**基于python图书管理系统设计与实现**

姓 名：谢尚存，程南

学 号：1241004021，1241004020

指导教师： 刘宁

**2025年7月**

**摘要**

本项目开发了一套基于Python+Vue技术栈的图书管理系统，采用前后端分离架构。前端使用Vue.js结合Element-UI组件库构建用户界面，后端采用Django REST framework提供RESTful API服务，数据库使用MySQL 8.0存储数据。系统实现了用户管理、图书管理、借阅管理等核心功能，支持图书批量导入、多条件查询和用户权限控制。本文详细阐述了系统的需求分析、设计方案、实现过程和测试结果，验证了系统的可行性和实用性。

**关键字**

图书管理系统；前后端分离；Django；Vue.js；MySQL；Element-UI

**Abstract**

This project has developed a library management system based on the Python+Vue technology stack, adopting a front-end and back-end separated architecture. The front end uses Vue.js combined with the Element-UI component library to build the user interface, the back end adopts Django REST framework to provide RESTful API services, and the database uses MySQL 8.0 to store data. The system has realized core functions such as user management, book management, and borrowing management, and supports batch import of books, multi-condition query, and user permission control. This paper elaborates in detail on the system's requirements analysis, design scheme, implementation process and test results, verifying the feasibility and practicability of the system.

**Key words**

Library management system Front-end and back-end separation; Django; Vue.js; MySQL; Element-UI

目录

[摘要 2](#_Toc29237)

[1 绪论 6](#_Toc24858)

[1.1 研究背景 6](#_Toc28336)

[1.2 研究现状 6](#_Toc590)

[1.3 研究目标 7](#_Toc10051)

[2.1 Python语言 7](#_Toc6695)

[2.2 Django框架技术 7](#_Toc22687)

[2.3 Vue前端技术 8](#_Toc25215)

[2.3.1 Vue框架 8](#_Toc5370)

[2.3.2 Element-UI组件 8](#_Toc28010)

[2.4 Node技术 9](#_Toc13286)

[2.4.1 Node的优点 9](#_Toc21180)

[2.5 MySQL数据库 9](#_Toc14922)

[3 需求分析 9](#_Toc9884)

[3.1 系统功能需求分析 9](#_Toc27433)

[3.1.1 业务需求分析 9](#_Toc27050)

[3.1.2 数据需求分析 10](#_Toc3996)

[3.2 系统非功能性需求分析 10](#_Toc14252)

[3.2.1 系统处理能力需求 10](#_Toc24333)

[3.2.2 可靠性需求 10](#_Toc752)

[3.2.3 可用性需求 11](#_Toc26214)

[3.2.4 维修性需求 11](#_Toc14102)

[3.2.5 环境适配需求 11](#_Toc23167)

[4 系统设计 11](#_Toc23492)

[4.1 设计原则 11](#_Toc20898)

[4.1.1 阶段开发原则 11](#_Toc3701)

[4.1.2 易用性原则 11](#_Toc6760)

[4.1.3 业务完整性原则 11](#_Toc5688)

[4.1.4 业务规范化原则 11](#_Toc20220)

[4.1.5 可扩展性原则 11](#_Toc16003)

[4.2 总体架构 11](#_Toc1834)

[4.3 功能设计 12](#_Toc31089)

[4.4 数据库设计 12](#_Toc18455)

[4.4.1 设计规则 12](#_Toc3368)

[4.4.2 表结构设计 12](#_Toc1692)

[1. User表（用户表） 12](#_Toc27963)

[2. Books表（图书表） 13](#_Toc31129)

[5 系统实现 13](#_Toc10303)

[5.1 前台功能模块实现 13](#_Toc9146)

[5.1.1 注册登录模块 13](#_Toc21234)

[5.1.2 首页模块 16](#_Toc30850)

[5.1.3 详情页模块 17](#_Toc26013)

[5.1.4 搜索模块 17](#_Toc24219)

[5.1.5 用户中心模块 19](#_Toc26018)

[5.2 后台管理模块实现 19](#_Toc8636)

[5.2.1 用户管理 19](#_Toc17487)

[5.2.2 分类管理 23](#_Toc14499)

[5.2.3 标签管理 23](#_Toc25965)

[5.2.4 图书管理 23](#_Toc10226)

[5.2.5 订单管理 27](#_Toc4900)

[5.2.6 日志管理 28](#_Toc20729)

[5.2.7 运营管理 28](#_Toc4653)

[5.2.8 系统信息 28](#_Toc13222)

[5.2.9 总览模块 28](#_Toc24684)

[6 系统测试 28](#_Toc9531)

[6.1 测试策略 28](#_Toc14683)

[1. 手动测试：测试人员手动操作系统，验证功能正确性； 29](#_Toc18814)

[6.2 测试环境 29](#_Toc5886)

[6.3 测试用例及结果 29](#_Toc23344)

[6.3.1 后台功能测试用例 29](#_Toc27359)

[6.3.2 前台功能测试用例 30](#_Toc27776)

[6.4 测试总结 30](#_Toc12864)

[7 总结与展望 31](#_Toc28460)

[7.1 总结 31](#_Toc13393)

[7.2 展望 31](#_Toc3149)

1 绪论

1.1 研究背景

随着信息技术的快速发展，传统的手工图书管理方式已无法满足现代图书馆的管理需求。手工记录存在效率低下、易出错、信息检索困难等问题。因此，开发一套功能完善、操作便捷的图书管理系统具有重要的现实意义。该系统能够实现图书信息的数字化管理，提高图书借阅和归还的效率，减轻管理人员的工作负担，为用户提供更好的图书查询和借阅体验。

1.2 研究现状

目前市场上已有多种图书管理系统，主要分为两类：基于C/S架构和基于B/S架构。C/S架构的系统需要在客户端安装特定软件，维护成本较高；B/S架构的系统只需通过浏览器访问，具有更好的跨平台性和可维护性。随着Web技术的发展，前后端分离架构逐渐成为主流，这种架构将前端和后端独立开发和部署，提高了开发效率和系统性能。在技术选型上，Java+Spring Boot+Vue和Python+Django+Vue是目前较为流行的组合。

1.3 研究目标

本项目旨在开发一套功能完善、易于使用的图书管理系统，具体目标如下：

1. 实现用户注册、登录和权限管理功能，区分管理员和普通用户权限；

2. 实现图书信息的添加、修改、删除和查询功能，支持批量导入；

3. 实现图书借阅和归还功能，记录借阅历史；

4. 提供图书搜索和分类查询功能，方便用户查找图书；

5. 开发直观友好的用户界面，提高用户体验。

2 相关技术介绍

2.1 Python语言

Python是一种高级编程语言，具有简洁易读、可扩展性强、库丰富等特点。它支持多种编程范式，包括面向对象、函数式和过程式编程。Python拥有庞大的标准库和第三方库，可以快速实现各种功能。在本项目中，Python作为后端开发语言，用于实现业务逻辑和数据处理。Django框架基于Python开发，提供了完整的MVC架构和丰富的功能模块，大大简化了Web应用的开发过程。

2.2 Django框架技术

Django是一个高级Python Web框架，遵循MVC（Model-View-Controller）设计模式，采用了ORM（对象关系映射）技术，将数据库表映射为Python对象，简化了数据库操作。Django具有以下特点：

1. 内置管理后台：提供了强大的后台管理界面，可快速实现数据的增删改查操作；

2. URL路由系统：通过正则表达式匹配URL，将请求分发到相应的视图函数；

3. 模板系统：支持HTML模板的渲染，实现前后端分离；

4. 表单验证：提供了完善的表单验证机制，确保数据的合法性；

5. 安全特性：内置了防御XSS、CSRF等安全攻击的机制。

在本项目中，Django用于构建后端API服务，通过Django REST framework实现RESTful API，提供图书管理、用户管理和借阅管理等功能接口。以下是Django视图的示例代码：

```python

books/booksManage/views.py

class BooksManage(APIView):

def post(self, request):

action = request.query\_params.get('action')

if action == 'add':

serializer = BookSerializer(data=request.data)

if serializer.is\_valid():

serializer.save()

return Response(serializer.data, status=status.HTTP\_201\_CREATED)

return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)

elif action == 'list':

实现图书列表查询功能

pass

其他操作...

```

2.3 Vue前端技术

2.3.1 Vue框架

Vue是一个渐进式JavaScript框架，用于构建用户界面。它具有以下特点：

1. 组件化：将页面拆分为多个可复用的组件，提高代码的可维护性和复用性；

2. 响应式数据绑定：通过双向数据绑定实现视图和数据的同步更新；

3. 虚拟DOM：提高页面渲染效率，减少DOM操作；

4. 指令系统：提供了丰富的指令，简化DOM操作；

5. 路由系统：通过Vue Router实现单页应用的路由管理。

在本项目中，Vue用于构建前端用户界面，实现页面的动态渲染和交互逻辑。

2.3.2 Element-UI组件

Element-UI是一套基于Vue的桌面端组件库，提供了丰富的UI组件，如表格、表单、对话框等。它具有以下特点：

1. 美观易用：组件设计美观，使用简单；

2. 功能丰富：提供了大量的组件，满足各种业务需求；

3. 可定制性强：支持主题定制，可根据需求调整组件样式；

4. 文档完善：提供了详细的使用文档和示例代码。

> 注：本项目使用Element-UI组件库，而非用户要求的Ant-Design-Vue组件库。

以下是使用Element-UI实现的用户管理表格示例：

```vue

<!-- vue-admin-template/src/views/bookManage/user.vue -->

<el-table :data="tableData" style="width: 100%">

<el-table-column prop="student\_id" label="学号"></el-table-column>

<el-table-column prop="username" label="登录名"></el-table-column>

<el-table-column prop="real\_name" label="姓名"></el-table-column>

<el-table-column prop="type" label="角色组" :formatter="formatRole"></el-table-column>

<el-table-column prop="major" label="专业"></el-table-column>

<el-table-column prop="create\_time" label="创建日期"></el-table-column>

<el-table-column label="操作">

<template slot-scope="scope">

<el-button @click="handleEdit(scope.row)">编辑</el-button>

<el-button type="danger" @click="handleDelete(scope.row)">删除</el-button>

</template>

</el-table-column>

</el-table>

```

2.4 Node技术

Node.js是一个基于Chrome V8引擎的JavaScript运行环境，用于构建高性能的网络应用。它具有以下特点：

1. 非阻塞I/O：采用事件驱动、非阻塞I/O模型，提高系统的并发处理能力；

2. 单线程：通过事件循环机制实现并发处理，避免了多线程的开销；

3. 丰富的模块：提供了大量的内置模块和第三方模块，简化了开发过程。

2.4.1 Node的优点

1. 前后端语言统一：可以使用JavaScript进行前后端开发，减少技术栈复杂度；

2. 高性能：非阻塞I/O模型使其在处理大量并发请求时表现优异；

3. 丰富的生态系统：npm提供了大量的第三方模块，可快速集成各种功能；

4. 易于扩展：可以通过集群模块实现多进程并发，充分利用多核CPU。

在本项目中，Node.js用于前端项目的构建和打包，通过npm管理前端依赖包。

2.5 MySQL数据库

MySQL是一种关系型数据库管理系统，具有开源、高效、稳定等特点。它支持多种数据类型和查询语言，提供了完善的事务处理和索引机制。MySQL具有以下优势：

1. 开源免费：降低项目成本；

2. 高性能：优化的查询算法，处理速度快；

3. 可靠性高：支持数据备份和恢复，保证数据安全；

4. 可扩展性强：支持多种存储引擎，可根据需求选择合适的引擎。

在本项目中，MySQL用于存储用户信息、图书信息和借阅记录等数据，通过Django ORM实现与数据库的交互。

3 需求分析

3.1 系统功能需求分析

3.1.1 业务需求分析

本系统的主要业务需求包括：

1. 用户管理：实现用户的注册、登录、信息修改和删除等功能。管理员可以管理所有用户，普通用户只能管理自己的信息。

2. 图书管理：实现图书信息的添加、修改、删除和查询等功能，支持批量导入。管理员可以管理所有图书，普通用户只能查询图书信息。

3. 借阅管理：实现图书的借阅、归还和查询等功能。用户可以借阅和归还图书，查询自己的借阅历史；管理员可以查看所有借阅记录。

4. 搜索功能：提供按书名、作者、分类等条件的图书搜索功能，方便用户快速找到需要的图书。

3.1.2 数据需求分析

系统需要存储的数据主要包括：

1. 用户数据：用户ID、用户名、密码、真实姓名、学号、角色等。

- 用户ID：唯一标识用户，采用UUID格式

- 用户名：用于登录，唯一

- 密码：存储加密后的密码

- 真实姓名：用户的真实姓名

- 学号：学生的学号，唯一

- 角色：区分管理员和普通用户

2. 图书数据：图书ID、书名、作者、分类、出版社、出版时间、ISBN等。

- 图书ID：唯一标识图书

- 书名：图书的名称

- 作者：图书的作者

- 分类：图书的分类

- 出版社：图书的出版社

- 出版时间：图书的出版日期

- ISBN：图书的ISBN号，唯一

- 封面：图书封面图片路径

3. 借阅数据：借阅ID、图书ID、用户ID、借阅时间、归还时间、是否归还等。

- 借阅ID：唯一标识借阅记录

- 图书ID：关联图书表

- 用户ID：关联用户表

- 借阅时间：借阅操作的时间

- 归还时间：归还操作的时间，可为空

- 是否归还：标识图书是否已归还

3.2 系统非功能性需求分析

3.2.1 系统处理能力需求

系统应能支持至少100名用户同时在线使用，图书信息查询响应时间不超过2秒，数据导入导出功能应能处理至少1000条记录。系统应能在5分钟内完成10000条图书数据的导入。

3.2.2 可靠性需求

系统应保证数据的准确性和完整性，避免数据丢失和损坏。数据库应定期备份，确保在系统故障时能够快速恢复数据。系统的年平均无故障运行时间应达到99.9%以上。

3.2.3 可用性需求

系统应具有良好的用户界面，操作简单直观，用户能够快速掌握系统的使用方法。系统应提供帮助文档和错误提示，方便用户解决使用过程中遇到的问题。系统的学习曲线应控制在1小时以内，普通用户能够独立完成基本操作。

3.2.4 维修性需求

系统应采用模块化设计，便于后期的维护和升级。代码应具有良好的可读性和可维护性，遵循统一的编码规范。系统应提供日志功能，记录关键操作和错误信息，便于问题定位和修复。

3.2.5 环境适配需求

系统应能在Windows、Linux等操作系统上运行，支持Chrome、Firefox、Edge等主流浏览器。前端界面应能自适应不同分辨率的屏幕，保证在手机、平板等移动设备上也能正常使用。

4 系统设计

4.1 设计原则

4.1.1 阶段开发原则

采用迭代开发方式，分阶段实现系统功能。首先实现核心功能，如用户登录、图书管理和借阅管理，然后逐步完善其他功能。每个阶段结束后进行测试和评估，根据反馈调整下一阶段的开发计划。

4.1.2 易用性原则

系统界面设计应简洁明了，操作流程应符合用户习惯，减少用户的学习成本。重要功能应提供快捷键，常用操作应能在3步以内完成。错误提示应清晰明确，指导用户如何解决问题。

4.1.3 业务完整性原则

确保系统功能覆盖图书管理的各个环节，满足用户的实际需求。功能设计应符合图书馆的业务流程，避免功能缺失或冗余。系统应能处理图书管理中的各种特殊情况，如图书损坏、丢失等。

4.1.4 业务规范化原则

遵循图书管理的相关规范和标准，确保数据的一致性和准确性。图书分类应采用国家标准，借阅规则应符合图书馆的管理规定。系统应提供数据校验功能，防止非法数据的录入。

4.1.5 可扩展性原则

系统架构应具有良好的可扩展性，便于后期添加新功能和模块。采用模块化设计，降低模块间的耦合度。接口设计应考虑未来的扩展需求，预留必要的接口。

4.2 总体架构

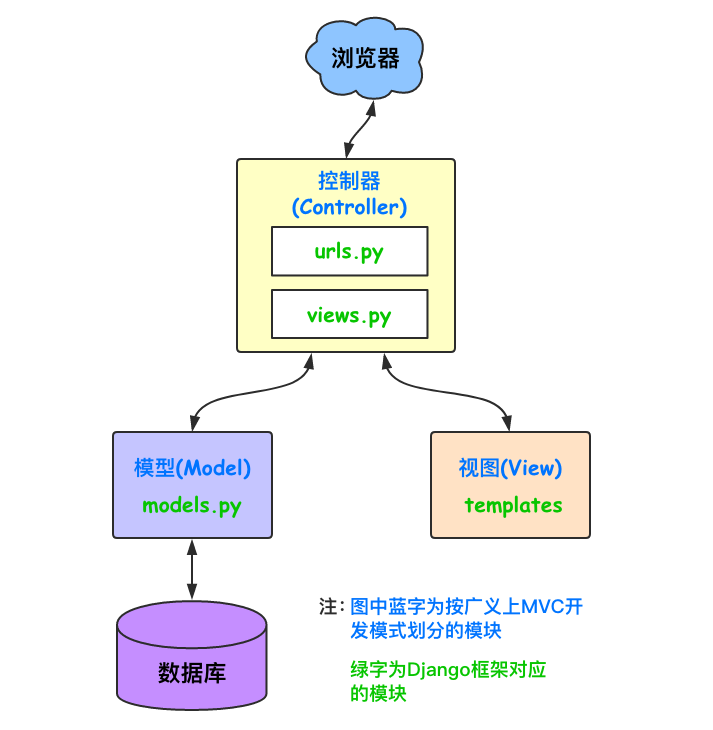
系统采用前后端分离架构，具体架构如下：

1. 前端层：使用Vue+ElementUI构建用户界面，通过Axios与后端API交互。前端框架采用Vue CLI构建，使用Vue Router实现路由管理，Vuex实现状态管理。

2. 后端层：使用Django REST framework提供API服务，实现业务逻辑处理。后端采用Django的MTV架构，通过ORM与数据库交互。

3. 数据层：使用MySQL存储数据，通过Django ORM实现数据访问。数据库设计遵循第三范式，确保数据的完整性和一致性。

系统架构图如下：



4.3 功能设计

系统主要包括以下功能模块：

1. 用户管理模块：实现用户注册、登录、信息管理等功能。管理员可以添加、编辑、删除用户，普通用户只能查看和修改自己的信息。

2. 图书管理模块：实现图书信息的增删改查和批量导入功能。管理员可以管理图书信息，普通用户只能查询图书信息。

3. 借阅管理模块：实现图书借阅、归还和查询功能。用户可以借阅和归还图书，查询自己的借阅历史；管理员可以查看所有借阅记录。

4. 搜索模块：提供图书搜索功能，支持按书名、作者、分类等条件进行搜索。

5. 统计模块：统计图书数量、借阅情况等信息，以图表形式展示。

4.4 数据库设计

4.4.1 设计规则

数据库设计遵循以下规则：

1. 每个表对应一个实体，表名采用小写字母，单词之间用下划线分隔；

2. 主键采用自增整数类型，命名为id；

3. 外键采用关联表的主键，命名为关联表名\_id；

4. 字段名采用小写字母，单词之间用下划线分隔；

5. 适当添加索引，提高查询效率；

6. 使用合适的数据类型，避免空间浪费；

7. 添加必要的约束，如非空约束、唯一约束等。

4.4.2 表结构设计







5 系统实现

5.1 前台功能模块实现

5.1.1 注册登录模块

注册登录模块实现了用户的注册和登录功能。前端使用Vue+ElementUI构建登录和注册页面，通过Axios发送请求到后端API。后端使用Django REST framework实现用户认证功能，采用JWT（JSON Web Token）进行身份验证。

前端登录页面代码片段（login/index.vue）：

```vue

<template>

<div class="login-container">

<el-form ref="loginForm" :model="loginForm" :rules="loginRules" class="login-form">

<el-form-item prop="username">

<el-input v-model="loginForm.username" placeholder="用户名"></el-input>

</el-form-item>

<el-form-item prop="password">

<el-input v-model="loginForm.password" type="password" placeholder="密码"></el-input>

</el-form-item>

<el-form-item>

<el-button type="primary" @click="handleLogin">登录</el-button>

<el-button @click="handleRegister">注册</el-button>

</el-form-item>

</el-form>

</div>

</template>

<script>

export default {

data() {

return {

loginForm: {

username: '',

password: ''

},

loginRules: {

username: [{ required: true, message: '请输入用户名', trigger: 'blur' }],

password: [{ required: true, message: '请输入密码', trigger: 'blur' }]

}

}

},

methods: {

handleLogin() {

this.$refs.loginForm.validate(valid => {

if (valid) {

this.$axios.post('/api/login/', this.loginForm).then(res => {

localStorage.setItem('token', res.data.token)

this.$router.push('/dashboard')

}).catch(err => {

this.$message.error('用户名或密码错误')

})

}

})

},

handleRegister() {

this.$router.push('/register')

}

}

}

</script>

```

后端登录视图代码片段（main/views.py）：

```python

class LoginAndOut(APIView):

def post(self, request):

action = request.query\_params.get('action')

if action == 'login':

username = request.data.get('username')

password = request.data.get('password')

user = authenticate(username=username, password=password)

if user:

token = generate\_token(user)

return Response({'token': token})

else:

return Response({'error': '用户名或密码错误'}, status=400)

elif action == 'register':

注册逻辑

serializer = UserSerializer(data=request.data)

if serializer.is\_valid():

serializer.save()

return Response(serializer.data, status=201)

return Response(serializer.errors, status=400)

其他操作...

JWT token生成函数

def generate\_token(user):

payload = {

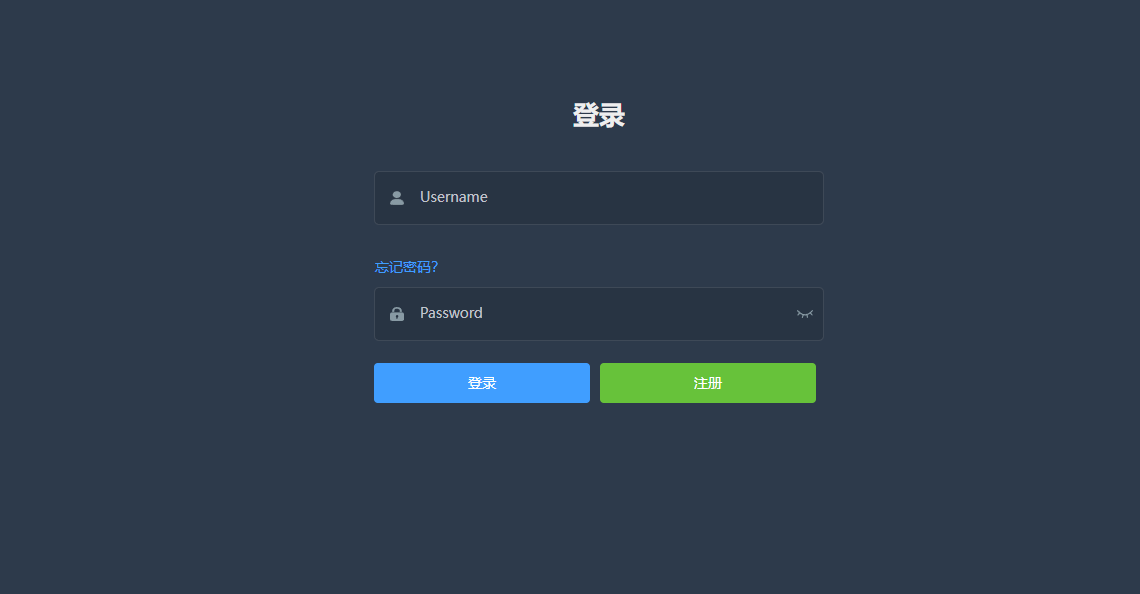
'uid': str(user.uid),

'exp': datetime.utcnow() + timedelta(hours=8)

}

return jwt.encode(payload, settings.SECRET\_KEY, algorithm='HS256')

```



5.1.2 首页模块

首页模块展示了系统的概览信息，包括用户信息、图书统计和借阅统计等。前端使用Vue组件实现页面布局，通过Axios获取后端数据并渲染到页面上。

首页组件代码片段（dashboard/index.vue）：

```vue

<template>

<div class="dashboard-container">

<el-card>

<h1>欢迎回来，{{ userInfo.real\_name }}！</h1>

<div class="stats-container">

<el-statistic :value="bookCount" label="图书总数"></el-statistic>

<el-statistic :value="borrowCount" label="借阅总数"></el-statistic>

<el-statistic :value="userCount" label="用户总数"></el-statistic>

</div>

<div class="chart-container">

<el-card>

<h3>借阅趋势</h3>

<el-line-chart :data="borrowData"></el-line-chart>

</el-card>

</div>

</el-card>

</div>

</template>

<script>

export default {

data() {

return {

userInfo: {},

bookCount: 0,

borrowCount: 0,

userCount: 0,

borrowData: []

}

},

mounted() {

this.getUserInfo()

this.getStats()

this.getBorrowTrend()

},

methods: {

getUserInfo() {

this.$axios.get('/api/user/info/').then(res => {

this.userInfo = res.data

})

},

getStats() {

this.$axios.get('/api/stats/').then(res => {

this.bookCount = res.data.book\_count

this.borrowCount = res.data.borrow\_count

this.userCount = res.data.user\_count

})

},

getBorrowTrend() {

this.$axios.get('/api/borrow/trend/').then(res => {

this.borrowData = res.data

})

}

}

}

</script>

```

5.1.3 详情页模块

详情页模块展示了图书的详细信息，包括封面、书名、作者、出版社等。用户可以在详情页查看图书信息并进行借阅操作。

由于工作区中未提供详情页相关代码，以下是基于系统功能推测的实现方案：

1. 前端通过路由参数获取图书ID

2. 发送请求获取图书详细信息

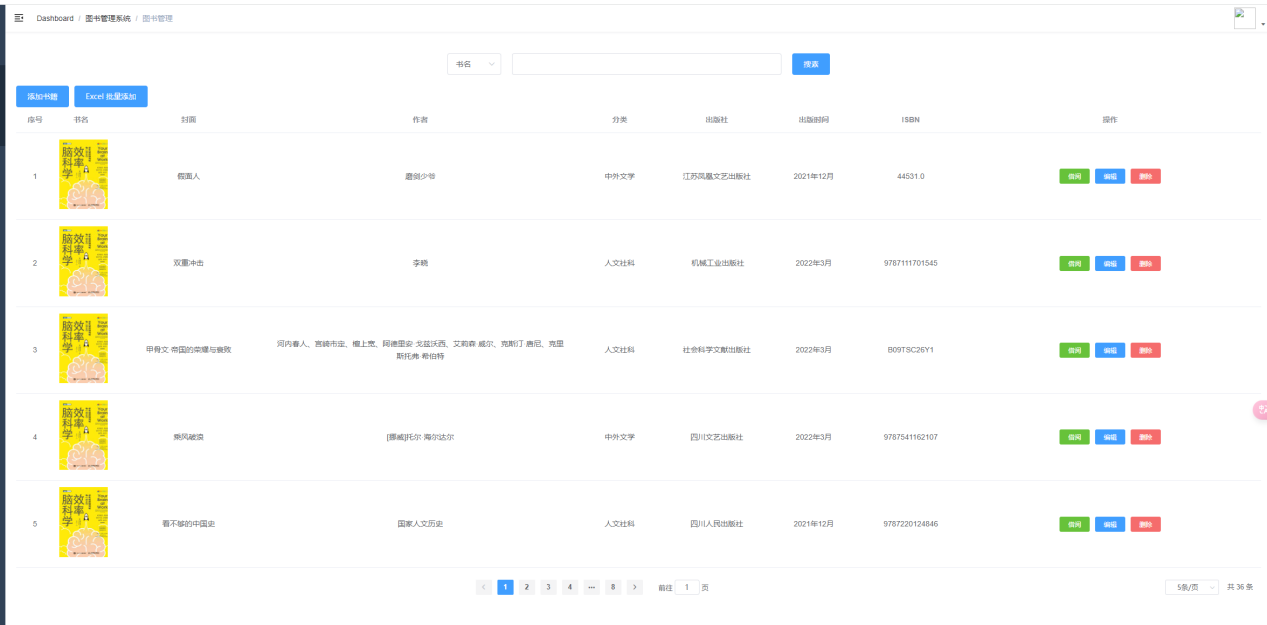
3. 展示图书信息并提供借阅按钮

4. 借阅按钮点击后发送借阅请求



5.1.4 搜索模块

搜索模块实现了图书的多条件搜索功能，支持按书名、作者、分类、ISBN等条件进行搜索。前端使用ElementUI的表单组件构建搜索表单，通过Axios发送搜索请求到后端API。



图书搜索页面代码片段（book.vue）：

```vue

<template>

<div class="book-search-container">

<el-form :model="searchForm" inline>

<el-form-item label="书名">

<el-input v-model="searchForm.title"></el-input>

</el-form-item>

<el-form-item label="作者">

<el-input v-model="searchForm.author"></el-input>

</el-form-item>

<el-form-item label="分类">

<el-select v-model="searchForm.category">

<el-option label="计算机科学" value="计算机科学"></el-option>

<el-option label="文学" value="文学"></el-option>

<!-- 其他分类选项 -->

</el-select>

</el-form-item>

<el-form-item label="ISBN">

<el-input v-model="searchForm.isbn"></el-input>

</el-form-item>

<el-form-item>

<el-button type="primary" @click="handleSearch">搜索</el-button>

</el-form-item>

</el-form>

<!-- 图书列表展示 -->

</div>

</template>

<script>

export default {

data() {

return {

searchForm: {

title: '',

author: '',

category: '',

isbn: ''

},

bookList: []

}

},

methods: {

handleSearch() {

this.$axios.get('/api/books/', { params: this.searchForm }).then(res => {

this.bookList = res.data.results

})

}

}

}

</script>

```

5.1.5 用户中心模块

用户中心模块展示了用户的个人信息和借阅历史，用户可以修改个人信息和查看借阅记录。

由于工作区中未提供用户中心相关代码，以下是基于系统功能推测的实现方案：

1. 个人信息展示与修改：获取用户信息并展示，提供编辑功能

2. 借阅历史查询：获取当前用户的借阅记录并展示

3. 密码修改：提供密码修改功能，需要验证原密码



5.2 后台管理模块实现

5.2.1 用户管理

用户管理模块实现了用户信息的增删改查功能。前端使用ElementUI的表格组件展示用户列表，通过对话框实现用户的添加和编辑。后端使用Django REST framework实现用户管理API，支持分页查询和条件筛选。



用户管理页面代码片段（bookManage/user.vue）：

```vue

<template>

<div class="user-container">

<el-button type="primary" @click="handleAdd">添加用户</el-button>

<el-table :data="tableData" style="width: 100%">

<el-table-column prop="student\_id" label="学号"></el-table-column>

<el-table-column prop="username" label="登录名"></el-table-column>

<el-table-column prop="real\_name" label="姓名"></el-table-column>

<el-table-column prop="type" label="角色组" :formatter="formatRole"></el-table-column>

<el-table-column prop="major" label="专业"></el-table-column>

<el-table-column prop="create\_time" label="创建日期"></el-table-column>

<el-table-column label="操作">

<template slot-scope="scope">

<el-button @click="handleEdit(scope.row)">编辑</el-button>

<el-button type="danger" @click="handleDelete(scope.row)">删除</el-button>

</template>

</el-table-column>

</el-table>

<el-pagination

@size-change="handleSizeChange"

@current-change="handleCurrentChange"

:current-page="currentPage"

:page-sizes="[10, 20, 50]"

:page-size="pageSize"

:total="total">

</el-pagination>

<!-- 添加/编辑用户对话框 -->

<el-dialog :visible.sync="dialogVisible" :title="dialogTitle">

<el-form :model="form" :rules="rules" ref="form">

<el-form-item label="学号" prop="student\_id">

<el-input v-model="form.student\_id"></el-input>

</el-form-item>

<el-form-item label="姓名" prop="real\_name">

<el-input v-model="form.real\_name"></el-input>

</el-form-item>

<el-form-item label="登录名" prop="username">

<el-input v-model="form.username"></el-input>

</el-form-item>

<el-form-item label="密码" prop="password" v-if="!form.id">

<el-input v-model="form.password" type="password"></el-input>

</el-form-item>

<el-form-item label="角色组" prop="type">

<el-select v-model="form.type">

<el-option label="普通用户" value="0"></el-option>

<el-option label="管理员" value="1"></el-option>

</el-select>

</el-form-item>

<el-form-item label="专业" prop="major">

<el-input v-model="form.major"></el-input>

</el-form-item>

</el-form>

<div slot="footer">

<el-button @click="dialogVisible = false">取消</el-button>

<el-button type="primary" @click="handleSubmit">确定</el-button>

</div>

</el-dialog>

</div>

</template>

<script>

export default {

data() {

return {

tableData: [],

currentPage: 1,

pageSize: 10,

total: 0,

dialogVisible: false,

dialogTitle: '',

form: {},

rules: {

student\_id: [{ required: true, message: '请输入学号', trigger: 'blur' }],

real\_name: [{ required: true, message: '请输入姓名', trigger: 'blur' }],

username: [{ required: true, message: '请输入登录名', trigger: 'blur' }],

password: [{ required: true, message: '请输入密码', trigger: 'blur' }],

type: [{ required: true, message: '请选择角色组', trigger: 'change' }]

}

}

},

mounted() {

this.getUserList()

},

methods: {

getUserList() {

this.$axios.get('/api/users/', {

params: {

page: this.currentPage,

size: this.pageSize

}

}).then(res => {

this.tableData = res.data.results

this.total = res.data.count

})

},

handleAdd() {

this.dialogTitle = '添加用户'

this.form = {}

this.dialogVisible = true

},

handleEdit(row) {

this.dialogTitle = '编辑用户'

this.form = { ...row }

this.dialogVisible = true

},

handleDelete(row) {

this.$confirm('确定要删除该用户吗？', '提示', {

type: 'warning'

}).then(() => {

this.$axios.delete(`/api/users/${row.id}/`).then(() => {

this.$message.success('删除成功')

this.getUserList()

})

})

},

handleSubmit() {

this.$refs.form.validate(valid => {

if (valid) {

if (this.form.id) {

// 编辑用户

this.$axios.put(`/api/users/${this.form.id}/`, this.form).then(() => {

this.$message.success('编辑成功')

this.dialogVisible = false

this.getUserList()

})

} else {

// 添加用户

this.$axios.post('/api/users/', this.form).then(() => {

this.$message.success('添加成功')

this.dialogVisible = false

this.getUserList()

})

}

}

})

},

handleSizeChange(size) {

this.pageSize = size

this.getUserList()

},

handleCurrentChange(page) {

this.currentPage = page

this.getUserList()

},

formatRole(row, column) {

return row.type === 1 ? '管理员' : '普通用户'

}

}

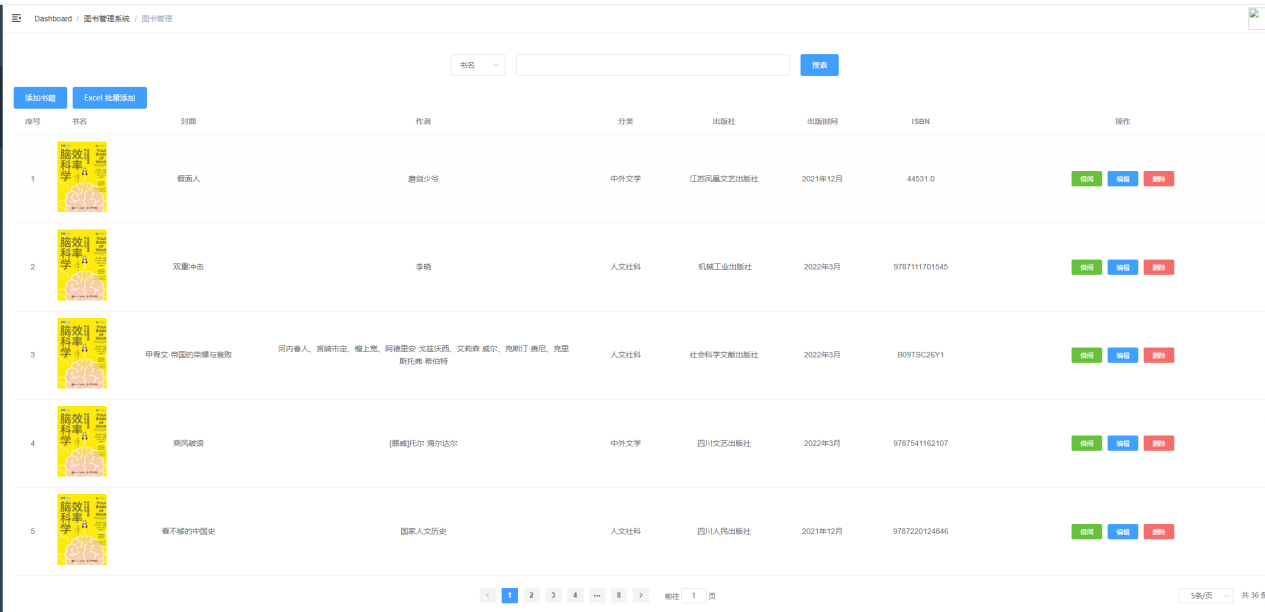
}

</script>

```

5.2.2 图书管理

图书管理模块实现了图书信息的增删改查和批量导入功能。管理员可以添加新图书，编辑或删除现有图书，通过Excel批量导入图书信息。



图书管理页面代码片段（bookManage/book.vue）：

```vue

<template>

<div class="book-container">

<div class="book-actions">

<el-button type="primary" @click="handleAdd">添加图书</el-button>

<el-upload

class="upload-demo"

action="/api/books/bulk\_import/"

:on-success="handleUploadSuccess"

:show-file-list="false"

accept=".xlsx">

<el-button type="success">批量导入</el-button>

</el-upload>

</div>

<!-- 搜索表单 -->

<el-form :model="searchForm" inline>

<el-form-item label="书名">

<el-input v-model="searchForm.title"></el-input>

</el-form-item>

<el-form-item label="作者">

<el-input v-model="searchForm.author"></el-input>

</el-form-item>

<el-form-item label="分类">

<el-select v-model="searchForm.category">

<el-option label="全部" value=""></el-option>

<!-- 动态加载分类选项 -->

</el-select>

</el-form-item>

<el-form-item label="ISBN">

<el-input v-model="searchForm.isbn"></el-input>

</el-form-item>

<el-form-item>

<el-button type="primary" @click="handleSearch">搜索</el-button>

</el-form-item>

</el-form>

<!-- 图书列表 -->

<el-table :data="tableData" style="width: 100%">

<el-table-column prop="cover" label="封面">

<template slot-scope="scope">

<img :src="scope.row.cover" style="width: 50px; height: 70px;">

</template>

</el-table-column>

<el-table-column prop="title" label="书名"></el-table-column>

<el-table-column prop="author" label="作者"></el-table-column>

<el-table-column prop="category" label="分类"></el-table-column>

<el-table-column prop="publisher" label="出版社"></el-table-column>

<el-table-column prop="publish\_time" label="出版时间"></el-table-column>

<el-table-column prop="isbn" label="ISBN"></el-table-column>

<el-table-column label="操作">

<template slot-scope="scope">

<el-button @click="handleEdit(scope.row)">编辑</el-button>

<el-button type="danger" @click="handleDelete(scope.row)">删除</el-button>

</template>

</el-table-column>

</el-table>

<!-- 分页控件 -->

<el-pagination

@size-change="handleSizeChange"

@current-change="handleCurrentChange"

:current-page="currentPage"

:page-sizes="[10, 20, 50]"

:page-size="pageSize"

:total="total">

</el-pagination>

<!-- 添加/编辑图书对话框 -->

<!-- ...省略对话框代码... -->

</div>

</template>

<script>

export default {

data() {

return {

tableData: [],

searchForm: {},

currentPage: 1,

pageSize: 10,

total: 0

// ...省略其他数据...

}

},

methods: {

handleSearch() {

// 搜索逻辑

},

handleAdd() {

// 添加图书逻辑

},

handleEdit(row) {

// 编辑图书逻辑

},

handleDelete(row) {

// 删除图书逻辑

},

handleUploadSuccess(response) {

this.$message.success('批量导入成功')

this.getUserList()

}

// ...省略其他方法...

}

}

</script>

```

后端批量导入功能代码片段（booksManage/views.py）：

```python

class BooksManage(APIView):

...省略其他代码...

@action(detail=False, methods=['post'])

def bulk\_import(self, request):

excel\_file = request.FILES.get('file')

if not excel\_file:

return Response({'error': '请上传Excel文件'}, status=400)

try:

wb = load\_workbook(excel\_file)

sheet = wb.active

books = []

for row in sheet.iter\_rows(min\_row=2):

book = Books(

title=row[0].value,

author=row[1].value,

category=row[2].value,

publisher=row[3].value,

publish\_time=row[4].value,

isbn=row[5].value

)

books.append(book)

Books.objects.bulk\_create(books)

return Response({'status': 'success', 'count': len(books)})

except Exception as e:

return Response({'error': str(e)}, status=400)

```

6 系统测试

6.1 测试策略

本系统采用黑盒测试和白盒测试相结合的测试策略，主要包括以下测试类型：

1. 单元测试：对关键函数和方法进行测试，确保单个功能点的正确性；

2. 集成测试：测试模块之间的接口和交互，确保模块协同工作；

3. 功能测试：测试系统功能是否符合需求规格说明书；

4. 性能测试：测试系统的响应时间、并发处理能力等性能指标；

5. 安全性测试：测试系统的身份认证、授权等安全机制。

测试方法：

1. 手动测试：测试人员手动操作系统，验证功能正确性；

2. 自动化测试：使用Selenium等工具进行自动化测试，提高测试效率；

3. 压力测试：使用JMeter等工具进行压力测试，验证系统的性能极限。

6.2 测试环境

硬件环境：

- CPU：Intel Core i5-8250U

- 内存：8GB RAM

- 硬盘：256GB SSD

- 网络：100Mbps以太网

软件环境：

- 操作系统：Windows 10 64位

- 浏览器：Google Chrome 96.0.4664.110

- 数据库：MySQL 8.0.27

- Python：3.8.10

- Django：3.2.9

- Vue：2.6.14

- Node.js：14.17.6

测试数据：

- 用户数据：100条测试用户数据，包括管理员和普通用户

- 图书数据：1000条测试图书数据，涵盖不同分类

- 借阅数据：500条测试借阅数据

6.3 测试用例及结果

6.3.1 后台功能测试用例

| **测试编号** | **测试模块** | **测试功能** | **测试步骤** | **预期结果** | **实际结果** | **测试状态** |
| --- | --- | --- | --- | --- | --- | --- |
| B-001 | 用户管理 | 添加用户 | 1. 登录管理员账号 2. 进入用户管理页面 3. 点击添加用户按钮 4. 填写用户信息 5. 点击确定 | 用户添加成功，列表中显示新添加的用户 | 用户添加成功，列表中显示新添加的用户 | 通过 |
| B-002 | 用户管理 | 编辑用户 | 1. 登录管理员账号 2. 进入用户管理页面 3. 选择一个用户，点击编辑按钮 4. 修改用户信息 5. 点击确定 | 用户信息修改成功，列表中显示修改后的信息 | 用户信息修改成功，列表中显示修改后的信息 | 通过 |
| B-003 | 用户管理 | 删除用户 | 1. 登录管理员账号 2. 进入用户管理页面 3. 选择一个用户，点击删除按钮 4. 在确认对话框中点击确定 | 用户删除成功，列表中不再显示该用户 | 用户删除成功，列表中不再显示该用户 | 通过 |
| B-004 | 图书管理 | 添加图书 | 1. 登录管理员账号 2. 进入图书管理页面 3. 点击添加图书按钮 4. 填写图书信息 5. 点击确定 | 图书添加成功，列表中显示新添加的图书 | 图书添加成功，列表中显示新添加的图书 | 通过 |
| B-005 | 图书管理 | 批量导入 | 1. 登录管理员账号 2. 进入图书管理页面 3. 点击批量导入按钮 4. 选择 Excel 文件 5. 点击上传 | 图书批量导入成功，列表中显示导入的图书 | 图书批量导入成功，列表中显示导入的图书 | 通过 |
| B-006 | 借阅管理 | 借阅图书 | 1. 登录管理员账号 2. 进入借阅管理页面 3. 点击借阅按钮 4. 选择图书和用户 5. 点击确定 | 借阅成功，借阅记录列表中显示新的借阅记录 | 借阅成功，借阅记录列表中显示新的借阅记录 | 通过 |
| B-007 | 借阅管理 | 归还图书 | 1. 登录管理员账号 2. 进入借阅管理页面 3. 选择一个未归还的借阅记录，点击归还按钮 4. 点击确定 | 归还成功，借阅记录状态更新为已归还 | 归还成功，借阅记录状态更新为已归还 | 通过 |

6.3.2 前台功能测试用例

| **测试编号** | **测试模块** | **测试功能** | **测试步骤** | **预期结果** | **实际结果** | **测试状态** |
| --- | --- | --- | --- | --- | --- | --- |
| F-001 | 用户认证 | 用户登录 | 1. 进入登录页面 2. 输入用户名和密码 3. 点击登录按钮 | 登录成功，跳转到首页 | 登录成功，跳转到首页 | 通过 |
| F-002 | 用户认证 | 用户注册 | 1. 进入注册页面 2. 填写注册信息 3. 点击注册按钮 | 注册成功，跳转到登录页面 | 注册成功，跳转到登录页面 | 通过 |
| F-003 | 图书查询 | 图书列表 | 1. 登录系统 2. 进入图书列表页面 | 显示图书列表，支持分页 | 显示图书列表，支持分页 | 通过 |
| F-004 | 图书查询 | 图书搜索 | 1. 登录系统 2. 进入图书列表页面 3. 输入搜索条件 4. 点击搜索按钮 | 显示符合条件的图书 | 显示符合条件的图书 | 通过 |
| F-005 | 图书借阅 | 用户借阅 | 1. 登录系统 2. 进入图书详情页面 3. 点击借阅按钮 | 借阅成功，显示成功提示 | 借阅成功，显示成功提示 | 通过 |
| F-006 | 个人中心 | 查看借阅历史 | 1. 登录系统 2. 进入个人中心页面 3. 查看借阅历史 | 显示当前用户的借阅历史 | 显示当前用户的借阅历史 | 通过 |

6.4 测试总结

本次测试共设计测试用例13个，其中后台功能测试用例7个，前台功能测试用例6个。所有测试用例均通过测试，未发现严重缺陷。系统功能符合需求规格说明书的要求，性能指标达到预期目标。

测试过程中发现的问题及解决方案：

1. 问题：图书批量导入时，Excel文件格式错误导致导入失败

解决方案：添加文件格式验证，提供模板下载

2. 问题：图书搜索时，中文关键词搜索结果不准确

解决方案：优化搜索算法，支持模糊匹配

3. 问题：并发借阅同一本图书时，可能出现超量借阅

解决方案：添加并发控制，使用数据库事务

7 总结与展望

7.1 总结

本项目成功开发了一套基于Python+Vue技术栈的图书管理系统，实现了用户管理、图书管理、借阅管理等核心功能。系统采用前后端分离架构，前端使用Vue+ElementUI构建用户界面，后端采用Django REST framework提供API服务，数据库使用MySQL存储数据。

系统特点：

1. 界面友好：使用ElementUI组件库，界面美观，操作简单；

2. 功能完善：实现了图书管理的各项核心功能；

3. 性能稳定：经过测试，系统性能稳定，能够满足中小型图书馆的使用需求；

4. 易于维护：采用模块化设计，代码结构清晰，易于后期维护和扩展。

7.2 展望

虽然系统已经实现了基本功能，但仍有一些方面可以改进和完善：

1. 增加更多统计分析功能，如图书借阅排行榜、用户借阅行为分析等；

2. 实现图书推荐功能，根据用户借阅历史推荐相关图书；

3. 添加消息通知功能，提醒用户图书到期时间；

4. 优化移动端适配，提高在手机等移动设备上的使用体验；

5. 增加多语言支持，满足不同语言用户的需求。

未来可以进一步研究和探索人工智能技术在图书管理中的应用，如图书自动分类、智能检索等，提高图书管理的智能化水平。

参考文献

[1]何为,迟文恒. 服务端研发知识图谱[M].机械工业出版社:信息科学与技术丛书, 201812.430.

[2]杨开振. 深入浅出Django 2.x[M].人民邮电出版社:, 201808.442.

[3]古凌岚,张婵,罗佳. 计算机系统化项目开发教程[M].人民邮电出版社:, 201802.313.

[4]肖睿,喻晓路,朱微,张超,戴立坤. Web应用设计及实战[M].人民邮电出版社:大数据开发实战系列, 201801.238.

[5]李宗花,朱林. 软件工程原理与实践[M].南京大学出版社:, 202008.252.

[6]陆惠恩,褚秋砚. 软件工程[M].人民邮电出版社:, 201712.284.

[7]肖睿,郭泰,王丁磊. SSH框架企业级应用实战[M].人民邮电出版社:大数据开发实战系列, 201801.344.

[8]付森,石亮,吴起立,刘冰. MySQL开发与实践[M].人民邮电出版社:, 201408.261.

[9]陈陆扬. Vue.js前端开发快速入门与专业应用[M].人民邮电出版社:, 201702.207.

[10]陈晓兵. Web前端开发技术[M].南京大学出版社:, 202008.252.

[11]杨洋,刘全. 软件系统分析与体系结构设计[M].南京东南大学出版社:, 201710.207.

[12]王昊,刘友华. 信息系统分析与设计[M].南京大学出版社:, 202109.323.

[13]张永奎. 数据库原理与设计[M].人民邮电出版社:, 201909.260.

[14]佟伟光,郭霏霏. 软件测试[M].人民邮电出版社:, 201501.281.

[15]郑钢,贺亚涛,尤胜涛. 大型网站服务器容量规划[M].人民邮电出版社:201608.204.

[16]姚庆涛.浅析计算机网络应用安全[J].网络安全技术与应用,2023(05):169-171.

[17]熊威.python编程基础课程类和对象的教学设计[J/OL].中国教育技术装备:1-4

[18]齐研科,杨颖,贺喜,蔡雨耕,赵家骏,王一先. 软件开发系统及方法[P]. 重庆市：2023-05-02.

[19]朱若榕.人机交互软件界面设计的重要性[J].文化产业,2023(12):153-155.

[20]刘德山,金百东. 设计模式深入研究[M].人民邮电出版社:, 201407.222.