

**<二手书交易平台>**

**<系统概要说明书>**

设计人员： 2022116481-张彦博

2022110173-贺鹏霖

2022112893-罗溢铭

2022112074-王诗军

指导教师 罗皓楠

|  |  |  |
| --- | --- | --- |
| **文件状态：**  **[√ ] 草稿**  **[ ] 正式发布**  **[ ] 正在修改** | **文件标识** | **1.0** |
| **当前版本** | **V1.0** |
| **拟 稿 人** | 张彦博、贺鹏霖、王诗军、罗溢铭 |
| **拟稿日期** | 4.15 |
| **审 核 人** | 张彦博 |
| **审核日期** | 4.29 |

编写说明

标题：二手书交易平台软件系统概要说明书

类别：文档

编辑软件：Miscrosoft Word 2000 中文版

版本历史：

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **作者** | **日期** | **备注** |
| V1.0.1 | 王诗军 | 4.10 | 系统概要说明文档框架完成 |
| V2.0.1 | 张彦博 | 4.18 | 体系结构设计完成 |
| V2.0.1 | 贺鹏霖 | 4.15 | 技术选型方案确认 |
| V2.0.2 | 罗溢铭  王诗军 | 4.23 | 数据库ER图定稿  API接口文档完成 |
| V3.0.1 | 张彦博 | 4.28 | 系统概要说明文档完成 |

完成情况分工：

|  |  |  |  |
| --- | --- | --- | --- |
| **学号** | **姓名** | **工作量** | **完成工作内容** |
| 2022116481 | 张彦博 | 25% | 体系结构设计 |
| 2022112893 | 罗溢铭 | 25% | 系统数据库设计 |
| 2022112074 | 王诗军 | 25% | 接口设计 |
| 2022110173 | 贺鹏霖 | 25% | 引言、系统出错处理设计 |
|  |  |  |  |

**目录**

[1、引言 5](#_Toc32611)

[1.1 编写目的 5](#_Toc31180)

[1.2 项目背景 5](#_Toc31925)

[1.3相关术语 6](#_Toc31739)

[1.4 参考资料 6](#_Toc2774)

[2、 系统体系结构设计 7](#_Toc2608)

[2.1 系统特点分析 7](#_Toc21132)

[2.2 系统体系结构设计 7](#_Toc1964)

[2.2.1 系统体系结构模式 7](#_Toc22883)

[2.2.2 系统体系结构设计 8](#_Toc32236)

[3、接口设计 13](#_Toc15483)

[3.1 外部接口 13](#_Toc20669)

[3.1.1用户端接口 13](#_Toc26324)

[3.2 内部接口 36](#_Toc9193)

[4、系统数据库设计 37](#_Toc18129)

[4.1 概念数据库设计 37](#_Toc1810)

[4.1.1实体型及属性 37](#_Toc25873)

[4.1.2联系型及属性 38](#_Toc25028)

[4.1.3关系转换 38](#_Toc3786)

[4.1.4 ER图 39](#_Toc10440)

[4.2 逻辑数据库设计 40](#_Toc6222)

[4.2.1 逻辑模型 40](#_Toc28463)

[4.2.2数据库sql语句 40](#_Toc8289)

[5、系统出错处理设计 45](#_Toc14525)

[5.1 出错信息 45](#_Toc17283)

[5.2 补救措施 47](#_Toc24724)

# 1、引言

## 1.1 编写目的

本《系统概要设计说明书》依据需求分析阶段的《需求规格说明书》编写，旨在明确校园二手书店网上管理系统的整体架构设计、模块划分和系统接口规范，为后续详细设计与开发工作提供技术指导依据

预期读者包括：项目开发团队成员、软件工程课程指导教师、系统验收评审老师以及未来系统维护人员。通过本文档，相关方能够清晰了解系统的技术实现方案与设计约束条件。

## 1.2 项目背景

在高校校园中，学生每年因课程变动或毕业会产生大量闲置教材和书籍。传统二手书交易方式（如线下摆摊、微信群交易）普遍存在信息不对称、交易成本高、资源浪费和信任风险等问题[1]。例如，买卖双方难以高效匹配需求，线下交易耗费时间精力，且缺乏统一的担保机制，导致交易纠纷频发。与此同时，大量书籍被废弃或低价处理，与可持续发展的理念相悖。针对这一现状，开发一个专注于校园场景的二手书在线交易平台具有迫切需求。

本项目的目标是构建一个校内可信的二手书交易系统，通过提供便捷的书籍发布、搜索和交易功能，降低交易成本，并依托校园身份认证系统实现用户实名制，结合信用评价机制提升交易安全性[2]。平台还将引入数据分析功能，优化用户体验，同时促进资源循环利用，响应绿色校园倡议。该平台不仅能为学生节省购书成本，还能推动校园数字化建设，减少资源浪费，并与社会层面的环保理念形成呼应。

在技术层面，系统采用成熟的Spring Boot后端框架与Vue.js前端技术，配合MySQL数据库实现轻量级架构设计，支持高并发访问与快速迭代。项目与学校政策紧密结合，例如教育部《绿色校园建设指南》中关于资源循环利用的要求，以及校内创新创业扶持政策，为其落地提供了可行性保障[3]。

## 1.3相关术语

|  |  |
| --- | --- |
| 术语 | 中文描述 |
| 概念模型 | 指对系统涉及的核心业务对象及其关系的抽象描述，不包含技术细节，仅聚焦业务本质。 |
| 逻辑模型 | 在概念模型基础上，详细定义业务对象的结构、属性及规则，但仍独立于具体技术实现。 |
| ER图 | 一种图形化工具，通过实体（矩形）、属性（椭圆）和关系连线（菱形）直观展示业务对象间的关联。 |
| 熔断恢复 | 就像自动开关：发现问题立刻断电保护，定期检测正常后自动恢复，防止系统崩溃。 |
| 幂等性 | 指对同一个操作执行一次或多次，最终得到的结果都相同 |

表 1-1

## 1.4 参考资料

1. 多抓鱼官方白皮书: 二手书循环经济模式分析[R]. 北京: 多抓鱼科技有限公司, 2022.
2. 李明, 王涛. 基于Spring Boot的校园二手交易平台设计与实现[J]. 计算机应用研究, 2021, 38(5): 132-137.
3. 教育部.教育部关于推进绿色校园建设的指导意见[EB/OL]. (2019-08-12)[2023-10-01]. [http://www.moe.gov.cn](http://www.moe.gov.cn/" \t "https://chat.deepseek.com/a/chat/s/_blank).

# 系统体系结构设计

## 2.1 系统特点分析

本系统为典型的基于事物交互的Web平台（B/S架构），聚焦学生高频、低成本、高信任度的二手书交易需求，通过用户、商品、交易的多维数据联动，实现二手书资源的优化配置。核心特点如下：

1. 精准校园场景适配：

​ 课程教材智能匹配：对接学校选课系统API，自动推荐相关课程教材（如《高等数学-李老师2023版》）。

​ 支持校园内二手书商：与校园内书商合作，提高课本回收速度。

2. 极简交易流程：

​ 扫码卖书：通过ISBN扫码自动填充书籍信息（含课程/版本备注），30秒完成上架。

搜索匹配：通过课程代码，匹配课程教材

1. 提升管理透明度

对书籍状态（新旧程度、价格波动、交易历史）、用户信用评级等数据进行统一管理，规范平台交易秩序。

## 2.2 系统体系结构设计

### 2.2.1 系统体系结构模式

本系统采用三层架构分层设计与分布式架构，基于B/S（Browser/Server）模式实现，将系统分为浏览器端以及服务器端，系统由以下三个核心层级构成：

| ​层级 | ​主要功能 | ​运行方式 |
| --- | --- | --- |
| ​前端 | 用户交互和数据显示 | ​基于浏览器运行 |
| ​后端 | 业务逻辑处理和数据存储 | 基于服务器部署运行 |
| ​数据库 | 数据的存储以及管理 | ​基于数据存储运行 |

表 2-2

### 2.2.2 系统体系结构设计

本系统采用三层架构设计模式，将不同功能模块分配到不同的层次上，保证系统的可扩展性和可维护性。通过将业务逻辑、数据管理和用户界面分离，使得修改其中任意部分不会对其他部分造成影响。

同时，系统的体系结构严格遵循软件设计基本原则，包括：高内聚低耦合：各模块功能独立，减少相互依赖，提高代码复用性。单一职责原则：每个模块仅负责一项核心功能，确保逻辑清晰、易于维护。这种分层设计使系统架构更加合理、清晰，便于后续功能扩展和代码维护。

#### （1）逻辑视图设计

本系统采用类图作为逻辑视图的核心表达方式，通过面向对象的方式呈现系统的主要实体及其交互关系。清晰定义了系统的核心业务实体及其交互逻辑，覆盖了从书籍发布、交易下单、物流跟踪到用户通信的全流程，同时通过状态枚举和权限控制保障系统的规范性与安全性。类图共包含七个核心类，分别对应系统的主要业务模块，各类间通过属性关联形成完整的业务逻辑链。

以下是我们所设计的核心类及关键属性/方法表格及类图：​

| 类名 | 关键属性/方法 | 核心关联关系 |
| --- | --- | --- |
| ​User​ | userID, username, status, updateProfile(), login() | 关联Order、Message、Notification |
| ​Book​ | bookID, title, status, publish() | 关联Order（卖家） |
| ​Order​ | orderID, status, paymentMethod, cancel(), requestRefund() | 关联User、Book、Shipping |
| ​Shipping​ | shippingID, address, status, updateStatus() | 关联Order（物流跟踪） |
| ​Message​ | messageID, content, send() | 关联User（收发双方） |
| ​Notification​ | notificationID, type, send(), isUrgent() | 关联User（接收方） |
| ​Admin​ | adminID, rolePermission | 关联User（管理权限） |

表 2-3

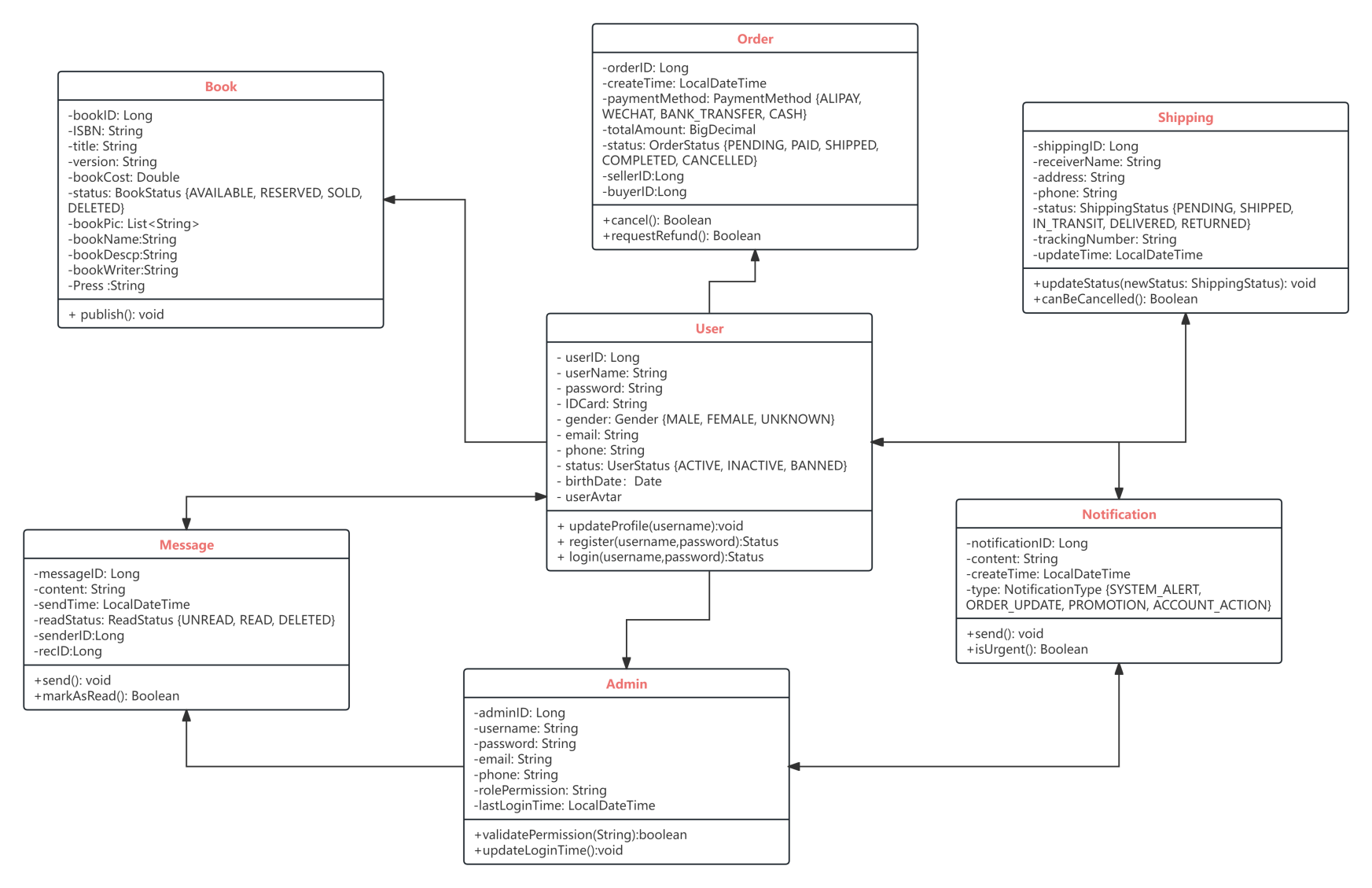


图 2-1 类图

#### 开发及运行视图设计

我们的二手书交易系统采用了三层架构，并且结合MVC架构，将系统自上而下划分为VIEW层、Controller层、Service层、Dao层和数据层。各层职责明确，通过标准化接口实现解耦，确保系统的可维护性和可扩展性。为了便于说明，我们这里通过表格以及系统架构图的形式进行展示

| 层级 | 核心功能 | 对应模块（二手书交易系统实现） |
| --- | --- | --- |
| ​VIEW层​ | 用户交互界面，接收输入并展示结果 | - 登录界面 - 首页推荐（热门书籍展示） - 消息通知（交易提醒） - 书籍搜索（关键词/分类筛选） - 个人管理（用户资料修改） - 书籍贩卖（发布二手书） - 订单查询（交易记录） - 客服售后（纠纷处理） |
| ​Controller层​ | 请求路由与参数校验，调用Service层处理业务逻辑 | 接收前端请求，校验参数后转发至Service模块 |
| ​Service层​ | 核心业务逻辑实现，组合Dao层操作完成复杂功能 | - 用户认证服务（登录/注册） - 书籍推荐与搜索服务（个性化推荐） - 数据获取服务（书籍详情） - 消息服务（买卖双方通信） - 交易服务（订单创建与支付） - 数据统计服务（交易量分析） |
| ​Dao层​ | 数据持久化操作，封装数据库访问细节 | 提供对MongoDB（非结构化数据）、MySQL（结构化数据）、Redis（缓存） |
| ​数据层​ | 数据存储与缓存，支持高并发访问 | MongoDB：存储书籍图片、用户聊天记录等非结构化数据 MySQL：存储用户信息、订单数据等核心业务数据 Redis：缓存热门书籍、会话状态等高频访问数据 |

表 2-4

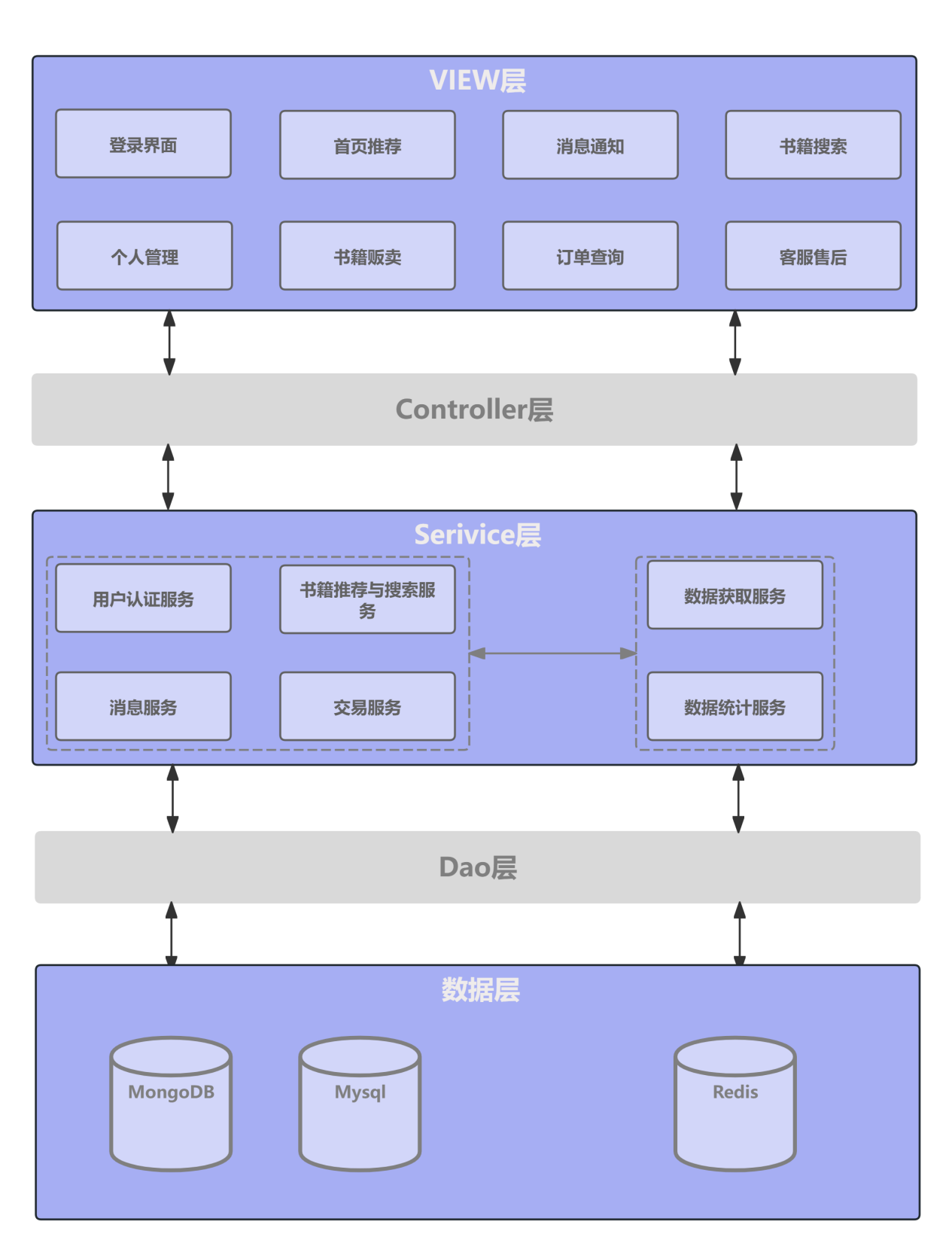


图 2-2 系统架构图

#### （3）部署视图设计

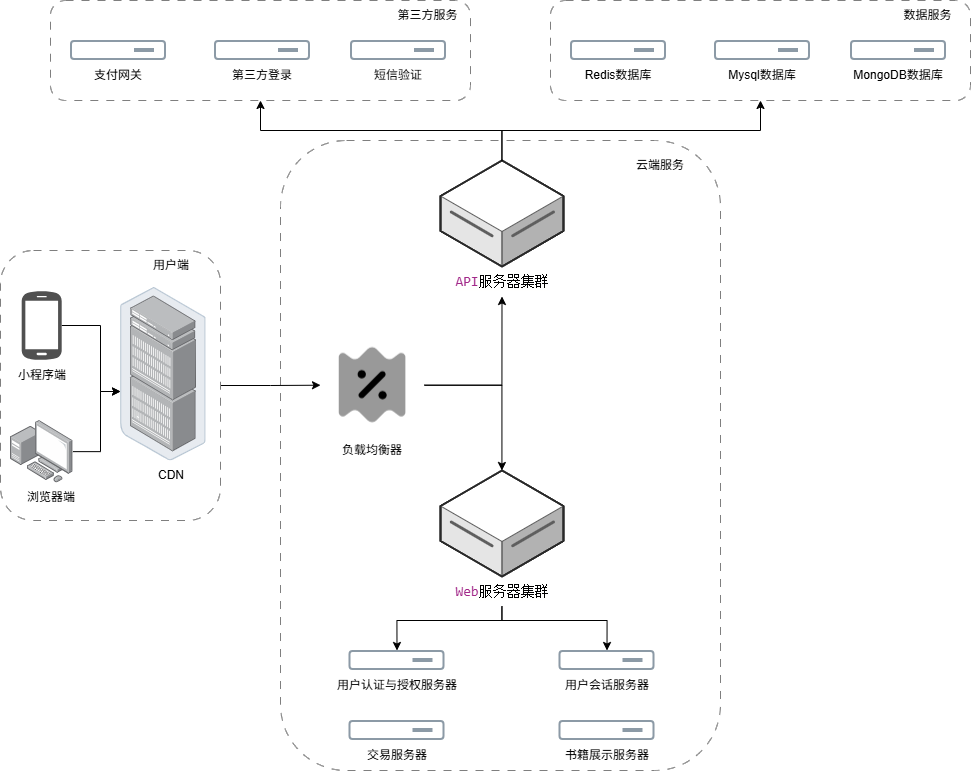
系统部署主要分为用户端以及云端两个部分。用户可以通过手机小程序端口或者电脑浏览器端口进行访问，用户请求首先会经过CDN加速，然后通过负载均衡器合理分配流量至云端。云端服务器主要分为两个部分：Web服务器集群以及API服务器集群。Web服务器集群主要负责网页内容展示以及拥有专门的功能服务器如：用户认证与授权服务器：管登录注册、用户会话服务器：记录用户登录状态等。而API服务器集群则主要是处理第三方服务支持以及数据库存储部分。

图 2-3 部署图

# 3、接口设计

## 3.1 外部接口

### 3.1.1用户端接口

|  |  |
| --- | --- |
| **API-01 用户登录API** | |
| **说明** | |
| **交互机制** | 普通用户登录时，填写相关表单，点击登录按钮调用本接口 |
| **接口技术** | Restfulapi，采用JAVA开发 |
| **访问地址** | <http://localhost:8088/user/login> |

表 3-1

续表 3-1

|  |  |
| --- | --- |
| **请求方式** | **post** |

|  |  |  |  |
| --- | --- | --- | --- |
| **请求参数** | **字段** | **类型** | **说明** |
| userName | String | 用户名/手机号 |
| password | String | 大于8位用户密码 |
| **返回数据** | 字段 | 类型 | 说明 |
| code | Integer | HTTP效应状态码 |
| message | Sting | 后台响应消息 |
| data | Json | 后台响应数据 |
| userId | Long | 用户的ID，自增 |
| token | String | 后台返回用户令牌 |
| **返回数据实例** | {  "code": 200,  "message": "Login successful",  "data": {  "userId": "1",  "token": "sjK2r9R5sGhvMwF2tPkbWzIAGvfS4ZpWaT\_vgCjF\_xc"  }  } | | |

|  |  |
| --- | --- |
| **API-02 用户注册API** | |
| **说明** | |
| **交互机制** | 同步请求，用户注册时，填写相关表单，点击登录按钮调用本接口 |
| **接口技术** | Restfulapi，采用JAVA开发 |
| **访问地址** | http://localhost:8088/user/register |
| **请求方式** | post |

|  |  |  |  |
| --- | --- | --- | --- |
| **请求参数** | **字段** | **类型** | **说明** |
| userName | String | 用户名 |
| password | String | 大于8位用户密码 |
| name | String | 姓名 |
| phone | String | 手机号 |
| email | String | 邮箱 |
| idNumber | String | 身份证号码 |
| gender | Integer | 0:男；1：女 |
| **返回数据** | 字段 | 类型 | 说明 |
| code | Integer | HTTP效应状态码 |
| message | Sting | 消息 |
| data | Json | 后台响应数据 |

表 3-2

续表 3-2

|  |  |  |  |
| --- | --- | --- | --- |
|  | **userId** | **Long** | **用户的ID，自增** |
| **返回数据实例** | {  "code": 200,  "message": "Sign successful",  "data": {  "userId": "1",    }  } | | |

|  |  |
| --- | --- |
| **API-03 注销API** | |
| **说明** | |
| **交互机制** | 同步请求，普通用户点击注销按钮调用本接口 |
| **接口技术** | Restfulapi，采用JAVA开发 |
| **访问地址** | <http://localhost:8088/user/logout> |
| **请求方式** | Post |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **请求参数** | | **字段** | **类型** | **说明** |
| userId | String | 用户ID |
| **返回数据** | | 字段 | 类型 | 说明 |
| code | Integer | HTTP效应状态码 |
| message | String | 消息 |
| data | JSON | 后台响应数据 |
| **返回数据实例** | {  "code": 200,  "message": "Login successful",  "data": null  } | | | |

表 3-3

|  |  |
| --- | --- |
| **API-04 图片上传API** | |
| **说明** | |
| **交互机制** | 用户点击图片上传，选择图片后，调用该接口图片传到服务器，服务器返回图片url回显 |
| **接口技术** | Restfulapi，采用JAVA开发 |
| **访问地址** | <http://localhost:8088/user/upload> |
| **请求方式** | post |

|  |  |  |  |
| --- | --- | --- | --- |
| **请求参数** | **字段** | **类型** | **说明** |
| imageFile | File | 图片文件 |

表 3-4

续表 3-4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **返回数据** | | **字段** | **类型** | **说明** |
| code | Integer | HTTP效应状态码 |
| message | String | 消息 |
| data | Json | 后台响应消息 |
| imageURL | String | 图片链接 |
| **返回数据实例** | {  "code": 200,  "message": "Image uploaded successfully",  "data": {  "imageUrl": "http://localhost:8088/images/2025/04/18/abc123.jpg",  }  } | | | |

|  |  |
| --- | --- |
| **API-05 发布书籍API** | |
| **说明** | |
| **交互机制** | 用户填写表单后，点击发布按钮调用该接口 |
| **接口技术** | Restfulapi，采用JAVA开发 |
| **访问地址** | <http://localhost:8088/user/postbook> |
| **请求方式** | Post |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **请求参数** | | **字段** | **类型** | | **说明** |
| userId | Long | | 用户的ID，自增 |
| title | String | | 书名 |
| author | String | | 作者 |
| publisher | String | | 出版社 |
| isbn | String | | isbn号 |
| description | String | | 书籍描述 |
| category | String | | 书籍分类 |
| price | Intger | | 价格 |
| imageFile | List<String> | | 图片URL数组 |
| **返回数据** | | 字段 | 类型 | 说明 | |
| code | Integer | HTTP效应状态码 | |
| message | String | 响应消息 | |
| data | Json | 后台响应消息 | |
| **返回数据实例** | {  "code": 200,  "message": " successful",  "data": null  } | | | | |

表 3-5

|  |  |
| --- | --- |
| **API-06 主页分页查询书籍API** | |
| **说明** | |
| **交互机制** | 用户主页选择分类后显示一页书籍缩略信息 |
| **接口技术** | Restfulapi，采用JAVA开发 |
| **访问地址** | <http://localhost:8088/user/home> |
| **请求方式** | get |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **请求参数** | | **字段** | **类型** | | **说明** |
| category | String | | 书籍种类 |
| sortOrder | String | | 排序方式(newest,hotest) |
| pageNum | Integer | | 当前页码(默认为10条/页) |
| **返回数据** | | 字段 | 类型 | 说明 | |
| code | Integer | HTTP效应状态码 | |
| message | String | 消息 | |
| data | Json | 后台响应消息(多本书籍，用书籍数组存) | |
| title | String | 书籍名称 | |
| imageFile | String | 书籍图片 | |
| bookID | String | 书籍ID | |
| userName | String | 发布者名称 | |
| userAvtar | String | 发布者头像 | |
| **返回数据实例** | {  "code": 200,  "message": "ok",  "data": [  {  "title": "算法导论",  "imageFile": "http://example.com/book1.jpg",  "bookID": "book-001",  "userName": "张三",  "userAvtar": "http://example.com/avatar1.png"  },  {  "title": "Java编程思想",  "imageFile": "http://example.com/book2.jpg",  "bookID": "book-002",  "userName": "李四",  "userAvtar": "http://example.com/avatar2.png"  } ]} | | | | |

表 3-6

|  |  |
| --- | --- |
| **API-07 搜索书籍浏览API** | |
| **说明** | |
| **交互机制** | 用户搜索栏输入内容后点击搜索按钮显示相关书籍信息 |
| **接口技术** | Restfulapi，采用JAVA开发 |
| **访问地址** | <http://localhost:8088/user/serach> |
| **请求方式** | get |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **请求参数** | **字段** | **类型** | | **说明** |
| serach | String | | 搜索内容 |
| sortOrder | String | | 排序方式(newest,hottest,priceDesc,priceAsc) |
| pageNum | Integer | | 当前页码(默认为10条/页) |
| priceMin | Double | | 价格区间较小值 |
| priceMax | Double | | 价格区间较大值 |
| **返回数据** | 字段 | 类型 | 说明 | |
| code | Integer | HTTP效应状态码 | |
| message | String | 消息 | |
| data | Json | 后台响应消息 | |
| title | String | 书籍名称(多本书籍，用书籍数组存) | |
| imageFile | String | 书籍一张图片 | |
| bookID | String | 书籍ID | |
| price | Double | 书籍价格 | |
| description | String | 书籍描述 | |
| userName | String | 发布者名称 | |
| userAvtar | String | 发布者头像 | |

表 3-7

续表 3-7

|  |  |
| --- | --- |
| **返回数据实例** | **{**  **"code": 200,**  **"message": "ok",**  **"data": [**  **{**  **"title": "算法导论",**  **"imageFile": "http://example.com/book1.jpg",**  **"bookID": "book-001",**  **"userName": "张三",**  **"userAvtar": "http://example.com/avatar1.png"**  **},**  **{**  **"title": "Java编程思想",**  **"imageFile": "http://example.com/book2.jpg",**  **"bookID": "book-002",**  **"userName": "李四",**  **"userAvtar": "http://example.com/avatar2.png"**  **}**  **]**  **}** |

|  |  |
| --- | --- |
| **API-08 书籍详情页API** | |
| **说明** | |
| **交互机制** | 用户点击书籍缩略图进入书籍详情页调用 |
| **接口技术** | Restfulapi，采用JAVA开发 |
| **访问地址** | <http://localhost:8088/user/book?bookID=> |
| **请求方式** | get |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **请求参数** | **字段** | **类型** | | **说明** |
| bookID | String | | 书籍ID |
| **返回数据** | 字段 | 类型 | 说明 | |
| code | Integer | HTTP效应状态码 | |
| message | String | 消息 | |
| data | Json | 后台响应消息 | |
| title | String | 书籍名称 | |
| imageFile | List<String> | 书籍图片 | |
| bookID | String | 书籍ID | |
| price | Double | 书籍价格 | |
| description | String | 书籍描述 | |
| userName | String | 发布者名称 | |
| userAvtar | String | 发布者头像 | |

表 3-8

续表 3-8

|  |  |
| --- | --- |
| **返回数据实例** | **{**  **"code": 200,**  **"message": "ok",**  **"data": {**  **"title": "算法导论",**  **"imageFile": [**  **"http://example.com/images/book123-cover.jpg",**  **"http://example.com/images/book123-back.jpg"**  **],**  **"bookID": "book-uuid-123456",**  **"price": 99.99,**  **"description": "经典的计算机算法教材，涵盖排序、图算法、动态规划等核心内容。",**  **"userName": "张三",**  **"userAvtar": "http://example.com/avatar/user123.png"**  **}**  **}** |

|  |  |
| --- | --- |
| **API-09 书籍添加到购物车API** | |
| **说明** | |
| **交互机制** | 用户在书籍详情页点击添加购物车调用此API，保存到购物车表 |
| **接口技术** | Restfulapi，采用JAVA开发 |
| **访问地址** | <http://localhost:8088/user/cartAdd> |
| **请求方式** | post |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **请求参数** | | **字段** | **类型** | | **说明** |
| userId | String | | 用户ID |
| bookID | String | | 书籍ID |
| **返回数据** | | 字段 | 类型 | 说明 | |
| code | Integer | HTTP效应状态码 | |
| message | String | 消息 | |
| data | Json | 后台响应消息 | |
| **返回数据实例** | {  "code": 200,  "message": "ok",  "data": null  } | | | | |

表 3-9

|  |  |
| --- | --- |
| **API-10 查看个人信息API** | |
| **说明** | |
| **交互机制** | 用户查看自己的个人信息时调用此API |
| **接口技术** | Restfulapi，采用JAVA开发 |
| **访问地址** | <http://localhost:8088/user/profile> |
| **请求方式** | get |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **请求参数** | | **字段** | **类型** | | **说明** |
| userId | String | | 用户ID |
| **返回数据** | | 字段 | 类型 | 说明 | |
| code | Integer | HTTP效应状态码 | |
| message | String | 消息 | |
| data | Json | 后台响应消息 | |
| userName | String | 用户名 | |
| phone | String | 电话 | |
| email | String | 邮箱 | |
| idNumber | String | 身份证号码 | |
| userAvtar | String | 头像 | |
| **返回数据实例** | {  "code": 200,  "message": "获取用户信息成功",  "data": {  "userName": "王小明",  "phone": "13800138000",  "email": "user@example.com",  "idNumber": "110101199001011234",  "userAvtar": "https://example.com/avatar.jpg"  }  } | | | | |

表 3-10

|  |  |
| --- | --- |
| **API-11 修改个人信息API** | |
| **说明** | |
| **交互机制** | 用户修改自己的个人信息时调用此API |
| **接口技术** | Restfulapi，采用JAVA开发 |
| **访问地址** | <http://localhost:8088/user/profile> |
| **请求方式** | post |

|  |  |  |  |
| --- | --- | --- | --- |
| **请求参数** | **字段** | **类型** | **说明** |
| userId | String | 用户ID |
| userName | String | 用户名 |
| phone | String | 电话 |
| email | String | 邮箱 |
| idNumber | String | 身份证号码 |

表 3-11

续表 3-11

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | **userAvtar** | **String** | | **头像** |
| **返回数据** | | 字段 | 类型 | 说明 | |
| code | Integer | HTTP效应状态码 | |
| message | String | 消息 | |
| data | Json | 后台响应消息 | |
| **返回数据实例** | {  "code": 200,  "message": "修改成功",  "data":null  } | | | | |

|  |  |
| --- | --- |
| **API-12 修改账号密码API** | |
| **说明** | |
| **交互机制** | 用户修改自己的密码时调用此API |
| **接口技术** | Restfulapi，采用JAVA开发 |
| **访问地址** | <http://localhost:8088/user/changePassword> |
| **请求方式** | get |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **请求参数** | | **字段** | **类型** | | **说明** |
| userName | String | | 用户名/手机号/邮箱 |
| oldPassword | String | | 原密码 |
| newPassword | String | | 新密码 |
| **返回数据** | | 字段 | 类型 | 说明 | |
| code | Integer | HTTP效应状态码 | |
| message | String | 消息 | |
| data | Json | 后台响应消息 | |
| **返回数据实例** | {  "code": 200,  "message": "修改成功",  "data":null  } | | | | |

表 3-12

|  |  |
| --- | --- |
| **API-13 查看购物车API** | |
| **说明** | |
| **交互机制** | 用户查看自己的购物车时调用此API |
| **接口技术** | Restfulapi，采用JAVA开发 |
| **访问地址** | [http://localhost:8088/user/cart?userId=](http://localhost:8088/user/cart?userID=) |
| **请求方式** | get |

|  |  |  |  |
| --- | --- | --- | --- |
| **请求参数** | **字段** | **类型** | **说明** |

表 3-13

续表 3-13

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | **userId** | **String** | | **用户ID** |
| **返回数据** | | 字段 | 类型 | 说明 | |
| code | Integer | HTTP效应状态码 | |
| message | String | 消息 | |
| data | Json | 后台响应消息 | |
| title | String | 书籍名称(多本书籍，用书籍数组存) | |
| imageFile | String | 书籍一张图片 | |
| bookID | String | 书籍ID | |
| price | Double | 书籍价格 | |
| description | String | 书籍描述 | |
| userName | String | 发布者名称 | |
| userAvtar | String | 发布者头像 | |
| **返回数据实例** | {  "code": 200,  "message": "ok",  "data": [  {  "title": "算法导论",  "imageFile": "http://example.com/book1.jpg",  "bookID": "book-001",  "userName": "张三",  "userAvtar": "http://example.com/avatar1.png"  },  {  "title": "Java编程思想",  "imageFile": "http://example.com/book2.jpg",  "bookID": "book-002",  "userName": "李四",  "userAvtar": "http://example.com/avatar2.png"  }  ]  } | | | | |

|  |  |
| --- | --- |
| **API-15 书籍购买创建订单API** | |
| **说明** | |
| **交互机制** | 用户在书籍详情页点击购买此API，进入创建订单界面，显示相关信息 |
| **接口技术** | Restfulapi，采用JAVA开发 |
| **访问地址** | <http://localhost:8088/user/orderCreate> |

表 3-15

续表 3-15

|  |  |
| --- | --- |
| **请求方式** | **Get** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **请求参数** | | **字段** | **类型** | | **说明** |
| bookID | String | | 书籍ID |
| userId | String | | 用户ID |
| **返回数据** | | 字段 | 类型 | 说明 | |
| code | Integer | HTTP效应状态码 | |
| message | String | 消息 | |
| data | Json | 后台响应消息 | |
| title | String | 书籍名称 | |
| imageFile | List<String> | 书籍一张图片 | |
| price | Double | 书籍价格 | |
| **返回数据实例** | {  "code": 200,  "message": "ok",  "data": {  "title": "算法导论",  "imageFile": [  "http://example.com/images/book123-cover.jpg"  ],  "price": 99.99  }  } | | | | |

|  |  |
| --- | --- |
| **API-16 书籍确认购买提交订单API** | |
| **说明** | |
| **交互机制** | 用户在确认订单页点击确认购买调用此API |
| **接口技术** | Restfulapi，采用JAVA开发 |
| **访问地址** | <http://localhost:8088/user/buy> |
| **请求方式** | post |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **请求参数** | **字段** | **类型** | | **说明** |
| bookID | String | | 书籍ID |
| userId | String | | 用户ID |
| recipientAddress | String | | 收货地址 |
| recipientName | String | | 收货人姓名 |
| recipientPhone | String | | 收货人电话 |
| **返回数据** | 字段 | 类型 | 说明 | |
| code | Integer | HTTP效应状态码 | |
| message | String | 消息 | |
| data | Json | 后台响应消息 | |

表 3-16

续表 3-16

|  |  |
| --- | --- |
| **返回数据实例** | **{**  **"code": 200,**  **"message": "ok",**  **"data": null**  **}** |

|  |  |
| --- | --- |
| **API-17 查看我的订单API** | |
| **说明** | |
| **交互机制** | 用户在主页面点击我的订单调用此API |
| **接口技术** | Restfulapi，采用JAVA开发 |
| **访问地址** | <http://localhost:8088/user/myOrder> |
| **请求方式** | get |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **请求参数** | **字段** | **类型** | | **说明** |
| userId | String | | 用户ID |
| sale | Boolean | | 是否卖家(true :我是卖家) |
| time1 | Date | | 时间较小值 |
| time2 | Date | | 时间较大值 |
| **返回数据** | 字段 | 类型 | 说明 | |
| code | Integer | HTTP效应状态码 | |
| message | String | 消息 | |
| data | Json | 后台响应消息 | |
| orderID | String | 订单ID(数组，多条订单记录) | |
| status | String | 订单状态(notSend,notReceive,complete,refund) | |
| bookID | String | 书籍ID | |
| userId | String | 另一个用户ID | |
| userName | String | 用户名称 | |
| title | String | 书籍名称 | |
| imageFile | List<String> | 书籍图片（一张） | |
| userAvatar | String | 用户头像 | |
| logisticsId | String | 物流编号 | |

表 3-17

续表 3-17

|  |  |
| --- | --- |
| **返回数据实例** | **{**  **"code": 200,**  **"message": "请求成功",**  **"data": [**  **{**  **"orderID": "ORD20231101",**  **"status": "notSend",**  **"bookID": "BK20231501",**  **"userId": "USER2001",**  **"userName": "张三",**  **"title": "深入理解计算机系统",**  **"imageFile": [**  **"https://example.com/book\_cover1.jpg"**  **],**  **"userAvatar": "https://example.com/user\_avatar1.jpg",**  **"logisticsId": "SF1122334455"**  **},**  **{**  **"orderID": "ORD20231102",**  **"status": "complete",**  **"bookID": "BK20231502",**  **"userId": "USER2002",**  **"userName": "李四",**  **"title": "JavaScript高级程序设计",**  **"imageFile": [**  **"https://example.com/book\_cover2.jpg"**  **],**  **"userAvatar": "https://example.com/user\_avatar2.jpg",**  **"logisticsId": "YT5566778899"**  **}**  **]**  **}** |

|  |  |
| --- | --- |
| **API-18 对订单进行发货API** | |
| **说明** | |
| **交互机制** | 用户在订单页面对订单进行发货填写相关表单调用此API |
| **接口技术** | Restfulapi，采用JAVA开发 |
| **访问地址** | <http://localhost:8088/user/myOrder/send> |
| **请求方式** | post |

|  |  |  |  |
| --- | --- | --- | --- |
| **请求参数** | **字段** | **类型** | **说明** |

表 3-18

续表 3-18

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | **orderID** | **String** | | **用户ID** |
| logisticsId | String | | 物流单号 |
| **返回数据** | | 字段 | 类型 | 说明 | |
| code | Integer | HTTP效应状态码 | |
| message | String | 消息 | |
| data | Json | 后台响应消息 | |
| **返回数据实例** | {  "code": 200,  "message": "ok",  "data":null  } | | | | |

|  |  |
| --- | --- |
| **API-19 对订单进行确认收货API** | |
| **说明** | |
| **交互机制** | 用户在订单页面对订单进行发货填写相关表单调用此API |
| **接口技术** | Restfulapi，采用JAVA开发 |
| **访问地址** | <http://localhost:8088/user/myOrder/ok> |
| **请求方式** | post |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **请求参数** | | **字段** | **类型** | | **说明** |
| orderID | String | | 用户ID |
| logisticsId | String | | 物流单号 |
| **返回数据** | | 字段 | 类型 | 说明 | |
| code | Integer | HTTP效应状态码 | |
| message | String | 消息 | |
| data | Json | 后台响应消息 | |
| **返回数据实例** | {  "code": 200,  "message": "ok",  "data":null  } | | | | |

表 3-19

|  |  |
| --- | --- |
| **API-20 对订单进行退款操作(须提供返回的物流编号)API** | |
| **说明** | |
| **交互机制** | 用户在订单页面对订单进行发货填写相关表单调用此API |
| **接口技术** | Restfulapi，采用JAVA开发 |
| **访问地址** | <http://localhost:8088/user/myOrder/refund> |
| **请求方式** | post |

表 3-20

续表 3-20

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **请求参数** | | **字段** | **类型** | | **说明** |
| orderID | String | | 用户ID |
| logisticsId | String | | 物流单号 |
| **返回数据** | | 字段 | 类型 | 说明 | |
| code | Integer | HTTP效应状态码 | |
| message | String | 消息 | |
| data | Json | 后台响应消息 | |
| **返回数据实例** | {  "code": 200,  "message": "ok",  "data":null  } | | | | |

|  |  |
| --- | --- |
| **API-21 用户聊天界面API** | |
| **说明** | |
| **交互机制** | 用户在聊天页面发送消息调用此API |
| **接口技术** | Restfulapi，采用JAVA开发 |
| **访问地址** | http://localhost:8088/user/chat |
| **请求方式** | Get |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **请求参数** | **字段** | **类型** | | **说明** |
| userId | String | | 用户ID |
| receiverUserId | String | | 接收用户ID |
| **返回数据** | 字段 | 类型 | 说明 | |
| code | Integer | HTTP效应状态码 | |
| message | String | 消息 | |
| data | Json | 后台响应消息 | |
| sendTime | DateTime | 所有消息（双方）发送时间 | |
| content | String | 发送内容 | |
| receiverUserId | String | 接收用户ID | |

表 3-21

续表 3-21

|  |  |
| --- | --- |
| **返回数据实例** | **{**  **"code": 200,**  **"message": "ok",**  **"data": {**  **[**  **{**  **"sendTime": "2023-08-15T10:25:30Z",**  **"content": "你好，请问产品使用上有什么问题吗？",**  **"receiverUserId": "user123456"**  **},**  **{**  **"sendTime": "2023-08-15T10:26:15Z",**  **"content": "我遇到了登录问题，一直提示密码错误",**  **"receiverUserId": "support001"**  **}**  **]**  **}**  **}** |

|  |  |
| --- | --- |
| **API-22 用户发送消息API** | |
| **说明** | |
| **交互机制** | 用户在聊天页面发送消息调用此API |
| **接口技术** | Restfulapi，采用JAVA开发 |
| **访问地址** | http://localhost:8088/user/chat/sned |
| **请求方式** | post |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **请求参数** | **字段** | **类型** | | **说明** |
| userId | String | | 用户ID |
| receiverUserId | String | | 接收用户ID |
|  | content | String | | 发送内容 |
| **返回数据** | 字段 | 类型 | 说明 | |
| code | Integer | HTTP效应状态码 | |
| message | String | 消息 | |
| data | Json | 后台响应消息 | |
| sendTime | DateTime | 发送时间 | |
| content | String | 发送内容 | |
| receiverUserId | String | 接收用户ID | |

表 3-22

续表 3-22

|  |  |
| --- | --- |
| **返回数据实例** | **{**  **"code": 200,**  **"message": "消息发送成功",**  **"data": {**  **"sendTime": "2023-08-16T09:15:22Z",**  **"content": "我已经解决了登录问题，谢谢帮助！",**  **"receiverUserId": "support001"**  **}**  **}** |

|  |  |
| --- | --- |
| **API-23 用户查看通知API** | |
| **说明** | |
| **交互机制** | 用户在查看通知调用此API |
| **接口技术** | Restfulapi，采用JAVA开发 |
| **访问地址** | http://localhost:8088/user/notice |
| **请求方式** | Get |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **请求参数** | | **字段** | **类型** | | **说明** |
| userId | String | | 用户ID |
| **返回数据** | | 字段 | 类型 | 说明 | |
| code | Integer | HTTP效应状态码 | |
| message | String | 消息 | |
| data | Json | 后台响应消息 | |
| sendTime | DateTime | 发送时间 | |
| content | String | 通知内容 | |
| **返回数据实例** | {  "code": 200,  "message": "ok",  "data": {  [  {  "sendTime": "2023-08-16T10:30:45Z",  "content": "您的账户已成功续费，有效期至2023-12-31"  },  {  "sendTime": "2023-08-15T14:25:18Z",  "content": "系统将于今晚23:00-24:00进行维护升级"  }  ]  }} | | | | |

表 3-23

**3.1.2管理员端接口**

|  |  |
| --- | --- |
| **API-24 列出待审核书籍API** | |
| **说明** | |
| **交互机制** | 管理员进入后台审核界面调用此API ,只查询status为0的书籍 |
| **接口技术** | Restfulapi，采用JAVA开发 |
| **访问地址** | <http://localhost:8088/admin/listBook> |
| **请求方式** | get |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **请求参数** | | **字段** | **类型** | | **说明** |
| pageNum | Integer | | 页码(固定每页大小为10) |
| **返回数据** | | 字段 | 类型 | 说明 | |
| code | Integer | HTTP效应状态码 | |
| message | String | 消息 | |
| data | Json | 后台响应消息 | |
| bookID | String | 书籍ID | |
| title | String | 书籍名称 | |
| description | String | 书记描述 | |
| imageFile | List<String> | 书籍图片 | |
| userId | String | 用户ID | |
| userName | String | 用户名称 | |
| userAvatar | String | 用户头像 | |
| **返回数据实例** | {  "code": 200,  "message": "ok",  "data": {  "bookID": "BK001",  "title": "深入理解计算机系统",  "description": "本书介绍计算机系统底层原理，涵盖操作系统、编译原理等内容。",  "imageFile": [  "https://example.com/book1.jpg",  "https://example.com/book1-back.jpg"  ],  "userId": "USR001",  "userName": "张三",  "userAvatar": "https://example.com/avatar1.jpg"  }  } | | | | |

表 3-24

|  |  |
| --- | --- |
| **API-25 通过审核书籍API** | |
| **说明** | |
| **交互机制** | 管理员在审核界面通过某个书籍调用此API ,修改书籍status=1 |
| **接口技术** | Restfulapi，采用JAVA开发 |
| **访问地址** | <http://localhost:8088/amdin/passBook> |
| **请求方式** | Post |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **请求参数** | | **字段** | **类型** | | **说明** |
| bookID | String | | 书籍ID |
| pass | Boolean | | 是否通过(true = 通过) |
| **返回数据** | | 字段 | 类型 | 说明 | |
| code | Integer | HTTP效应状态码 | |
| message | String | 消息 | |
| data | Json | 后台响应消息 | |
| **返回数据实例** | {  "code": 200,  "message": "ok",  "data": null  } | | | | |

表 3-25

|  |  |
| --- | --- |
| **API-26 列出未完成订单API** | |
| **说明** | |
| **交互机制** | 管理员进入订单管理界面调用API，列出所有订单(status=0) |
| **接口技术** | Restfulapi，采用JAVA开发 |
| **访问地址** | <http://localhost:8088/amdin/listOrder> |
| **请求方式** | get |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **请求参数** | **字段** | **类型** | | **说明** |
| pageNum | String | | 页码(固定每页大小为10) |
| **返回数据** | 字段 | 类型 | 说明 | |
| code | Integer | HTTP效应状态码 | |
| message | String | 消息 | |
| data | Json | 后台响应消息 | |
| orderID | String | 订单ID | |
| status | String | 订单状态 | |
| userId | String | 用户ID | |
| userName | String | 用户名称 | |
| bookID | String | 书籍ID | |
| title | String | 书籍名称 | |

表 3-26

续表 3-26

|  |  |
| --- | --- |
| **返回数据实例** | **{**  **"code": 200,**  **"message": "ok",**  **"data": [**  **{**  **"orderID": "ORD202403011234",**  **"status": "paid",**  **"userId": "USR1001",**  **"userName": "张三",**  **"bookID": "BK5001",**  **"title": "深入理解计算机系统"**  **},**  **{**  **"orderID": "ORD202403011235",**  **"status": "shipped",**  **"userId": "USR1002",**  **"userName": "李四",**  **"bookID": "BK5002",**  **"title": "JavaScript高级程序设计"**  **}**  **]**  **}** |

|  |  |
| --- | --- |
| **API-27 对订单同意/拒绝退款操作API** | |
| **说明** | |
| **交互机制** | 管理员在订单管理界面对订单调用API |
| **接口技术** | Restfulapi，采用JAVA开发 |
| **访问地址** | <http://localhost:8088/amdin/pass>Refund |
| **请求方式** | post |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **请求参数** | | **字段** | **类型** | | **说明** |
| orderID | String | | 订单ID |
| agree | Boolean | | 是否同意 |
| **返回数据** | | 字段 | 类型 | 说明 | |
| code | Integer | HTTP效应状态码 | |
| message | String | 消息 | |
| data | Json | 后台响应消息 | |
| **返回数据实例** | { "code": 200,  "message": "ok",  "data":null} | | | | |

表 3-27

|  |  |
| --- | --- |
| **API-28 管理员列出用户API** | |
| **说明** | |
| **交互机制** | 管理员在订单管理界面对订单调用API， |
| **接口技术** | Restfulapi，采用JAVA开发 |
| **访问地址** | <http://localhost:8088/amdin/listUser> |
| **请求方式** | get |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **请求参数** | | **字段** | **类型** | | **说明** |
| pageNum | Integer | | 页码(固定每页大小为10) |
| userName | String | | 用户名称 |
| **返回数据** | | 字段 | 类型 | 说明 | |
| code | Integer | HTTP效应状态码 | |
| message | String | 消息 | |
| data | Json | 后台响应消息 | |
| userId | String | 用户ID | |
| userName | String | 用户名称 | |
| userAvatar | String | 用户头像 | |
| userPhone | String | 用户手机号 | |
| **返回数据实例** | {  "code": 200,  "message": "ok",  "data": {  "userId": "USR10001",  "userName": "张三",  "userAvatar": "https://example.com/avatars/usr10001.jpg",  "userPhone": "13800138000"  }  } | | | | |

表 3-28

|  |  |
| --- | --- |
| **API-29 管理员对用户禁用/启用操作API** | |
| **说明** | |
| **交互机制** | 管理员在订单管理界面对订单调用API， |
| **接口技术** | Restfulapi，采用JAVA开发 |
| **访问地址** | <http://localhost:8088/amdin/banUser> |
| **请求方式** | post |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **请求参数** | **字段** | **类型** | | **说明** |
| userId | Integer | | 页码 |
| ban | Boolean | | 是禁用还是启用(true： 禁用) |
| **返回数据** | 字段 | 类型 | 说明 | |
| code | Integer | HTTP效应状态码 | |
| message | String | 消息 | |

表 3-29

续表 3-29

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | **data** | **Json** | **后台响应消息** |
| **返回数据实例** | {  "code": 200,  "message": "ok",  "data": null  } | | | |

|  |  |
| --- | --- |
| **API-30 管理员发送通知API** | |
| **说明** | |
| **交互机制** | 管理员在向用户发送通知时发送调用此API |
| **接口技术** | Restfulapi，采用JAVA开发 |
| **访问地址** | http://localhost:8088/admin/notice/send |
| **请求方式** | post |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **请求参数** | | **字段** | **类型** | | **说明** |
| adimID | String | | 管理员ID |
| receiverUserId | String | | 接收用户ID |
|  | | content | String | | 发送内容 |
| **返回数据** | | 字段 | 类型 | 说明 | |
| code | Integer | HTTP效应状态码 | |
| message | String | 消息 | |
| data | Json | 后台响应消息 | |
| sendTime | DateTime | 发送时间 | |
| content | String | 发送内容 | |
| receiverUserId | String | 接收用户ID | |
| **返回数据实例** | {  "code": 200,  "message": "通知发送成功",  "data": {  "sendTime": "2023-08-15T14:30:45Z",  "content": "您的退款申请成功",  "receiverUserId": "user123456"  }  } | | | | |

表 3-30

|  |  |
| --- | --- |
| **API-31 管理员登录API** | |
| **说明** | |
| **交互机制** | 管理员登录时，填写相关表单，点击登录按钮调用本接口 |
| **接口技术** | Restfulapi，采用JAVA开发 |

表 3-31

续表 3-31

|  |  |
| --- | --- |
| **访问地址** | **<http://localhost:8088/admin/login>** |
| **请求方式** | post |

|  |  |  |  |
| --- | --- | --- | --- |
| **请求参数** | **字段** | **类型** | **说明** |
| userName | Long | 管理员用户名 |
| password | String | 大于8位用户密码 |
| **返回数据** | 字段 | 类型 | 说明 |
| code | Integer | HTTP效应状态码 |
| message | String | 后台响应消息 |
| data | Json | 后台响应数据 |
| adminID | Long | 管理员ID |
| token | String | 后台返回用户令牌 |
| **返回数据实例** | {  "code": 200,  "message": "Login successful",  "data": {  " adminID ": "540888",  "token": "sjK2r9R5sGhvMwF2tPkbWzIAGvfS4ZpWaT\_vgCjF\_xc"  }  } | | |

## 3.2 内部接口

|  |  |
| --- | --- |
| **内部接口1 管理员对用户禁用/启用自动发送消息** | |
| **说明** | |
| **交互机制** | 对用户禁用/启用自动发送消息 |
| **接口技术** | 内部服务调用（非暴露给前端的API） |
| **访问地址** | <http://localhost:8088/amdin/banUser> |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **输入** | **字段** | **类型** | | **说明** |
| userId | Integer | | 页码 |
| ban | Boolean | | 是禁用还是启用(true： 禁用) |
| **输出** | 字段 | 类型 | 说明 | |
| content | Srting | “userId 已被禁用/启用” | |

表 3-32

|  |  |
| --- | --- |
| **内部接口2 管理员对书籍审核通过自动发送消息** | |
| **说明** | |
| **交互机制** | 书籍审核通过自动发送消息 |

表 3-33

续表 3-33

|  |  |
| --- | --- |
| **接口技术** | **内部服务调用（非暴露给前端的API）** |
| **访问地址** | <http://localhost:8088/amdin/banUser> |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **输入** | **字段** | **类型** | | **说明** |
| bookID | String | | 书籍ID |
| pass | Boolean | | 是否通过(true = 通过) |
| **输出** | 字段 | 类型 | 说明 | |
| content | Srting | “你的bookID 已/未通过审核” | |

|  |  |
| --- | --- |
| **内部接口3 管理员订单同意/拒绝退款操作自动发送消息** | |
| **说明** | |
| **交互机制** | 订单同意/拒绝退款操作自动发送消息 |
| **接口技术** | 内部服务调用（非暴露给前端的API） |
| **访问地址** | <http://localhost:8088/amdin/banUser> |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **输入** | **字段** | **类型** | | **说明** |
| orderID | String | | 订单ID |
| agree | Boolean | | 是否同意 |
| **输出** | 字段 | 类型 | 说明 | |
| content | Srting | “你的orderID 已/未通过退款” | |

表 3-34

# 4、系统数据库设计

## 4.1 概念数据库设计

## 4.1.1实体型及属性

用户：用户id，用户名, 身份证号, 性别 , 上次登录时间, 密码 , 邮箱 , 电话

信息：消息id，内容, 发送时间，接收用户id，是否已读

订单：订单id，支付方式, 状态, 创建时间，卖家id，支付时间，总金额

书籍：书籍id，书名, 作者, 出版社, 状态, 描述, 版本，价格，书籍分类，ISBN号

管理员：管理员id ，管理员名, 密码, 电话, 密码，邮箱，电话，角色权限，上次登陆时间

物流信息: 物流信息id ，收件人地址, 收件人姓名, 收件人电话, 运单号，发货时间，物流公司

图片： 图片id，图片文件

系统通知： 通知编号，标题，内容

### 4.1.2联系型及属性

发送——关联实体：用户、消息

交易——关联实体：用户、订单

属于——关联实体：订单、书籍

发布——关联实体：用户、书籍 属性：发布时间

采用——关联实体：订单、物流信息

属于——关联实体：书籍、图片

反馈——关联实体：用户、用户反馈

接收——关联实体：用户、系统通知

发送——关联实体：管理员、系统通知 属性：发送时间

### 4.1.3关系转换

用户：用户id，用户名, 身份证号, 性别 , 上次登录时间, 密码 , 邮箱 , 电话

信息：消息id，内容, 发送时间，接收用户id，是否已读，用户id，其中用户id为外键

订单：订单id，支付方式, 状态, 创建时间，卖家id，支付时间，总金额，用户id，书籍id，其中用户id和书籍id为外键

书籍：书籍id，书名, 作者, 出版社, 状态, 描述, 版本，价格，书籍分类，ISBN号，用户id，发布时间，其中用户id为外键

物流信息: 物流信息id ，收件人地址, 收件人姓名, 收件人电话, 运单号，发货时间，物流公司，订单编号，其中订单编号为外键

图片： 图片id，图片文件，书籍id，其中书籍id为外键

审核：书籍id，管理员id，操作，审核时间，操作原因，其中书籍id，管理员id为外键

系统通知： 通知编号，标题，内容

接收表：用户id，通知编号，其中用户id，通知编号为外键

发送表：管理员id，通知编号，发送时间，其中管理员id和通知编号为外键

### 4.1.4 ER图

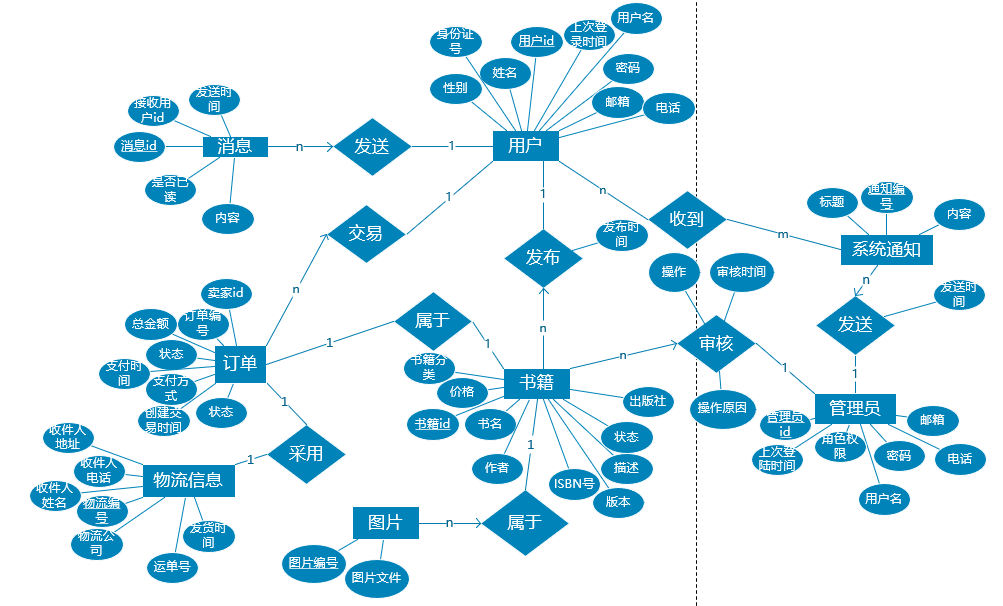


图4-1 ER图

## 4.2 逻辑数据库设计

### 4.2.1 逻辑模型

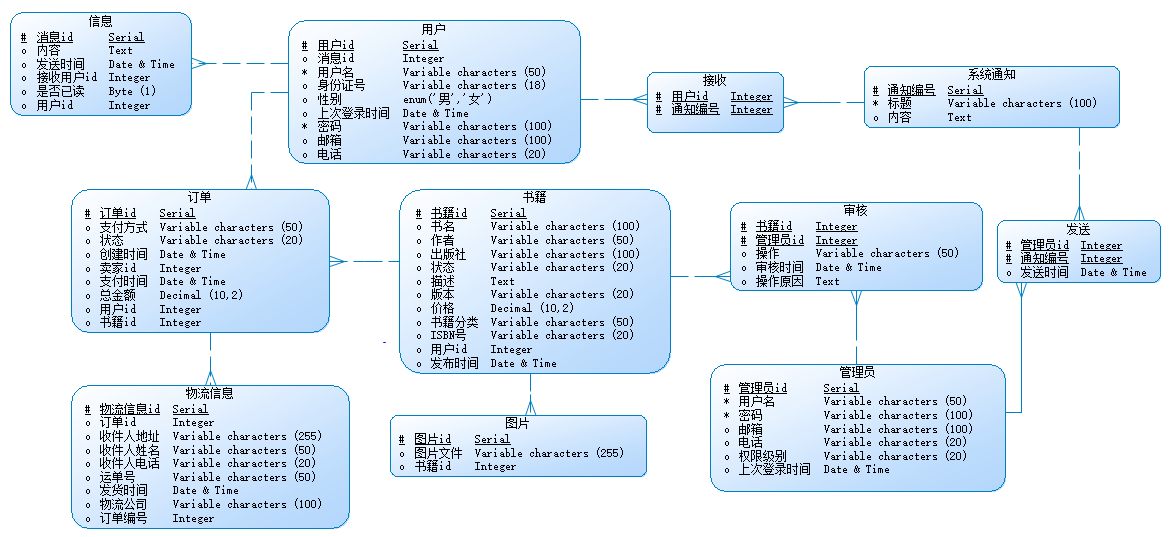


图4-2 逻辑模型

### 4.2.2数据库sql语句

CREATE TABLE user (

userId INT PRIMARY KEY AUTO\_INCREMENT,

userName VARCHAR(50) NOT NULL,

idNumber VARCHAR(18) UNIQUE,

gender ENUM('男', '女') DEFAULT '男',

lastLoginTime DATETIME,

password VARCHAR(100) NOT NULL,

email VARCHAR(100),

phone VARCHAR(20)

);

CREATE TABLE admin (

adminId INT PRIMARY KEY AUTO\_INCREMENT,

userName VARCHAR(50) NOT NULL,

password VARCHAR(100) NOT NULL,

email VARCHAR(100),

phone VARCHAR(20),

permissionLevel VARCHAR(20),

lastLoginTime DATETIME

);

CREATE TABLE systemNotice (

noticeId INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(100) NOT NULL,

content TEXT

);

CREATE TABLE book (

bookId INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(100),

author VARCHAR(50),

publisher VARCHAR(100),

status VARCHAR(20),

description TEXT,

version VARCHAR(20),

price DECIMAL(10,2),

category VARCHAR(50),

isbn VARCHAR(20),

userId INT,

publishTime DATETIME,

FOREIGN KEY (userId) REFERENCES user(userId)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE image (

imageId INT PRIMARY KEY AUTO\_INCREMENT,

imageFile VARCHAR(255),

bookId INT,

FOREIGN KEY (bookId) REFERENCES book(bookId)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE orders (

orderId INT PRIMARY KEY AUTO\_INCREMENT,

paymentMethod VARCHAR(50),

status VARCHAR(20),

createTime DATETIME,

sellerId INT,

paymentTime DATETIME,

totalAmount DECIMAL(10,2),

userId INT,

bookId INT,

FOREIGN KEY (userId) REFERENCES user(userId)

ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (bookId) REFERENCES book(bookId)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE logistics (

logisticsId INT PRIMARY KEY AUTO\_INCREMENT,

recipientAddress VARCHAR(255),

recipientName VARCHAR(50),

recipientPhone VARCHAR(20),

trackingNumber VARCHAR(50),

deliveryTime DATETIME,

logisticsCompany VARCHAR(100),

orderId INT,

FOREIGN KEY (orderId) REFERENCES orders(orderId)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE message (

messageId INT PRIMARY KEY AUTO\_INCREMENT,

content TEXT,

sendTime DATETIME,

receiverUserId INT,

isRead BOOLEAN DEFAULT FALSE,

userId INT,

FOREIGN KEY (userId) REFERENCES user(userId)

ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (receiverUserId) REFERENCES user(userId)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE review (

bookId INT,

adminId INT,

action VARCHAR(50),

reviewTime DATETIME,

reason TEXT,

PRIMARY KEY (bookId, adminId),

FOREIGN KEY (bookId) REFERENCES book(bookId)

ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (adminId) REFERENCES admin(adminId)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE receive (

userId INT,

noticeId INT,

PRIMARY KEY (userId, noticeId),

FOREIGN KEY (userId) REFERENCES user(userId)

ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (noticeId) REFERENCES systemNotice(noticeId)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE send (

adminId INT,

noticeId INT,

sendTime DATETIME,

PRIMARY KEY (adminId, noticeId),

FOREIGN KEY (adminId) REFERENCES admin(adminId)

ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (noticeId) REFERENCES systemNotice(noticeId)

ON DELETE CASCADE ON UPDATE CASCADE

);

# 5、系统出错处理设计

## 5.1 出错信息

|  |  |  |  |
| --- | --- | --- | --- |
| 可能出现的错误 | 系统输出信息形式 | 系统输出信息含义 | 出错提醒界面 |
| 账号不存在 | 消息提示 | 用户账号输入错误 | QQ20250427-204237 |
| 密码错误 | 消息提示 | 用户密码输入错误 | QQ20250427-204450 |
| 请求资源不存在 | 错误页面 | 用户请求资源不存在 | QQ20250427-204450 |
| 服务器内部错误 | 错误页面 | 服务器内部出错 |  |
| 网络错误 | 错误页面 | 网络通信失败，请重试 |  |
| 资源重定向 | 错误页面 | 服务器资源重定向 |  |
| 响应时间过长 | 错误页面 | 服务器相应时间过长，点击重试 |  |

表 5-1

## 5.2 补救措施

1.异常捕获：系统采用全局异常拦截机制，自动捕获业务异常、系统异常和未知错误，并记录详细日志（包括错误堆栈、请求参数、用户信息等关键数据），同时支持自定义异常类型扩展。

2.异常处理：建立分级处理策略：对用户展示可配置的友好提示（支持多语言），提供明确操作指引。

3.自动重试：针对瞬时性故障建立智能重试策略，采用渐进式延迟算法，设置最大重试次数限制。系统自动记录每次重试的详细日志信息，包含完整的请求上下文，同时支持根据不同错误类型配置差异化的重试策略。

4.服务降级：设计多层次的服务降级策略，根据系统负载和依赖服务状态动态调整响应方式。降级级别从完整服务逐步过渡到核心数据返回，最终保障最基本功能可用。所有降级操作实时反映在监控系统，并提供手动干预接口。

5.数据补偿：实现异步补偿任务管理机制，自动捕获失败操作并生成补偿任务。任务执行确保幂等性，支持按业务优先级调度，提供完整的补偿历史追踪和手动触发功能，关键业务补偿操作设置安全确认环节。

6.熔断恢复：基于实时系统指标动态调整熔断阈值和恢复策略。在服务熔断期间实施健康状态探测，根据系统恢复情况智能调整探测频率，服务恢复后优先处理积压的重要业务请求。

7.事务回滚：构建完善的事务管理体系，支持分布式环境下的数据一致性保障。回滚操作保留完整的执行上下文，提供灵活的回滚策略选择，关键业务数据变更前自动生成保护性快照。

8.流量控制：实施动态流量调控机制，根据系统实时负载智能调整请求处理策略。包括建立多级请求缓冲队列，实现业务优先级调度，保障核心业务通道的可用性。

总之系统出错处理的目标是快速发现问题、最小化影响、高效恢复服务。通过分层捕获、分类处理和多级保障机制，确保系统在异常情况下仍能提供可控的服务能力。