



**课程设计报告**

**课程名称 计算机网络**

**题目名称**

**学生学院**

**专业班级**

**学 号**

**学生姓名**

**指导教师 梁路**

**2022年5月**

# 绪论

## 选题背景

这些年来，互联网技术突飞猛进，电子商务随之兴起，创新出了一种利用网络购物的方式。为顺应时代潮流，开始发展或转变电商业务的传统行业与日俱增。在成熟的互联网技术的支持下，电子商务系统有了俱全、友好的平台为广大消费者提供网上购物服务。网上购物渐渐被越来越多人接受，甚至成为多数人购物时的首选。淘宝、天猫、京东和拼多多等综合型电商购物平台以及当当网、亚马逊图书、小米商城等专注于特定领域的电商购物平台都尤其热门。

网上书城是目前电子商务系统应用最为成功的具体表现形式之一。在网上书城上购书的方式，相对于在实体店购书的方式，不仅质量和效率更高而且还更加方便快捷。网上书城运用信息化的科技手段，商家可以更加合理化、科学化地进行经营管理。

## 系统开发方法

网上书城系统的结构是B/S（Browser/Server）结构。B/S结构有成本低、系统维护开销小、开发和升级简单、统一客户端、用户数量大等优点。前端的浏览器网页通过Vue.js框架开发，Vue的学习和使用难度都比较低。后端的Web服务器通过Node.js和Express框架开发。Node.js是让前端开发语言JavaScript能在服务端运行的开发平台。Node.js平台上可以使用快速、开放、极简的Web开发框架Express。数据库系统采用MySQL。MySQL是一个多线程、多用户和健壮的SQL数据库服务器，有体积小、速度快、成本低、开源等优点。

## 系统开发目标

网上书城系统的开发须达成以下目标：

1. 用户界面要简洁且人性化，交互操作要方便且容易。
2. 系统要运行稳妥、功能俱全。
3. 系统数据库设计要合理规范。

## 本文的主要工作

本文针对大部分商家对网上销售图书和用户网上买书的需求，设计与实现了一个基于Vue和Node.js的网上书城。主要研究的内容包括以下几个方面：

1. 了解网上书城的基本构成，调查对比其他网上购书网站的发展现状。
2. 完成网上书城的需求分析，主要使用用例建模与详述等方法。
3. 完成网上书城的总体架构、界面草图、功能模块和数据库的设计。
4. 完成网上书城的数据库、界面、功能模块的实现过程的说明。
5. 完成网上书城的各功能模块测试用例的设计和测试。

## 论文的组织结构

本文共划分为以下章节：

第一章：说明网上书城的相关背景、开发方法与目标以及主要工作。

第二章：简述开发网上书城所采用的相关技术。

第三章：对网上书城的经济可行性和技术可行性进行分析。

第四章：对网上书城进行详细的需求分析和用例建模。

第五章：基于网上书城的需求分析，设计架构、界面、功能模块和数据库。

第六章：基于网上书城的设计，对数据库、界面和功能模块做详细的实现介绍。

第七章：对网上书城进行功能模块测试。

第八章：对本文的工作进行总结。

# 相关技术概述

## MVC和MVVM

MVC模式包括三个部分：模型(Mode1)、视图(View)和控制器(Controller)，分别对应于内部数据、数据表示和输入输出控制部分[1]。View通过事件通知到Controller，Controller经过对事件的处理完成相关业务逻辑，要求Model改变数据状态，Model再将新数据更新到View。MVC模式结构如图2.1所示。



图2.1 MVC模式结构

MVVM（Model-ViewModel-View）代表模型-视图模型-视图模式，它是一种简化用户界面的事件驱动编程方式。MVVM模式结构如图2.2所示。

在MVVM架构中，是不允许Model和View直接通信的，只能通过ViewModel来通信。ViewModel是连接View和Model的中间件。ViewModel能够观察到Model数据的变化，并对View对应的内容进行更新。ViewModel能够监听到View的变化，并能够通知Model数据发生变化。

View中的视图逻辑通过双向数据绑定和命令绑定到ViewModel的属性上，通过ViewModel得到Model中的数据，双向数据绑定实现了视图和数据模型的自动同步机制[2]。

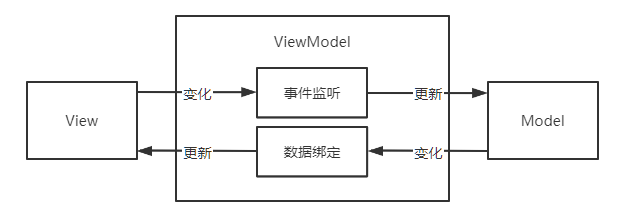


图2.2 MVVM模式结构

## Vue.js

Vue.js 是近年来最热门的前端框架技术之一，容易学习和使用，本身的体积小。与其他重量级框架不同的是，Vue是一套构建用户界面的渐进式框架，采用自底向上增量开发的设计方式，是更加灵活、开放的解决方案，架构更加简单，适合开发人员快速掌握其全部特性并投入使用，还便于与第三方库或既有项目整合[3]。渐进式框架的特点是可以只使用框架中你需要用的功能特性，Vue不强制要求你一次性接受并使用它的全部功能特性。

Vue.js是一个MVVM模式的实现者。Vue通过监听器劫持并监听要成为响应式的数据属性，将这些数据属性定义为访问器属性，定义setter函数和getter函数并分配一个订阅者管理器Dep。Vue通过模板解释器对于绑定了响应式数据的DOM元素，创建其订阅者Watcher。响应式数据的getter负责将依赖它的订阅者收集到订阅者管理器中，这将在订阅者创建时进行。数据更新时因为调用了数据属性的setter函数，此时会通知订阅者管理器Dep调用订阅者的更新函数。Vue通过监听原生DOM事件，在视图发生变化的时候就更新相应的数据属性。Vue响应式原理如图2.3所示。

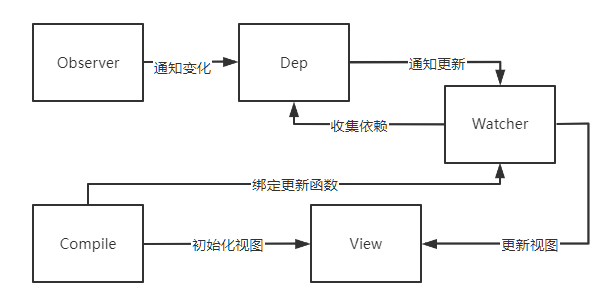


图2.3 Vue响应式原理

## Node.js和Express

Node.js是服务器端的JavaScript运行环境，它具有无阻塞(non-blocking)和事件驱动(event-driven)等的特色，Node.Js采用V8引擎，同样，Node.js实现了类似Apache和nginx的web服务，让你可以通过它来搭建基于JavaScript的WebApp[4]。如果需要执行读取或写入操作时，例如对数据库进行增删改查或增加文件到本地文件系统时，Node.js不会阻塞线程去等待操作完成，而是异步执行，通过回调来处理操作完成的结果。因此，只需要一个Web服务器，就可以处理成千个并发连接请求。

Express是用于Node.js(Node)平台的最小但灵活且强大的Web开发框架[5]。Express有易上手、高性能和扩展性强等优点，通过路由配置可以更有效地开发和管理接口，通过各种中间件可以对各种HTTP请求进行逻辑处理。

## B/S结构

把传统C/S模式中的服务器部分解为一个数据服务器与一个或多个应用服务器(Web服务器)，从而构成一个三层结构的客户服务器体系[6]。B/S（Browser/Server）结构如图2.4所示。浏览器处理用户的交互行为，向Web服务器发送HTTP请求，处理响应的数据进行显示。Web服务器解析请求，处理相应业务逻辑，请求数据库服务器进行读写数据操作，进一步处理后响应浏览器请求。数据库服务器根据操作请求处理数据库内部的数据，返回处理结果。

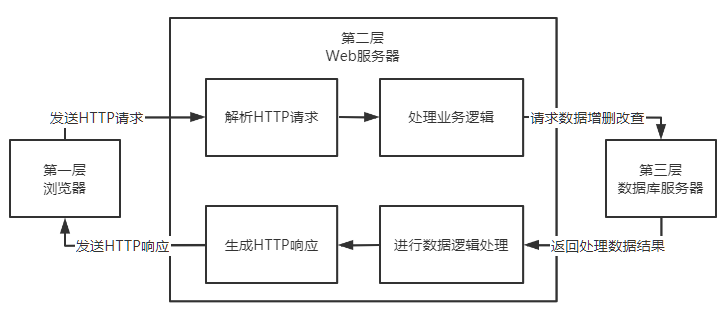


图2.4 B/S结构

# 可行性分析

## 经济可行性

该文的网上书城系统开发主要采用的相关技术都考虑到了经济成本。采用的数据库系统MySQL、Web服务器Node.js、前端框架Vue.js、开发工具Visual Studio Code都是开源的，在开发的费用上是经济可行的。

## 技术可行性

Vue框架在近几年都非常热门流行，学习和使用它的开发者人员数量非常庞大。关于Vue的教程文档、网课和例子随处可见，所以使用Vue框架来开发会更加容易。开发前端页面和采用Node.js开发Web服务器都是使用JavaScript语言，所以前端和后端都只需学习一门语言就能实现。VS Code（Visual Studio Code）开发工具既可以开发前端也可以开发Web服务器，而且支持JavaScript的语法提示和代码补全，降低了开发难度和提高了容错率，也支持插件扩展。通过扩展，任何平台的开发人员都可以将VS Code与任何语言和框架一起使用，以开发任何类型的应用程序[7]。

本网上书城系统所采用的MySQL、Node.js、Vue.js技术都是当前热门、成熟、可靠的系统解决方案。网上书城系统的功能和结构并不算太复杂，技术难度不会太高，而且可以参考借鉴许多相似的文献和项目。

# 需求分析

## 需求获取

根据网上书城使用群体的意愿及功能分析，网上书城被划分为前台和后台系统。前台包括用户登录或注册网上书城、按照需要筛选图书、查看图书的信息、加购物车、添加地址和提交订单等功能。后台包括增删改查图书、增删改查类别、修改订单状态和修改账户状态等功能。

## 总体需求

### 功能性需求

1. 前台：
2. 登录：用户点击顶部菜单栏的登录菜单项时跳转到登录页面；登录失败时提示错误信息，比如密码不正确或账号不存在或账号被冻结；登录成功后自动跳转前台首页。
3. 注册：用户点击顶部菜单栏的注册菜单项时跳转到注册页面；注册失败时提示错误信息，比如账号已存在或创建账号失败；注册时的用户名和密码长度有最短和最长限制，由数字和字母组成；注册成功后自动登录。
4. 修改密码：用户点击顶部菜单栏的修改我的密码菜单项，打开的对话框内显示表单，用户填写现在、新修改和二次确认的密码后，进行密码修改就点击修改按钮；密码长度有最短和最长限制，由数字和字母组成；密码修改成功后提示成功信息；如果修改密码失败，则提示错误信息，比如密码错误或修改失败。
5. 筛选图书：用户在搜索栏输入书名、作者或ISBN可以对图书进行模糊查询；类别选择栏中展示图书的二级分类名，用户在类别选择栏点击图书类别可以筛选出特定类别的图书，支持图书类别多选；用户点击排序操作栏中的按默认、销量、价格升序或降序可以对图书进行相应排序；默认情况下网页展示10本图书，用户可以点击分页器的页码选择剩下的图书，可以选择网页能展示多少本图书。
6. 图书信息：图书的封面、名称、ISBN、作者、类别、定价和销量信息要能在页面上查看。
7. 收货信息：用户点击顶部菜单栏的收货信息菜单项时，打出的对话框内展示用户已保存的姓名、电话、地址这些收货的相关信息；点击对话框的新增按钮显示输入框，填写姓名、电话、地址后可以进行新增操作；可以修改收货信息的姓名、电话、地址；点击删除按钮可以删除该收货信息。
8. 订单：用户点击图书的下单按钮或购物车的一键下单按钮打开确认订单对话框，可以选择订单的收货信息、图书购买数量；用户点击顶部菜单栏的订单菜单项时打出我的订单对话框，对话框内展示用户所有订单信息，包含编号、下单时间、付款时间、发货时间、总价格、状态、快递单号；用户点击订单打开订单详情信息对话框，对话框内展示该订单信息，包含收货信息、购买的图书信息、编号、下单时间、总价格、状态、快递单号；
9. 购物车：可以加购特定数量的图书；点击顶部菜单栏的我的购物车菜单项，打出的对话框内展示用户已添加的图书；点击相应图书的编辑按钮可以改变该图书的本数；点击图书的删除按钮可以取消加购该图书；点击对话框的一键下单按钮可以一次性购买购物车里的所有图书。
10. 后台：
11. 登录：登录成功时自动跳转到后台首页。登录失败时提示例如密码不正确或账号被冻结等错误信息；
12. 管理员管理：可以输入账号名、姓名、手机号和选择状态进行筛选；可以对管理员账户进行分页查看；可以添加管理员包含账号名、密码、姓名、手机号码、等级和状态的账户，账号名、密码有最短和最长限制，由数字和字母组成，等级分1到5级，状态为正常和冻结两种；可以编辑账号的信息；可以删除账号。
13. 用户管理：可以输入用户名和账号状态进行检索；可以分页查看用户账号；可以更改用户账号的状态为正常或冻结。
14. 分类管理：可以根据类名进行筛选；可以新增一级或二级图书类别；可以对类别进行改名操作；可以对图书类别进行移除。
15. 图书管理：可以输入图书的各种信息进行筛选；可以新增图书；可以更改图书的相关信息；可以分页查看图书。
16. 订单管理：可以根据订单编号、所属用户和状态进行筛选；可以分页查看所有订单；可以查看订单的所属用户、收货信息、图书信息、总价格、下单时间、付款时间、发货时间、状态和快递单号，订单有未付款、未发货、已发货三种状态；打开发货对话框并提供订单的快递单号就能进行发货操作。
17. 权限控制：管理员账户的等级共分成5级，订单管理需要1级以上，分类管理需要2级以上，图书管理需要3级以上，用户账户管理需要4级以上，管理员账户管理需要5级以上。
18. 数据报表：后台首页展示待发货的订单数量、图书销量排行、图书库存为空数量的数量。

### 非功能性需求

1. 网页主题明确，重点突出；页面设计简洁、人性化，易于用户操作；网页风格设计统一，布局合理大方；网页提供深浅两种颜色主题。
2. HTML合理使用语义化标签和减少嵌套层次，CSS样式尽量使用ID和类名选择器，提高代码可读性和渲染速度。
3. 提高用户与网页交互的响应速度，操作反馈要及时。
4. 组件化、模块化开发，方便系统扩展功能，降低代码之间的耦合度。

## 参与者

参与者是某些具有行为的事务，可以是人，计算机系统或组织[8]。后台系统管理员和前台普通用户是网上书城系统的参与者。参与者活动表如表4.1所示。

表4.1 参与者活动表

|  |  |
| --- | --- |
| 参与者 | 活动 |
| 系统管理员 | 1. 登录后台管理系统 2. 查询订单、查询图书、查询类别、查询用户、查询管理员 3. 修改订单、修改图书、修改类别、修改用户、修改管理员 4. 添加图书、添加类别、添加管理员 5. 删除图书、删除类别、删除管理员 |
| 普通用户 | 1. 登录或注册前台系统 2. 查看图书、查看购物车里的图书、查看订单相关信息、查看收货信息 3. 将图书加入购物车、下单购买图书、取消订单 4. 修改购物车里的图书、修改收货信息、修改账号密码 |

## 用例建模

用例建模通过用例图可以体现参与者的需求。管理员对网上书城后台系统的用例如图4.1所示。普通用户对网上书城前台系统的用例如图4.2所示。

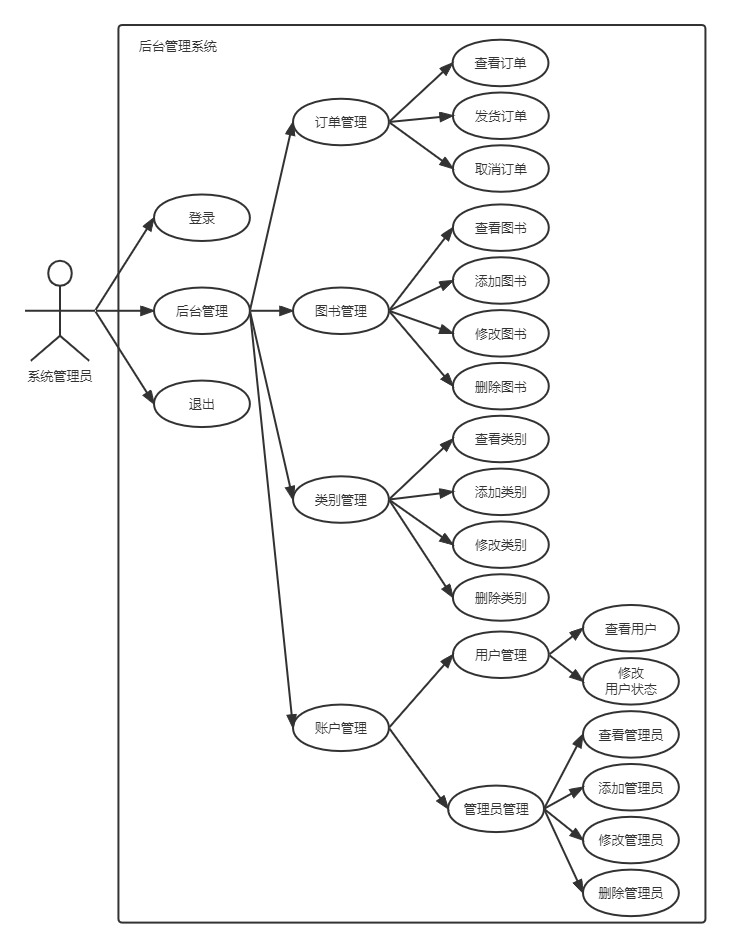


图4.1 系统管理员用例图

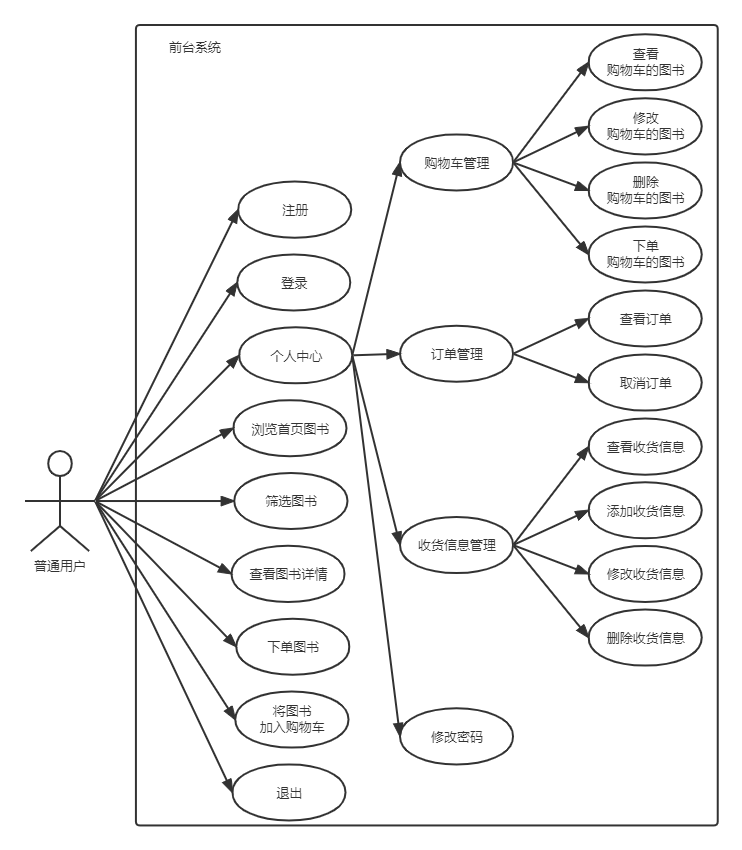


图4.2 普通用户用例图

## 用例详述

用例详述能够详细地描述网上书城系统的功能含义。以下主要对后台的添加图书、查询图书、修改图书和删除图书进行用例详述。

1. 添加图书的用例详述如表4.2所示。

表4.2 添加图书用例详述表

|  |  |
| --- | --- |
| 用例名称 | 添加图书 |
| 参与者 | 系统管理员 |
| 用例说明 | 在后台管理系统填写添加图书需要的信息，并添加到系统数据库中 |
| 前置条件 | 管理员已成功登录后台管理系统且管理员账户等级为3级以上 |
| 基本事件流 | 1. 管理员点击后台管理系统左侧菜单栏中的“图书管理”选项 2. 系统显示“图书管理”页内容 3. 管理员点击 “添加图书”按钮 4. 系统打开“添加图书”对话框 5. 管理员填写添加图书所需要的表单内容，选择图书所属类别，上传图书封面图片 6. 管理员点击“确定”按钮，完成添加图书的操作 7. 用例终止 |
| 其他事件流 | 在管理员点击“确定”按钮前，管理员关闭该“添加图书”对话框，系统不会对表单中已存在的内容进行任何操作 |
| 异常事件流 | 填写的图书信息格式有误或不全，提示错误的地方，管理员点击“确定”按钮，表单内容不会提交 |
| 后置条件 | 系统完成图书添加，提示成功，新增图书能在“图书管理”页进行查看 |

1. 查询图书的用例详述如表4.3所示。

表.3 查询图书用例详述表

|  |  |
| --- | --- |
| 用例名称 | 查询图书 |
| 参与者 | 系统管理员 |
| 用例说明 | 管理员输入或选择特定的查询条件，系统根据查询条件到数据库中进行查询，并把结果以表格和分页的形式展示到页面上 |
| 前置条件 | 管理员已成功登录后台管理系统且管理员账户等级为3级以上 |
| 基本事件流 | 1. 管理员点击后台管理系统左侧菜单栏中的“图书管理”选项 2. 系统显示“图书管理”页内容 3. 管理员根据需要在查询表单中填写或选择所需要的查询条件 4. 点击“筛选”按钮，完成查询操作，页面显示查询结果 5. 用例终止 |
| 其他事件流 | 如果不填写任何查询条件，直接点击“筛选”按钮或点击“重置”按钮清空查询表单内容，就直接查询所有的图书 |
| 异常事件流 | 填写的图书信息格式有误，管理员点击“确定”按钮，表单内容不会提交 |
| 后置条件 | 系统完成查询，并把结果以表格和分页的形式展示到页面上 |

1. 修改图书的用例详述如表4.4所示。

表4.4 修改图书用例详述表

|  |  |
| --- | --- |
| 用例名称 | 修改图书 |
| 参与者 | 系统管理员 |
| 用例说明 | 在后台管理系统填写修改图书需要的信息，并到系统数据库中进行修改 |
| 前置条件 | 管理员已成功登录后台管理系统且管理员账户等级为3级以上 |
| 基本事件流 | 1. 管理员在“图书管理”页的表格中点击需要修改的图书的“编辑”按钮 2. 系统打开“修改图书”对话框 3. 管理员根据需要修改图书的信息，修改对话框内表单的内容 4. 管理员点击“确定”按钮，完成修改图书的操作 5. 用例终止 |
| 其他事件流 | 在管理员点击“确定”按钮前，管理员关闭该“修改图书”对话框，系统清空表单中已存在的内容 |
| 异常事件流 | 填写的图书信息格式有误，管理员点击“确定”按钮，表单内容不会提交 |
| 后置条件 | 系统完成图书修改，提示成功，修改后的图书能在“图书管理”页进行查看 |

1. 删除图书的用例详述如表4.5所示。

表4.5 删除图书详述表

|  |  |
| --- | --- |
| 用例名称 | 删除图书 |
| 参与者 | 系统管理员 |
| 用例说明 | 在后台管理系统删除图书，并到系统数据库中进行删除，且无法恢复 |
| 前置条件 | 管理员已成功登录后台管理系统且管理员账户等级为3级以上 |
| 基本事件流 | 1. 管理员在“图书管理”页的表格中点击需要删除的图书的“删除”按钮 2. 系统打开“删除图书”确认框 3. 管理员确认后，进行删除操作 4. 用例终止 |
| 异常事件流 | 删除失败时，提示错误 |
| 后置条件 | 系统完成图书删除，提示成功，删除后的图书不能在“图书管理”页进行查看 |

# 系统设计

## 架构设计

为方便用户使用和管理员管理，网上书城系统应设计成B/S结构。该结构的Web服务器被划分成视图层、控制层、数据层，设计成MVC模式。视图层提供接受客户端请求的接口，请求通过相应的配置路由到相应的控制器，等控制器处理完后发送结果给浏览器端。控制层会对请求进行数据验证、计算等逻辑处理，需要拿到相应数据可以调用数据层的相关操作接口，处理完接口返回的数据后传给视图层。数据层连接MySQL数据库系统并对数据表中的数据进行查询、更新等操作，提供接受操作请求的接口给控制层。系统结构图如图5.1所示。

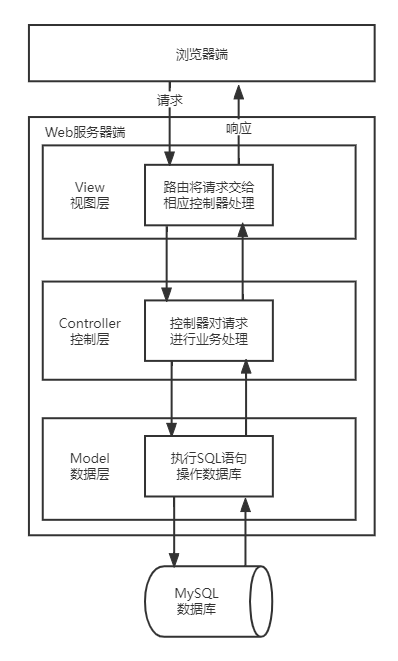


图.1 系统结构图

## 界面设计

1. 前台：
2. 前台首页界面要有顶部导航栏，图书筛选栏，图书的基本信息展示。首页设计草图如图5.2所示。



图5.2 前台首页界面设计草图

1. 图书详情界面使用对话框展示，包括图书的详情信息，底部要有对该图书的相应操作栏。图书详情设计草图如图5.3所示。

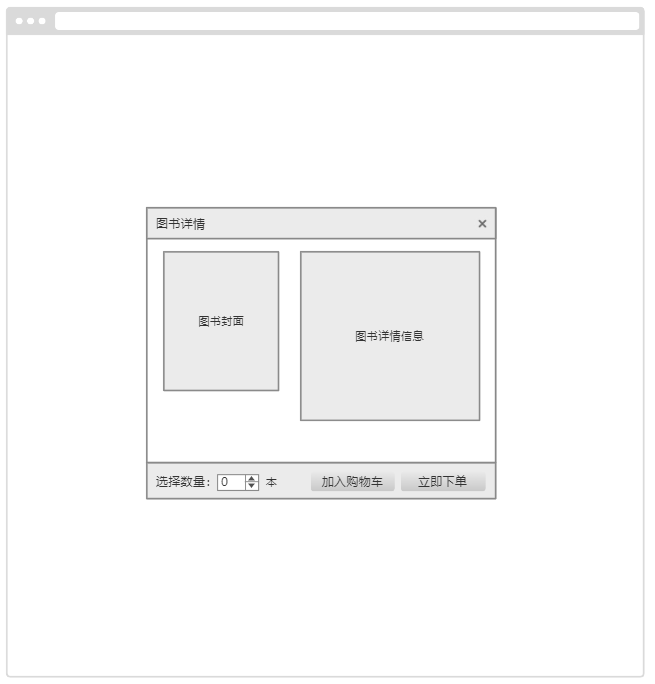


图5.3 前台图书详情界面设计草图

1. 我的购物车界面使用对话框展示，包含已加入购物车的图书信息，有对购物车里图书的相应操作栏。我的购物车设计草图如图5.4所示。

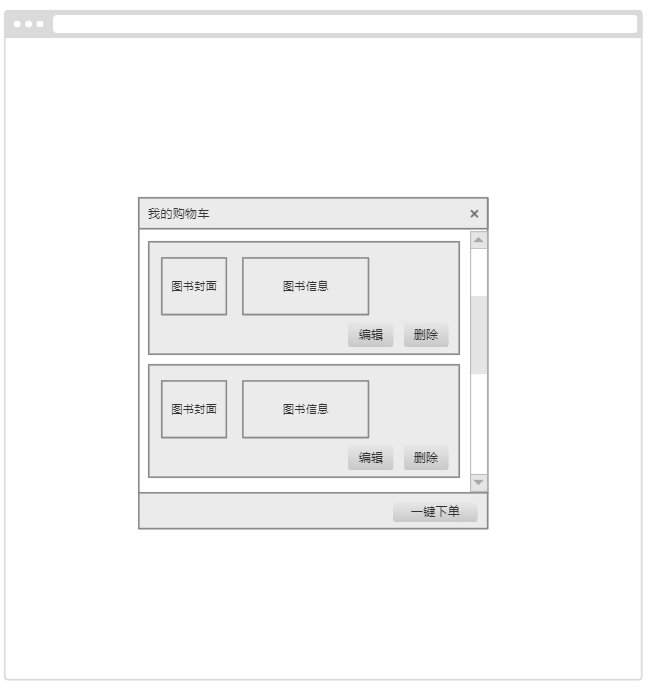


图5.4 前台我的购物车界面设计草图

1. 确认订单界面使用对话框展示，包含订单的收货信息和图书信息，有订单的价格信息和操作按钮。确认订单设计草图如图5.5所示。



图5.5 前台确认订单界面设计草图

1. 我的订单界面使用对话框展示，包含用户的订单信息和相应操作栏。我的订单设计草图如图5.6所示。

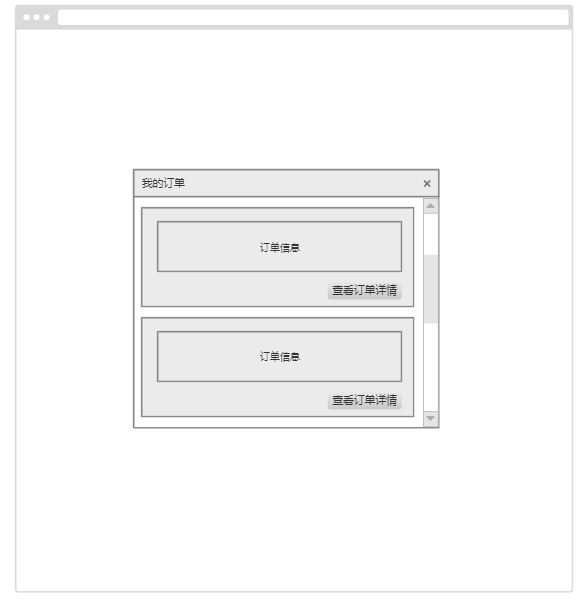


图5.6 前台我的订单界面设计草图

1. 我的订单详情界面使用对话框展示，要有该订单的收货信息，下单的图书信息和订单相关信息。我的订单详情设计草图如图5.7所示。

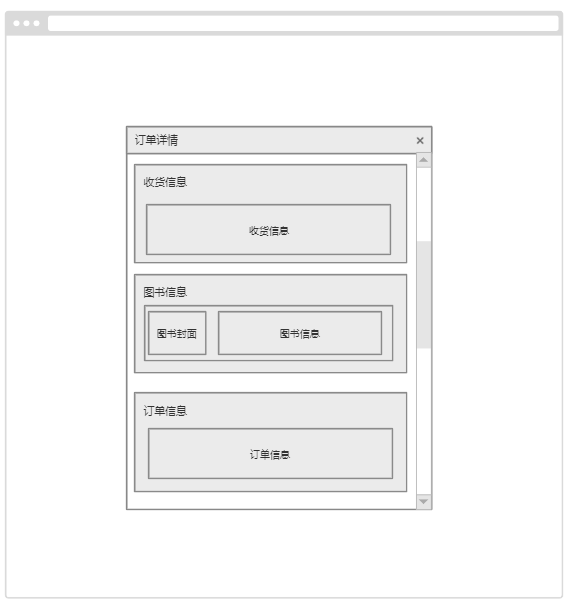


图5.7 前台我的订单详情设计草图

1. 后台：
2. 后台管理首页界面包含待发货订单和图书库存为空的数量信息，通过柱状图展示各图书的销量排行。后台首页界面设计草图如图5.8所示。

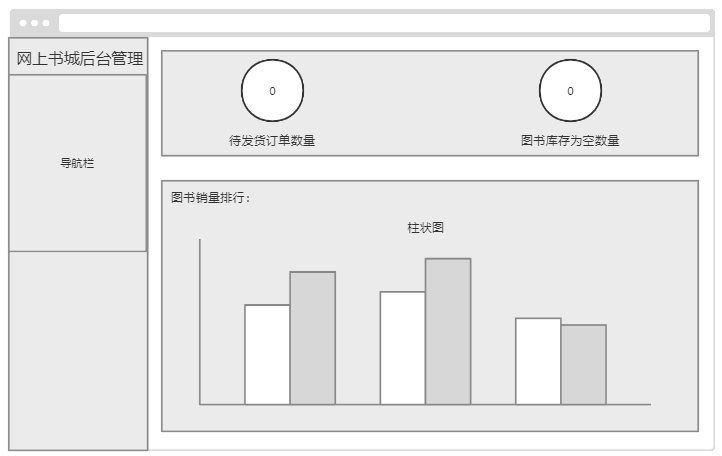


图5.8 后台管理首页界面设计草图

1. 图书管理界面包含搜索栏，通过表格展示图书信息，有图书相应的操作按钮。图书管理界面设计草图如图5.9所示。



图5.9 后台图书管理界面设计草图

1. 添加图书界面使用对话框展示，包含添加图书各种信息的表单控件和操作栏。添加图书界面设计草图如图5.10所示。

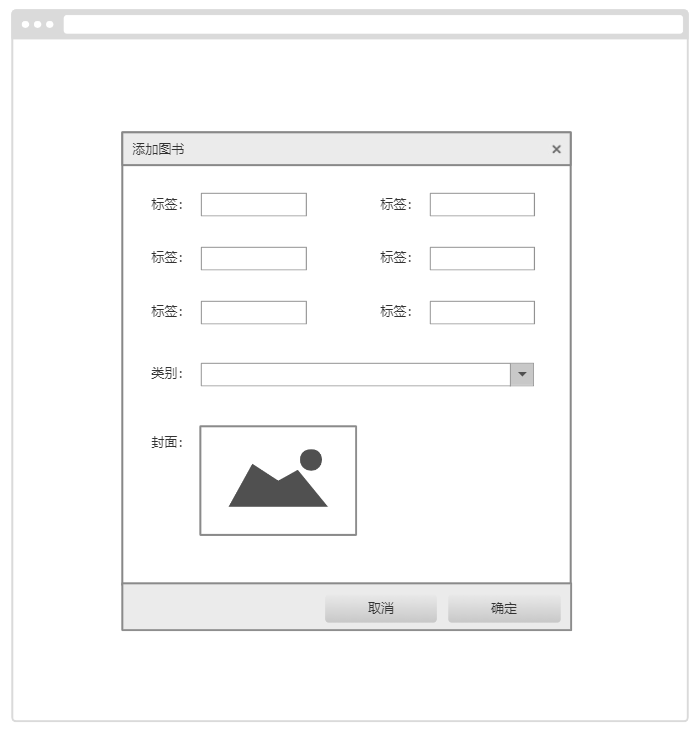


图5.10 后台添加图书界面设计草图

## 功能模块设计

将系统划分为若干个功能模块，可以更加灵活方便地实现各项业务功能。模块化就是把程序划分成独立命名且可独立访问的模块，每个模块完成一个子功能，把这些模块集成起来构成一个整体，可以完成指定的功能满足用户的需求[9]。网上书城系统划分成了前台的用户和后台的管理员两个模块，用户的功能描述如图5.11所示，管理员的功能描述如图5.12所示。

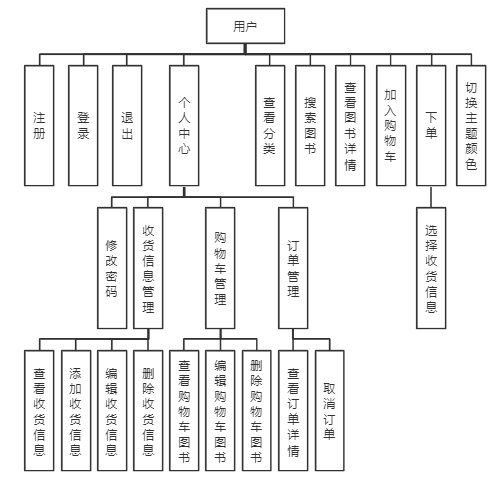


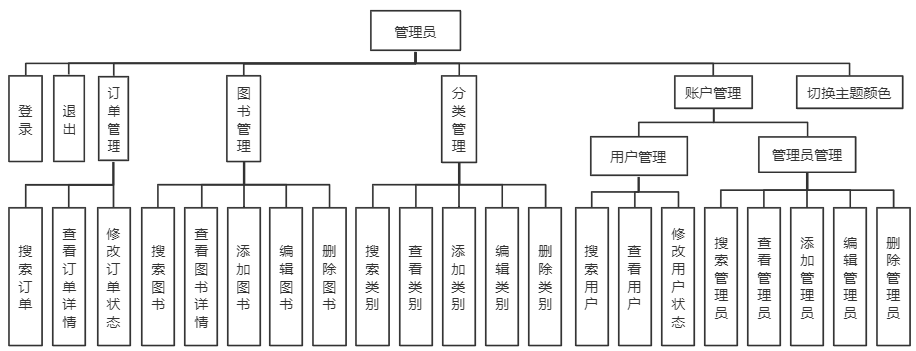
图.11 用户功能模块图

图5.12 管理员功能模块图

## 数据库设计

### 概念结构设计

将需求分析得到的用户需求抽象为信息结构(即概念模型)的过程就是概念结构设计[10]。实体关系图的实体是实际的事物，实体的属性就是其一特性，实体间存在一个实体比一个实体、一个实体比多个实体、多个实体比多个实体三种联系。概念模型能通过E-R实体关系图来进行描述。

1. 本系统有如下7个实体：
2. 网上书城管理员：属性有编号、账号名、密码、状态、等级、姓名、电话。
3. 网上书城用户：属性有编号、账号名、密码、状态。
4. 用户收货信息：属性有编号、收货人的手机号、地址、姓名。
5. 用户订单：属性有编号、合计价格、收货人的手机号、地址、姓名、状态、下单时间、付款时间、发货时间、快递单号。
6. 订单细则：属性有编号、该细则的图书数量、该细则的图书ISBN、该细则的书名、该细则的图书价格、该细则的图书封面。
7. 图书：属性有编号、ISBN、书名、作者、出版社、封面、数量、定价、销量。
8. 类别：属性有编号、父类编号、类名。
9. 这些实体间的联系有如下5种：
10. 用户和收货信息之间是一个实体比多个实体的联系。一个用户能添加多个收货信息，一个收货信息只能属于一个用户。
11. 用户和图书之间是多个实体比多个实体的联系。一个用户能将多个图书实体加入购物车，一个图书实体可以被多个用户加入购物车。
12. 用户和订单之间是一个实体比多个实体的联系。一个用户能下多个订单，一个订单只能属于一个用户。
13. 订单和订单细则之间是一个实体比多个实体的联系。一个订单能由多个订单细则组成，一个订单细则只能属于一个订单。
14. 图书和类别之间是多个实体比多个实体的联系。一本图书能被划分成多个类别，一个类别可以包含多本图书。

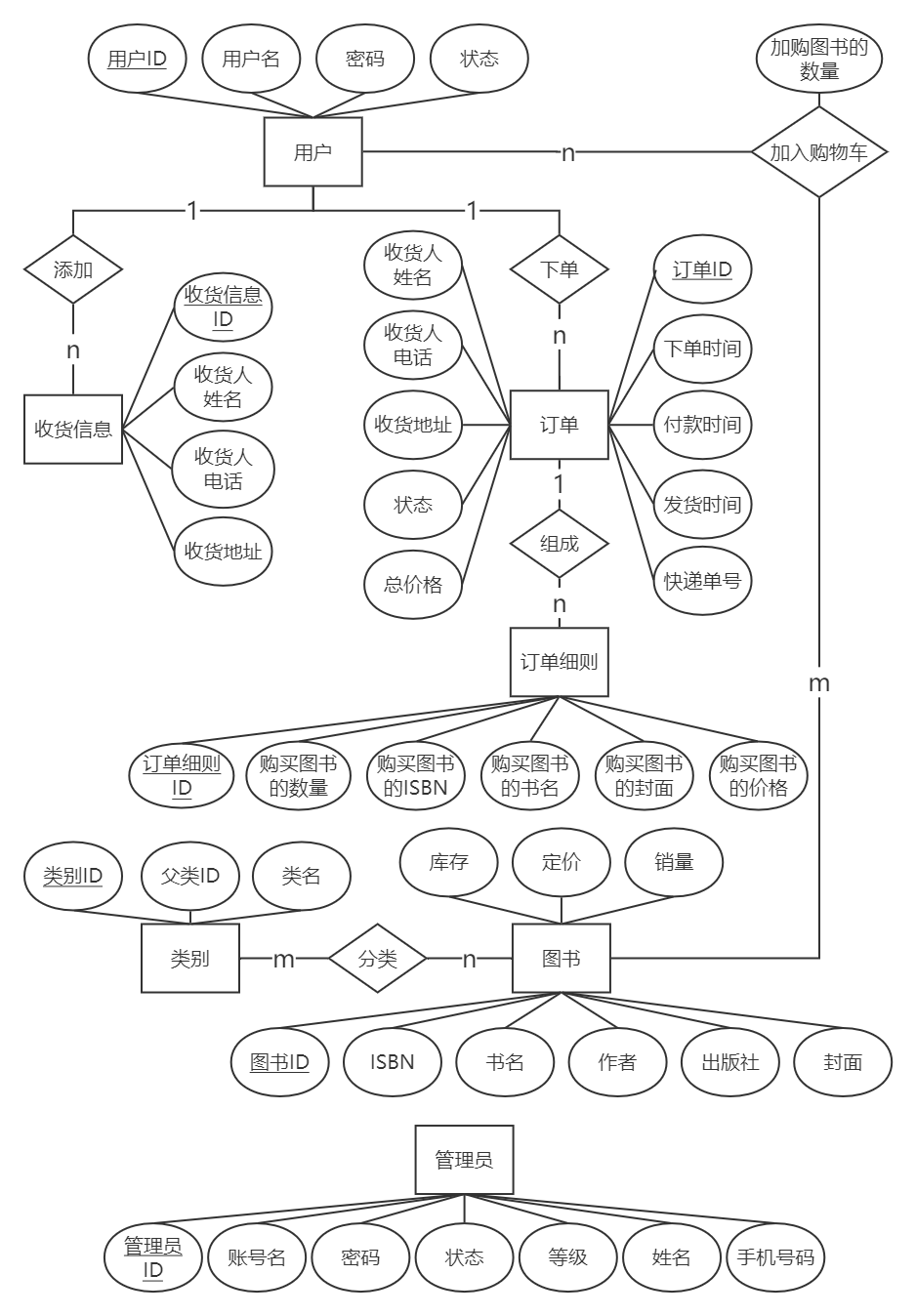
网上书城的数据库的完整E-R实体关系图如图5.13所示。

图5.13 网上书城E-R图

### 逻辑结构设计

概念结构是独立于任何一种数据模型的信息结构，逻辑结构设计的任务就是把概念结构设计阶段设计好的基本E-R图转换为与选用数据库管理系统产品所支持的数据模型相符合的逻辑结构[10]。分析网上书城的完整E-R图，一个实体应作为一个逻辑结构表，该表的主键是实体的码，该表的字段是实体的属性。一个实体比多个实体的联系（1：n）中与1对应的实体的码是与n对应的实体的逻辑结构表的外键。多个实体比多个实体的联系（m：n）可以单独作为一个逻辑结构表，该表的主键和外键是与该联系相连的各实体的码，表的字段是该联系上的属性。以下是各逻辑结构表的设计：

1. book图书逻辑结构表如表5.1所示。

表5.1 图书逻辑结构表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 备注 | 数据类型 | 字段长度 | 约束 |
| ID | 图书ID | Int | 11 | 主键，自增 |
| ISBN | 国际标准书号 | Char | 13 | 非空，唯一 |
| bookname | 书名 | Varchar | 20 | 非空 |
| author | 作者 | Varchar | 20 | 非空 |
| press | 出版社 | Varchar | 20 | 无 |
| price | 定价 | Float | 11 | 非空 |
| amount | 库存 | Int | 11 | 非空 |
| sale | 销量 | Int | 11 | 非空 |
| cover | 封面 | Varchar | 255 | 无 |

1. category类别逻辑结构表如表5.2所示。

表5.2 类别逻辑结构表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 备注 | 数据类型 | 字段长度 | 约束 |
| ID | 类别ID | Int | 11 | 主键，自增 |
| category | 类别名 | Varchar | 10 | 非空 |
| superID | 父类ID | Int | 11 | 外键 |

1. book\_category图书分类逻辑结构表如表5.3所示。

表5.3 图书分类逻辑结构表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 备注 | 数据类型 | 字段长度 | 约束 |
| bookID | 图书ID | Int | 11 | 主键，外键 |
| book\_categoryID | 类别ID | Int | 11 | 主键，外键 |

1. admin管理员逻辑结构表如表5.4所示。

表5.4 管理员逻辑结构表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 备注 | 数据类型 | 字段长度 | 约束 |
| ID | 管理员ID | Int | 11 | 主键，自增 |
| admin | 账号名 | Varchar | 20 | 非空，唯一 |
| password | 密码 | Varchar | 20 | 非空 |
| name | 姓名 | Varchar | 20 | 无 |
| phone | 手机号码 | Char | 11 | 无 |
| role | 级别 | Int | 1 | 非空，只能为1、2、3、4、5 |
| status | 状态 | Char | 2 | 非空，只能为“正常”、“冻结” |

1. user用户逻辑结构表如表5.5所示。

表5.5 用户逻辑结构表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 备注 | 数据类型 | 字段长度 | 约束 |
| ID | 用户ID | Int | 11 | 主键，自增 |
| username | 用户名 | Varchar | 20 | 非空，唯一 |
| password | 密码 | Varchar | 20 | 非空 |
| status | 状态 | Char | 2 | 非空，只能为“正常”、“冻结” |

1. delivery收货信息逻辑结构表如表5.6所示。

表5.6 收货信息逻辑结构表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 备注 | 数据类型 | 字段长度 | 约束 |
| ID | 收货信息ID | Int | 11 | 主键，自增 |
| delivery\_userID | 用户ID | Int | 11 | 外键 |
| name | 收货人姓名 | Varchar | 20 | 非空 |
| phone | 收货人电话 | Char | 11 | 非空 |
| address | 收货地址 | Varchar | 255 | 非空 |

1. orders订单逻辑结构表如表5.7所示。

表5.7 订单逻辑结构表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 备注 | 数据类型 | 字段长度 | 约束 |
| ID | 订单ID | Int | 11 | 主键，自增 |
| orders\_userID | 用户ID | Int | 11 | 外键 |
| createdTime | 下单时间 | Timestamp | -- | 非空 |
| payTime | 付款时间 | Timestamp | -- | 无 |
| sendTime | 发货时间 | Timestamp | -- | 无 |
| totalPrice | 总金额 | Float | 11 | 非空 |
| name | 收货人姓名 | Varchar | 20 | 非空 |
| phone | 收货人电话 | Char | 11 | 非空 |
| address | 收货地址 | Varchar | 255 | 非空 |
| status | 订单状态 | Char | 3 | 非空，只能为“未付款”，“未发货”，“已发货”，“已取消” |
| trackingNumber | 快递单号 | Int | 11 | 无 |

1. order\_item订单细则逻辑结构表如表5.8所示。

表5.8 订单细则逻辑结构表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 备注 | 数据类型 | 字段长度 | 约束 |
| ID | 订单细则ID | Int | 11 | 主键,自增 |
| item\_orderID | 订单ID | Int | 11 | 外键 |
| bookID | 购买图书ID | Int | 11 | 非空 |
| num | 购买图书数量 | Int | 11 | 非空 |
| ISBN | 购买图书  国际标准书号 | Char | 13 | 非空,唯一 |
| bookname | 购买图书书名 | Varchar | 20 | 非空 |
| price | 购买图书价格 | Float | 11 | 非空 |
| cover | 购买图书封面 | Varchar | 255 | 无 |

1. shopcart购物车细则逻辑结构表如表5.9所示。

表5.9 购物车细则逻辑结构表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 备注 | 数据类型 | 字段长度 | 约束 |
| shopcart\_userID | 用户ID | Int | 11 | 主键，外键 |
| bookID | 图书ID | Int | 11 | 主键，外键 |
| num | 加购图书数量 | Int | 11 | 非空 |

# 系统实现

## 数据库实现

对设计好的数据结构模型在MySQL数据库软件使用sql脚本进行建表。

1. 图书表book的sql实现脚本：

CREATE TABLE `book` (

`ID` int NOT NULL AUTO\_INCREMENT,

`ISBN` char(13) NOT NULL,

`bookname` varchar(20) NOT NULL,

`author` varchar(20) NOT NULL,

`press` varchar(20) DEFAULT NULL,

`cover` varchar(255) DEFAULT NULL,

`amount` int NOT NULL,

`price` float(11,2) NOT NULL,

`sale` int NOT NULL DEFAULT '0',

PRIMARY KEY (`ID`),

UNIQUE KEY `ISBN` (`ISBN`)

) ENGINE=InnoDB AUTO\_INCREMENT=118 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

1. 类别表category的sql实现脚本：

CREATE TABLE `category` (

`ID` int NOT NULL AUTO\_INCREMENT,

`category` varchar(10) NOT NULL,

`superID` int DEFAULT NULL,

PRIMARY KEY (`ID`),

KEY `superID` (`superID`),

CONSTRAINT `superID` FOREIGN KEY (`superID`) REFERENCES `category` (`ID`) ON DELETE RESTRICT ON UPDATE RESTRICT

) ENGINE=InnoDB AUTO\_INCREMENT=112 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

1. 图书分类表book\_category的sql实现脚本：

CREATE TABLE `book\_category` (

`bookID` int NOT NULL,

`book\_categoryID` int NOT NULL,

PRIMARY KEY (`bookID`,`book\_categoryID`),

KEY `book\_categoryID` (`book\_categoryID`),

CONSTRAINT `bookID` FOREIGN KEY (`bookID`) REFERENCES `book` (`ID`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `book\_categoryID` FOREIGN KEY (`book\_categoryID`) REFERENCES `category` (`ID`) ON UPDATE CASCADE ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

1. 管理员表:admin的sql实现脚本：

CREATE TABLE `admin` (

`ID` int NOT NULL AUTO\_INCREMENT,

`admin` varchar(20) NOT NULL,

`password` varchar(20) NOT NULL,

`name` varchar(20) DEFAULT NULL,

`phone` char(11) DEFAULT NULL,

`status` char(2) NOT NULL DEFAULT '正常',

`role` int NOT NULL DEFAULT '0',

PRIMARY KEY (`ID`),

UNIQUE KEY `admin` (`admin`)

) ENGINE=InnoDB AUTO\_INCREMENT=39 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

1. 用户表user的sql实现脚本：

CREATE TABLE `user` (

`ID` int NOT NULL AUTO\_INCREMENT,

`username` varchar(20) NOT NULL,

`password` varchar(20) NOT NULL,

`status` char(2) NOT NULL DEFAULT '正常',

PRIMARY KEY (`ID`),

UNIQUE KEY `username` (`username`)

) ENGINE=InnoDB AUTO\_INCREMENT=29 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

1. 收货信息表delivery的sql实现脚本：

CREATE TABLE `delivery` (

`ID` int NOT NULL AUTO\_INCREMENT,

`delivery\_userID` int NOT NULL,

`name` varchar(20) NOT NULL,

`phone` char(11) NOT NULL,

`address` varchar(255) NOT NULL,

PRIMARY KEY (`ID`),

KEY ` delivery\_userID` (`delivery\_userID`) USING BTREE,

CONSTRAINT ` delivery\_userID` FOREIGN KEY (`delivery\_userID`) REFERENCES `user` (`ID`) ON UPDATE CASCADE ON DELETE CASCADE

) ENGINE=InnoDB AUTO\_INCREMENT=13 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

1. 订单表orders的sql实现脚本：

CREATE TABLE `orders` (

`ID` int AUTO\_INCREMENT,

`orders\_userID` int NOT NULL,

`name` varchar(20) NOT NULL,

`phone` char(11) NOT NULL,

`address` varchar(255) NOT NULL,

`totalPrice` float(11,2) NOT NULL,

`createdTime` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP,

`payTime` timestamp NULL DEFAULT NULL,

`sendTime` timestamp NULL DEFAULT NULL,

`status` char(3) NOT NULL DEFAULT '未付款',

`trackingNumber` int DEFAULT NULL,

PRIMARY KEY (`ID`),

KEY `orders\_userID` (`orders\_userID`),

CONSTRAINT `orders\_userID` FOREIGN KEY (`orders\_userID`) REFERENCES `user` (`ID`) ON UPDATE CASCADE ON DELETE CASCADE

) ENGINE=InnoDB AUTO\_INCREMENT=58 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

1. 订单细则表order\_item的sql实现脚本：

CREATE TABLE `order\_item` (

`ID` int NOT NULL AUTO\_INCREMENT,

`item\_orderID` int NOT NULL,

`bookID` int NOT NULL,

`num` int NOT NULL,

`ISBN` char(13) NOT NULL,

`bookname` varchar(20) NOT NULL,

`price` float(11,2) NOT NULL,

`cover` varchar(255),

PRIMARY KEY (`ID`),

KEY ` item\_orderID` (`item\_orderID`),

CONSTRAINT ` order\_item\_orderID` FOREIGN KEY (`item\_orderID`) REFERENCES `orders` (`ID`) ON UPDATE CASCADE ON DELETE CASCADE

) ENGINE=InnoDB AUTO\_INCREMENT=66 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

1. 购物车细则表shopcart的sql实现脚本：

CREATE TABLE `shopcart` (

`shopcart\_userID` int NOT NULL,

`bookID` int NOT NULL,

`num` int NOT NULL,

PRIMARY KEY (`shopcart\_userID`,`bookID`),

KEY `shopcart\_bookID` (`bookID`) USING BTREE,

CONSTRAINT `shopcart\_bookID` FOREIGN KEY (`bookID`) REFERENCES `book` (`ID`) ON UPDATE CASCADE ON DELETE CASCADE,

CONSTRAINT `shopcart\_userID` FOREIGN KEY (`shopcart\_userID`) REFERENCES `user` (`ID`) ON UPDATE CASCADE ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

## 界面实现

1. 前台：
2. 首页界面顶部的导航栏可以让用户操作购物车、订单和个人的信息，筛选操作栏可以让用户通过输入图书的相关信息、选择类别和排序方式来检索图书，图书的信息通过卡片来展示。首页界面如图6.1所示。



图6.1 前台首页界面

1. 图书详情界面在对话框内展示图书的所有信息，在对话框底部，用户可以对图书进行加入购物车或立即购买操作，先在输入框中输入具体多少本图书，然后可以通过相应的按钮进行操作。图书详情界面如图6.2所示。



图6.2 前台图书详情界面

1. 购物车界面显示了图书的基本信息和加购的数量，有对其中图书的编辑和删除按钮和对所有图书进行购买的按钮。购物车界面如图6.3所示。

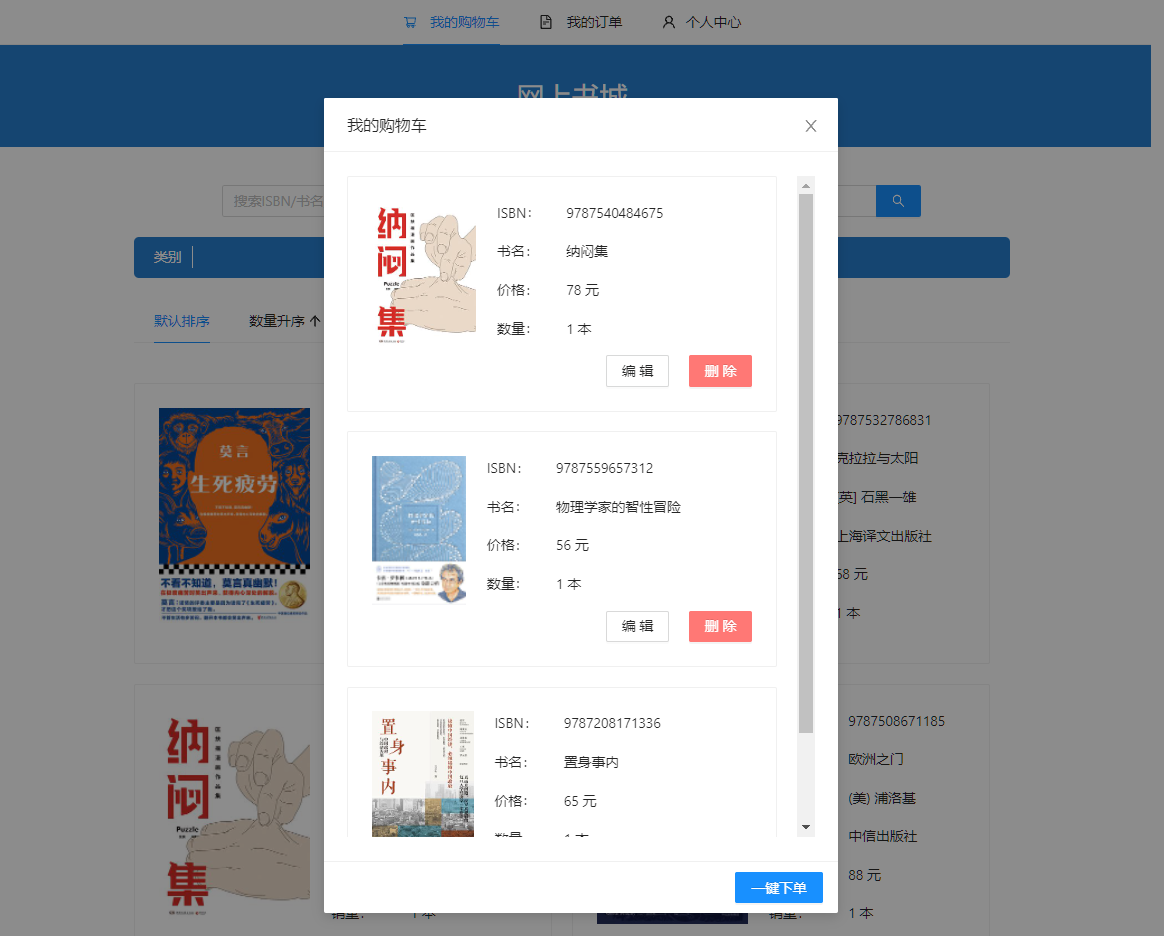


图6.3 前台购物车界面

1. 下单界面包含收货信息栏给用户提供选择，要下单的图书信息、订单总价格和确认下单按钮。下单界面如图6.4所示。

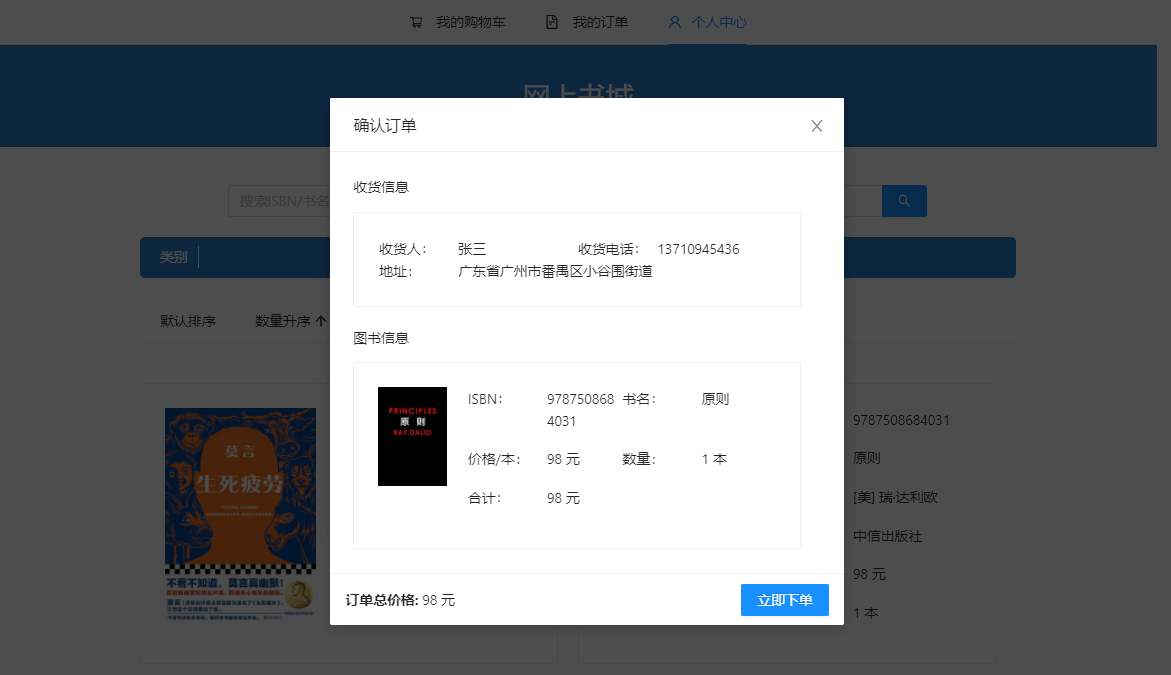


图6.4 前台下单界面

1. 我的订单界面包含订单的编号、下单图书、总价格、状态和时间基本信息。前台我的订单界面如图6.5所示。



图6.5 前台我的订单界面

1. 我的订单详情界面展示该订单的收货相关信息、购买的图书信息和其他订单相关信息。我的订单详情界面如图6.6所示。

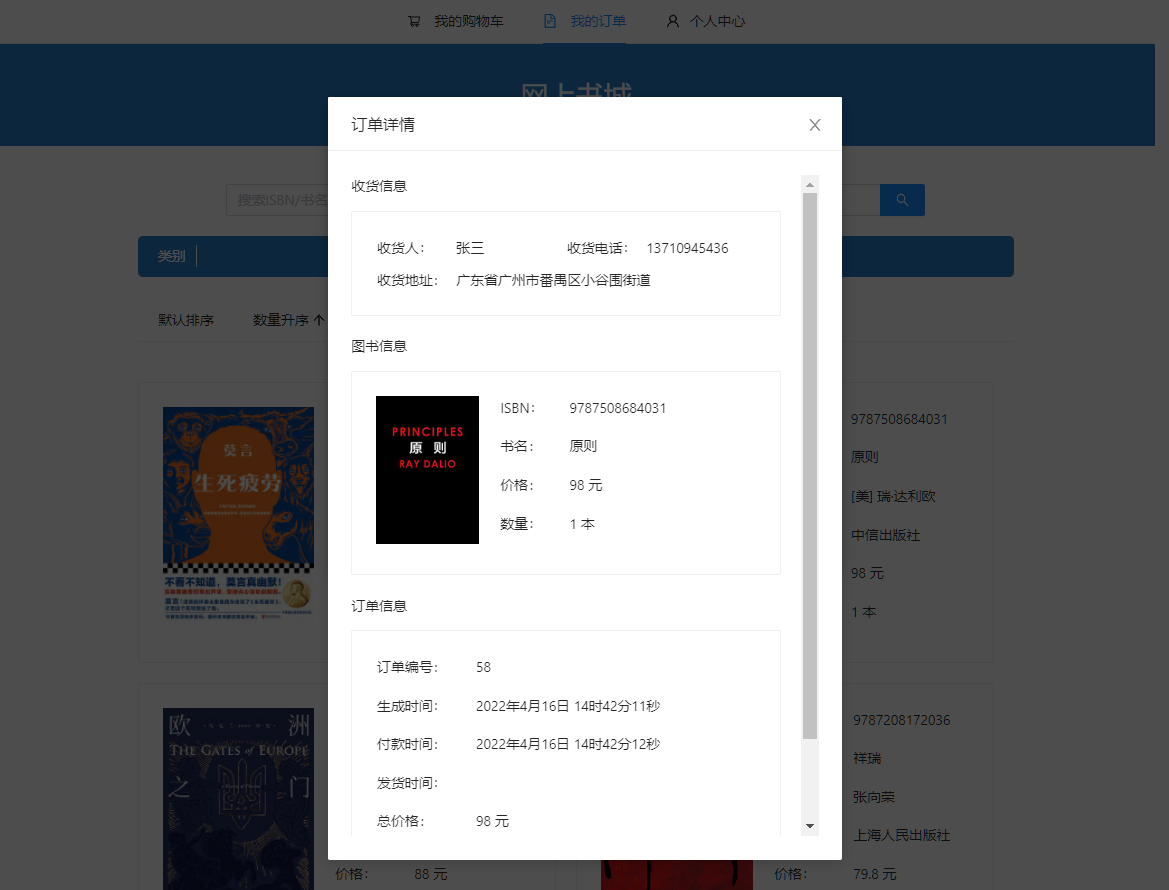


图6.6 前台我的订单详情界面

1. 后台：
2. 后台管理首页提示了待发货的订单数量和图书库存为空的数量，通过柱状图展示了各种图书的销量排行信息。管理首页界面如图6.7所示。



图6.7 后台管理首页界面

1. 图书管理界面有对图书进行检索的筛选栏，通过表格展示了图书的信息，展开表格的行能展示相应图书的更多信息，提供了对图书进行增删改的按钮。图书管理界面如图6.8所示。

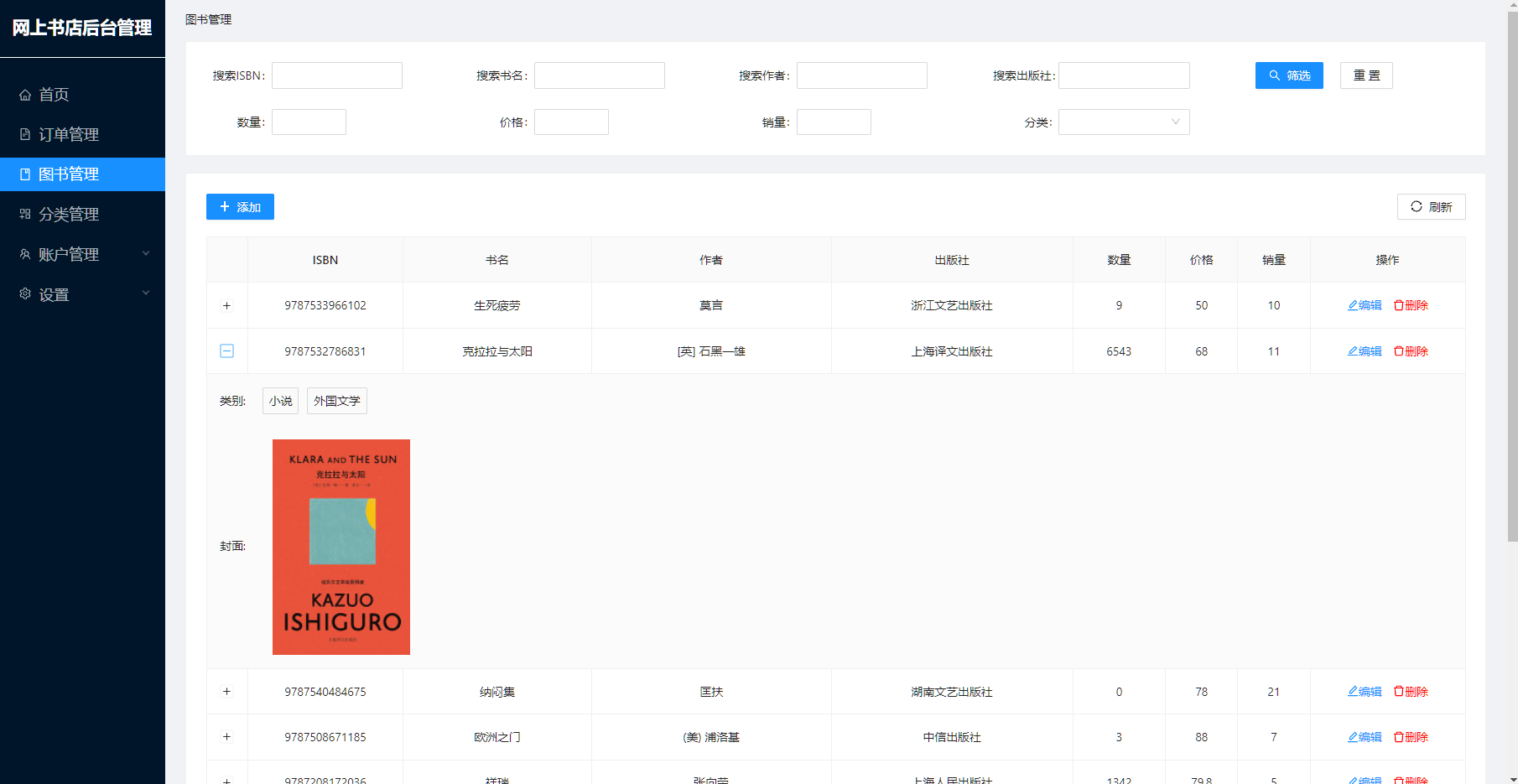


图6.8 后台图书管理界面

1. 添加图书界面使用对话框进行显示，包含填写图书信息的输入框，选择图书类别的选择框和对图书封面进行操作的一系列按钮。添加图书界面如图6.9所示。

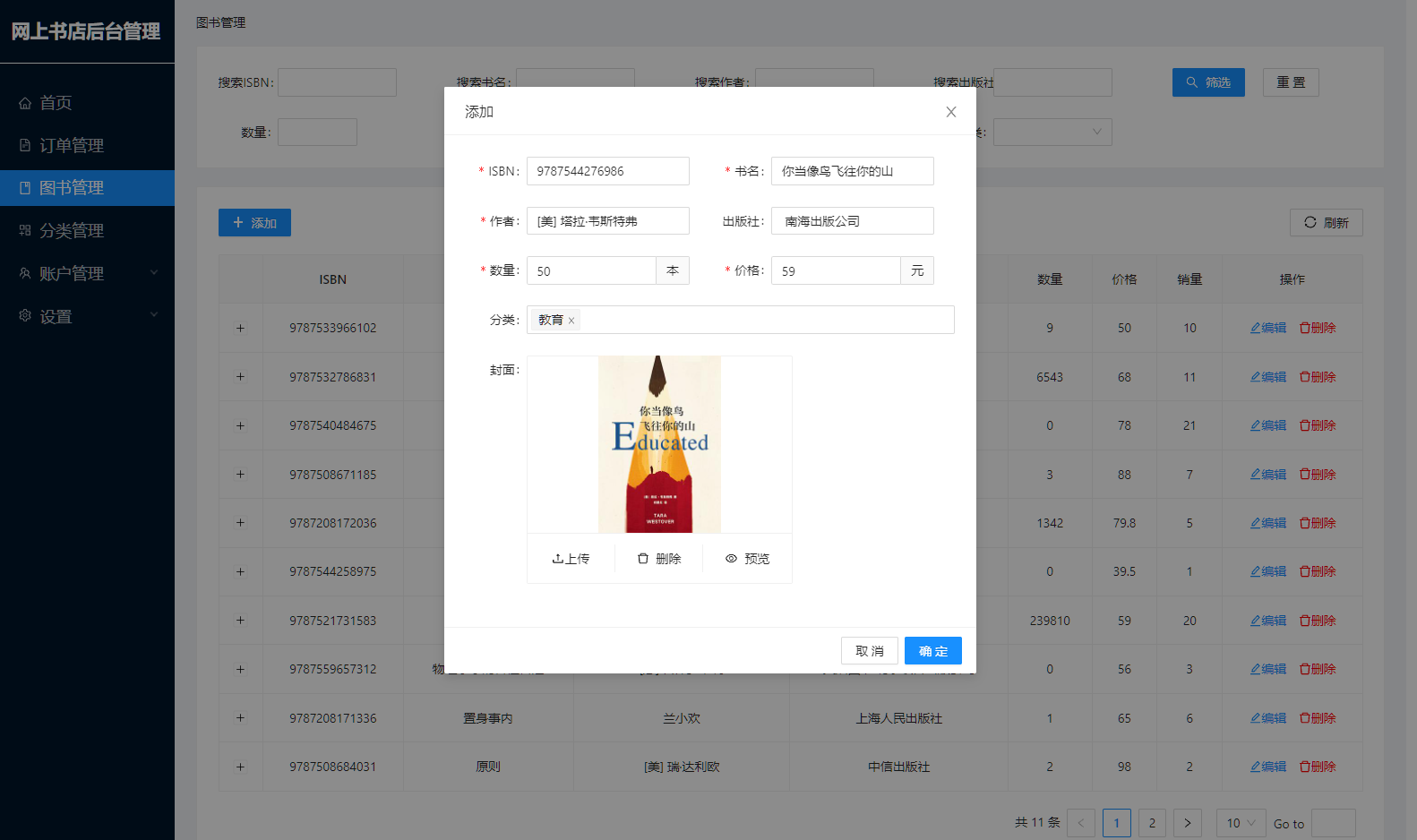


图6.9 后台添加图书界面

## 功能模块实现

### 前台功能模块

1. 实现前台图书搜索功能：
2. 前端将query数据通过Vue框架的v-model指令绑定到页面上的输入框上，query数据会同步用户输入的内容。监听用户选择类别的事件触发时往category对象上添加包含类别ID和类名的属性。监听用户选择排序方式的事件触发时设置orderBy数据和orderHow数据。监听用户选择页码和每页显示数量的事件触发时设置pageNum数据和pageSize数据。
3. 当用户点击搜索按钮或选择了类别或选择了排序或切换了页面时，前端向后端服务器发送HTTP请求，发送查询数据query、category、orderBy、orderHow、pageNum、pageSize。
4. 后端服务器接受到HTTP请求，视图层Express框架的路由负责解析请求，根据请求的URL转交给获取图书的控制器进行业务处理。
5. 控制器验证请求中的查询数据是否为空还有类型是否正确，错误则返回包含错误信息的HTTP响应，正确则调用模型层的查询接口获取数据。
6. 模型层执行sql查询语句，将图书分类表左连接到图书表，通过where匹配表的数据和查询数据，对查询到的数据进行排序，最后返回查询到数据给控制器。
7. 控制器对模型层的查询接口返回的结果按照页码和每页显示数量进行分页处理并生成HTTP响应将图书数据返回给前端。
8. 前端收到HTTP响应后，把响应体的图书数据赋值给book数组，book数组因为绑定到了页面的列表上，所以页面上的图书列表会自动更新。

实现前台图书搜索功能的部分代码参见附录A。

1. 实现前台下单功能：
2. 前端获取用户选择的收货信息ID，购买的图书ID，输入的数量。
3. 用户执行下单操作，前端向后端发送HTTP请求，发送获取到的数据。
4. 后端接受到前端的HTTP请求，视图层Express框架的路由解析请求，调用验证用户账号的中间件，根据请求的URL转交给下单的控制器。
5. 控制器验证请求中的查询数据是否为空还有类型是否正确，错误则返回包含错误信息的HTTP响应，正确则调用模型层的添加订单接口。
6. 模型层获取数据库连接，开始事务，先根据收货信息ID在收货信息表中获取对应的收货人姓名、电话、地址数据，再根据图书ID在图书表中获取库存、价格数据，当库存不小于购买数量时，向订单表中插入一条包含用户编号、收货信息、订单里图书总价格的数据，最后向订单细则表插入包含订单ID、图书ID、购买数量的数据。全部插入成功后，提交事务，否则回滚。
7. 控制器收到模型层的操作结果，判断是否操作成功，是则向前端发送包含成功信息的HTTP响应。
8. 前端判断返回的HTTP响应是否包含成功信息，是则提示用户下单成功，否则提示下单失败。

实现前台下单功能的部分代码参见附录B。

### 后台功能模块

实现后台图书添加功能：

1. 前端将ISBN、书名、作者、出版社、数量、价格、类别的这些添加图书的数据通过Vue框架的v-model指令绑定到页面上的输入框上，添加图书的数据会同步用户输入的内容。管理员上传封面文件到服务器的临时文件夹后，前端获取服务器返回的临时封面文件的文件名。
2. 管理员确定添加图书时，前端向后端发送包含要添加图书数据的HTTP请求。
3. 后端接受到前端的HTTP请求，视图层Express框架的路由解析请求，调用验证管理员账号和权限的中间件，根据请求的URL转交给添加图书的控制器。
4. 控制器验证添加图书的数据是否为空或类型是否正确，错误则返回包含错误信息的HTTP响应，正确则将临时文件夹的封面文件移动到永久保存封面的文件夹，调用模型层添加图书的接口。
5. 模型层获取数据库连接，开始事务，先在图书表插入一条包含ISBN、书名、作者、出版社、数量、价格、封面文件名的数据。插入成功后，根据图书ID和选择的类别ID在图书分类表插入数条数据。全部插入成功后，提交事务，否则回滚。
6. 控制器收到模型层的操作结果，判断是否操作成功，是则向前端发送包含成功信息的HTTP响应。
7. 前端判断返回的HTTP响应是否包含成功信息，是就提示图书已成功保存了。

实现后台图书添加功能的部分代码参见附录C。

# 系统测试

## 测试用例

测试用例（test case）就是为了特定测试目的（如考察特定程序路径或验证某个产品特性）而设计的测试条件、测试数据及与之相关的操作过程序列的一个特定的使用实例或场景[11]。以下设计了网上书城系统的主要功能模块的测试用例：

1. 登录网上书城的测试用例如表7.1所示。

表7.1 登录功能模块的测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 测试目的 | 输入数据 | 期望结果 | 实际结果 |
| login\_1 | 不输入账号名，验证登录是否有正确的响应 | 空账号名和密码 | 登录失败，账号名输入框下面提示“账号名不能为空” | 测试通过 |
| login\_2 | 不输入密码，验证登录是否有正确的响应 | 账号名和空密码 | 登录失败，密码输入框下面提示“密码不能为空” | 测试通过 |
| login\_3 | 输入账号不存在的账号名，验证登录是否有正确的响应 | 账号不存在的账号名和密码 | 登录失败，页面顶部弹出消息框提示“登录失败，该账号不存在” | 测试通过 |
| login\_4 | 输入错误的密码，验证登录有正确的响应 | 账号存在的账号名和错误的密码 | 登录失败，页面顶部弹出消息框提示“登录失败，密码错误” | 测试通过 |
| login\_5 | 登录状态为冻结的账号，验证是否有正确的响应 | 账号存在且状态为冻结的账号名和正确的密码 | 登录失败，页面顶部弹出消息框提示“该账号已冻结” | 测试通过 |
| login\_6 | 登录状态为正常的账号，验证是否有正确的响应 | 账号存在且状态为正常的账号名和正确的密码 | 登录成功，页面跳转到首页，页面顶部弹出消息框提示“登录成功” | 测试通过 |

1. 所有模块的表单输入和提交功能的测试用例如表7.2所示。

表7.2 所有模块的表单输入和提交功能的测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 测试目的 | 输入数据 | 期望结果 | 实际结果 |
| form\_1 | 不输入带\*的必填表单项，验证提交表单是否有正确的响应 | 空的带\*的必填表单项 | 提交失败，带\*的必填项下面提示“该项不能为空” | 测试通过 |
| form\_2 | 对于要求输入数字类型的表单项，输入非数字类型的数据，验证提交表单是否有正确的响应 | 非数字类型的数据 | 提交失败，该表单项下面提示“该项不能输入非数字类型” | 测试通过 |
| form\_3 | 对于要求输入字符串类型的表单项，输入非字符串类型的数据，验证提交表单是否有正确的响应 | 非字符串类型的数据 | 提交失败，该表单项下面提示“该项不能输入非字符串类型” | 测试通过 |
| form\_4 | 不输入不带\*的可填表单项，验证提交表单是否有正确的响应 | 合法的带\*的必填表单项和空的不带\*的可填表单项 | 提交成功，页面顶部弹出消息框提示“提交成功” | 测试通过 |
| form\_5 | 输入所有表单项，验证提交表单是否有正确的响应 | 合法的带\*的必填表单项和合法的不带\*的可填表单项 | 提交成功，页面顶部弹出消息框提示“提交成功” | 测试通过 |

1. 上传封面图片功能模块的测试用例如表7.3所示。

表7.3 上传封面图片功能模块的测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 测试目的 | 输入数据 | 期望结果 | 实际结果 |
| upload\_1 | 上传非图片格式的文件，验证是否有正确的响应 | 非图片格式的文件 | 阻止上传，并提示“只能上传图片格式的文件” | 测试通过 |
| upload\_2 | 上传超过额定大小的图片文件，验证是否有正确的响应 | 超过额定大小的图片文件 | 阻止上传，并提示“上传的图片文件太大” | 测试通过 |
| upload\_3 | 上传正常的格式、大小符合的图片文件，验证是否有正确的响应 | 正常格式、大小符合的图片文件 | 上传成功，显示图片的预览图 | 测试通过 |
| upload\_4 | 已上传成功，再次上传正确的图片文件，验证是否有正确的响应 | 正常格式、大小符合的图片文件 | 上传成功，显示最后一次上传的图片的预览图 | 测试通过 |

1. 所有模块删除功能的测试用例如表7.4所示。

表7.4 所有模块删除功能的测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 测试目的 | 输入数据 | 期望结果 | 实际结果 |
| remove\_1 | 点击删除按钮，验证是否弹出确认框 | 点击删除按钮 | 弹出确认删除确认框 | 测试通过 |
| remove\_2 | 点击删除按钮，点击确认框的确定按钮，验证是否有正确的响应 | 点击删除按钮和点击确认框的确定按钮 | 删除成功，页面顶部弹出消息框提示“删除成功” | 测试通过 |
| remove\_3 | 点击删除按钮，点击确认框的取消按钮，验证是否有正确的响应 | 点击删除按钮和点击确认框的取消按钮 | 取消删除成功，页面顶部弹出消息框提示“已取消删除” | 测试通过 |

1. 下单功能模块的测试用例如表7.5所示。

表7.5 下单功能模块的测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 测试目的 | 输入数据 | 期望结果 | 实际结果 |
| order\_1 | 验证是否能输入超过图书库存的购买数量 | 超过图书库存的数量 | 输入的购买数量，不能超过图书库存，超过则自动改为库存最大数。 | 测试通过 |
| order\_2 | 点击下单按钮，验证是否弹出确认订单对话框 | 点击下单按钮 | 弹出确认订单对话框 | 测试通过 |
| order\_3 | 不选择收货信息，点击确认订单按钮，验证是否有正确的响应 | 空的收货信息和点击确认订单按钮 | 下单失败，页面顶部弹出消息提示框“请选择订单的收货信息” | 测试通过 |
| order\_4 | 选好收货信息，点击确认订单按钮，验证是否有正确的响应 | 收货信息和点击确认订单按钮 | 下单成功，页面顶部弹出消息提示框“下单成功” | 测试通过 |

## 系统测试结果分析

经过测试，网上书城系统基本都通过了设计的测试用例，使实现功能的正常运行得到了保证，整体功能模块都满足了系统的设计要求。

# 总结

这次毕业设计，设计的网上书城系统是自互联网兴起以来，最热门的电子商务系统之一。为独立完成该系统的开发，综合运用了多方面的当前互联网比较热门、前沿的技术知识，结合软件工程的开发思想，对系统进行分析、设计和实现。在系统设计阶段，通过对需求进行详细分析，确定了具体功能和开发方法，查阅和参考相关教材和文献，花费了大量时间和精力来学习开发技术知识。在完成了系统设计工作后进入系统开发过程，期间进行了许多次代码的修改优化和功能测试工作。

通过不懈的努力和不断的尝试，我终于实现了网上书城系统的所有功能。不过本系统也存在一些欠缺，比如前台页面设计得差强人意，代码冗余等，因此在页面设计上需要多参照优秀的设计模板，代码实现上需要可以提取相同的代码封装成函数调用。经历过了设计和开发的每一步后，我明白了只有明确需求和功能才能更好的进行编码实现，同时收获了全面及深刻的实践经验，编程开发能力得到了很大的提升。

# 参考文献

1. 任中方, 张华, 闫明松, 陈世福. MVC模式研究的综述[J]. 计算机应用研究, 2004, 21(10):1.
2. 孙连山, 李云倩. MVVM框架在Web前端的应用研究[J]. 电脑知识与技术：学术版, 2016, 0(2X):45.
3. 朱二华. 基于Vue.js的Web前端应用研究[J]. 科技与创新, 2017, 0(20):119.
4. 骆文亮. Node.js服务器技术初探[J]. 无线互联科技, 2014(3):227.
5. Hage Yaapa. Express Web Application Development[M]. Packt Publishing, 2013:5.
6. 侯淑英. B/S模式和C/S模式优势比较[J]. 沈阳教育学院学报, 2007, 9(2):98.
7. Ovais Mehboob Ahmed Khan, Khusro Habib, Chris Dias. Developing Multi-Platform Apps with Visual Studio Code[M]. Packt Publishing, 2020:37.
8. Craig Larman. Applying UML and Patterns[M]. 3rd Edition. Prentice Hall, 2004:47.
9. 张海藩, 牟永敏. 软件工程导论[M]. 第6版. 清华大学出版社, 2013:94.
10. 王珊, 萨师煊. 数据库系统概论[M]. 第5版. 高等教育出版社, 2014:215-231.
11. 朱少民. 软件测试[M]. 第2版. 人民邮电出版社, 2016:46.

# 致谢

在编写毕业论文和开发系统的过程中，我遇到过诸多大大小小的困难和挫折。在此期间，我特别感谢梁路指导老师。从毕业设计的选题到系统的设计开发再到毕业论文的编写，梁路老师都对我悉心教导，给予了我很大帮助。在每次的面对面的讨论中，梁路老师都能敏锐指出我的系统存在的缺点，还耐心地提出了相应独特的建议，解开了许多我遇到的困惑。

大学四年将在本次毕业设计上划上句号，本次毕业设计的完成离不开每一个教导过我的老师们和帮助过我的同学们，我能掌握扎实的计算机知识得益于您们的辛苦付出，非常感谢您们。

# 附录A

前台图书搜索功能的前端相关代码：

const queryForm = reactive({

pageNum: 1,

pageSize: 10,

query: "",

category: {},

orderBy: "",

orderHow: "",

});

const total = ref(0);

const dataList = ref([]);

const getDataList = () => {

const category = []

for (const key in queryForm.category) {

category.push(key);

}

bookAPI.getList({ ...queryForm, category })

.then((data) => {

dataList.value = data.dataList;

total.value = data.total;

queryForm.pageNum = data.pageNum;

});

}

onBeforeMount(getDataList);

const selectCategory = (category) => {

queryForm.category[category.value] = category.label;

getDataList();

}

const removeCategory = (categoryID) => {

delete queryForm.category[categoryID];

getDataList();

}

const currentSort = ref(['默认']);

const selectSort = () => {

if (currentSort.value[0] === '默认') {

queryForm.orderBy = '';

queryForm.orderHow = '';

} else {

queryForm.orderBy = currentSort.value[0];

queryForm.orderHow = queryForm.orderHow === 'desc' ? 'asc' : 'desc';

}

getDataList();

}

getList: (queryForm) => {

return new Promise((resolve, reject) => {

axios.post(`/book/getList`, queryForm)

.then(({ data: { status, msg, data } }) => {

if (status === 200) {

const { dataList, total, pageNum } = data;

resolve({ dataList, total, pageNum });

} else {

reject(msg);

}

})

.catch(() => {

reject("接口调用失败");

});

});

},

前台图书搜索功能的后端相关代码：

const getList = async ({ body }, res) => {

const { query, category, orderBy, orderHow, pageNum, pageSize } = body;

if (typeof query !== 'string' || !Array.isArray(category) || typeof orderBy !== 'string' || typeof orderHow !== 'string'

|| typeof pageNum !== 'number' || typeof pageSize !== 'number'

) {

return res.json({ status: 400, msg: "参数错误" });

}

const data = await bookModel.getList({ query, category, orderBy, orderHow });

const total = data.length;

if (total === 0) return res.json({ status: 200, msg: "成功", data: { dataList: [], total, pageNum: 1 } });

const maxPageNum = Math.ceil(total / pageSize);

const currentPage = pageNum > maxPageNum ? maxPageNum : pageNum;

const offset = (currentPage - 1) \* pageSize;

res.json({ status: 200, msg: "成功", data: { dataList: data.splice(offset, pageSize), total, pageNum: currentPage } });

}

const getList = ({ query, category, orderBy, orderHow }) => {

return new Promise((resolve, reject) => {

const sql =

`select book.\* from book

left join book\_category on book.ID = book\_category.bookID

${(query || category.length > 0) ? `where` : ''}

${(query) ? `( ISBN like '%${query}%' or bookname like '%${query}%' or author like '%${query}%' or press like '%${query}%' )` : ''}

${(query && category.length > 0) ? `and` : ''}

${(category.length > 0) ? `book\_category. book\_categoryID in (${category.join(',')})` : ''}

group by book.ID

${(category.length > 0) ? `having count(\*) = ${category.length}` : ''}

${(orderBy && orderHow) ? `order by ${orderBy} ${orderHow}` : ''}

`;

pool.query(sql, (err, result) => err ? reject(err) : resolve(result));

})

}

# 附录B

前台下单功能的前端相关代码：

const bookOrderModalVisible = ref(false);

const bookOrderList = ref([]);

const bookOrderTotalPrice = ref(0);

const fromShopcart = ref(false);

const showBookOrderModal = (bookList, from) => {

bookOrderList.value = bookList;

if (from === "fromShopcart") fromShopcart.value = true;

bookOrderModalVisible.value = true;

let totalPrice = 0;

for (const { num, price } of bookOrderList.value) {

totalPrice += num \* price;

}

bookOrderTotalPrice.value = totalPrice;

}

const orderBook = (selectDeliveryID) => {

if (!selectDeliveryID) return message.error("收货信息不能为空");

orderAPI

.add({ book: bookOrderList.value, deliveryID: selectDeliveryID, fromShopcart: fromShopcart.value })

.then(({ orderID }) => {

message.success("下单成功");

closeBookOrderModal();

showOrderPayModal(orderID);

})

.catch(() => {

message.error("下单失败");

});

}

add: (addForm) => {

return new Promise((resolve, reject) => {

axios

.post(`/order/add`, addForm)

.then(({ data: { status, msg, data } }) => {

if (status === 200) {

resolve(data);

} else {

reject(msg);

}

})

.catch(() => {

reject("接口调用失败");

});

});

},

前台下单功能的后端相关代码：

const useVerifyUser = async (req, res, next) => {

try {

await verifyUser(req.headers.authorization);

next();

} catch (err) {

res.json({ status: 401, msg: err });

}

}

router.post('/add', useVerifyUser, controller.add);

const add = async ({ headers: { authorization }, body }, res) => {

const userID = jwt.verify(authorization, "tokenSecret");

const { deliveryID, book, fromShopcart } = body;

if (typeof deliveryID !== "number" || !Array.isArray(book) || book.length === 0) {

return res.json({ status: 400, msg: "参数错误" });

}

try {

const orderID = await orderModel.add({ userID, deliveryID, book, fromShopcart });

res.json({ status: 200, msg: "成功", data: { orderID } });

} catch (err) {

res.json({ status: 500, msg: err });

}

}

const add = ({ userID, deliveryID, book, fromShopcart }) => {

return new Promise(async (resolve, reject) => {

const connection = await new Promise((resolve) => {

pool.getConnection((err, connection) => resolve(err ? false : connection));

});

if (!connection) return reject("连接数据库出错");

const transaction = await new Promise((resolve) => {

connection.beginTransaction((err) => resolve(err ? false : true));

});

if (!transaction) {

connection.release();

return reject("事务开始出错");

}

const [delivery] = await new Promise((resolve) => {

const sql = `select \* from delivery where delivery\_userID = ${userID} and ID = ${deliveryID}`;

console.log(sql);

connection.query(sql, (err, result) => resolve(err ? [] : result));

});

if (!delivery) {

connection.release();

return reject("获取收货信息出错");

}

let totalPrice = 0;

const bookList = [];

for (const { ID, num, price } of book) {

const [book] = await new Promise((resolve) => {

const sql = `select \* from book where ID = ${ID}`;

console.log(sql);

connection.query(sql, (err, result) => resolve(err ? [] : result));

});

if (!book && book.price !== price && book.amount < num) {

connection.release();

return reject("获取下单的图书信息出错");

}

else {

totalPrice += book.price \* num;

bookList.push({ ...book, num });

}

}

const { affectedRows = 0, insertId: orderID } = await new Promise((resolve) => {

const sql =

`insert into orders(orders\_userID, name, phone, address, totalPrice)

values (${userID}, '${delivery.name}', '${delivery.phone}', '${delivery.address}', ${totalPrice})`;

console.log(sql);

connection.query(sql, (err, result) => resolve(err ? {} : result));

});

if (affectedRows === 0) {

connection.release();

return reject("添加订单出错");

}

for (const { ID, ISBN, bookname, price, cover, num } of bookList) {

const { affectedRows = 0 } = await new Promise((resolve) => {

const sql =

`insert into order\_item(item\_orderID, bookID, ISBN, bookname, price, cover, num)

values (${orderID}, ${ID}, '${ISBN}', '${bookname}', ${price}, '${cover}', ${num})`;

console.log(sql);

connection.query(sql, (err, result) => resolve(err ? {} : result));

});

if (affectedRows === 0) {

return connection.rollback(() => {

connection.release();

reject("添加订单细则出错");

});

}

{

const { affectedRows = 0 } = await new Promise((resolve) => {

const sql = `update book set amount = amount - ${num} , sale = sale + ${num} where ID = ${ID} and amount >= ${num}`

console.log(sql);

connection.query(sql, (err, result) => resolve(err ? {} : result));

});

if (affectedRows === 0) {

return connection.rollback(() => {

connection.release();

reject("更新图书信息出错");

});

}

}

}

if (fromShopcart) {

const sql = `delete from shopcart where shopcart\_userID = ${userID}`

console.log(sql);

connection.query(sql);

}

return connection.commit(() => {

connection.release();

resolve(orderID);

});});}

# 附录C

后台图书添加功能的前端相关代码：

const addFormRef = ref(null);

const form = reactive({

ISBN: "",

bookname: "",

author: "",

press: "",

amount: "",

price: "",

cover: "",

category: [],

removedCategory: [],

removeCoverList: [],

});

const add = async () => {

await addFormRef.value.validateFields();

bookAPI

.add(form)

.then(() => {

$message.success("添加成功");

closeAddModal();

getDataList();

})

.catch((err) => {

$message.error(`添加失败,${err}`);

});

};

add: (addForm) => {

return new Promise((resolve, reject) => {

axios

.post(`/${baseUrl}/add`, addForm)

.then(({ data: { status, msg } }) => {

if (status === 200) {

resolve();

} else {

reject(msg);

}

})

.catch(() => {

reject("接口调用失败");

});

});

},

后台图书添加功能的后端相关代码：

const useVerifyAdmin = async ({ headers }, res, next) => {

try {

await verifyAdmin({ headers, requireRole: 3 });

next();

} catch (err) {

res.json({ status: 401, msg: err });

}

}

router.post('/add', useVerifyAdmin, controller.add);

const add = async ({ body }, res) => {

const { ISBN, bookname, author, press, amount, price, cover, category, removeCoverList } = body ?? {};

if (typeof ISBN !== 'string' || typeof bookname !== 'string' || typeof author !== 'string' || typeof press !== 'string'

|| typeof amount !== 'number' || typeof price !== 'number' || !Array.isArray(category) || !Array.isArray(removeCoverList)

) {

return res.json({ status: 400, msg: "参数错误" });

}

if (typeof cover === 'string' && cover !== '' && fs.existsSync(`tmp/${cover}`)) {

try {

fs.renameSync(`tmp/${cover}`, `public/cover/${cover}`);

} catch {

return res.json({ status: 500, msg: "失败" });

}

}

for (const cover of removeCoverList) {

typeof cover === 'string' && cover !== '' && fs.existsSync(`tmp/${cover}`) && fs.rmSync(`tmp/${cover}`);

}

try {

await bookModel.add({ ISBN, bookname, author, press, amount, price, cover, categoryList: category });

return res.json({ status: 200, msg: "成功" });

} catch {

typeof cover === 'string' && cover !== '' && fs.existsSync(`public/cover/${cover}`) && fs.rmSync(`public/cover/${cover}`);

return res.json({ status: 500, msg: "失败" });

}

}

const add = ({ ISBN, bookname, author, press, amount, price, cover, categoryList }) => {

return new Promise(async (resolve, reject) => {

const connection = await new Promise((resolve) =>

pool.getConnection((err, connection) => resolve(connection))

);

if (!connection) return reject();

const transaction = await new Promise((resolve) => {

connection.beginTransaction((err) => !err && resolve(true));

});

if (!transaction) return reject();

const { affectedRows, insertId } = await new Promise((resolve) => {

const sql = `insert into book ${NAME\_VALUES({ ISBN, bookname, author, press, amount, price, cover })}`;

console.log(sql);

connection.query(sql, (err, result) => resolve(err? {} : result));

});

if (affectedRows !== 1) {

return connection.rollback(() => {

connection.release();

reject();

});

}

for (const categoryID of categoryList) {

if (typeof categoryID !== 'number') continue;

const { affectedRows } = await new Promise((resolve) => {

const sql = `insert into book\_category ${NAME\_VALUES({ bookID: insertId, book\_categoryID: categoryID })}`;

console.log(sql);

connection.query(sql, (err, result) => resolve(err? {} : result));

});

if (affectedRows !== 1) {

return connection.rollback(() => {

connection.release();

reject();

});

}

}

connection.commit((err) => {

if (err) {

connection.rollback(() => {

connection.release();

reject();

});

} else {

connection.release();

resolve();

}

})

})

}