华北科技学院

**基于SpringBoot + Vue的图书管理系统的设计与实现**

学 院： 计算机学院

专 业 名 称： 计算机科学与技术

小 组 成 员： 孟畅、张雨彤、王茜、肖文飞、李嘉欣、侯依涵、李雯、马鸿媛、王可佳

完成日期：2025年 7月9日

目录

[1 引言 3](#_Toc202978369)

[1.1 编写目的 3](#_Toc202978370)

[1.2 项目背景 3](#_Toc202978371)

[2 总体设计 3](#_Toc202978372)

[2.1 需求概述 3](#_Toc202978373)

[2.2 软件结构 4](#_Toc202978374)

[3 模块设计 4](#_Toc202978375)

[3.1 功能概述 4](#_Toc202978376)

[3.2 模块处理逻辑 5](#_Toc202978377)

[4 数据库设计 9](#_Toc202978378)

[4.1 ER图表 9](#_Toc202978379)

[4.2 表设计 9](#_Toc202978380)

[4.2.1 图书信息 9](#_Toc202978381)

[4.2.2 图书类型 10](#_Toc202978382)

[4.2.3 借阅信息 10](#_Toc202978383)

[4.2.4 用户信息 10](#_Toc202978384)

[5 接口设计 11](#_Toc202978385)

[5.1 外部接口 11](#_Toc202978386)

[5.1.1 登录界面 11](#_Toc202978387)

[5.1.2首页 11](#_Toc202978388)

[5.1.3 用户管理界面 12](#_Toc202978389)

[5.1.4 图书列表界面 12](#_Toc202978390)

[5.1.5 图书类型列表界面 13](#_Toc202978391)

[5.1.6 图书借阅管理界面 13](#_Toc202978392)

[5.2 内部接口 13](#_Toc202978393)

[5.2.1 登录接口 13](#_Toc202978394)

[5.2.2 注册接口 14](#_Toc202978395)

[5.2.3 图书管理接口 14](#_Toc202978396)

[5.2.4 图书类型管理接口 15](#_Toc202978397)

[5.2.5 图书借阅管理接口 17](#_Toc202978398)

[5.2.6 用户管理接口 20](#_Toc202978399)

[6 性能 22](#_Toc202978400)

[6.1 精度 22](#_Toc202978401)

[6.2 时间特性 22](#_Toc202978402)

[6.3 灵活性 22](#_Toc202978403)

[7 测试 22](#_Toc202978404)

# 1 引言

## 1.1 编写目的

本报告的目的是对图书管理系统进行详细设计说明，以便用户及项目开发人员了解产品详细的设计与实现，为开发人员提供开发参考书。以下叙述将结合文字描述、伪代码，图表等来描述图书管理系统的详细设计和相关的模块描述。本报告的预期读者有客户、项目经理、开发人员以及跟该项目相关的其他人员。

## 1.2 项目背景

高校图书管理系统由孟畅、张雨彤、王茜、肖文飞、李嘉欣、侯依涵、李雯、马鸿媛、王可佳进行开发和实施工作，该项目适用于图书馆的管理人员和读者。

# 2 总体设计

## 2.1 需求概述

(1) 图书信息的管理，包括图书信息的条件查询、录入、修改和删除。

(2) 图书类型的管理，包括建立图书类型、查询图书类型、修改图书类型和删除图书类型。

(3) 用户信息管理。包括对系统管理员和读者的信息进行维护管理，可以新增、修改、删除和条件分页查询用户信息。

(4) 图书借阅与归还，记录图书借阅与归还信息，能判断该书是否可借，供读者选择图书并借阅，对借阅的图书执行归还操作。

(5) 用户注册和登录，读者可以通过登录页面注册新账号，管理员和读者可以使用用户名和密码登录图书管理系统。

## 2.2 软件结构

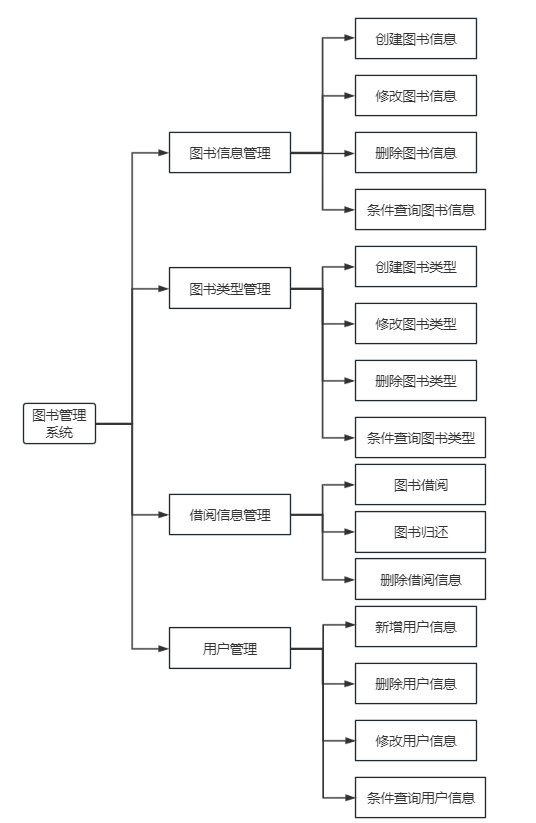


        图1 高校图书管理系统的软件结构图

描述：图书管理系统向下划分为图书管理、图书类型管理、用户信息管理和借阅管理四个模块。图书管理主要负责对图书的信息进行管理，以及管理员进行图书的查询；图书类型管理主要关联图书信息进行分类，可以对类型进行管理，以及对类型进行条件查询；借阅管理主要负责读者进行借书、还书操作，以及查询读者历史借阅信息；用户信息管理可以针对图书管理员和读者的信息进行管理。

# 3 模块设计

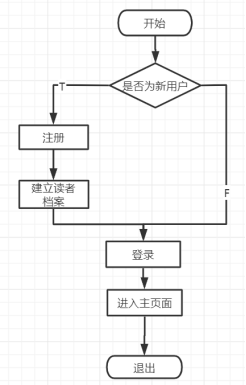
## 3.1 功能概述

表1 功能描述

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 模块 | 功能描述 | 操作角色 |
| 1 | 读者管理 | 建立读者档案 | 管理员 |
| 2 | 读者管理 | 建立新的读者信息 | 管理员 |
| 3 | 读者管理 | 修改读者信息 | 管理员 |
| 4 | 读者管理 | 读者修改个人信息 | 读者 |
| 5 | 读者管理 | 注销读者信息 | 管理员 |
| 6 | 读者管理 | 读者注销个人信息 | 读者 |
| 7 | 读者管理 | 发送借阅信息 | 系统 |
| 8 | 读者管理 | 查询图书信息 | 读者 |
| 9 | 图书管理 | 录入图书信息 | 管理员 |
| 10 | 图书管理 | 管理员录入图书信息 | 管理员 |
| 11 | 图书管理 | 修改图书信息 | 管理员 |
| 12 | 图书管理 | 管理员修改图书部分信息 | 管理员 |
| 13 | 图书管理 | 删除图书信息 | 管理员 |
| 14 | 图书管理 | 管理员删除图书全部信息 | 管理员 |
| 15 | 图书管理 | 查询图书信息 | 读者/管理员 |
| 16 | 图书管理 | 判断图书逾期 | 系统 |
| 17 | 图书管理 | 管理员根据读者是否逾期借书发送罚款信息 | 管理员 |
| 18 | 系统管理 | 管理员登录 | 管理员 |
| 19 | 借阅管理 | 借书（读者选择借书操作） | 读者 |
| 20 | 借阅管理 | 还书（读者选择还书操作） | 读者 |
| 21 | 借阅管理 | 图书挂失（读者选择挂失操作） | 读者 |

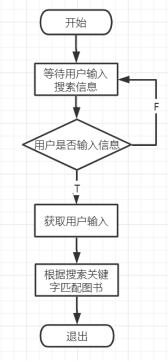
  如表1所示，该表对模块的功能进行了概述。

## 3.2 模块处理逻辑



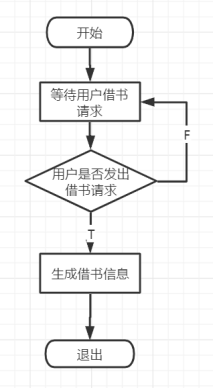
读者管理程序流程图

图2为读者管理的程序流程图，新读者要先注册，然后建立读者档案，之后再选择登录，已经注册过的读者可以直接进行登录，登录之后进入主页面。



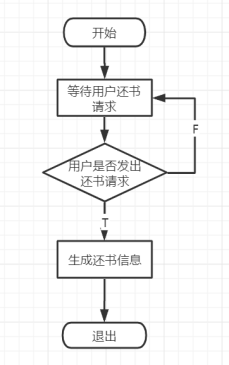
          查询图书信息程序流程图

图3为查询图书信息的查询流程图，该模块首先进入等待用户输入搜索信息状态。如果用户已经输入则获取用户输入并根据输入的搜索关键字在图书库中匹配图书最后退出。如果用户未输入则模块继续处于等待用户输入搜索信息状态。



读者借书程序流程图

图4是读者借书的程序流程图，该模块首先进入等待用户借书请求状态。如果用户发出请求则生成借书信息并写入图书借阅库，如果用户未发出请求则模块继续处于等待用户借书请求状态。

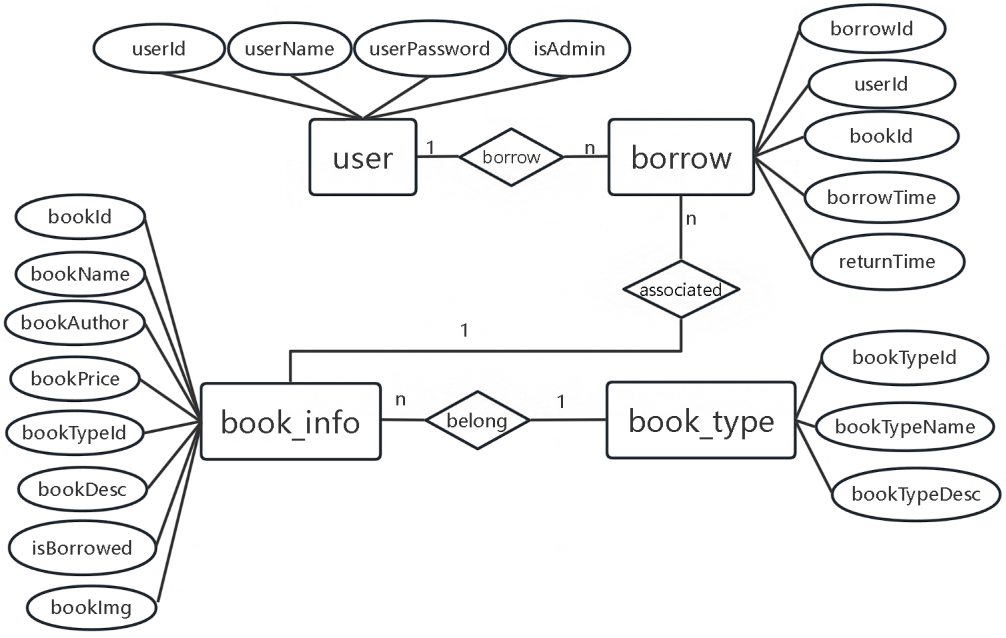


读者还书程序流程图

  图5是读者还书的程序流程图，该模块首先进入等待用户还书请求状态。如果用户发出请求则生成还书信息并写入图书借阅库，如果用户未发出请求则模块继续处于等待用户还书请求状态。

# 4 数据库设计

## 4.1 ER图表



## 4.2 表设计

### 4.2.1 图书信息

|  |  |  |
| --- | --- | --- |
| 数据名称 | 数据类型 | 数据描述 |
| bookId | Int | 图书id |
| bookName | Varchar | 图书名称 |
| bookAuthor | Varchar | 图书作者 |
| bookPrice | Decimal | 图书价格 |
| bookTypeId | Int | 图书类型id |
| bookDesc | Varchar | 图书描述 |
| isBorrowed | Tinyint | 借阅标记。1表示借出，0表示归还 |
| bookImg | Varchar | 图书封面图片链接 |

### 4.2.2 图书类型

|  |  |  |
| --- | --- | --- |
| 数据名称 | 数据类型 | 数据描述 |
| bookTypeId | Int | 类型id |
| bookTypeName | Varchar | 图书类型名称 |
| bookTypeDesc | Varchar | 图书类型描述 |

### 4.2.3 借阅信息

|  |  |  |
| --- | --- | --- |
| 数据名称 | 数据类型 | 数据描述 |
| borrowId | Varchar | 借阅信息id |
| userId | Int | 借阅读者的id |
| bookId | Int | 借阅图书信息id |
| borrowTime | Datetime | 借阅时间 |
| returnTime | Datetime | 归还时间 |

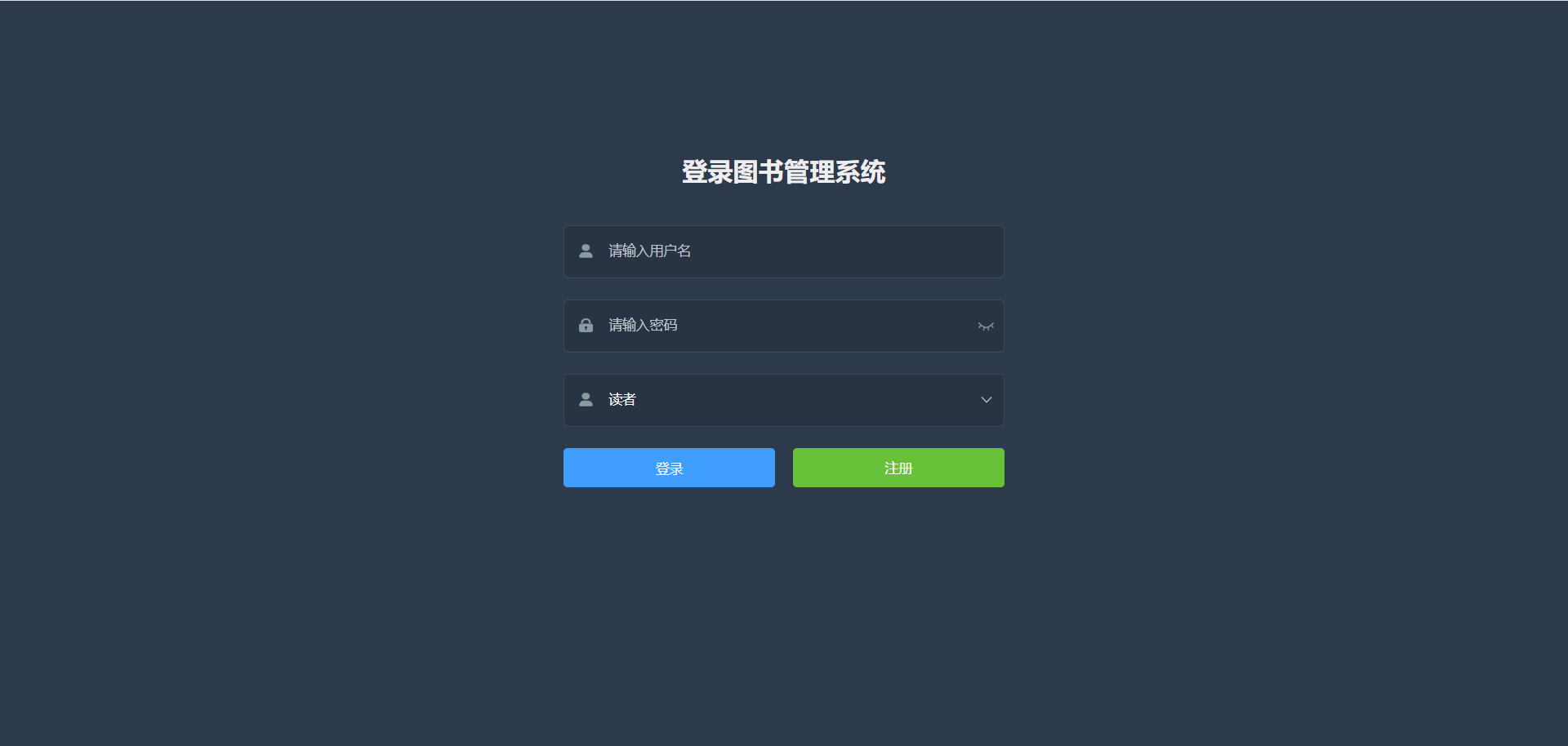
### 4.2.4 用户信息

|  |  |  |
| --- | --- | --- |
| 数据名称 | 数据类型 | 数据描述 |
| userId | Int | 用户id |
| username | Varchar | 用户名 |
| userPassword | Varchar | 用户密码 |
| isAdmin | Tinyint | 是否管理员标记。1是管理员，0是读者 |

# 5 接口设计

## 5.1 外部接口

### 5.1.1 登录界面



5.1.2首页



### 5.1.3 用户管理界面



### 5.1.4 图书列表界面



### 5.1.5 图书类型列表界面



### 5.1.6 图书借阅管理界面



## 5.2 内部接口

### 5.2.1 登录接口

*// 登录*@RequestMapping(value = "/login")  
public Map<String, Object> login(@RequestBody User user) {  
 *// 登录* User userObj = userService.login(user);  
 if(userObj == null) { *// 账号或密码错误  
 // 返回结果对象* return MyResult.*getResultMap*(420, "账号或密码错误");  
 } else { *// 账号密码正确  
 // 创建token* String token = TokenProcessor.*getInstance*().makeToken();  
 *// 保存到Redis* userService.saveUser(token, userObj);  
 *// 返回结果对象* return MyResult.*getResultMap*(200, "登录成功",  
 new HashMap<String, String>(){{ put("token", token); }});  
 }  
}

### 5.2.2 注册接口

*// 注册*@RequestMapping(value = "/register")  
public Integer register(String username, String password){  
 return userService.register(username, password);  
}

### 5.2.3 图书管理接口

*// 获取图书数量*@GetMapping(value = "/getCount")  
public Integer getCount(){  
 return bookInfoService.getCount();  
}  
  
*// 查询所有图书信息*@GetMapping(value = "/queryBookInfos")  
public List<BookInfo> queryBookInfos(){  
 return bookInfoService.queryBookInfos();  
}  
  
*// 分页搜索查询图书信息 params: {page, limit, bookname, bookauthor, booktypeid}*@GetMapping(value = "/queryBookInfosByPage")  
public Map<String, Object> queryBookInfosByPage(@RequestParam Map<String, Object> params){  
 MyUtils.*parsePageParams*(params);  
 int count = bookInfoService.getSearchCount(params); *// 获得总数* List<BookInfo> bookInfos = bookInfoService.searchBookInfosByPage(params); *// 分页查询* return MyResult.*getListResultMap*(0, "success", count, bookInfos);  
}  
  
*// 添加图书信息*@PostMapping(value = "/addBookInfo")  
public Integer addBookInfo(@RequestBody BookInfo bookInfo){  
 return bookInfoService.addBookInfo(bookInfo);  
}  
  
*// 删除图书信息*@DeleteMapping(value = "/deleteBookInfo")  
public Integer deleteBookInfo(@RequestBody BookInfo bookInfo){  
 return bookInfoService.deleteBookInfo(bookInfo);  
}  
  
*// 删除一些图书信息*@DeleteMapping(value = "/deleteBookInfos")  
public Integer deleteBookInfos(@RequestBody List<BookInfo> bookInfos){  
 return bookInfoService.deleteBookInfos(bookInfos);  
}  
  
*// 更新图书信息*@PutMapping(value = "/updateBookInfo")  
public Integer updateBookInfo(@RequestBody BookInfo bookInfo){  
 return bookInfoService.updateBookInfo(bookInfo);  
}

### 5.2.4 图书类型管理接口

*// 获得数量*@GetMapping(value = "/getCount")  
public Integer getCount(){  
 return bookTypeService.getCount();  
}  
  
*// 查询所有类型*@GetMapping(value = {"/queryBookTypes", "/reader/queryBookTypes"})  
public List<BookType> queryBookTypes(){  
 return bookTypeService.queryBookTypes();  
}  
  
*// 分页查询图书类型 params: {page, limit, booktypename}*@GetMapping(value = "/queryBookTypesByPage")  
public Map<String, Object> queryBookTypesByPage(@RequestParam Map<String, Object> params){  
 MyUtils.*parsePageParams*(params);  
 int count = bookTypeService.getSearchCount(params);  
 List<BookType> bookTypes = bookTypeService.searchBookTypesByPage(params);  
 return MyResult.*getListResultMap*(0, "success", count, bookTypes);  
}  
  
*// 添加类型*@PostMapping(value = "/addBookType")  
public Integer addBookType(@RequestBody BookType bookType){  
 return bookTypeService.addBookType(bookType);  
}  
  
*// 删除类型*@DeleteMapping(value = "/deleteBookType")  
public Integer deleteBookType(@RequestBody BookType bookType){  
 return bookTypeService.deleteBookType(bookType);  
}  
  
*// 删除一些类型*@DeleteMapping(value = "/deleteBookTypes")  
public Integer deleteBookTypes(@RequestBody List<BookType> bookTypes){  
 return bookTypeService.deleteBookTypes(bookTypes);  
}  
  
*// 更新类型*@PutMapping(value = "/updateBookType")  
public Integer updateBookType(@RequestBody BookType bookType){  
 return bookTypeService.updateBookType(bookType);  
}

### 5.2.5 图书借阅管理接口

*// 分页查询借阅 params: {page, limit, userid, bookid}*@RequestMapping(value = "/queryBorrowsByPage")  
public Map<String, Object> queryBorrowsByPage(@RequestParam Map<String, Object> params){  
 MyUtils.*parsePageParams*(params);  
 int count = borrowService.getSearchCount(params);  
 List<Borrow> borrows = borrowService.searchBorrowsByPage(params);  
 return MyResult.*getListResultMap*(0, "success", count, borrows);  
}  
  
*// 添加借阅*@RequestMapping(value = "/addBorrow")  
public Integer addBorrow(@RequestBody Borrow borrow){  
 return borrowService.addBorrow(borrow);  
}  
  
*// 获得数量*@RequestMapping(value = "/getCount")  
public Integer getCount(){  
 return borrowService.getCount();  
}  
  
*// 删除借阅*@RequestMapping(value = "/deleteBorrow")  
public Integer deleteBorrow(@RequestBody Borrow borrow){  
 return borrowService.deleteBorrow(borrow);  
}  
  
*// 删除一些借阅*@RequestMapping(value = "/deleteBorrows")  
public Integer deleteBorrows(@RequestBody List<Borrow> borrows){  
 return borrowService.deleteBorrows(borrows);  
}  
  
*// 更新借阅*@RequestMapping(value = "/updateBorrow")  
public Integer updateBorrow(@RequestBody Borrow borrow){  
 return borrowService.updateBorrow(borrow);  
}  
  
*// 借书*@RequestMapping(value = {"/borrowBook", "/reader/borrowBook"})  
@Transactional  
public Integer borrowBook(Integer userid, Integer bookid){  
 try{  
 *// 查询该书的情况* BookInfo theBook = bookInfoService.queryBookInfoById(bookid);  
  
 if(theBook == null) { *// 图书不存在* throw new NullPointerException("图书" + bookid + "不存在");  
 } else if(theBook.getIsborrowed() == 1) { *// 已经被借* throw new NotEnoughException("图书" + bookid + "库存不足（已经被借走）");  
 }  
  
 *// 更新图书表的isBorrowed* BookInfo bookInfo = new BookInfo();  
 bookInfo.setBookid(bookid);  
 bookInfo.setIsborrowed((byte) 1);  
 Integer res2 = bookInfoService.updateBookInfo(bookInfo);  
 if(res2 == 0) throw new OperationFailureException("图书" + bookid + "更新被借信息失败");  
  
 *// 添加一条记录到borrow表* Borrow borrow = new Borrow();  
 borrow.setUserid(userid);  
 borrow.setBookid(bookid);  
 borrow.setBorrowtime(new Date(System.*currentTimeMillis*()));  
 Integer res1 = borrowService.addBorrow2(borrow);  
 if(res1 == 0) throw new OperationFailureException("图书" + bookid + "添加借阅记录失败");  
  
 } catch (Exception e) {  
 System.*out*.println("发生异常，进行手动回滚");  
 TransactionAspectSupport.*currentTransactionStatus*().setRollbackOnly();  
 e.printStackTrace();  
 return 0;  
 }  
 return 1;  
}  
  
*// 还书*@RequestMapping(value = {"/returnBook", "/reader/returnBook"})  
@Transactional  
public Integer returnBook(Integer borrowid, Integer bookid){  
 try {  
 *// 查询该书的情况* BookInfo theBook = bookInfoService.queryBookInfoById(bookid);  
 *// 查询借书的情况* Borrow theBorrow = borrowService.queryBorrowsById(borrowid);  
  
 if(theBook == null) { *// 图书不存在* throw new NullPointerException("图书" + bookid + "不存在");  
 } else if(theBorrow == null) { *//结束记录不存在* throw new NullPointerException("借书记录" + bookid + "不存在");  
 } else if(theBorrow.getReturntime() != null) { *// 已经还过书* throw new NotEnoughException("图书" + bookid + "已经还过了");  
 }  
  
 *// 更新图书表的isBorrowed* BookInfo bookInfo = new BookInfo();  
 bookInfo.setBookid(bookid);  
 bookInfo.setIsborrowed((byte) 0);  
 Integer res2 = bookInfoService.updateBookInfo(bookInfo);  
 if(res2 == 0) throw new OperationFailureException("图书" + bookid + "更新被借信息失败");  
  
 *// 更新Borrow表，更新结束时间* Borrow borrow = new Borrow();  
 borrow.setBorrowid(borrowid);  
 borrow.setReturntime(new Date(System.*currentTimeMillis*()));  
 Integer res1 = borrowService.updateBorrow2(borrow);  
 if(res1 == 0) throw new OperationFailureException("图书" + bookid + "更新借阅记录失败");  
  
 } catch (Exception e) {  
 System.*out*.println("发生异常，进行手动回滚");  
 TransactionAspectSupport.*currentTransactionStatus*().setRollbackOnly();  
 e.printStackTrace();  
 return 0;  
 }  
 return 1;  
}

### 5.2.6 用户管理接口

*// 查看用户信息*@RequestMapping(value = "/info")  
public Map<String, Object> info(String token) {  
 *// 从redis中取用户* User user = userService.getUser(token);  
 if(user == null) { *// 获取失败* return MyResult.*getResultMap*(420, "获取用户信息失败");  
 } else { *// 获取成功* return MyResult.*getResultMap*(200, "获取用户信息成功", user);  
 }  
}  
  
*// 退出登录*@RequestMapping(value = "/logout")  
public Map<String, Object> logout(String token) {  
 *// 从redis中移除用户* userService.removeUser(token);  
 return MyResult.*getResultMap*(200, "退出登录成功" );  
}  
  
*// 修改密码*@RequestMapping(value = {"/alterPassword", "reader/alterPassword"})  
public Integer alterPassword(Integer userid, String username, Byte isadmin, String oldPassword, String newPassword){  
 *//检查旧密码是否正确* User userObj = new User();  
 userObj.setUserid(userid);  
 userObj.setUsername(username);  
 userObj.setUserpassword(oldPassword);  
 userObj.setIsadmin(isadmin);  
  
 User user = userService.login(userObj);  
 if(user == null) { *//旧密码不正确* return 0;  
 } else { *//旧密码正确，设置新密码* userService.setPassword(userObj.getUserid(), newPassword);  
 return 1;  
 }  
}  
  
*// 获得数量*@GetMapping(value = "/getCount")  
public Integer getCount(){  
 return userService.getCount();  
}  
  
*// 查询所有用户*@GetMapping(value = "/queryUsers")  
public List<User> queryUsers(){  
 return userService.queryUsers();  
}  
  
*// 分页查询用户 params: {page, limit, username}*@GetMapping(value = "/queryUsersByPage")  
public Map<String, Object> queryUsersByPage(@RequestParam Map<String, Object> params){  
 MyUtils.*parsePageParams*(params);  
 int count = userService.getSearchCount(params);  
 List<User> users = userService.searchUsersByPage(params);  
 return MyResult.*getListResultMap*(0, "success", count, users);  
}  
  
*// 添加用户*@PostMapping(value = "/addUser")  
public Integer addUser(@RequestBody User user){  
 return userService.addUser(user);  
}  
  
*// 删除用户*@DeleteMapping(value = "/deleteUser")  
public Integer deleteUser(@RequestBody User user){  
 return userService.deleteUser(user);  
}  
  
*// 删除一些用户*@DeleteMapping(value = "/deleteUsers")  
public Integer deleteUsers(@RequestBody List<User> users){  
 return userService.deleteUsers(users);  
}  
  
*// 更新用户*@RequestMapping(value = "/updateUser")  
public Integer updateUser(@RequestBody User user){  
 return userService.updateUser(user);  
}

# 6 性能

## 6.1 精度

要按照严格的数据格式输入，不能输入非法字符，否则系统不给予响应进行处理，查询时要保证准确率为100%，所有包含查询关键字的书籍都应能查到，不能有遗漏。

## 6.2 时间特性

（1）响应时间：用户任意操作后5秒内系统给予反馈信息。

（2）更新处理时间：由系统运行状态来决定。

（3）数据的转换和传送时间：能够在20秒内完成。

## 6.3 灵活性

当需求发生某些变化时，该软件的基本操作、数据结构、运行环境等等基本不会发生变化，只是对系统的数据库的文件和记录进行处理。

# 7 测试

|  |  |  |
| --- | --- | --- |
| 功能点 | 测试用例 | 输出结果 |
| 读者登录 | 选择读者角色，输入读者的用户名和密码，点击登录 | 跳转到图书管理系统首页，并展示读者相关功能菜单 |
| 读者注册 | 点击注册，跳转注册页，填入用户名、密码、确认密码，点击注册 | 未填写完整表单，提示对应异常，填写完整表单，提示注册成功 |
| 管理员登录 | 选择管理员角色，输入管理员用户名和密码，点击登录 | 跳转到图书管理系统首页，并展示管理员相关功能菜单 |
| 图书信息新增 | 点击新增，输入图书信息，点击确认新增 | 未填写完整表单，提示对应异常，填写完整表单，提示新增成功 |
| 图书信息查询 | 输入对应查询条件，点击查询 | 图书信息列表 |
| 图书信息修改 | 选中图书信息，弹出图书信息修改框，修改对应信息，点击修改 | 未填写完整表单，提示对应异常，填写完整表单，提示修改成功 |
| 图书信息删除 | 选中删除的数据 | 提示删除成功 |
| 图书类型新增 | 点击新增，输入图书类型信息，点击确认新增 | 未填写完整表单，提示对应异常，填写完整表单，提示新增成功 |
| 图书类型查询 | 输入对应查询条件，点击查询 | 图书类型列表 |
| 图书类型修改 | 选中图书类型信息，弹出图书类型修改框，修改对应信息，点击修改 | 未填写完整表单，提示对应异常，填写完整表单，提示修改成功 |
| 图书类型删除 | 选中删除的数据 | 提示删除成功 |
| 图书借阅 | 读者选中图书，点击借阅图书 | 提示借阅成功，并生成借阅时间 |
| 图书归还 | 读者选中借阅中的图书 | 提示归还成功，并更新归还时间 |
| 用户新增 | 点击新增，输入用户信息，点击确认新增 | 未填写完整表单，提示对应异常，填写完整表单，提示新增成功 |
| 用户信息查询 | 输入对应查询条件，点击查询 | 用户信息列表 |
| 用户修改 | 选中用户信息，弹出用户信息修改框，修改对应信息，点击修改 | 未填写完整表单，提示对应异常，填写完整表单，提示修改成功 |
| 用户删除 | 选中删除的数据 | 提示删除成功 |