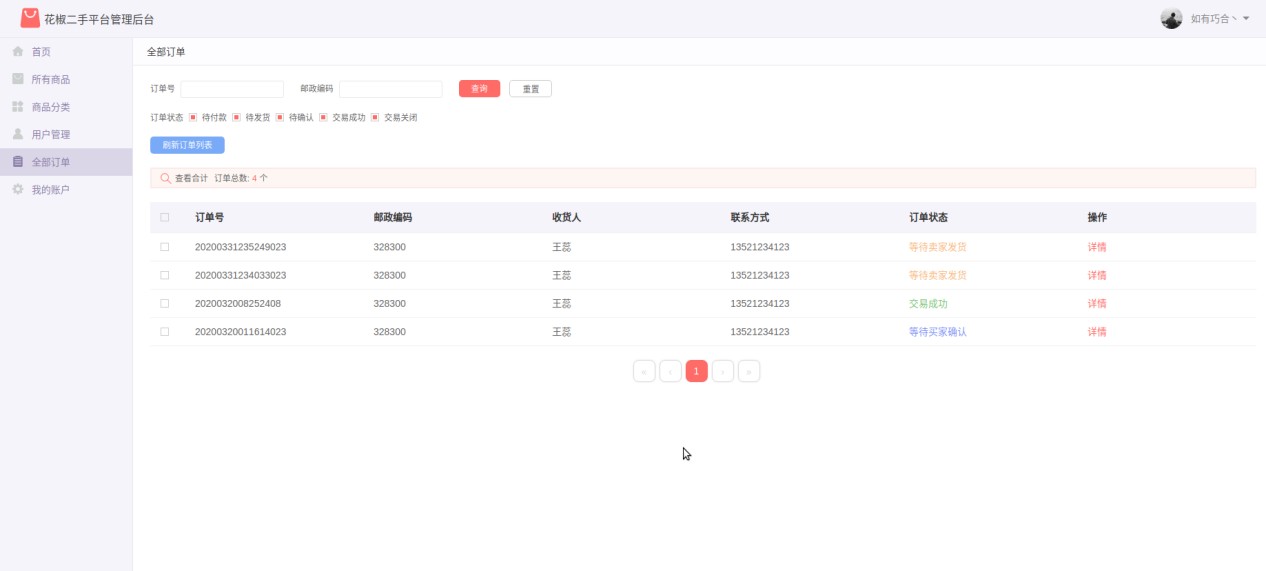


图 5-19 订单管理页面

* + - 1. 管理员个人中心页面

管理员个人中心页面显示管理员个人信息详情。如图 5-20 所示：

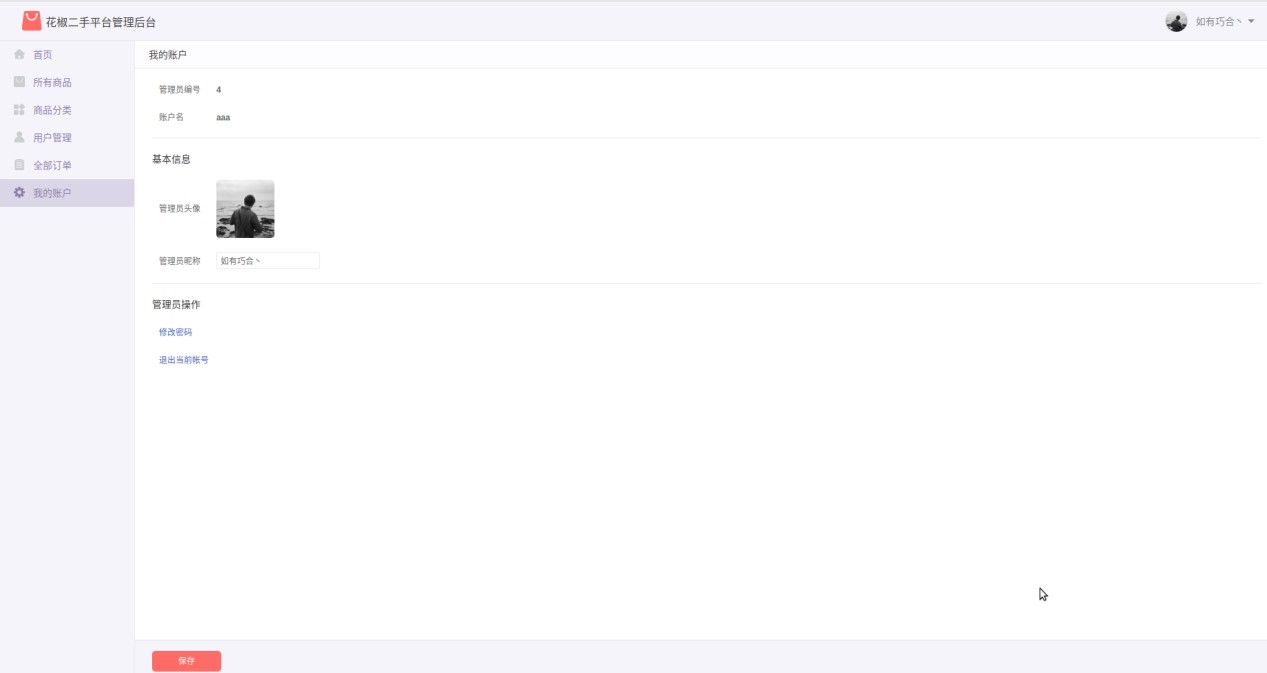


图 5-20 管理员个人中心页面

# 详细设计

## 系统主要功能模块介绍

1. 商品信息模块：此模块有三个功能分别是商品信息发布、商品搜索、商品信息修改、商品下架。
2. 收藏夹模块：该模块包含的功能为增加商品、删除商品、查询商品。
3. 订单模块：该模块包括了已售商品、已购商品、评价三个子模块。

① 已售商品子模块包含查看已售商品、查看资产两个功能。

② 已购商品子模块包含查看已购商品、查看资产两个功能。

③ 评价功能子模块：用户可以对自己的已售订单和已购订单进行评价和追加评价。

1. 用户信息模块：此模块有三个功能分别是登录注册、修改个人信息和查看个人信息。
2. 商品分类模块：即商品按类别显示，供用户浏览。
3. 商品信息管理模块：该模块包括了商品审核、商品删除、商品查询。
4. 用户信息管理模块：该模块包括了用户查询、用户删除、用户信用度修改。
5. 商品分类管理模块：该模块包含增加商品分类、删除商品分类和查询商品分类三个功能。

## 功能模块设计

* + 1. 商品信息模块
       1. 功能结构设计

商品信息模块功能结构设计如图 6-1 所示：

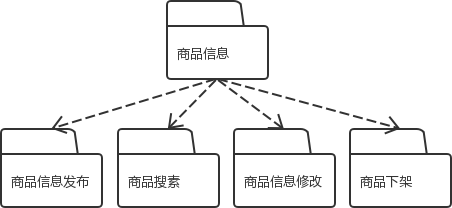


图 6-1 商品信息模块功能结构设计

1. 商品信息发布：卖家可以发布二手商品的信息，对商品进行描述，并且添

加图片增加可信度，也方便买家查看相应的信息。

1. 商品搜索：实现本站内特有的搜索引擎，可以输入字段，模糊查询合适的商品并且显示出来。
2. 商品信息修改：卖家可以对自己发布的商品信息进行修改，例如修改商品名称，修改商品的单价和数量等，修改之后将新的数据提交到数据库，数据库执行相应的修改操作。
3. 商品下架：卖家可以下架自己所发布的商品，下架后商品的状态信息变成“已下架”，其他买家无法购买此商品，但数据库中仍保存此商品的各项信息。
   * + 1. 类图设计

商品信息模块类设计如图 6-2 所示：

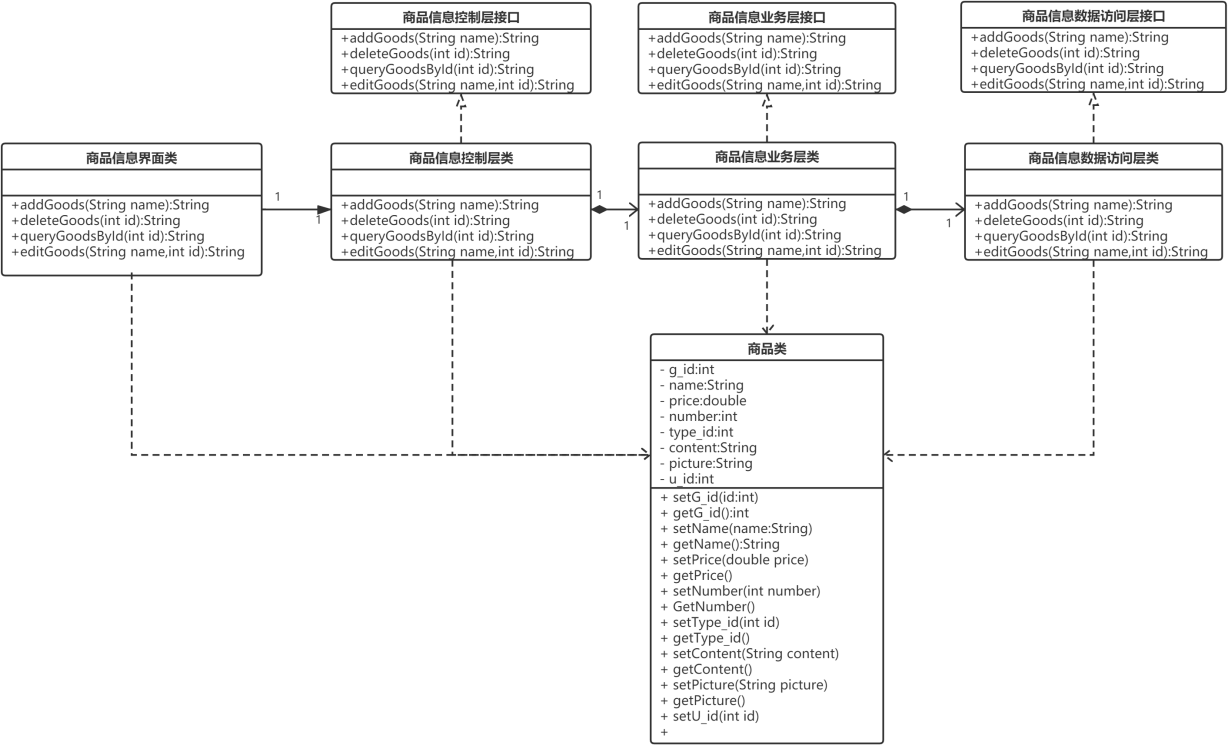


图 6-2 商品信息模块类设计

商品信息类描述如表 6-1 所示：

表 6-1 商品信息类描述

|  |  |  |
| --- | --- | --- |
| 商品信息类 | | |
| 类的描述：负责响应页面发出的对商品信息操作请求 | | |
| 操作描述 | addGoods(String name) | 发布商品 |
| deleteGoods(int id) | 下架商品 |
| queryGoodsById(int id) | 根据商品 id 搜索商品 |
| editGoods(String name,int id) | 修改商品 |

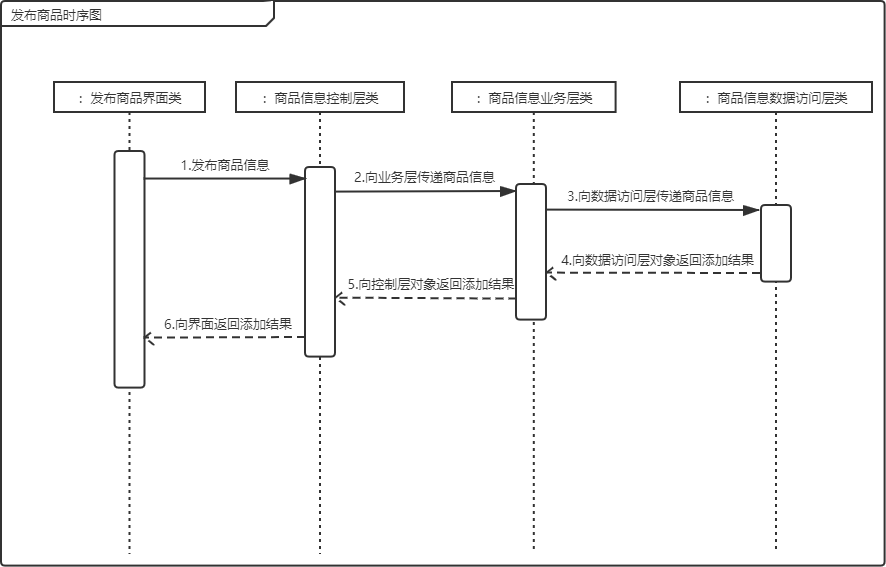
商品类描述如表 6-2 所示：

表 6-2 商品类描述

|  |  |  |
| --- | --- | --- |
| 商品类 | | |
| 类的描述：商品实体类 | | |
| 属性描述 | g\_id | 商品 id |
| name | 商品名称 |
| price | 单价 |
| number | 库存量 |
| type\_id | 商品类别 id |
| content | 商品内容 |
| picture | 商品图片 |
| u\_id | 所属卖家 id |
| 操作描述 | setG\_id(int id) | 设置商品 id |
| getG\_id() | 获取商品 id |
| setName(String name) | 设置商品名称 |
| GetName() | 获取商品名称 |
| setPrice(double price) | 设置商品价格 |
| getPrice() | 获取商品价格 |
| setNumber(int number) | 设置商品库存 |
| GetNumber() | 获取商品库存 |
| setType\_id(int id) | 设置商品类别 id |
| getType\_id() | 获取商品类别 id |
| setContent(String content) | 设置商品描述 |
| getContent() | 获取商品描述 |
| setPicture(String picture) | 设置商品图片 |
| getPicture() | 获取商品图片 |
| setU\_id(int id) | 设置商品所属卖家 |
| getU\_id() | 获取商品所属卖家 |

* + - 1. 顺序图设计

商品信息发布模块时序图如图 6-3 所示：



* + - 1. 核心处理流程设计

图 6-3 商品信息发布时序图

商品信息模块核心处理流程设计如图 6-4 所示：

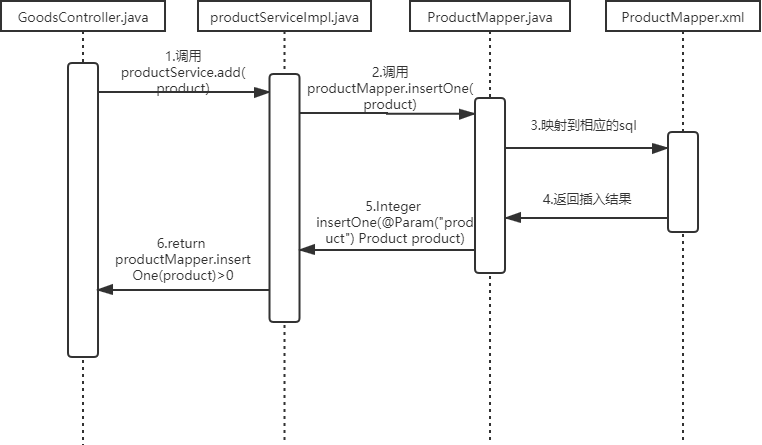


图 6-4 商品信息发布业务流程图

* + 1. 收藏夹模块
       1. 功能结构设计

收藏夹模块功能结构设计如图 6-5 所示：

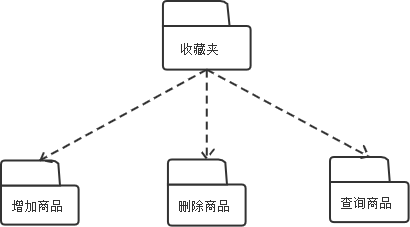


图 6-5 收藏夹模块功能结构设计

1. 添加收藏商品：用户可以将自己心仪的商品添加收藏到收藏夹。
2. 取消收藏商品：当用户取消其中一个的收藏物品时，实时在收藏夹列表里删除该物品。
3. 查询已收藏商品：用户可以查询已收藏的商品，系统显示用户加入收藏夹的所有商品。
   * + 1. 类图设计

收藏夹模块类设计如图 6-6 所示：

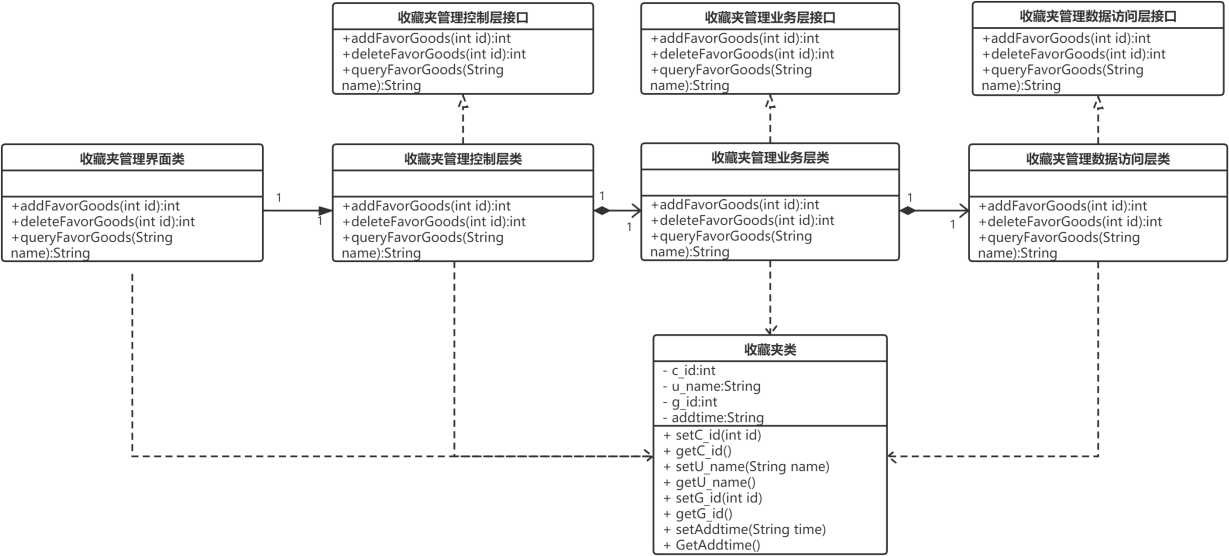


图 6-6 收藏夹模块类设计

收藏夹管理界面类描述如表 6-3 所示：

表 6-3 收藏夹管理界面类描述

|  |  |  |
| --- | --- | --- |
| 收藏夹管理界面类 | | |
| 类的描述：负责响应页面发出的对收藏夹的操作请求 | | |
| 操 | addFavorGoods(int id) | 添加商品至收藏夹 |

表 6-3 收藏夹管理界面类描述（续）

|  |  |  |
| --- | --- | --- |
| 作描述 | deleteFavorGoods(int id) | 取消收藏商品 |
| queryFavorGoods(String name) | 查询收藏夹商品 |

收藏夹类描述如表 6-4 所示：

表 6-4 收藏夹类描述

|  |  |  |
| --- | --- | --- |
| 收藏夹类 | | |
| 类的描述：收藏夹实体类 | | |
| 属性描述 | c\_id | 收藏夹 id |
| u\_name | 用户名 |
| g\_id | 商品 id |
| addtime | 添加时间 |
| 操作描述 | setC\_id(int id) | 设置收藏夹 id |
| getC\_id() | 获取收藏夹 id |
| setU\_name(String name) | 设置用户名 |
| getU\_name() | 获取用户名 |
| setG\_id(int id) | 设置商品 id |
| getG\_id() | 获取商品 id |
| setAddtime(String time) | 设置添加时间 |
| GetAddtime() | 获取添加时间 |

* + - 1. 顺序图设计

收藏夹模块时序图如图 6-7 所示：

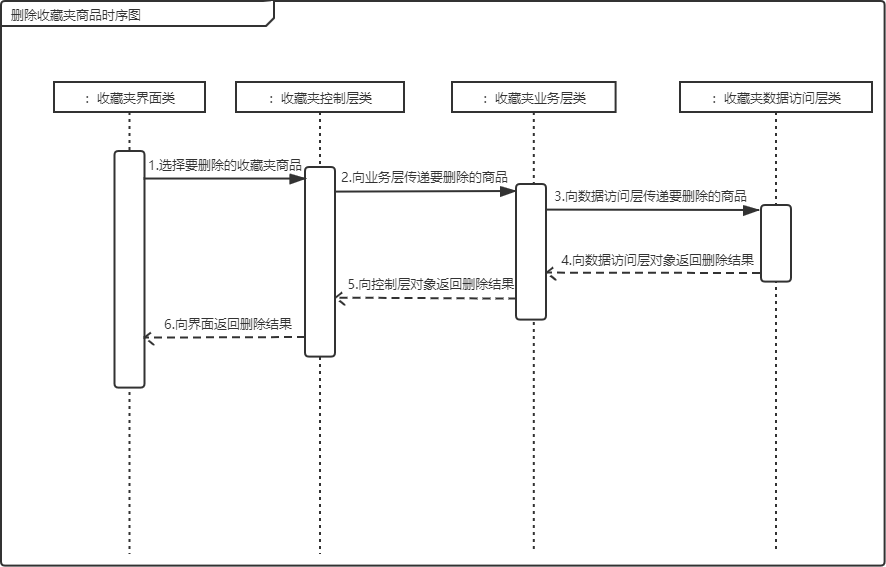


图 6-7 删除收藏夹商品时序图

* + - 1. 核心处理流程设计

收藏夹模块核心处理流程设计如图 6-8 所示：

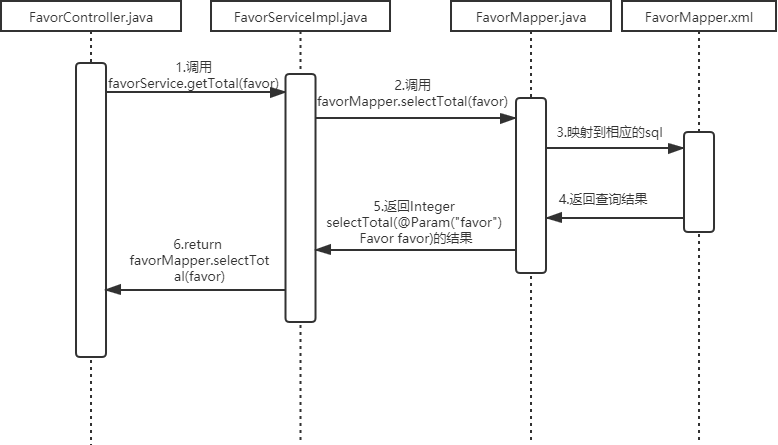


图 6-8 查询收藏夹商品业务流图

* + 1. 用户信息模块
       1. 功能结构设计

用户信息模块功能结构设计如图 6-9 所示：

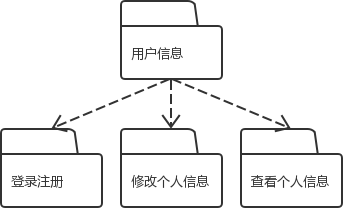


图 6-9 用户信息模块功能结构设计

1. 登录注册：用户通过账号和密码登录，登录之后显示首页。当用户没有账号的时候，点击注册，从登录界面切换至注册界面，填写正确的信息，后台验证通过则注册成功，用户注册完后跳转登录界面。
2. 修改个人信息：用户可在个人信息界面根据个人情况修改个人信息并在后台更新在页面同步显示。
3. 查看个人信息：用户可在个人信息界面查看信息，例如用户名、真实姓名、邮箱、手机号码、QQ 号等。
   * + 1. 类图设计

用户信息模块类设计如图 6-10 所示：

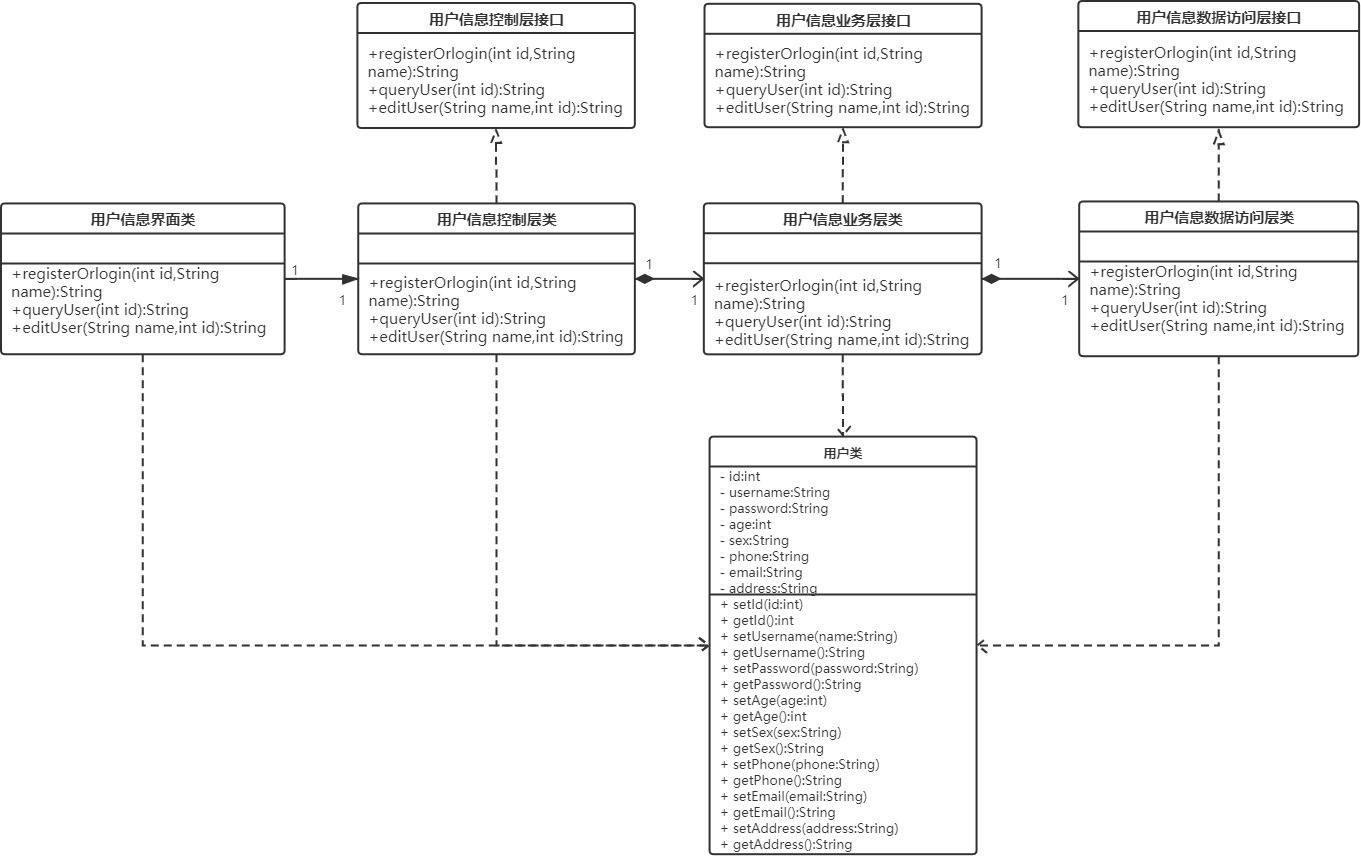


图 6-10 用户信息模块类设计

用户信息界面类描述如表 6-5 所示：

表 6-5 用户信息界面类描述

|  |  |  |
| --- | --- | --- |
| 用户信息界面类 | | |
| 类的描述：负责响应页面发出的对用户信息操作请求 | | |
| 操作描述 | registerOrlogin(String name) | 注册登录 |
| queryUser(int id) | 查看个人信息 |
| editUser(String name,int id) | 修改个人信息 |

用户类描述如表 6-6 所示：

表 6-6 用户类描述

|  |  |  |
| --- | --- | --- |
| 用户类 | | |
| 类的描述：用户实体类 | | |
| 属性描述 | id | 用户 id |
| username | 用户名 |
| password | 密码 |

表 6-6 用户类描述（续）

|  |  |  |
| --- | --- | --- |
|  | age | 年龄 |
| sex | 性别 |
| phone | 手机号 |
| email | 邮箱 |
| address | 地址 |
| 操作描述 | setId(int id) | 设置用户 id |
| getId() | 获取用户 id |
| setUsername(String name) | 设置用户名 |
| GetUsername() | 获取用户名 |
| setPassword(String password) | 设置密码 |
| getPassword() | 获取密码 |
| setAge(int age) | 设置年龄 |
| getAge() | 获取年龄 |
| setSex(String sex) | 设置性别 |
| getSex() | 获取性别 |
| setPhone(String phone) | 设置手机号 |
| getPhone() | 获取手机号 |
| setEmail(String email) | 设置电子邮箱 |
| getEmail() | 获取电子邮箱 |
| setAddress(String address) | 设置地址 |
| getAddress() | 获取地址 |

* + - 1. 顺序图设计

用户信息模块时序图如图 6-11 所示：

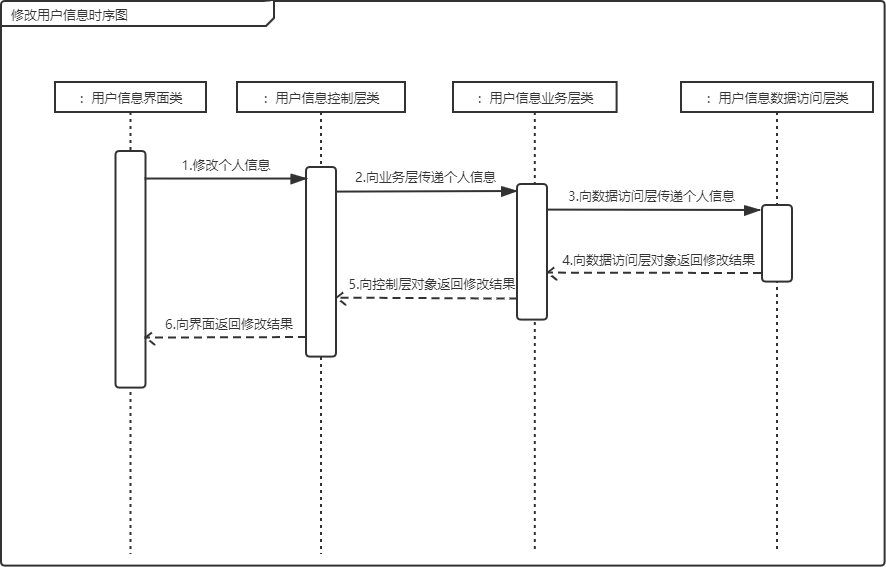


图 6-11 修改用户信息时序图

* + - 1. 核心处理流程设计

用户信息模块核心处理流程设计如图 6-12 所示：

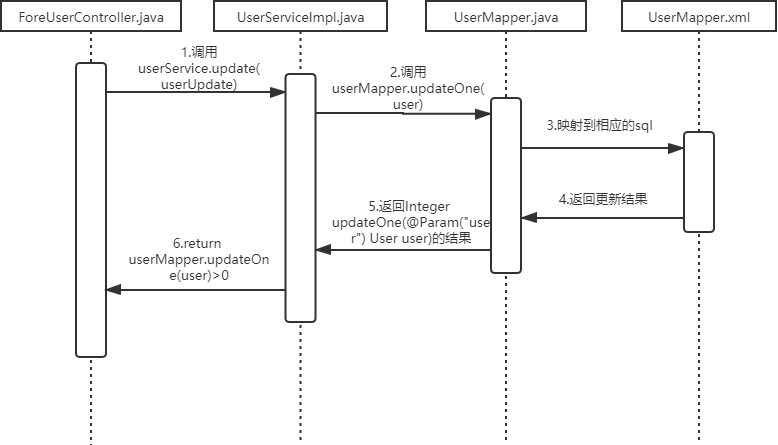


图 6-12 修改用户信息业务流图

* + 1. 订单模块
       1. 功能结构设计

订单模块功能结构设计如图 6-13 所示：

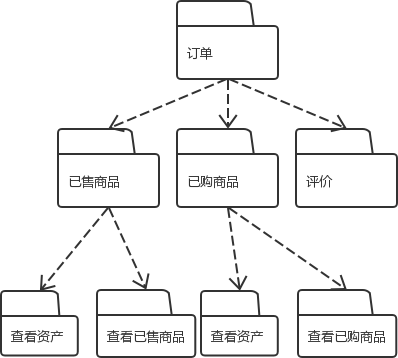


图 6-13 订单模块功能结构设计

1. 已售商品子模块包含查看已售商品、查看资产两个功能。

① 查看已售商品：用户可以查看自己的所有出售的商品的历史记录，每条记录都可以显示 详细的订单信息。

② 查看资产：用户可以查看自己每次出售商品后的的资产变化，每条记录都可以显示详细的订单信息。

1. 已购商品子模块包含查看已购商品、查看资产两个功能。

① 查看已购商品：用户可以查看自己的所有购买过的商品的历史记录，每条记录都可以显 示详细的订单信息。

② 查看资产：用户可以查看自己每次购买商品后的的资产变化，每条记录都可以显示详细的订单信息。

1. 评价功能子模块：用户可以对自己的已售订单和已购订单进行评价和追加评价。
   * + 1. 类图设计

订单模块类设计如图 6-14 所示：

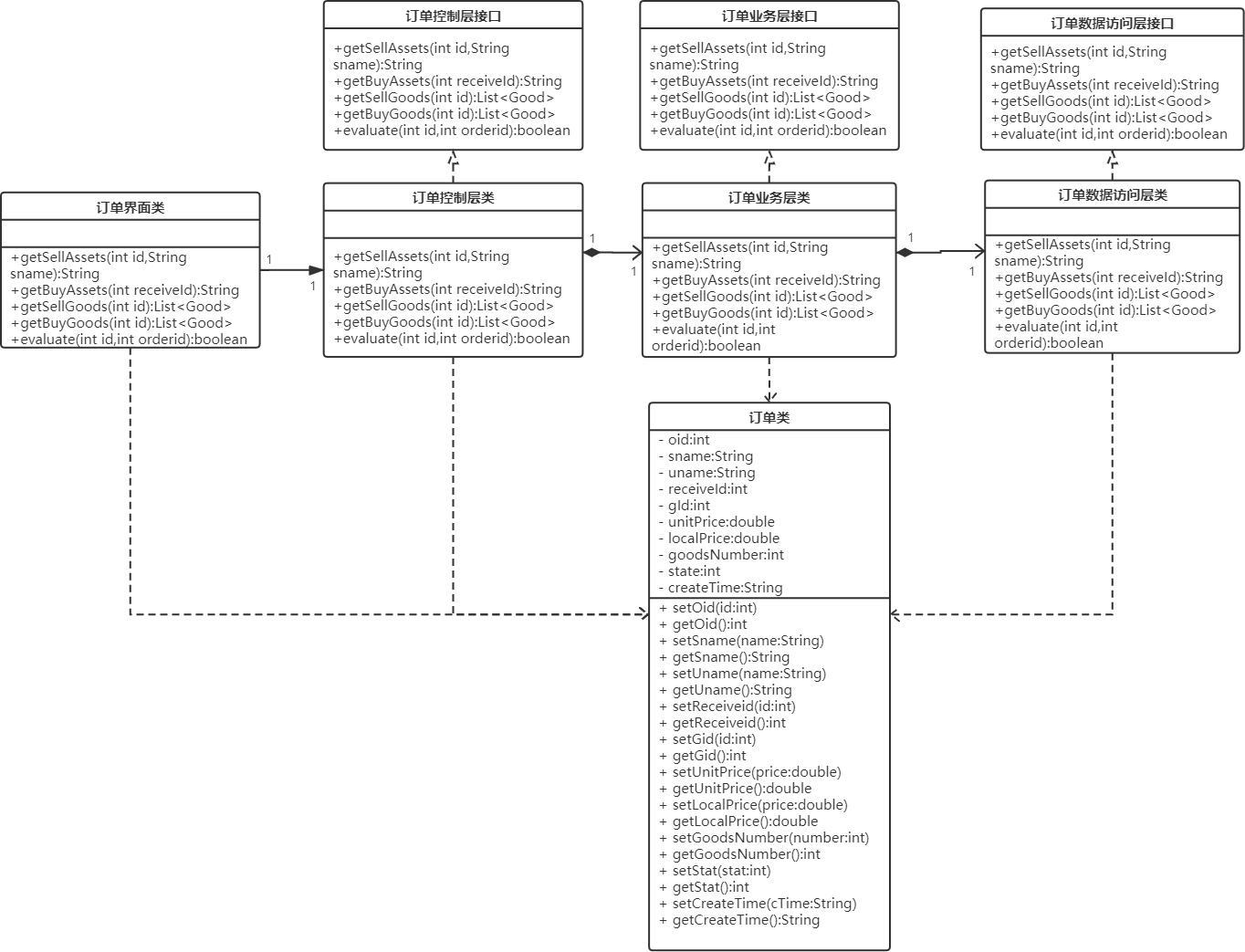


图 6-14 订单模块类设计

订单管理界面类描述如表 6-7 所示：

表 6-7 订单管理界面类描述

|  |  |  |
| --- | --- | --- |
| 订单管理界面类 | | |
| 类的描述：负责响应页面发出的对订单操作请求 | | |
| 操作描述 | getSellAssets(int id,String sname) | 查看已售资产 |
| getBuyAssets(int receiveId) | 查看已购资产 |
| getSellGoods(int id) | 查看已售商品 |
| getBuyGoods(int id) | 查看已购商品 |
| evaluate(int id,int orderid) | 评价订单 |

订单类描述如表 6-8 所示：

表 6-8 订单类描述

|  |  |  |
| --- | --- | --- |
| 订单类 | | |
| 类的描述：订单实体类 | | |
| 属性描述 | oid | 订单 id |
| sname | 卖家姓名 |
| uname | 买家姓名 |
| receiveId | 买家编号 |
| gid | 商品编号 |
| unitPrice | 总价 |
| localPrice | 单价 |
| goodsNumber | 商品数量 |
| state | 订单状态 |
| createTime | 生成时间 |
| 操作描述 | setOid(id:int) | 设置订单 id |
| getOid() | 获取订单 id |
| setSname(name:String) | 设置卖家姓名 |
| getSname() | 获取卖家姓名 |
| setUname(name:String) | 设置买家姓名 |
| getUname() | 获取买家姓名 |
| setReceiveid(id:int) | 设置买家编号 |
| getReceiveid() | 获取买家编号 |
| setGid(id:int) | 设置商品编号 |
| getGid() | 获取商品编号 |
| setUnitPrice(price:double) | 设置总价 |
| getUnitPrice() | 获取总价 |
| setLocalPrice(price:double) | 设置单价 |
| getLocalPrice() | 获取单价 |
| setGoodsNumber(number:int) | 设置商品数量 |

表 6-8 订单类描述（续）

|  |  |  |
| --- | --- | --- |
|  | getGoodsNumber() | 获取商品数量 |
| setStat(stat:int) | 设置订单状态 |
| getState() | 获取订单状态 |
| setCreateTime(cTime:String) | 设置生成时间 |
| getCreateTime() | 获取生成时间 |

* + - 1. 顺序图设计

订单模块时序图如图 6-15 所示：

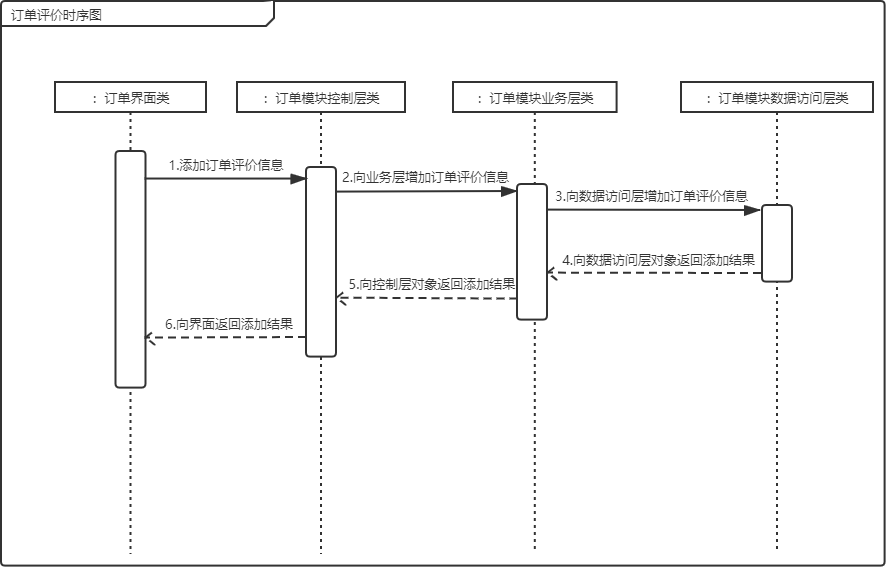


图 6-15 订单评价时序图

* + - 1. 核心处理流程设计

订单模块核心处理流程设计如图 6-16 所示：

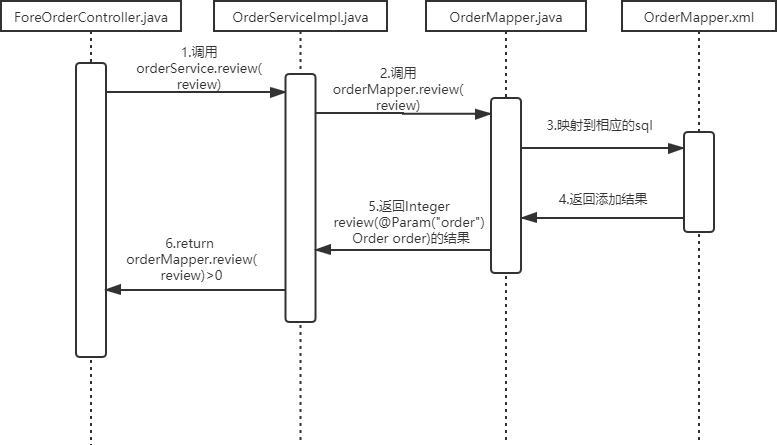


图 6-16 查询已购商品业务流图

* + 1. 商品信息管理模块
       1. 功能结构设计

商品信息管理模块功能结构设计如图 6-17 所示：

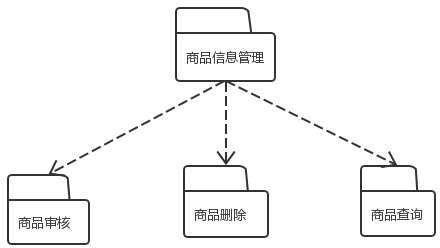


图 6-17 商品信息管理模块结构设计

1. 商品审核：包含了管理员对于卖家发布的商品的详细信息内容的审核，如存在违规操作，对则管理员有权限删除该商品，该权限仅限管理员。
2. 商品删除：对于存在违规操作，含有违规内容的商品进行删除，用户无法在页面搜索到该商品。
3. 商品查询：包含了平台所有商品的情况及状态，仅限管理员可以在后台查询。

* + - 1. 类图设计

商品信息管理模块类设计如图 6-18 所示：

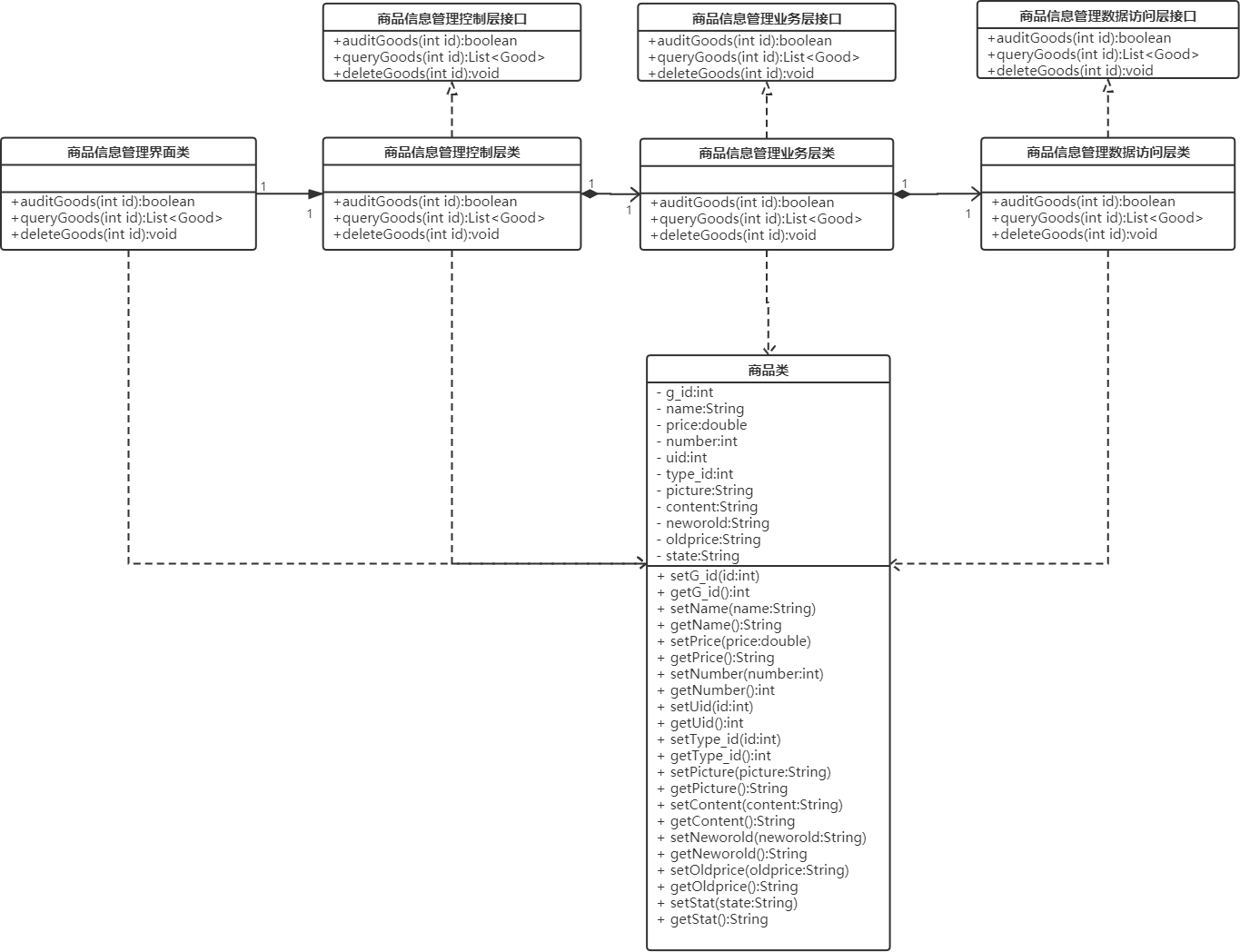


图 6-18 商品信息管理模块类设计

商品信息管理界面类描述如表 6-9 所示：

表 6-9 商品信息管理界面类描述

|  |  |  |
| --- | --- | --- |
| 商品信息管理界面类 | | |
| 类的描述：负责响应管理员对商品信息操作请求 | | |
| 操作描述 | auditGoods(int id) | 商品审核 |
| queryGoods(int id) | 商品查询 |
| deleteGoods(int id) | 商品删除 |

商品类描述如表 6-10 所示：

表 6-10 商品类描述

|  |  |  |
| --- | --- | --- |
| 商品类 | | |
| 类的描述：商品信息实体类 | | |
| 属性描述 | g\_id | 商品 id |
| name | 商品名称 |
| price | 单价 |
| number | 库存量 |
| type\_id | 商品类别 id |
| content | 商品内容 |
| picture | 商品图片 |
| u\_id | 所属卖家 id |
| 操作描述 | setG\_id(int id) | 设置商品 id |
| getG\_id() | 获取商品 id |
| setName(String name) | 设置商品名称 |
| GetName() | 获取商品名称 |
| setPrice(double price) | 设置商品价格 |
| getPrice() | 获取商品价格 |
| setNumber(int number) | 设置商品库存 |
| GetNumber() | 获取商品库存 |
| setType\_id(int id) | 设置商品类别 id |
| getType\_id() | 获取商品类别 id |
| setContent(String content) | 设置商品描述 |
| getContent() | 获取商品描述 |
| setPicture(String picture) | 设置商品图片 |
| getPicture() | 获取商品图片 |
| setU\_id(int id) | 设置商品所属卖家 |
| getU\_id() | 获取商品所属卖家 |

* + - 1. 顺序图设计

商品信息管理模块时序图如图 6-19 所示：

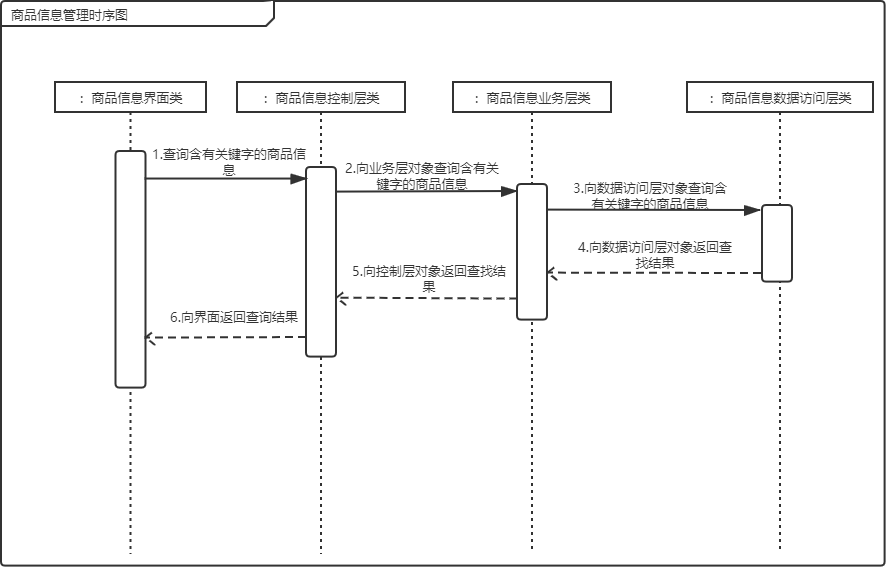


图 6-19 商品信息查询时序图

* + - 1. 核心处理流程设计

商品信息管理模块核心处理流程设计如图 6-20 所示：

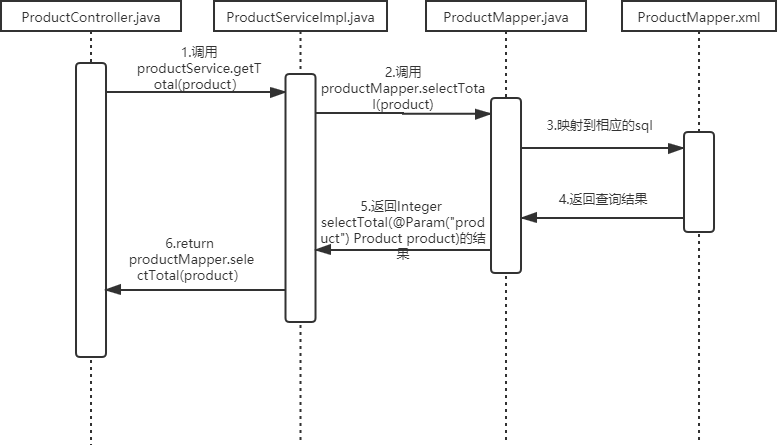


图 6-20 管理员查询商品业务流图

* + 1. 商品分类管理模块
       1. 功能结构设计

商品分类管理模块功能结构设计如图 6-21 所示：

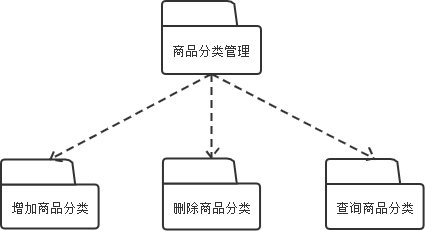


图 6-21 商品分类管理模块结构设计

* + - 1. 类图设计

商品分类管理模块功能结构设计如图 6-22 所示：

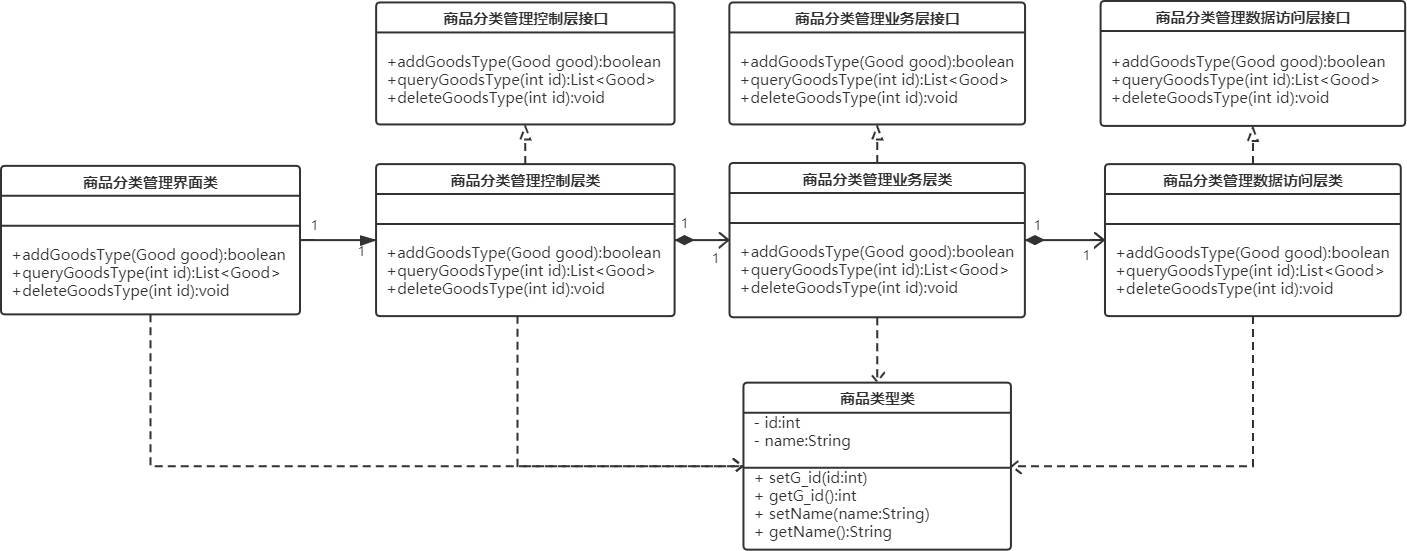


图 6-22 商品分类管理模块类设计

商品封面类管理界面类如表 6-11 所示：

表 6-11 商品分类管理界面类描述

|  |  |  |
| --- | --- | --- |
| 商品分类管理类 | | |
| 类的描述：负责响应页面发出的对商品分类信息操作请求 | | |
| 操作描述 | addGoodsType(Good good) | 添加商品分类 |
| queryGoodsType(int id) | 查询商品分类 |
| deleteGoodsType(int id) | 删除商品分类 |

商品类型类描述如表 6-12 所示：

表 6-12 商品类型类描述

|  |  |  |
| --- | --- | --- |
| 商品类型类 | | |
| 类的描述：商品类型实体类 | | |
| 属性描  述 | id | 商品 id |
| name | 商品类型 |
| 操作描述 | setG\_id(int id) | 设置商品 id |
| getG\_id() | 获取商品 id |
| setName(String name) | 设置商品类型 |
| GetName() | 获取商品类型 |

* + - 1. 顺序图设计

商品分类管理模块时序图如图 6-23 所示：

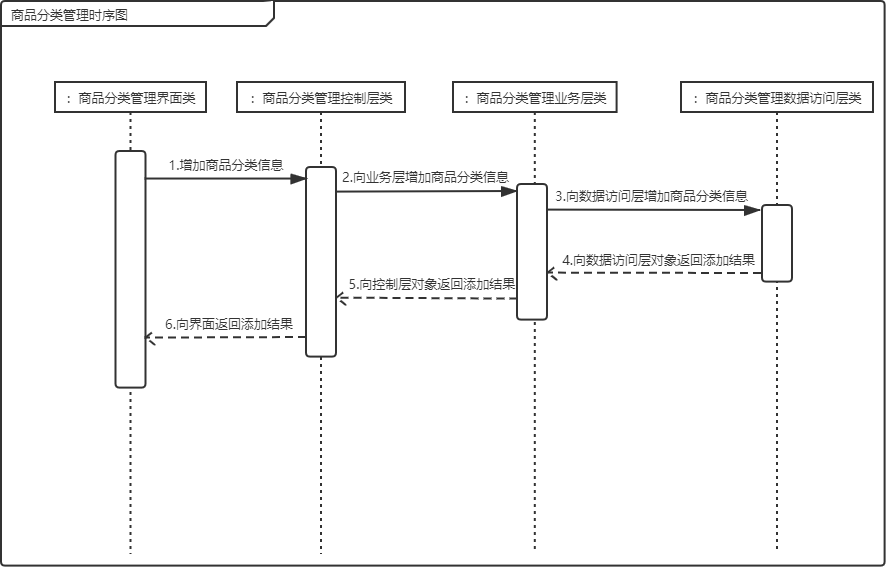


图 6-23 商品添加分类信息时序图

* + - 1. 核心处理流程设计

商品分类管理模块核心处理流程设计如图 6-24 所示：

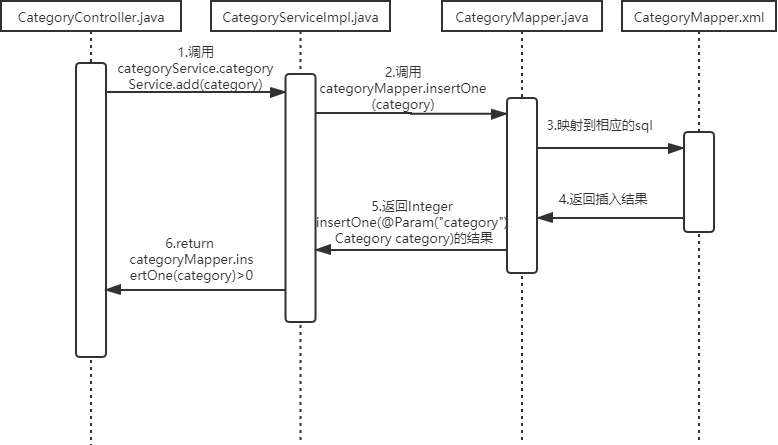


图 6-24 管理员增加商品分类业务流图

* + 1. 用户信息管理模块
       1. 功能结构设计

用户信息管理模块功能结构设计如图 6-25 所示：

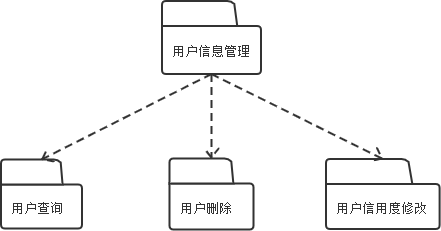


图 6-25：用户信息管理模块结构设计

1. 用户查询：管理员可对平台内的所有用户进行查询，该权限仅限管理员。
2. 用户删除：对于信用度极低且存在违规操作，发布含有违规内容的用户进行删除。
3. 用户信用度修改：包含了管理员对用户信用度的评定，若用户存在违规操作，管理员可通过降低该用户的信用度对用户进行管理。
   * + 1. 类图设计

用户信息管理模块类设计如图 6-26 所示：

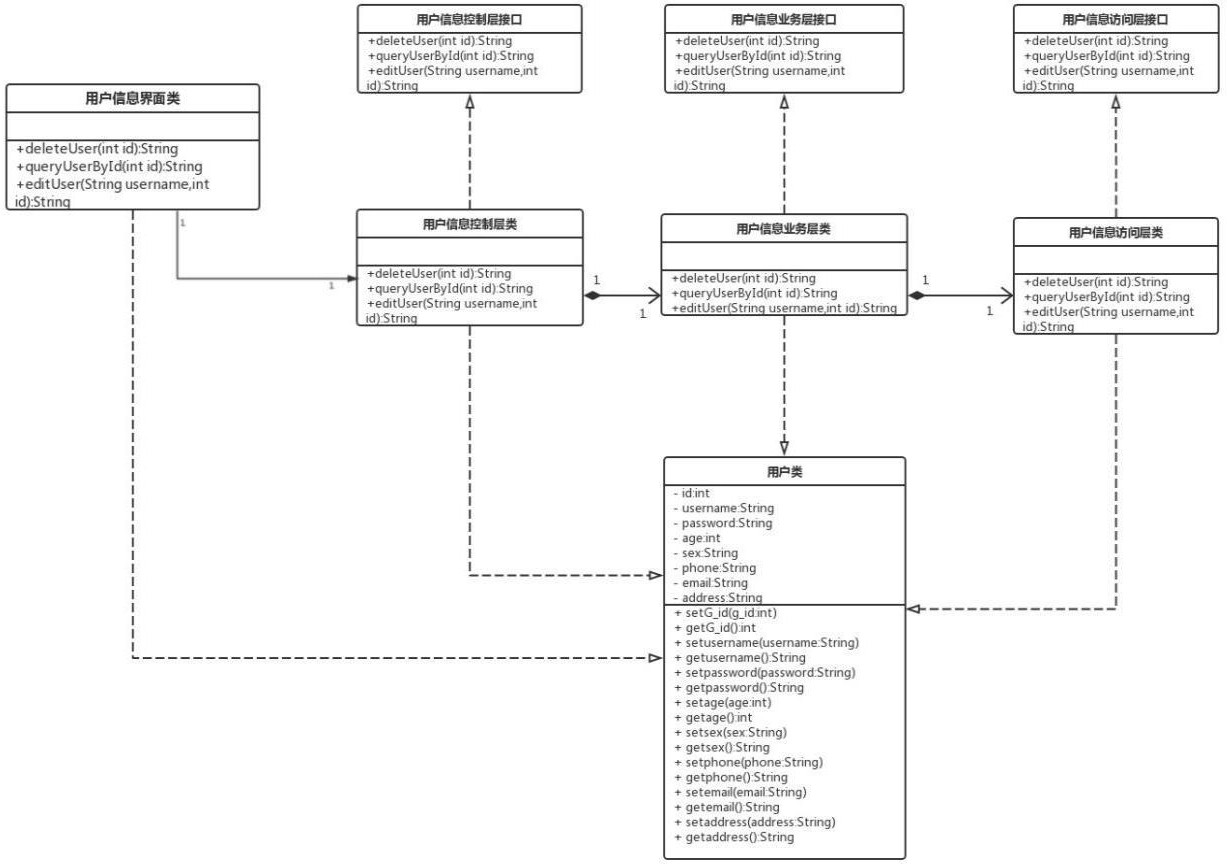


图 6-26 用户信息管理模块类设计

用户信息界面类描述如表 6-13 所示：

表 6-13 用户信息界面类描述

|  |  |  |
| --- | --- | --- |
| 用户信息界面类 | | |
| 类的描述：负责响应页面发出的对用户信息操作请求 | | |
| 操作描述 | editUser(String username,int id) | 用户信用度修改 |
| deleteUser(int id) | 删除用户信息 |
| queryUserById(int id) | 查询用户信息 |

用户类描述如表 6-14 所示：

表 6-14 用户类描述

|  |  |  |
| --- | --- | --- |
| 用户类 | | |
| 类的描述：用户实体类 | | |
| 属性描 | id | 用户 id |
| username | 用户名 |

表 6-14 用户类描述（续）

|  |  |  |
| --- | --- | --- |
| 述 | password | 密码 |
| age | 年龄 |
| sex | 性别 |
| phone | 手机号 |
| email | 邮箱 |
| address | 地址 |
| 操作描述 | setId() | 设置用户 id |
| getId() | 获取用户 id |
| setUsername() | 设置用户名 |
| GetUsername() | 获取用户名 |
| setPassword() | 设置密码 |
| getPassword() | 获取密码 |
| setAge() | 设置年龄 |
| getAge() | 获取年龄 |
| setSex() | 设置性别 |
| getSex() | 获取性别 |
| setPhone() | 设置手机号 |
| getPhone() | 获取手机号 |
| setEmail() | 设置电子邮箱 |
| getEmail() | 获取电子邮箱 |
| setAddress() | 设置地址 |
| getAddress() | 获取地址 |

* + - 1. 顺序图设计

用户信息管理模块时序图如图 6-27 所示：

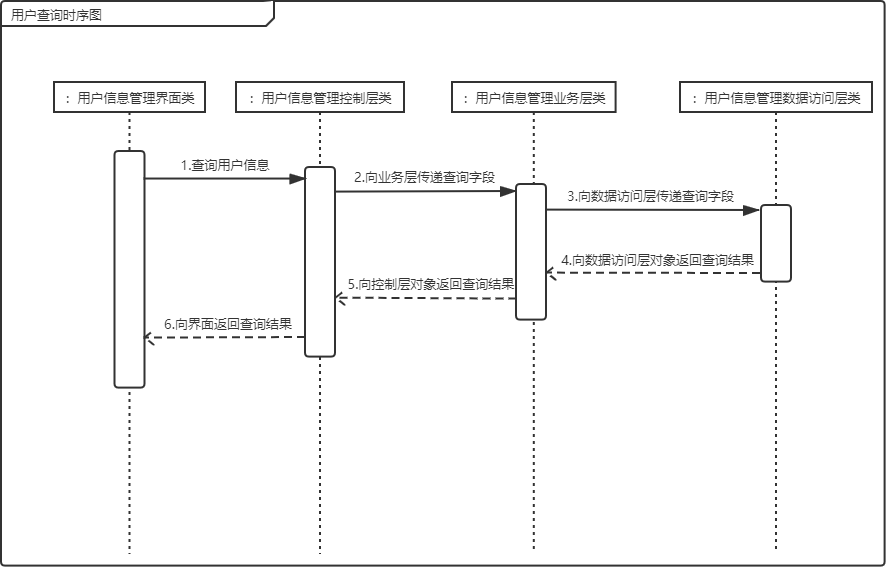


图 6-27 管理员用户查询时序图

* + - 1. 核心处理流程设计

用户信息管理模块核心处理流程设计如图 6-28 所示：

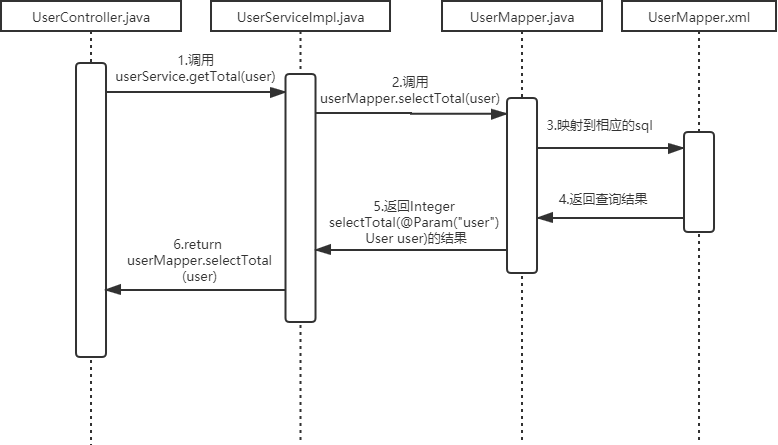


图 6-28 管理员查询用户业务流图

# 编码

## 代码实现与核心算法

发布商品部分代码如下，首先检查用户是否已经登录，若未登录跳转到登录页面；若已登陆跳转到发布商品页面。

@RequestMapping(value = "publishGoods", method = RequestMethod.GET) public String publishGoods(Map<String, Object> map, HttpSession session) {

logger.info("检查用户是否登录");

Integer userId = (Integer) checkUser(session); User user;

if (userId != null) {

logger.info("获取用户信息");

user = userService.get(Integer.*parseInt*(userId.toString())); map.put("user", user);

} else {

return "redirect:/login";

}

logger.info("获取产品分类列表");

List<Category> categoryList = categoryService.getList(null,null); map.put("categoryList",categoryList);

return "fore/pubGoods";

}

提交发布的商品信息部分代码如下，获取用户填写的商品信息的各参数，保存到数据库中。

@ResponseBody

@RequestMapping(value = "publishGoodsSubmit",method = RequestMethod.POST,produces = "application/json;charset=utf-8")

public String publishGoodsSubmit(HttpSession session,

@RequestParam String product\_name, @RequestParam String product\_price, @RequestParam String product\_sale\_price, @RequestParam String catelogId, @RequestParam String product\_chengse, @RequestParam String product\_title, @RequestParam(required = false) String[]

productSingleImageList*/\**产品预览图片名称数组*\*/*,

@RequestParam(required = false) String[] productDetailsImageList*/\**产品详情图片名称数组*\*/*) throws Exception {

JSONObject jsonObject=new JSONObject();

*//*查询出当前用户 *user* 对象，便于使用 *id*

Integer userId = (Integer) session.getAttribute("userId");

Category category = categoryService.get(Integer.*valueOf*(catelogId));

*//*当前日期

Date date=new Date();

String s\_product\_title= URLDecoder.*decode*(product\_title, "UTF-8"); String s\_product\_name=URLDecoder.*decode*(product\_name, "UTF-8");

String s\_product\_chengse=URLDecoder.*decode*(product\_chengse, "UTF-8"); System.*out*.println(s\_product\_title);

Product product=new Product(s\_product\_name,s\_product\_title,Double.*valueOf*(product\_price),Double.*valueOf*(product

\_sale\_price),

date,category,0,s\_product\_chengse,userId); logger.info("将新增的一条商品信息插入数据库"); boolean yn = productService.add(product);

if (!yn) {

logger.warn("产品添加失败！事务回滚"); jsonObject.put("success", false);

throw new RuntimeException();

}

int product\_id = lastIDService.selectLastID();

logger.info("添加成功！,新增产品的 ID 值为：{}", product.getProduct\_id());

+ 1))

if (productSingleImageList != null && productSingleImageList.length > 0) { logger.info("整合商品子信息-商品预览图片");

List<ProductImage> productImageList = new ArrayList<>(5); for (String imageName : productSingleImageList) {

productImageList.add(new ProductImage()

.setProductImage\_type(0)

.setProductImage\_src(imageName.substring(imageName.lastIndexOf("/")

.setProductImage\_product(new Product().setProduct\_id(product\_id))

);

}

logger.info("共有{}条产品预览图片数据", productImageList.size()); for (ProductImage productImage:productImageList){

yn = productImageService.add(productImage);

}

if (yn) {

logger.info("产品预览图片添加成功！");

} else {

logger.warn("产品预览图片添加失败！事务回滚"); jsonObject.put("success", false);

throw new RuntimeException();

}

}

+ 1))

if (productDetailsImageList != null && productDetailsImageList.length > 0) { logger.info("整合产品子信息-产品详情图片");

List<ProductImage> productImageList = new ArrayList<>(5); for (String imageName : productDetailsImageList) {

productImageList.add(new ProductImage()

.setProductImage\_type(1)

.setProductImage\_src(imageName.substring(imageName.lastIndexOf("/")

.setProductImage\_product(new Product().setProduct\_id(product\_id))

);

}

logger.info("共有{}条产品详情图片数据", productImageList.size()); yn = productImageService.addList(productImageList);

if (yn) {

logger.info("产品详情图片添加成功！");

} else {

logger.warn("产品详情图片添加失败！事务回滚"); jsonObject.put("success", false);

throw new RuntimeException();

}

}

logger.info("产品信息及其子信息添加成功！");

jsonObject.put("success", true); jsonObject.put("product\_id", product\_id);

return jsonObject.toJSONString();

}

展示买家订单部分代码如下：

@RequestMapping(value = "order/{index}/{count}", method = RequestMethod.GET) public String goToPage(HttpSession session, Map<String, Object> map,

@RequestParam(required = false) Byte status, @PathVariable("index") Integer index*/\** 页数 *\*/*, @PathVariable("count") Integer count*/\** 行数*\*/*) {

logger.info("检查用户是否登录"); Object userId = checkUser(session); User user;

if (userId != null) {

logger.info("获取用户信息");

user = userService.get(Integer.*parseInt*(userId.toString()));

map.put("user", user);

} else {

return "redirect:/login";

}

Byte[] status\_array = null; if (status != null) {

status\_array = new Byte[]{status};

}

PageUtil pageUtil = new PageUtil(index, count); logger.info("根据用户ID:{}获取已购订单列表", userId);

List<ProductOrder> productOrderList = productOrderService.getList(new ProductOrder().setProductOrder\_user(new User().setUser\_id(Integer.*valueOf*(userId.toString()))), status\_array, new OrderUtil("productOrder\_id", true), pageUtil);

*//*统计已购商品订单总金额

double totalprice=0;

*//*订单总数量

Integer orderCount = 0;

if (productOrderList.size() > 0) {

orderCount = productOrderService.getTotal(new ProductOrder().setProductOrder\_user(new User().setUser\_id(Integer.*valueOf*(userId.toString()))), status\_array);

logger.info("获取订单项信息及对应的产品信息"); for (ProductOrder order : productOrderList) {

List<ProductOrderItem> productOrderItemList = productOrderItemService.getListByOrderId(order.getProductOrder\_id(), null);

if (productOrderItemList != null) {

for (ProductOrderItem productOrderItem : productOrderItemList) {

*//*统计已购商品订单总金额

Integer product\_id = productOrderItem.getProductOrderItem\_product().getProduct\_id();

totalprice =totalprice+productOrderItem.getProductOrderItem\_price(); Product product = productService.get(product\_id);

product.setSingleProductImageList(productImageService.getList(product\_id, 0, new PageUtil(0, 1)));

productOrderItem.setProductOrderItem\_product(product); if (order.getProductOrder\_status() == 3) {

productOrderItem.setIsReview(reviewService.getTotalByOrderItemId(productOrderItem.getProd uctOrderItem\_id()) > 0);

}

}

}

order.setProductOrderItemList(productOrderItemList);

}

}

pageUtil.setTotal(orderCount);

logger.info("获取商品分类列表信息");

List<Category> categoryList = categoryService.getList(null, new PageUtil(0, 5));

logger.info("统计已购商品订单总金额"+totalprice); map.put("totalBuyedPrice",totalprice);

map.put("pageUtil", pageUtil); map.put("productOrderList", productOrderList); map.put("categoryList", categoryList); map.put("status", status);

logger.info("转到前台-订单列表页"); return "fore/orderListPage";

}

## 代码优化分析

将页面相同的部分单独写成一个页面，然后再页面里引用。如：

<nav>

<%@ include file="include/navigator.jsp" %>

<div class="header">

<img src="${pageContext.request.contextPath}/res/images/fore/WebsiteImage/logo.png" style="height: 64px ;width: 200px" >

<div class="home\_nav\_title">

<img src="${pageContext.request.contextPath}/res/images/fore/WebsiteImage/header\_nav\_title.png">

<span>商品分类</span>

</div>

</div>

</nav>

# 测试

## 测试方案设计

### 测试策略

测试策略主要是设计功能测试和系统测试环节。功能测试：

测试系统各功能在正常情况下是否能够实现其功能。功能测试策略表如表 8-1 所示：

表 8-1 功能测试策略表

|  |  |
| --- | --- |
| 测试目标 | 确保测试的功能正常，包括商品信息发布、商品搜索、商品信息修改、商品下架、添加收藏商品、取消收藏商品、查询已收藏商品、登录注册、修改个人信息、查看  已售商品、查看资产、评价功能 |
| 测试范围 | 商品信息发布、商品搜索、商品信息修改、商品下架、添加收藏商品、取消收藏商品、查询已收藏商品、登录注册、修改个人信  息、查看已售商品、查看资产、评价功能 |
| 技术 | 利用有效的和无效的数据来执行各个功能，以核实一下内容：  在使用有效数据时得到预期结果。  在使用无效数据时显示错误消息或提示消息。  各业务规则都得到了正确的应用。 |
| 开始标准 | 进入网站首页 |
| 完成标准 | 缺陷 95%得到修改并且通过复测 |
| 测试重点和优先级 | 商品信息发布、添加收藏商品 |
| 需考虑的特殊事项 | 测试环境、网络问题、使用次数 |

系统测试：

主要目的检测系统是否达到需求对业务流程及数据流的处理是否符合标准， 检测系统对业务流处理是否存在逻辑不严谨及错误，检测需求是否存在不合理的标准及要求。此阶段基于功能测试的完成。

系统测试策略表如表 8-2 所示：

表 8-2 系统测试策略表

|  |  |
| --- | --- |
| 测试目标 | 检测需求中业务流程，数据流的正确性 |
| 测试范围 | 需求中明确的业务流程，或组合不同功能  模块而形成一个大的功能 |
| 技术 | 利用有效的和无效的数据来执行各个功能，以核实一下内容：  在使用有效数据时得到预期结果。 |

|  |  |
| --- | --- |
|  | 在使用无效数据时显示错误消息或提示消息。  各业务规则都得到了正确的应用。 |
| 开始标准 | 在完成某个集成测试时必须达到标准 |
| 完成标准 | 所计划的测试已全部执行  所发现的等级高的缺陷已经全部解决 |
| 测试重点和优先级 | 商品信息发布、商品下架、添加收藏商品 |
| 需考虑的特殊事项 | 测试环境、网络问题、使用次数 |

### 测试进度安排

测试进度安排表如表 8-3 所示：

表 8-3 测试进度安排表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试阶段 | 测试任务 | 工作量估计 | 人员分配 | 起止时间 |
| 第一阶段功能测试 | 1. 商品信息 2. 商品搜索 3. 商品信息修改 4. 商品下架 5. 添加收藏商品   是否能正确实现其功能，是否有  操作错误 | 4 日 |  | 2019 年 4 月  11 日至 2019  年 4 月 14 日 |
| 第二阶段系统测试 | 1）完成所有模块的组合测试2）确定所有业务流向和数据都  是正确的 | 3 日 |  | 2019 年 4 月  15 日至 2019  年 4 月 17 日 |
| 第三阶段  性能测试 | 多用户访问，交  替进行测试 | 1 日 |  | 2019 年 4 月  18 日 |
| 第四阶段  安装部署测试 | 对与项目部署进行测试 | 2 日 |  | 2019 年 4 月  19 日至 2019  年 4 月 20 日 |
| 第五阶段  兼容测试 | 软件的运行情况 | 1 日 |  | 2019 年 4 月  21 日 |

### 测试资源

1. 测试环境

ubuntu 操作系统，1G 独立显卡，CPU 为 INTEL CORE i5 处理器，硬盘 1T， 内存 8G，Tomcat7，Google Chrome，mysql。

1. 测试工具

Google Chrome，IntelliJ IDEA

1. 测试文档

软件测试计划文档、软件测试用例、缺陷报告文档。

### 关键测试点

在功能测试阶段中，注意商品信息发布操作测试、添加收藏商品操作测试。

## 测试用例构建

### 测试用例编写约定

用例编号： 唯一标识，与需求编号对应，为多对一关系用例名称： 对测试项简短的描述

前提条件： 执行用例时需要的预置条件操作步骤： 执行该动作需要完成的操作

预期结果： 执行完该动作后程序的表现结果实际结果： 实际输出的结果

问题描述： 执行该用例出现后系统显示的错误验证结果： 该测试用例是否执行通过

BUG 编号： 填写 bug 库中对应此用例的 BUG 编号

测试结果与结论：此次测试是否通过测试执行者：执行测试的人员

### 测试用例设计

商品信息发布测试用例如表 8-4 所示：

表 8-4 商品信息发布测试用例表

|  |  |
| --- | --- |
| 用例编号 | 01 |
| 用例名称 | 商品信息发布 |
| 前置条件 | 用户进入商品发布页面 |
| 操作步骤 | 1. 令标题为空，其他的均填写 2. 令商品描述为空，其他的均填写 3. 不上传商品图片，其他的均填写 4. 不选择商品分类，其他的均填写 5. 不填写商品新旧程度，其他的均填写 6. 所有的均填写 |
| 预期结果 | 1. 提示标题不能为空 2. 提示商品描述不能为空 3. 提示图片不能为空 4. 提示商品分类没有选择 |

|  |  |
| --- | --- |
|  | 1. 提示商品新旧程度没有选择 2. 弹出提示框显示发布成功 |
| 实际结果 | 与预期结果相符 |
| 问题描述 | 暂无 |
| BUG 编号 | 暂无 |
| 测试结果与结论 | 测试通过 |
| 测试执行者 |  |

商品搜索测试用例如表 8-5 所示：

表 8-5 商品搜索测试用例表

|  |  |
| --- | --- |
| 用例编号 | 02 |
| 用例名称 | 商品搜索 |
| 前置条件 | 用户进入首页 |
| 操作步骤 | 1. 在搜索框中输入不存在的关键词 2. 在搜索框中输入存在的关键字 |
| 预期结果 | 1. 提示没有相应的商品 2. 将查询出的商品显示在网页上 |
| 实际结果 | 与预期结果相符 |
| 问题描述 | 暂无 |
| BUG 编号 | 暂无 |
| 测试结果与结论 | 测试通过 |
| 测试执行者 |  |

商品信息修改测试用例如表 8-6 所示：

表 8-6 商品信息修改测试用例表

|  |  |
| --- | --- |
| 用例编号 | 03 |
| 用例名称 | 商品信息修改 |
| 前置条件 | 用户进入商品信息修改页 |
| 操作步骤 | 1. 将标题修改为空，其他的均不修改 2. 将商品描述为空，其他的均不修改 3. 删除商品图片，其他的均不修改 4. 不选择商品分类，其他的均不修改 5. 不填写商品新旧程度，其他的均不修改 6. 修改任意一个但不为空 |

|  |  |
| --- | --- |
| 预期结果 | 1. 提示标题不能为空 2. 提示商品描述不能为空 3. 提示图片不能为空 4. 提示商品分类没有选择 5. 提示商品新旧程度没有选择 6. 弹出提示框显示修改成功 |
| 实际结果 | 与预期结果相符 |
| 问题描述 | 暂无 |
| BUG 编号 | 暂无 |
| 测试结果与结论 | 测试通过 |
| 测试执行者 |  |

商品下架测试用例如表 8-7 所示：

表 8-7 商品下架测试用例表

|  |  |
| --- | --- |
| 用例编号 | 04 |
| 用例名称 | 商品下架 |
| 前置条件 | 用户进入已发布页面 |
| 操作步骤 | 点击下架按钮 |
| 预期结果 | 提示下架成功，并跳转到已下架页面 |
| 实际结果 | 与预期结果相符 |
| 问题描述 | 暂无 |
| BUG 编号 | 暂无 |
| 测试结果与结论 | 测试通过 |
| 测试执行者 |  |

添加收藏商品测试用例如表 8-8 所示：

表 8-8 添加收藏商品测试用例表

|  |  |
| --- | --- |
| 用例编号 | 05 |
| 用例名称 | 添加收藏商品 |
| 前置条件 | 用户进入商品详情页面 |
| 操作步骤 | 点击收藏按钮 |
| 预期结果 | 提示收藏成功，并跳转到收藏夹页面 |
| 实际结果 | 与预期结果相符 |
| 问题描述 | 暂无 |
| BUG 编号 | 暂无 |
| 测试结果与结论 | 测试通过 |
| 测试执行者 |  |

### 关键测试用例

商品信息发布测试用例表如表 8-9 所示：

表 8-9 商品信息发布测试用例表

|  |  |
| --- | --- |
| 用例编号 | 01 |
| 用例名称 | 商品信息发布 |
| 前置条件 | 用户进入商品发布页面 |
| 操作步骤 | 1. 令标题为空，其他的均填写 2. 令商品描述为空，其他的均填写 3. 不上传商品图片，其他的均填写 4. 不选择商品分类，其他的均填写 5. 不填写商品新旧程度，其他的均填写 6. 所有的均填写 |
| 预期结果 | 1. 提示标题不能为空 2. 提示商品描述不能为空 3. 提示图片不能为空 4. 提示商品分类没有选择 5. 提示商品新旧程度没有选择 6. 弹出提示框显示发布成功 |
| 实际结果 | 与预期结果相符 |
| 问题描述 | 暂无 |
| BUG 编号 | 暂无 |
| 测试结果与结论 | 测试通过 |
| 测试执行者 |  |

### 测试用例维护

1. 删除过时的测试用例

因为需求的改变等原因可能会使一个基线测试用例不再适合被测试系统，这些测试用例就会过时。例如，某个变量的界限发生了改变，原来针对边界值的测试就无法完成对新边界测试。所以，在软件的每次修改后都应进行相应的过时测试用例的删除。

1. 删除冗余的测试用例

如果存在两个或者更多个测试用例针对一组相同的输入和输出进行测试，那么这些测试用例是冗余的。冗余测试用例的存在降低了回归测试的效率。所以需要定期的整理测试用例库，并将冗余的用例删除掉。

1. 增添新的测试用例

如果某个程序段、构件或关键的接口在现有的测试中没有被测试，那么应该开发新测试用例重新对其进行测试。并将新开发的测试用例合并到基线测试包中。

# 总结与展望

## 设计工作总结

在此次毕设中我主要分为需求分析阶段，详细设计阶段，编码阶段，测阶阶段来完成这个系统。

需求分析阶段通过对业务需求，用户需求和系统需求进行分析，产生软件需求规格说明书，以及系统总体架构，各子系统及各功能模块。

详细设计阶段运用流程图，时序图，伪代码等描述某一模块内部的处理流程， 开发方法和编码技巧等，具体到各功能模块内部，分析、建立数据库表，以及各功能流程。

编码阶段采用 B/S 架构，利用 SSM 框架，采用 Java 编程语言，根据需求分析内容和详细设计内容编写系统代码，根据对数据结构、算法分析和模块实现等方面的设计要求，分别实现各模块的功能，从而实现目标系统的功能、性能、接口、界面等方面的功能。

测试阶段分为单元测试、集成测试、系统测试。当每写完一个功能模块时， 进行单元测试，不同模块之间进行集成测试，各模块均实现后，进行系统测试。

## 未来工作展望

经过这次毕业论文的编写，让我对整个软件开发流程有了更深刻的了解，知道整个软件系统是怎么开发出来的，也让我对 ssm 框架的使用更加熟练。虽然再功能上已经比较完善，但是在用户体验度上还不够完美，页面还需要美化。因此未来打算从页面，功能方面继续维护，完善该系统，并进行推广。若学生用户使用量增加，则会考虑将该系统做成分布式系统。

# 谢辞

# 引用文献

1. 贾晓芳，沈泽刚.Java Web 应用开发中的常见乱码形式及解决方法[J]. 软件导刊，2017,

（04）

1. 籍慧文.Web 应用开发中 JAVA 编程语言的应用探讨[J].科技创新与应用，2017, (07)
2. 苟文博，于强.基于 MySQL 的数据管理系统设计与实现[J].电子设计工程，2017,（06）
3. 杜丁超.计算机软件 Java 编程特点及其技术分析[J].电脑知识与技术,2017,13(36)
4. 蔡创.计算机软件开发中JAVA 编程语言的分析和思考[J].信息技术与信息化,2017(12) [6]张永强. 计算机软件 Java 编程特点及其技术分析[J]. 计算机产品与流通,2019(01):23.
5. 王越 . JAVA 编程语言在计算机软件开发中的应用 [J]. 电子技术与软件工程,2019(01):35.
6. 叶欣,李建佳,温煜,赖舒婷. 基于 Java 语言的医疗质量管理系统设计与实现[J]. 中国数字医学,2018,13(12):54-55+104.
7. 曹文 渊 . JAVA 语言 在 计算 机软 件开 发中 的 应用 [J]. 电子 技术 与 软件 工程,2019(02):53-54.
8. 倪海顺. 计算机软件开发的 Java 编程语言应用探讨[J]. 信息与电脑( 理论版),2019(02):60-61.
9. Rongrong Gong, Shijian Luo, Ji He.Use Case Based Innovative Design of E-commerce Website ，Journal of Computer Science and Technology 2012，5：32-35
10. Jan Hatzius，Goldman Sachs Investment Research. Internet B2B eCommerce B2B:2B or Not 2B? Version1.1,2010,12: 68-71.
11. 李荣国，王见.MySQL 数据库在自动测试系统中的应用[J].计算机应用，2019（31）：

169-171.

1. 宋国柱，陈俊杰.基于 Mysql 数据库的设计与实现[J].计算机工程与设计，2019（24）：

5771-5773.

1. 邵志远，金海，唐晓辉.高性能高可用 MySQL 数据库集群[J].计算机研究与发展，2019，

42（6）：1006-1012.

[16] 李书振.MySQL 数据库的安全机制[J].计算机应用，2019，22（6）：51-53. [17] 赵廷涛，刘冰.linux 下的 MySQL 数据库编程[M].北京：清华大学出版社，2019. [18] 马建红.JSP 应用与开发技术.第三版.清华大学出版社.2018

[19] 罗如为. Java Web 开发技术与项目实战.水利水电出版社.2019 [20] 明日科技.Java 从入门到精通.第五版.清华大学出版社.2019 [21] 杨占胜.JSP WEB 应用程序开发教程.电子工业出版社.2018

[22] 赵钢. JSP Servlet+EJB 的 Web 模式应用研究[J]. 电子设计工程, 2013, 21(13):47-49 [23] 肖英. 解决 JSP/Servlet 开发中的中文乱码问题[J]. 科技传播, 2011, (1)11-25

1. 黄艳峰. 在Java 语言中实施“案例教学”的研究与探索[J]. 电脑知识与技术, 2010, 6(5):1148-1149
2. 邓子云.JSP 网络编程从基础到实践[M].北京:电子工业出版社,2009. [26] Todd Cook. JSP 从入门到精通[W].北京：电子工业出版社，2003·

[27] Bruce Eckel. Thinking in java(English Edition). 北京：机械工业出版社,2006· [28] 赵安君,曾应员,徐邦海，常春藤.网络安全技术于应用[M].北京:人民邮电出版

社,2007.

[29] 范明,叶阳东,邱保志,职为梅.数据库原理教程[M].北京:科学出版社,2008. [30] 孙涌.现代软件工程[M].北京:北京希望电子出版社,2003.

# 附录 A 外文翻译—原文部分

外文出处：An Introduction to Database Systems

An Introduction to Database Systems

A Database Management System (DBMS) consists of a collection of interrelated data and a set of programs to access those data. A database is a collection of data organized to server many applications efficiently by centralizing the data and minimizing redundant data. The primary goal of a DBMS is to provide an environment that is both convenient and efficient to use in retrieving and storing database information.

Database systems are designed to manage large bodies of information. The management of data involves both the definition of structures for the storage of information and the provision of mechanisms for the manipulation of information stored, despite system must avoid possible anomalous results.

The important of information in most organizations, which determines the value of the database, has led to the development of a large body of concepts and techniques for the database, has led to the development of a large body of concepts and techniques for the efficient management of data.

The typical file-processing system is supported by a conventional operating system. Permanent records are stored in various files, and different application programs are written to extract records from, and to add records to, the appropriate files. Before the advent of DBMSs, organizations typically stored information using such systems.

For the system to be usable, it must retrieve data efficiently. This concern has led to the design of complexity from users through several levels of abstraction, to simplify users’ interactions with the system:

A: Physical levels. The lowest level of abstraction describes how the data are actually stored. At the physical level, complex low-level data structures are described in detail.

B: Logical level. The next-higher level of abstraction describes what data are stored in database, and what relationships exist among those data. The entire database is thus described in terms of a small number of relatively simple structures. Although implementation of the simple structures at the logical level may involve complex physical-level structures, the user of the logical level does not need to be aware of this

complexity. The logical level of abstraction is used by database administrators, who must decide what information is to be kept in the database.

C: View level. The highest level of abstraction describes only part of the entire database. Despite the use of simpler structures at the logical level, some complexity remains, because of the large size of the database. Many users of the database system will not be concerned with all this information. Instead, such users need to access only a part of the database. So that their interaction with the system is simplified, the view level of abstraction is defined. The system may provide many views for the same database.

The ability to modify a schema definition in one level without affecting a schema definition in the next higher level is called data independence. There are two levels of data independence:

Physical data independence is the ability to modify the physical schema without causing application programs to be rewritten. Modifications at the physical level are occasionally necessary to improve performance.

Logical data independence is the ability to modify the logical schema without causing application programs to be rewritten. Modifications at the logical level are necessary whenever the logical structure of the database is altered.

Logical data independence is more difficult to achieve than physical data independence, since application programs are heavily dependent on the logical structure of the data that they access.

The concept of data independence is similar in many respects to the concept of abstract data types in modem programming languages. Both hide implementation details from the users, to allow users to concentrate on the general structure, rather than on low-level implementation details.

The structured query language (SQL) is the most widely used and standard query language for relational database management systems. It is a kind of non-procedural language. The SQL language has several parts:

Data-definition language (DDL). The SQL DDL provides commands for defining relation schemas, deleting relations, creating indices, and modifying relation schemas. Interactive data-manipulation language (DML). The SQL DML include a query language based on both the relational algebra and the tuple relational calculus. It includes also commands to insert tuples into, delete tuples from, and modify tuples in

the database.

Embedded DML. The embedded form of SQL is designed for use within

general-purpose programming languages.

View definition. The SQL DDL include commands for defining views.

Authorization. The SQL DDL includes commands for specifying access rights to relations and views.

Integrity. The SQL DDL includes commands for specifying integrity constraints that the data stored in the database must satisfy. Updates that violate integrity constraints are disallowed.

Transaction control. SQL includes commands for specifying the beginning and ending of transactions. Several implementations also allow explicit locking of data for concurrency control.

A transaction is a collection of operations that performs a single logical function in a database application. Each transaction is a unit of both atomicity and consistency. Thus, we require that transaction do not violate any database consistency constraints. That is, if the database was consistent when a transaction started, the database must be consistent when the transaction successfully terminates. However, during the execution of a transaction, it may be necessary temporarily to allow inconsistency. This temporary inconsistency, although necessary, may lead to difficulty if a failure occurs.

Using database system to manage data has the following characteristics:

1. Data structure

When describing data, it is necessary to describe not only the data itself, but also the connections between the data. Data structuring is one of the main characteristics of databases, and also the essential difference between database systems and file systems. In the same database, there is a connection between the various data files, and they follow a certain structural form as a whole (Figure).

1. High data sharing, low redundancy and easy expansion

High sharing: The database system views and describes the data from an overall perspective. The data is no longer for an application but for the entire system, which can reduce data redundancy and save storage space. That is, the data in the database can be used by multiple users and multiple applications simultaneously.

Low redundancy: Redundancy refers to the degree to which the same data is stored repeatedly. Due to the structure of the data in the database system, the redundancy may be reduced to the lowest level.

Easy to expand: Because the design mainly considers data structure, that is, it is oriented to the system, not to an application, so it is easy to expand.

Data sharing and reducing redundancy can also avoid incompatibility and inconsistency between data.

Incompatibility: For example, someone worked in two departments, 1986-1990 in department A, 19901996 in department B, when writing archives, department A was written as 1986-1990, because information was not shared, plus Work negligence, Department B was written from 1989 to 1996, which caused incompatibility, and the two departments repeated for 1 year.

Inconsistency: For example, if an employee is "Li Ping", the name entered by the department is "Li Ping" because the information is not shared, but the name entered by the salary record is "Li Ping", which causes inconsistency.

1. High data independence

Since the application access is not directly accessed from the database, but indirectly accessed through the DBMS, it maintains the physical and logical independence of the application and database data. The so-called physical independence means that when the storage structure (or physical structure) of the data changes, the corresponding change of the DBMS can keep the logical structure of the data unchanged, so that the application does not have to change. Logical independence means that when the overall logical structure of the data changes, the local logical structure of the data can be kept unchanged through the corresponding changes of the DBMS. The application is written according to the local logical structure of the data, so the application does not have to change.

1. The data is managed and controlled by the DBMS

Since the sharing of the database is concurrent sharing, that is, multiple users can access the data in the database at the same time, or even access the same data in the database at the same time. To this end, the DBMS must provide the following data control functions:

1. Data security protection refers to the protection of data to prevent data leakage and destruction caused by illegal use of data, so that each user can only access and process certain data in certain ways according to regulations.
2. Data integrity check refers to the correctness, validity and compatibility of data. That is, control the data within the effective range, or require a certain relationship between the data.

Correctness: If you enter the salary, you should enter the value, but the actual input of the character is incorrect.

Validity: If the age of the person is lost, the data between 0 and 150 should be lost, but

the actual input of -5 is invalid.

Compatibility: When entering the feed proportion percentage, the sum of the five raw material percentages should add up to 100, and the actual input data should be greater than 100, which is incompatible.

1. Concurrency control refers to controlling multiple users to simultaneously access and modify data in the database to ensure the integrity of the database. For example, multiple users can read at the same time, but only one user can write data at a time.
2. Database recovery refers to the function of recovering the database from an error state to a known correct state (also called a complete state or a consistent state). Computer system hardware failures, software failures, operator errors, and deliberate destruction will affect the accuracy of the data in the database, and even cause the loss of some or all of the data in the database.

In the late 1960s, the scale of data processing using electronic computers became larger and larger. At the same time, large-capacity and inexpensive disk storage appeared, and the operating system became mature. In order to solve the problem of data independence, realize the unified management of data, and achieve the purpose of data sharing, database technology has emerged. Database technology was born from the mid-1960s to today. It has a history of more than 30 years, has experienced three generations of evolution, and has developed into a subject with rich content based on data modeling and DBMS core technology; it has driven a huge software industry ——DBMs products and related tools and solutions. Database technology has achieved great success, from the first-generation hierarchical and network database system and the second-generation relational database system to the third-generation new-generation database system with the object-oriented database model as the main feature.

1. The first generation database system

Is a hierarchical and mesh database system developed in the 1970s. In 1963, the IDS system designed and developed by Bachman began to operate. It can share databases for multiple COBOL programs; in 1968, mesh database systems such as TOTAL began to appear; in 1969, the hierarchical database system developed by McGee et al. Of IBM Corporation Published by the IMS system, it allows multiple programs to share the database. In the month of 1969, the CODASYL database developer proposed a network model database system specification report, which made the database system begin to be standardized and standardized. In 1971, the database task group of the American Database System Language Association conducted a

systematic research and discussion on the network database method, and put forward several reports called the DBTG report. The DBTG report identified and established many concepts, methods and methods of the network database system. Technology, proposed three levels of abstract mode, that is, the external mode that describes the part of the data structure required by the application, the conceptual mode that describes the data structure of the entire guest system, and the internal mode that describes the data storage structure, which solves the data independence Problem, marking the theoretical maturity of the database. Because of this, many experts believe that database technology originated in the late 1960s.

1. The second generation database system

Is a relational database system. In 1970, IBM Codd published a paper entitled "Relational Models for Large-Scale Shared Database Data", proposed a relational data model, and pioneered the relational database method and relational database theory. The relational method has played a vital role in the development of database technology due to its theoretical perfection and simple structure, and successfully laid a theoretical foundation for the relational database technology. The 1970s was an era of relational database theoretical research and prototype system development. The research on relational database system has achieved a series of results, including the following aspects:

1. lays the theoretical foundation of the relationship model, and gives a specification of the relationship model that is generally accepted by people.
2. proposed relational data languages, such as relational algebra, relational calculus, SQL language, QBE, etc. These descriptive languages have changed the process-oriented styles of previous programming languages and meshed, hierarchical database languages, and have been welcomed by end users for their ease of learning and understanding, laying the foundation for database language standardization in the 1980s.
3. has developed a large number of relational database system prototypes, and has overcome a series of key technologies such as query optimization, concurrency control, and fault recovery in system implementation. Not only does it greatly enrich the database management system implementation technology and database theory, but more importantly, it promotes the vigorous development and wide application of relational database system products.
4. The third generation database system

Since the 1980s, database theory and applications have entered a period of mature

development. With the development of computing technology and computer networks, the field of computer applications has rapidly expanded, and the field of database applications has also continued to expand. The huge achievements of database technology in the business field have stimulated the rapid growth of demand for databases in other fields. On the one hand, new database application areas, such as computer-aided design / management (CAD / CAM), process control, office automation systems, geographic information systems (GIS), computer manufacturing systems (CIMS), etc., have opened up new applications for databases On the other hand, the new demands of management in practical applications also directly promote the research and development of database technology.

Traditional databases represented by relational databases have been difficult to meet the needs of new fields, because new applications require the database to handle data with higher complexity, such as processing time-related attributes, and even require the database to be dynamic and proactive . In this way, there must be a new database technology to meet the actual needs. In order to meet the needs of modern applications, the database technology must be combined with other modern data processing technologies (such as object-oriented technology, timing and real-time processing technology, artificial intelligence technology, multimedia technology) Perfect integration to form "new generation database technology", also known as "modern database technology", such as temporal database technology, real-time database technology and multimedia database technology. The main type of database in the third-generation database system is the object-oriented database system (OODBS), which is the product of the combination of database technology and object-oriented programming methods. It is both a DBMS and an object-oriented system. Therefore, it not only has DBMS characteristics, such as persistence, auxiliary management, data sharing (concurrency), data reliability (transaction management and recovery), query processing and mode modification, but also has object-oriented characteristics, such as type / class, encapsulation Features such as sex

/ data abstraction, inheritance, object identification, compound objects, and scalability.

# 附录 B 外文翻译—译文部分

对于数据库系统的介绍

一个数据库管理系统由一些相关的数据和一组用来访问那些数据的程序组成。一个数据库是一个为了有效地服务各种应用软件的数据组织的集合通过集中并减少数据的冗余。数据库管理系统最主要的一个目标就是去提供一个既方便又有效地查询、存储数据信息系统的环境。

数据库系统被设计用来管理拥有很大数据量的信息。这些数据的管理包括存储信息结构的定义和处理存储信息的规范机制，尽管系统必须避免可能的不规则的结果。

在大多数的机构里信息有着重要性，这点决定了数据库的价值，让数据库的观念和技术有了很大的发展，让有效的数据管理观念和技术有了很大的发展。典型的文件处理系统被传统的操作系统支持。永久性的记录存储在各种各样

的文件中，并且，不同应用程序被编写用来读取记录和添加记录在恰当得文件里。在数据库管理系统出现以前，组织代表性的信息存储用那些系统来实现。

为了使系统可用，它必须有效地检索数据。这种担忧导致了从用户到多个抽象级别的复杂性设计，以简化用户与系统的交互：

A：物理级别。最低的抽象级别描述了数据的实际存储方式。在物理级别， 将详细描述复杂的低级数据结构。

B：逻辑级别。更高的抽象级别描述了哪些数据存储在数据库中以及这些数据之间存在什么关系。因此，根据少量的相对简单的结构来描述整个数据库。尽管在逻辑级别上实现简单结构可能会涉及复杂的物理级别结构，但是逻辑级别的用户并不需要意识到这种复杂性。数据库管理员使用逻辑抽象级别，管理员必须决定要在数据库中保留哪些信息。

C：视图级别。提取的最高级别只描述整体的数据库。尽管简单结构的使用在逻辑级，一些复杂的残余因为数据库过大。许多数据库系统的用户没有参与到所有的这些信息。取而代之的是，这些用户需要访问的只是数据库的一部分。因此，他们与系统的交互作用是单一化的，提取的视图级别被定义。系统可以为相同的数据库提供很多视图。

在一个级别中修改模式定义而不影响下一个更高级别的模式定义的能力称为数据独立性。数据独立性有两个级别：

物理数据独立性是在不导致应用程序程序被重写的情况下修改物理模式的能力。有时需要在物理级别进行修改以提高性能。

逻辑数据独立性是在不导致应用程序被重写的情况下修改逻辑架构的能力。每当更改数据库的逻辑结构时，都需要在逻辑级别进行修改。

逻辑数据独立性比物理数据独立性更难实现，因为应用程序在很大程度上依赖于它们访问的数据的逻辑结构。

数据独立性的概念在许多方面与现代编程语言中的抽象数据类型的概念相似。两者都向用户隐藏了实施细节，以使用户可以专注于总体结构，而不是底层

的实施细节。

结构化查询语言（SQL）是关系数据库管理系统中使用最广泛的标准查询语言。它是一种非过程语言。SQL 语言包含几个部分：

* 1. 数据定义语言（DDL）。SQL DDL 提供了用于定义关系架构，删除关系，创建索引以及修改关系架构的命令。
  2. 交互式数据处理语言（DML）。SQL DML 包括基于关系代数和元组关系演算的查询语言。它还包括用于在数据库中插入元组，从中删除元组以及修改元组的命令。
  3. 嵌入式 DML。SQL 的嵌入式形式设计为在通用编程语言中使用。
  4. 查看定义。SQL DDL 包含用于定义视图的命令。
  5. 授权。SQL DDL 包含用于指定对关系和视图的访问权限的命令。
  6. 诚信 SQL DDL 包括用于指定数据库中存储的数据必须满足的完整性约束的命令。禁止违反完整性约束的更新。
  7. 交易控制。SQL 包含用于指定事务开始和结束的命令。几种实现方式还允许显式锁定数据以进行并发控制。

事务处理是一组操作的集合，在数据库应用程序中执行单独的逻辑功能。每一个事务处理是一组集合即包括原子数又包括连接数。因此，我们需要哪些食物处理程序不要违反任何数据库连接约束。更确切的说，如果当一个事物处理开始时数据库是协调一致的，当这个事务处理结束时，数据库也应该是协调一致的。然而，在一个事物处理执行期间，它可能临时的允许不一致。这种暂时性的不一致，尽管是必须得，如果一个故障发生，可能导致困难形成。

用数据库系统来管理数据具有如下特点：

1.数据结构化

在描述数据时不仅要描述数据本身，还要描述数据之间的联系。数据结构化是数据库的主要特征之一，也是数据库系统与文件系统的本质区别。在同一数据库中，各个数据文件存在联系，且在整体上服从一定的结构形式(图)。

2.数据共享性高、冗余度低、易扩充

共享性高：数据库系统从整体角度看待和描述数据，数据不再面向某个应用而是面向整个系统，这样可以减少[数据冗余](https://wiki.mbalib.com/wiki/%E6%95%B0%E6%8D%AE%E5%86%97%E4%BD%99)，节约存储空间。即数据库中的数据可以被多个用户、多个应用同时使用。

冗余度低：冗余度是指同一数据被重复存储的程度，数据库系统由于数据结构化，使得冗余度可能降到最低程序。

易扩充：由于设计时主要考虑数据结构化，即面向系统，而不是面向某个应用，所以容易扩充。

数据共享和减少冗余还能避免数据之间的不相容性和不一致性。

不相容性：例如，某人工作了两个部门，1986-1990 年在甲部门，19901996 年在乙部门，在写档案材料时，甲部门写成了 1986-1990 年，由于信息不共享，

加之工作疏忽，乙部门写成了 1989-1996 年，即造成了不相容，两部门间重复了

1 年。

不一致性：例如，某职工为“李萍”，由于信息不共享，系别输入的姓名为“李萍”，但工资记录输入的姓名为“李平”，即造成了不一致。

3.数据独立性高

由于应用程序取数不是直接从数据库中取数，而是通过 DBMS 间接取数，故保持了应用与数据库数据的物理独立性和逻辑独立性。所谓物理独立性是指：当数据的存储结构(或物理结构)改变时，通过 DBMS 的相应改变可以保持数据的逻辑结构不变，从而应用程序也不必改变。逻辑独立性是指：当数据的总体逻辑结构改变时，通过 DBMS 的相应改变可以保持数据的局部逻辑结构不变，应用程序是依据数据的局部逻辑结构编写的，所以应用程序不必改变。

4.数据由 DBMS 统一管理和控制

由于数据库的共享是并发的共享，即多个用户可以同时存取数据库中的数据， 甚至可以同时存取数据库中同一个数据。为此 DBMS 必须提供以下几方面的数据控制功能：

(1)数据的安全性保护指保护数据，防止不合法使用数据造成数据的泄密和破坏，使每个用户只能按规定，对某些数据以某些方式进行访问和处理。

(2)数据的完整性检查指数据的正确性、有效性和相容性。即将数据控制在有效的范围内，或要求数据之间满足一定的关系。

正确性：如输入工资时，应该输入数值，而实际输入了字符，即不正确。

有效性：如输人年龄时，应该输人 0～150 之间的数据，而实际输入了-5， 即无效。

相容性：如输入饲料配比百分比时，应该 5 种原料百分比之和加起来为 100， 而实际输入数据加起来大于 100，即不相容。

(3)并发控制指控制多个用户同时存取、修改数据库中的数据，以保证数据库的完整性。例如，多个用户可以同时读，但同一时间只能允许一个用户写数据。

(4)数据库恢复指将数据库从错误状态恢复到某一已知的正确状态(亦称为完整状态或一致状态)的功能。计算机系统的硬件故障、软件故障、操作员的失误，以及故意的破坏均会影响数据库中数据的正确性，甚至造成数据库部分或全部数据的丢失。

20 世纪 60 年代后期，使用电子计算机进行数据处理的规模越来越大，同时出现了大容量且价格低廉的磁盘存储器，操作系统也日渐成熟。为了解决数据的独立性问题，实现数据的统一管理，达到数据共享的目的，出现了数据库技术。数据库技术从 20 世纪 60 年代中期产生到今天，仅仅 30 多年的历史，经历了三代演变，发展成为以数据建模和 DBMS 核心技术为主，内容丰富的一门学科；带动了一个巨大的软件产业——DBMs 产品及其相关工具和解决方案。数据库技术取得了巨大的成功，从第一代层次与网络数据库系统和第二代关系数据库系统，发展到第三代以面向对象数据库模型为主要特征的新一代数据库系统。

1.第一代数据库系统

是 20 世纪 70 年代研制的层次和网状数据库系统。1963 年，Bachman 设计开发的 IDS 系统开始投入运行，它可以为多个 COBOL 程序共享数据库；1968 年， TOTAL 等网状数据库系统开始出现；1969 年，IBM 公司 McGee 等人开发的层次式数据库系统的 IMS 系统发表，它可以让多个程序共享数据库。1969 年 lo 月， CODASYL 数据库研制者提出了网络模型数据库系统规范报告，使数据库系统开始走向规范化和标准化。1971 年，美国数据库系统语言协会下属的数据库任务组对网络数据库方法进行了系统的研究、探讨，提出了称为 DBTG 报告的若干报告， DBTG 报告确定并建立了网络数据库系统的许多概念、方法和技术，提出了三级抽象模式，即对应用程序所需的那部分数据结构描述的外模式，对整个客体系统数据结构描述的概念模式，对数据存储结构描述的内模式，解决了数据独立性的问题，标志着数据库在理论上的成熟。正因为如此，许多专家认为数据库技术起源于 20 世纪 60 年代末。

2.第二代数据库系统

是关系数据库系统。1970 年 IBM 公司 Codd 发表了题为“大型共享数据库数据的关系模型”的论文，提出了关系数据模型，开创了关系数据库方法和关系数据库理论。关系方法由于其理论上的完美和结构上的简单，对数据库技术的发展起了至关重要的作用，成功地为关系数据库技术奠定了理论基础。20 世纪 70 年代是关系数据库理论研究和原型系统开发的时代。关系数据库系统的研究取得了一系列的成果，主要包括以下几个方面：

(1)奠定了关系模型的理论基础，给出了被人们普遍接受的关系模型的规范说明。

(2)提出了关系数据语言，如关系代数、关系演算、SQL 语言、QBE 等。这些描述性语言一改以往程序设计语言和网状、层次数据库语言的面向过程的风格， 以其易学易懂的优点得到了最终用户的欢迎，为 20 世纪 80 年代数据库语言标准化打下了基础。

(3)研制了大量的关系数据库系统原型，攻克了系统实现中查询优化、并发控制、故障恢复等一系列关键技术。不仅大大丰富了数据库管理系统实现技术和数据库理论，更重要的是促进了关系数据库系统产品的蓬勃发展和广泛应用。

3.第三代数据库系统

20 世纪 80 年代以来，数据库理论和应用进入成熟发展时期。随着计算技术和计算机网络的发展，计算机应用领域迅速扩展，数据库应用领域也在不断地扩大，数据库技术在商业领域的巨大成就刺激了其他领域对数据库需求的迅速增长。一方面，新的数据库应用领域，如计算机辅助设计／管理(CAD／CAM)、过程控制、 办公自动化系统、地理信息系统(GIS)、计算机制造系统(CIMS)等，为数据库的 应用开辟了新的天地；另一方面，在实际应用中管理方面的新需求也直接推动了数据库技术的研究与发展。

以关系数据库为代表的传统数据库已经很难胜任新领域的需求，因为新的应用要求数据库能处理复杂性较高的数据，如处理与时间有关的属性，甚至还要求

数据库有动态性和主动性。这样就必须有新的数据库技术才能够满足现实需要， 为了满足现代应用的需求，必须将数据库技术与其他现代数据处理技术(如面向对象技术、时序和实时处理技术、人工智能技术、多媒体技术)完善地集成，以形成“新一代数据库技术”，也可称为“现代数据库技术”，如时态数据库技术、实时数据库技术和多媒体数据库技术等。第三代数据库系统中最主要的数据库类型是面向对象数据库系统(objectorienteddatabasesystem，简称 OODBS)，它是数据库技术与面向对象程序设计方法相结合的产物，它既是一个 DBMS，又是一个面向对象系统。因而既具有 DBMS 特性，如持久性、辅助管理、数据共享(并发性)、数据可靠性(事务管理和恢复)、查询处理和模式修改等，又具有面向对象的特征，如类型／类、封装性／数据抽象、继承性、对象标识、复合对象和可扩充等特性。

# 附录 C 软件使用说明书

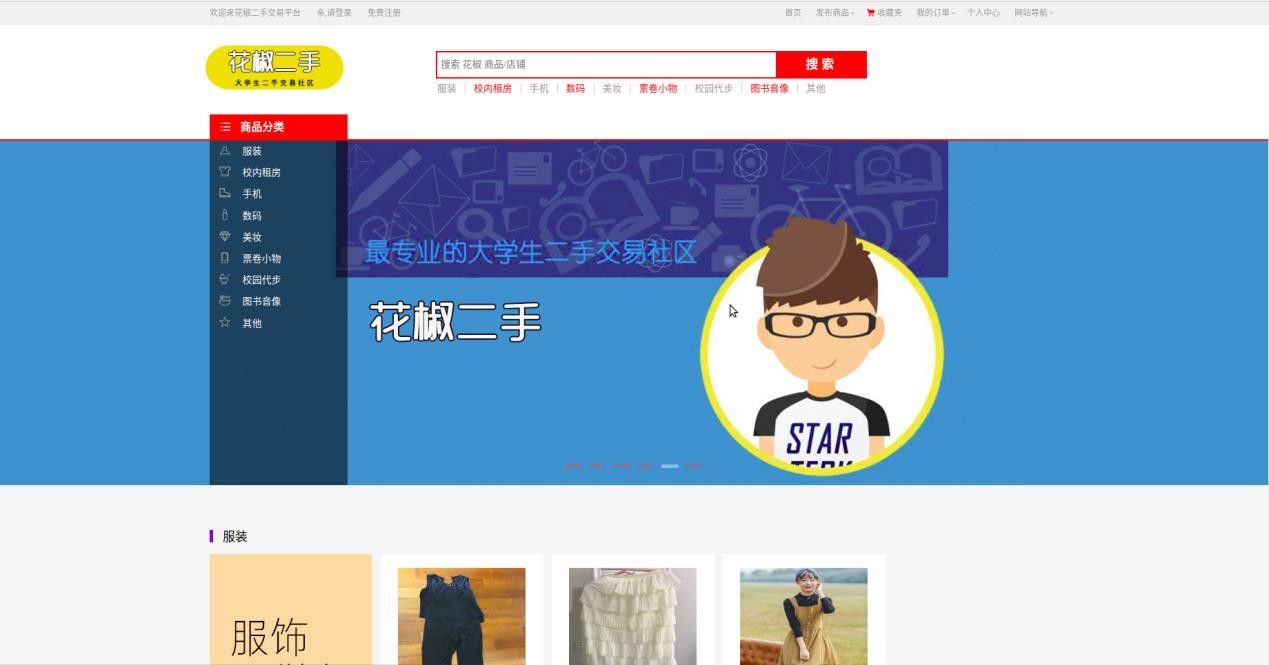
用户进入首页后可以浏览首页的商品。首页如图 1 所示：

图 1 首页

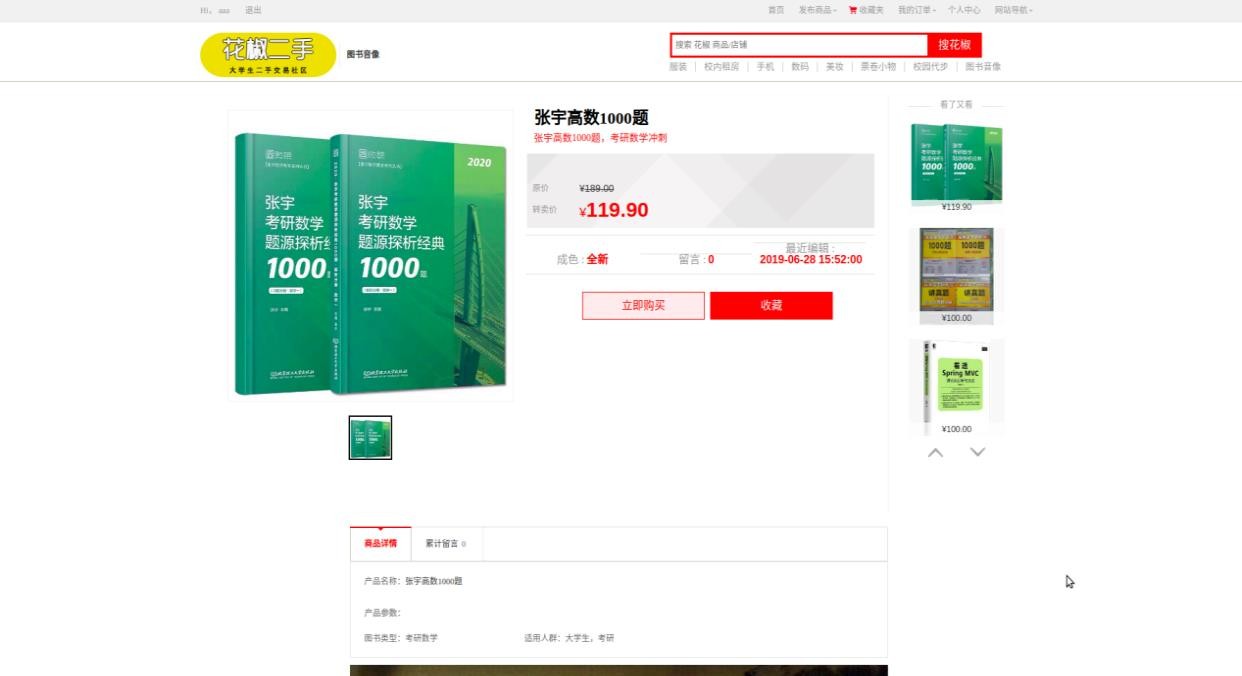
用户在首页看到心仪的商品可以点击进去查看商品详情。商品详情页如图 2 所示：

图 2 商品详情页

但是要加入购物车，发布商品或者购买商品必须要先登录，登录页面如图 3 所示：

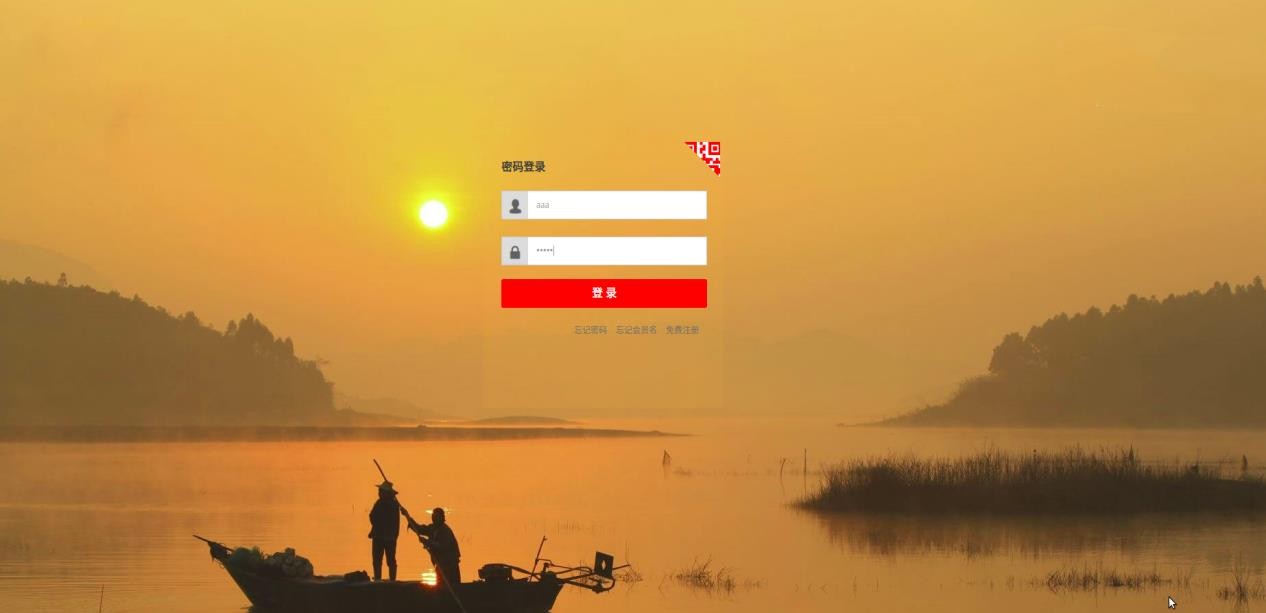


图 3 登录页

用户登陆后，可以点击商品详情页的收藏按钮加到个人收藏夹。收藏夹页面如图

4 所示：

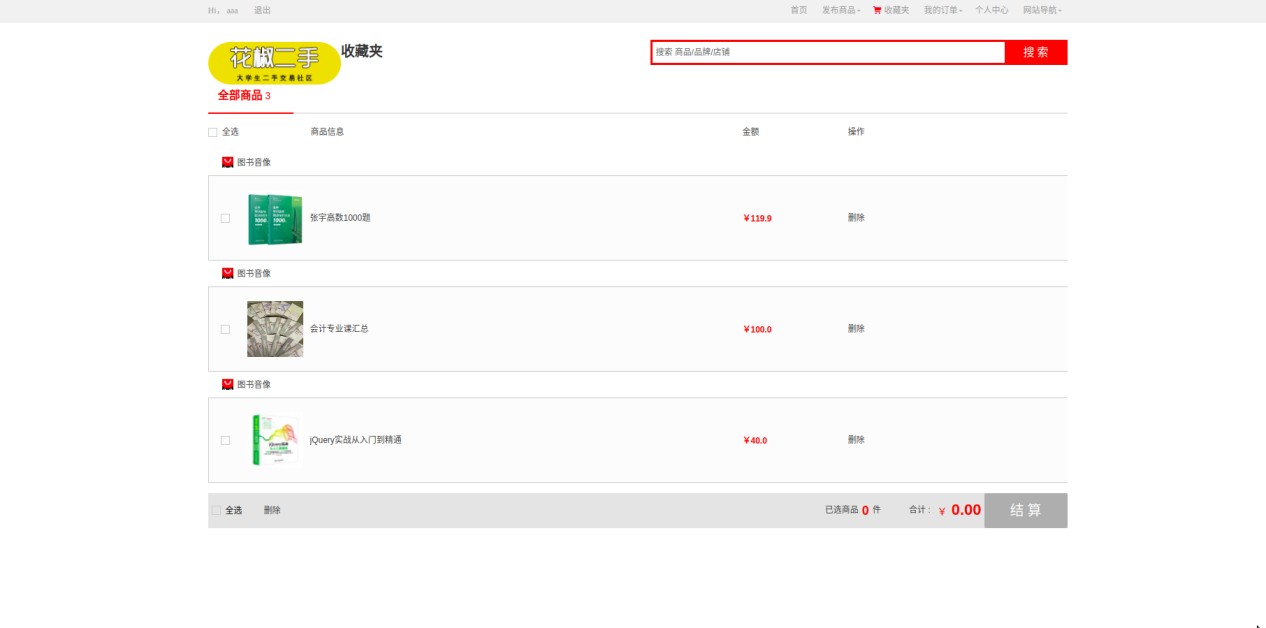


图 4 收藏夹页

用户登录后也可以点击立即购买按钮，从而跳转到收货地址页面，用户在收货地址页面编写收货地址信息。收货地址页面如图 5 所示：

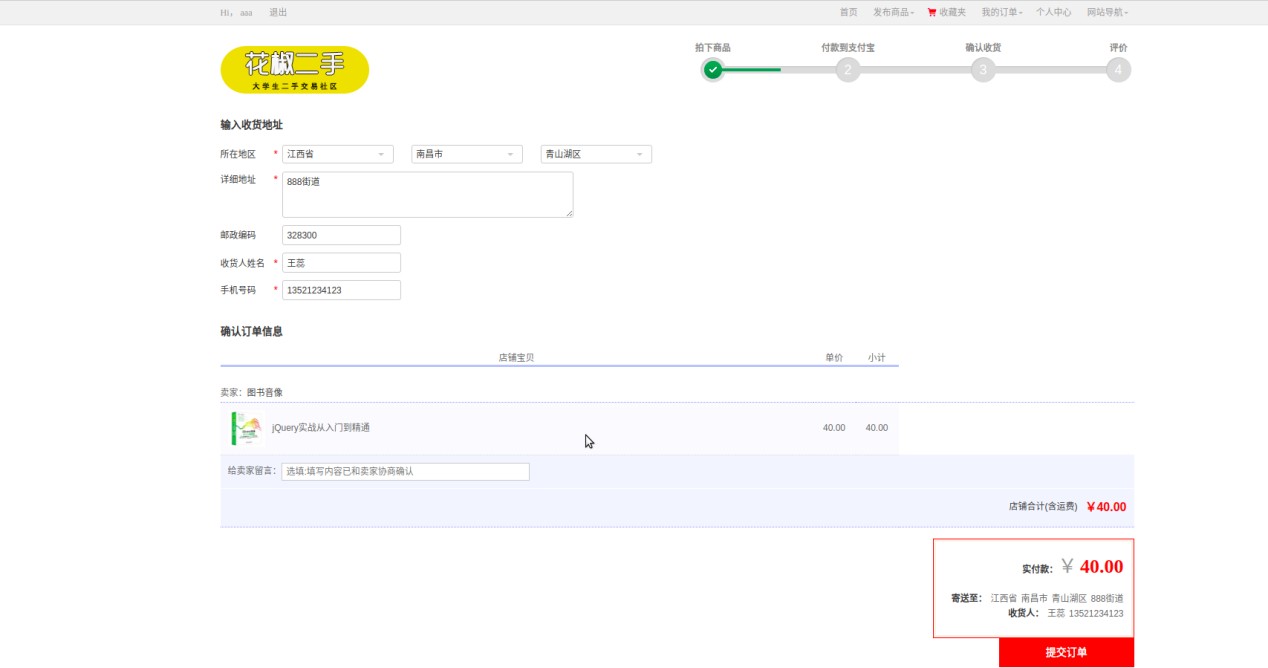


图 5 收货地址页

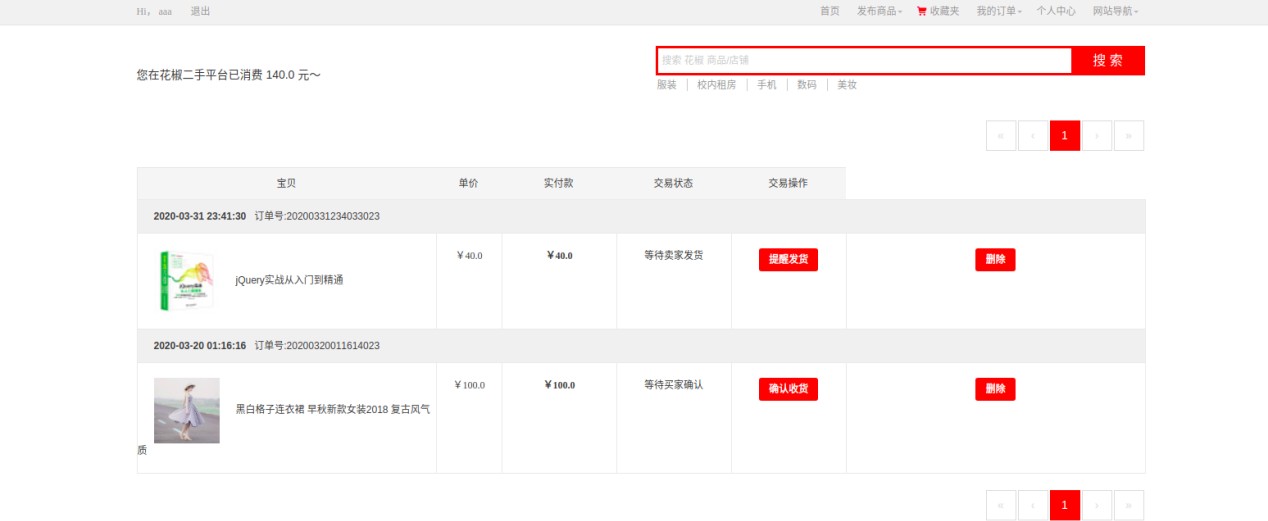
用户购买成功后，可以在订单页面看到自己的订单信息以及订单状态。订单页面如图 6 所示：

图 6 订单页

用户作为买家在登录后可以发布信息，在发布商品页面填写商品信息。发布物品页面如图 7 所示：

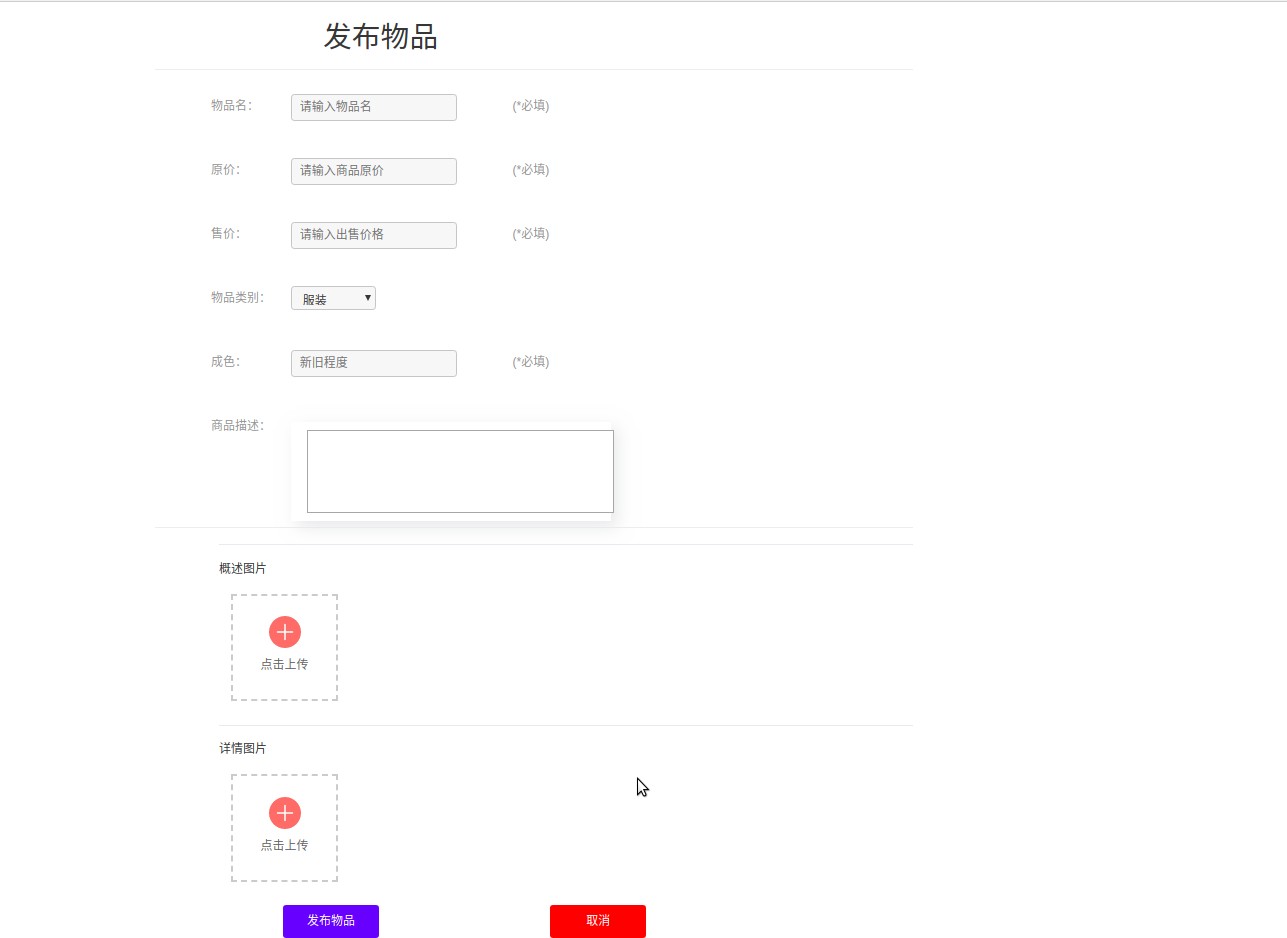


图 7 发布物品页

在发布商品之后可以点击页面导航已发布按钮查看已发布商品信息。已发布商品页面如图 8 所示：

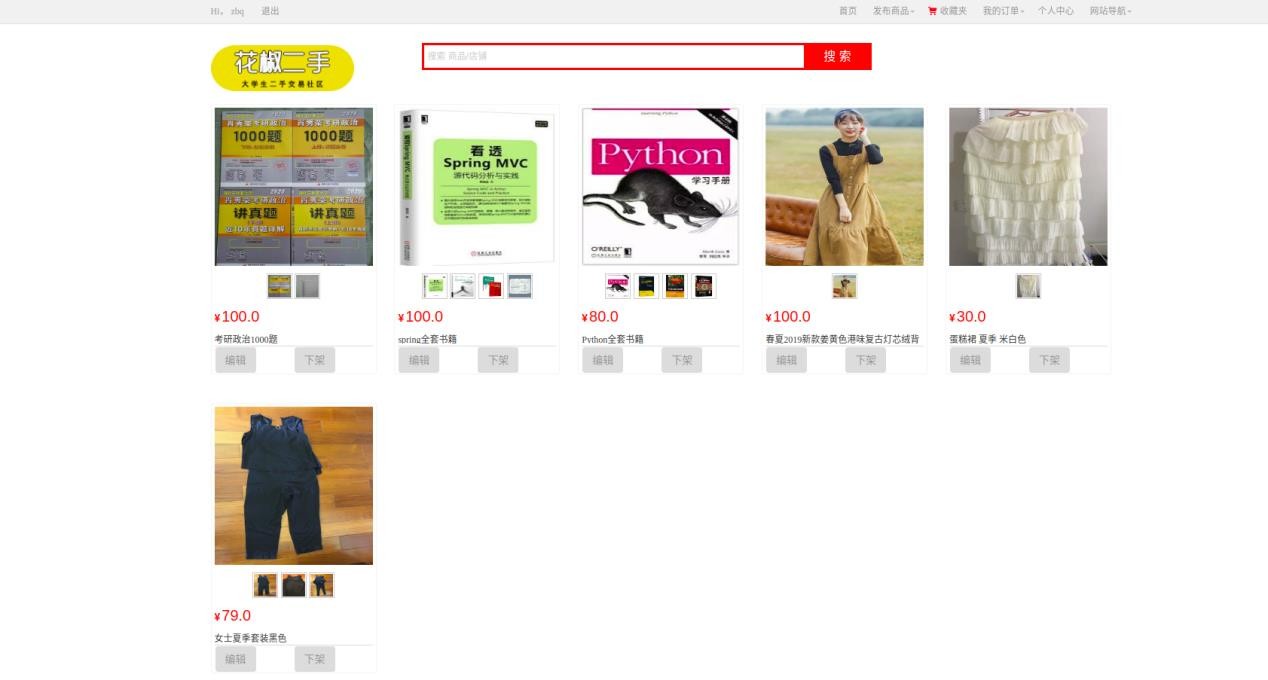


图 8 已发布物品页

# 附录 D 主要源代码

发布商品部分代码如下:

@RequestMapping(value = "publishGoods", method = RequestMethod.GET) public String publishGoods(Map<String, Object> map, HttpSession session) {

logger.info("检查用户是否登录");

Integer userId = (Integer) checkUser(session); User user;

if (userId != null) {

logger.info("获取用户信息");

user = userService.get(Integer.*parseInt*(userId.toString())); map.put("user", user);

} else {

return "redirect:/login";

}

logger.info("获取产品分类列表");

List<Category> categoryList = categoryService.getList(null,null); map.put("categoryList",categoryList);

return "fore/pubGoods";

}

提交发布的商品信息部分代码如下：

@ResponseBody

@RequestMapping(value = "publishGoodsSubmit",method = RequestMethod.POST,produces = "application/json;charset=utf-8")

public String publishGoodsSubmit(HttpSession session,

@RequestParam String product\_name, @RequestParam String product\_price, @RequestParam String product\_sale\_price, @RequestParam String catelogId, @RequestParam String product\_chengse, @RequestParam String product\_title, @RequestParam(required = false) String[]

productSingleImageList*/\**产品预览图片名称数组*\*/*,

@RequestParam(required = false) String[] productDetailsImageList*/\**产品详情图片名称数组*\*/*) throws Exception {

JSONObject jsonObject=new JSONObject();

*//*查询出当前用户 *user* 对象，便于使用 *id*

Integer userId = (Integer) session.getAttribute("userId");

Category category = categoryService.get(Integer.*valueOf*(catelogId));

*//*当前日期

Date date=new Date();

String s\_product\_title= URLDecoder.*decode*(product\_title, "UTF-8"); String s\_product\_name=URLDecoder.*decode*(product\_name, "UTF-8");

String s\_product\_chengse=URLDecoder.*decode*(product\_chengse, "UTF-8"); System.*out*.println(s\_product\_title);

Product product=new Product(s\_product\_name,s\_product\_title,Double.*valueOf*(product\_price),Double.*valueOf*(product

\_sale\_price),

date,category,0,s\_product\_chengse,userId); logger.info("将新增的一条商品信息插入数据库"); boolean yn = productService.add(product);

if (!yn) {

logger.warn("产品添加失败！事务回滚"); jsonObject.put("success", false);

throw new RuntimeException();

}

int product\_id = lastIDService.selectLastID();

logger.info("添加成功！,新增产品的 ID 值为：{}", product.getProduct\_id());

+ 1))

if (productSingleImageList != null && productSingleImageList.length > 0) { logger.info("整合商品子信息-商品预览图片");

List<ProductImage> productImageList = new ArrayList<>(5); for (String imageName : productSingleImageList) {

productImageList.add(new ProductImage()

.setProductImage\_type(0)

.setProductImage\_src(imageName.substring(imageName.lastIndexOf("/")

.setProductImage\_product(new Product().setProduct\_id(product\_id))

);

}

logger.info("共有{}条产品预览图片数据", productImageList.size()); for (ProductImage productImage:productImageList){

yn = productImageService.add(productImage);

}

if (yn) {

logger.info("产品预览图片添加成功！");

} else {

logger.warn("产品预览图片添加失败！事务回滚"); jsonObject.put("success", false);

throw new RuntimeException();

}

}

+ 1))

if (productDetailsImageList != null && productDetailsImageList.length > 0) { logger.info("整合产品子信息-产品详情图片");

List<ProductImage> productImageList = new ArrayList<>(5); for (String imageName : productDetailsImageList) {

productImageList.add(new ProductImage()

.setProductImage\_type(1)

.setProductImage\_src(imageName.substring(imageName.lastIndexOf("/")

.setProductImage\_product(new Product().setProduct\_id(product\_id))

);

}

logger.info("共有{}条产品详情图片数据", productImageList.size()); yn = productImageService.addList(productImageList);

if (yn) {

logger.info("产品详情图片添加成功！");

} else {

logger.warn("产品详情图片添加失败！事务回滚"); jsonObject.put("success", false);

throw new RuntimeException();

}

}

logger.info("产品信息及其子信息添加成功！");

jsonObject.put("success", true); jsonObject.put("product\_id", product\_id);

return jsonObject.toJSONString();

}

展示买家订单部分代码如下：

@RequestMapping(value = "order/{index}/{count}", method = RequestMethod.GET) public String goToPage(HttpSession session, Map<String, Object> map,

@RequestParam(required = false) Byte status, @PathVariable("index") Integer index*/\** 页数 *\*/*, @PathVariable("count") Integer count*/\** 行数*\*/*) {

logger.info("检查用户是否登录"); Object userId = checkUser(session); User user;

if (userId != null) {

logger.info("获取用户信息");

user = userService.get(Integer.*parseInt*(userId.toString())); map.put("user", user);

} else {

return "redirect:/login";

}

Byte[] status\_array = null; if (status != null) {

status\_array = new Byte[]{status};

}

PageUtil pageUtil = new PageUtil(index, count); logger.info("根据用户ID:{}获取已购订单列表", userId);

List<ProductOrder> productOrderList = productOrderService.getList(new ProductOrder().setProductOrder\_user(new User().setUser\_id(Integer.*valueOf*(userId.toString()))), status\_array, new OrderUtil("productOrder\_id", true), pageUtil);

*//*统计已购商品订单总金额

double totalprice=0;

*//*订单总数量

Integer orderCount = 0;

if (productOrderList.size() > 0) {

orderCount = productOrderService.getTotal(new ProductOrder().setProductOrder\_user(new User().setUser\_id(Integer.*valueOf*(userId.toString()))), status\_array);

logger.info("获取订单项信息及对应的产品信息"); for (ProductOrder order : productOrderList) {

List<ProductOrderItem> productOrderItemList = productOrderItemService.getListByOrderId(order.getProductOrder\_id(), null);

if (productOrderItemList != null) {

for (ProductOrderItem productOrderItem : productOrderItemList) {

*//*统计已购商品订单总金额

Integer product\_id = productOrderItem.getProductOrderItem\_product().getProduct\_id();

totalprice =totalprice+productOrderItem.getProductOrderItem\_price(); Product product = productService.get(product\_id);

product.setSingleProductImageList(productImageService.getList(product\_id, 0, new PageUtil(0, 1)));

productOrderItem.setProductOrderItem\_product(product); if (order.getProductOrder\_status() == 3) {

productOrderItem.setIsReview(reviewService.getTotalByOrderItemId(productOrderItem.getProd uctOrderItem\_id()) > 0);

}

}

}

order.setProductOrderItemList(productOrderItemList);

}

}

pageUtil.setTotal(orderCount);

logger.info("获取商品分类列表信息");

List<Category> categoryList = categoryService.getList(null, new PageUtil(0, 5));

logger.info("统计已购商品订单总金额"+totalprice); map.put("totalBuyedPrice",totalprice);

map.put("pageUtil", pageUtil); map.put("productOrderList", productOrderList); map.put("categoryList", categoryList); map.put("status", status);

logger.info("转到前台-订单列表页"); return "fore/orderListPage";

}