目录

一、题目

图书馆管理系统

二、绪论

2.1 研究背景及意义

随着信息技术的飞速发展，图书管理系统作为高校信息化建设的重要组成部分，在图书馆管理、图书借阅、用户服务等方面发挥着越来越重要的作用。传统的图书管理方式存在着效率低、数据管理不便、信息反馈不及时等问题，已难以满足现代高校对于图书资源管理智能化、信息化、高效化的需求。因此，开发一套基于现代信息技术、功能完善、界面友好、操作简便的图书管理系统，具有重要的现实意义和应用价值。

特别是在“数字校园”建设日益深化的背景下，高校图书馆需要借助现代信息技术，实现从传统服务模式向智慧服务模式的转型。一个高效的图书管理系统不仅能够显著提升图书馆的工作效率，还能够为师生提供更加便捷的图书借阅体验，推动高校教学科研水平的提高。

2.2 主要研究目标与内容

本系统旨在构建一个面向高校师生的图书管理平台，具备普通用户在线注册登录、图书信息查询、借阅归还操作等基本功能，同时为管理员提供对图书和用户信息的增删改查、借阅审核、数据统计等高级功能，全面支持图书馆的业务处理流程。

系统采用前后端分离架构，前端基于 Vue3 与 Element Plus 框架，结合 Node.js 与 npm 实现高效构建与模块管理，提升开发与维护效率。后端则使用 Spring Boot 框架整合 Spring Security 与 JWT 实现权限控制和安全认证，并通过 MyBatis-Plus 简化数据库操作逻辑，配合 MySQL 数据库实现数据的高效存储与管理。此外，系统集成 Redis 缓存技术以提升响应速度，增强并发性能。

在系统设计与实现过程中，充分考虑用户体验与系统可扩展性，通过 Swagger

文档与单元测试保障系统的可维护性与开发质量。同时，系统预留与 RFID、图书自助借还设备等智慧硬件的对接接口，为今后扩展智慧图书馆功能奠定技术基础。

本报告将从关键技术选型、需求分析、系统设计、功能实现与测试等多个方面，对系统的开发全过程进行详尽阐述，旨在为同类型信息管理系统的研究与开发提供理论参考与实践指导。

关键词：web 技术、Vue3、Spring Boot、图书管理系统

三、系统所用关键技术

3.1 总体设计

系统采用前后端分离架构，前端负责界面渲染，通过 Axios 与后端通信，后端负责运营逻辑和数据处理。

Node.js 与 npm：

Node.js 是一个基于 Chrome V8 引擎的 JavaScript 运行环境，常用于前端项目构建工具链的搭建。npm 是其包管理工具，项目中通过 npm 安装 Element Plus、Axios 等依赖。

```
C:\Users\陈>node -v
v20.12.0

C:\Users\陈>npm -v
10.5.0
```

3.2 前端

Vue3 是一种现代化的 JavaScript 框架，适用于构建用户界面。它具有响应式、组件化等特性，结合 Element Plus 提供优雅的 UI 界面。本系统前端采用 Vue3 构建，用户体验良好。

```
C:\Users\陈\vue_test>cd..

C:\Users\陈>npm install element-plus @element-plus/icons-vue

added 45 packages, changed 1 package, and audited 94 packages in 16s

18 packages are looking for funding
  run `npm fund` for details
```

3.3 后端

Java 开发语言：

Java 是一种面向对象、可以跨平台的开发语言，具有稳定性好、安全性高等特点，是开发网站和后端服务器应用程序的重要工具。

本项目使用 JDK18，适配 Spring-Boot3

```
C:\Users\陈>java -version
java version "18.0.2.1" 2022-08-18
Java(TM) SE Runtime Environment (build 18.0.2.1+1-1)
Java HotSpot(TM) 64-Bit Server VM (build 18.0.2.1+1-1, mixed mode, sharing)
```

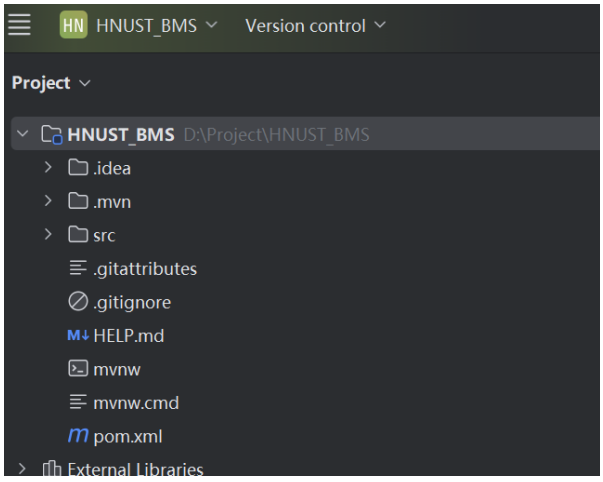
```
C:\Users\陈>mvn -v
Apache Maven 3.9.9 (8e8579a9e76f7d015ee5ec7bfcdc97d260186937)
Maven home: D:\apache-maven-3.9.9
Java version: 18.0.2.1, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-18.0.2.1
Default locale: zh_CN, platform encoding: UTF-8
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"
```

Spring 框架：

Spring 是一套优秀的组件化应用编程框架，提供了 IOC 和 AOP 等核心功能，用于解耦系统级组件之间的耗耘性强耦性绑定问题，提高系统扩展性和维护性。

Spring Boot 开发框架：

Spring Boot 是 Spring 公司推出的方便性框架，通过自动配置和简化编程，大大降低了开发性术门槛，最终实现 Java Web 快速开发。

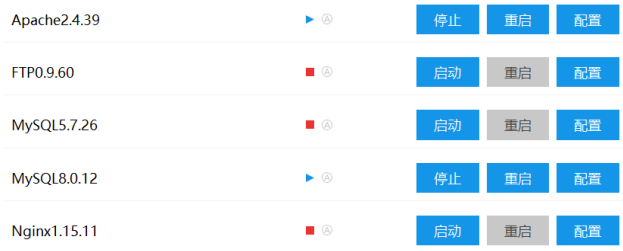


MyBatis 持久层框架：

MyBatis 是一个优秀的开源 ORM 框架，方便将 SQL 语句和实体对象完美映射，提高开发效率。本系统基于 MyBatis-Plus，展示更优秀的体验效果。

MySQL 数据库：

MySQL 是应用最常见的开源关系型数据库，具有安全稳定、效率高和扩展性强等优点，是后端应用系统中的实用选择。本项目使用 MySQL8.0.12



四、系统需求分析与设计

4.1 功能分析

4.1.1 角色分析

- 超级管理员：

  增删查改“普通管理员”

  查看/修改自身信息

- 普通管理员：

  增删查改“用户”

  增删查改“图书”

  查看借阅记录

  查看/修改自身信息

- 用户：

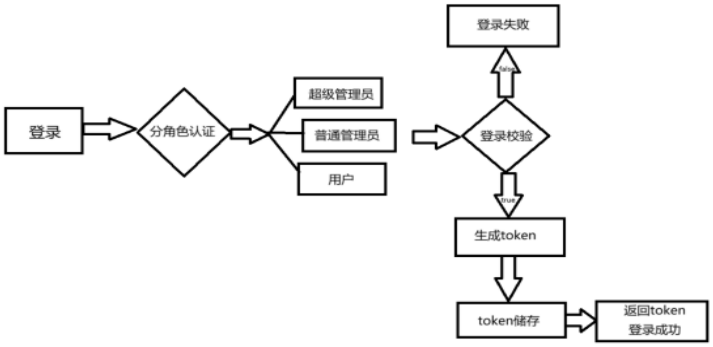
  借阅/归还/查看图书信息

  查看借阅/归还记录

  查看自身信息

  注册、修改自身账号

4.1.2 登录流程



4.2 数据库表创建

管理员信息

用户信息

图书列表

图书借阅表

图书归还表

```
Current database: HNUST_BMS_DB

+-------------------------+
| Tables_in_hnust_bms_db |
+-------------------------+
| admin                   |
| announcement            |
| book                    |
| book_borrow_info        |
| book_category           |
| borrow_record           |
| carousel_map            |
| return_record           |
| sys_token               |
| token                   |
| user                    |
| user_return_book_info   |
+-------------------------+
```

五、系统实现

5.1 数据库

5.1.1 普通管理员信息表

```
mysql> show columns from admin;
+-------------+-------------+------+-----+---------+----------------+
| Field       | Type        | Null | Key | Default | Extra          |
+-------------+-------------+------+-----+---------+----------------+
| id          | bigint(20)  | NO   | PRI | NULL    | auto_increment |
| role        | int(2)      | YES  |     | 1       |                |
| username    | varchar(20) | NO   | UNI | NULL    |                |
| password    | varchar(64) | NO   |     | NULL    |                |
| nick_name   | varchar(20) | NO   | UNI | NULL    |                |
| create_time | datetime    | YES  |     | NULL    |                |
+-------------+-------------+------+-----+---------+----------------+
```

5.1.2 用户信息表

```
mysql> show columns from user;
+----------------+-------------+------+-----+---------+----------------+
| Field          | Type        | Null | Key | Default | Extra          |
+----------------+-------------+------+-----+---------+----------------+
| id             | bigint(20)  | NO   | PRI | NULL    | auto_increment |
| student_number | varchar(32) | NO   | UNI | NULL    |                |
| password       | varchar(64) | NO   |     | NULL    |                |
| name           | varchar(20) | NO   |     | NULL    |                |
| sex            | tinyint(1)  | YES  |     | 1       |                |
| college        | varchar(50) | NO   |     | NULL    |                |
| grade          | varchar(20) | NO   |     | NULL    |                |
| phone          | varchar(11) | NO   |     | NULL    |                |
| dormitory      | varchar(50) | YES  |     | NULL    |                |
| create_time    | datetime    | YES  |     | NULL    |                |
| update_time    | datetime    | YES  |     | NULL    |                |
+----------------+-------------+------+-----+---------+----------------+
```

5.1.3 图书信息表

```
mysql> show columns from book;
+-----------------+---------------+------+-----+---------+----------------+
| Field           | Type          | Null | Key | Default | Extra          |
+-----------------+---------------+------+-----+---------+----------------+
| id              | bigint(20)    | NO   | PRI | NULL    | auto_increment |
| book_number     | varchar(20)   | NO   | UNI | NULL    |                |
| cover_url       | varchar(500)  | NO   |     | NULL    |                |
| book_name       | varchar(50)   | NO   | UNI | NULL    |                |
| author          | varchar(20)   | NO   |     | NULL    |                |
| category_id     | bigint(20)    | NO   |     | NULL    |                |
| price           | decimal(10,2) | YES  |     | 0.00    |                |
| publishing_house | varchar(50)  | NO   |     | NULL    |                |
| publication_date | date         | NO   |     | NULL    |                |
| total           | int(4)        | NO   |     | 0       |                |
| extant_total    | int(4)        | NO   |     | 0       |                |
| intro           | varchar(1000) | YES  |     | NULL    |                |
| create_time     | datetime      | YES  |     | NULL    |                |
| creator         | varchar(50)   | YES  |     | NULL    |                |
| update_time     | datetime      | YES  |     | NULL    |                |
| modifier        | varchar(50)   | YES  |     | NULL    |                |
+-----------------+---------------+------+-----+---------+----------------+
```

5.1.4 图书类别表

```
mysql> show columns from book_category;
+---------------+-------------+------+-----+---------+----------------+
| Field         | Type        | Null | Key | Default | Extra          |
+---------------+-------------+------+-----+---------+----------------+
| id            | bigint(20)  | NO   | PRI | NULL    | auto_increment |
| category_name | varchar(50) | NO   | UNI | NULL    |                |
| create_time   | datetime    | YES  |     | NULL    |                |
| creator       | varchar(20) | YES  |     | NULL    |                |
| update_time   | datetime    | YES  |     | NULL    |                |
| modifier      | varchar(20) | YES  |     | NULL    |                |
+---------------+-------------+------+-----+---------+----------------+
```

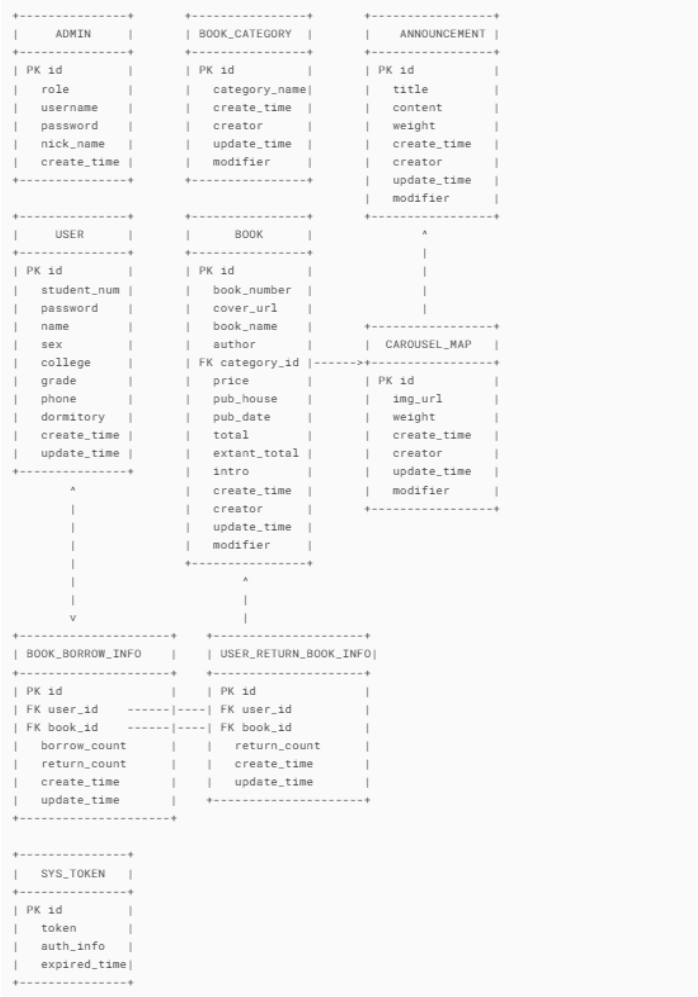
5.1.5 图书借阅表

```
mysql> show columns from book_borrow_info;
+--------------+------------+------+-----+---------+----------------+
| Field        | Type       | Null | Key | Default | Extra          |
+--------------+------------+------+-----+---------+----------------+
| id           | bigint(20) | NO   | PRI | NULL    | auto_increment |
| user_id      | bigint(20) | NO   |     | NULL    |                |
| book_id      | bigint(20) | NO   |     | NULL    |                |
| borrow_count | int(4)     | NO   |     | 0       |                |
| return_count | int(4)     | YES  |     | 0       |                |
| create_time  | datetime   | YES  |     | NULL    |                |
| update_time  | datetime   | YES  |     | NULL    |                |
+--------------+------------+------+-----+---------+----------------+
```

5.1.6 图书归还表

```
mysql> show columns from user_return_book_info;
+--------------+------------+------+-----+---------+----------------+
| Field        | Type       | Null | Key | Default | Extra          |
+--------------+------------+------+-----+---------+----------------+
| id           | bigint(20) | NO   | PRI | NULL    | auto_increment |
| user_id      | bigint(20) | NO   |     | NULL    |                |
| book_id      | bigint(20) | NO   |     | NULL    |                |
| return_count | int(4)     | NO   |     | NULL    |                |
| create_time  | datetime   | YES  |     | NULL    |                |
| update_time  | datetime   | YES  |     | NULL    |                |
+--------------+------------+------+-----+---------+----------------+
```

5.1.7E-R 图



5.2 后端实现

5.2.1 结果集处理

· 格式对象统一：

result：Boolean 类型，true 表示此次请求成功，反之表示失败

code：后端返回给前端的状态码，方便区分错误类型

message：一个字符串类型的消息说明

· 成功处理：

采用 AOP 思想实现响应统一包装

通过注解(@ResponseResult)控制是否启用包装

兼容多种返回类型(String/null/普通对象)

保持响应内容类型(Content-Type)的一致性

- 条件判断(supports 方法):
  - 检查控制器类或方法是否标注了@ResponseResult 注解

- 排除 FileSystemResource 类型的返回
- 根据注解决定是否进行响应包装

- 响应包装(beforeBodyWrite 方法):
  - 将原始数据包装成 SuccessInfo 统一格式
  - 特殊处理 String 类型响应，确保返回 JSON 格式
  - 特殊处理 null 值响应，确保返回 JSON 格式
  - 设置响应 Content-Type 为 application/json
- 异常处理:
  - 使用@SneakyThrows 自动处理可能的异常
  - 通过@ControllerAdvice 实现全局处理

- 失败处理：

  采用 Spring 的全局异常处理机制

  统一的错误响应格式

  完善的错误日志记录(简略信息+完整堆栈)

  分层异常处理(特定异常优先于通用异常)

  - 全局异常处理机制:

    使用@RestControllerAdvice 注解标记为全局异常处理器

    通过@ExceptionHandler 指定处理的异常类型

  - 异常处理流程:

    捕获到异常后，首先记录错误日志

    将异常信息包装成统一的 FailInfo 格式返回

  - 异常处理策略:

    Exception.class 作为兜底处理，捕获所有未被特殊处理的异常

    SysException.class 专门处理自定义系统异常

5.2.2 登录

- 用户登录

通过传入的账号，密码去数据库中找数据，找到则登录成功，反之则登录失败。

@AllArgsConstructor
@Slf4j

```
@Service
public class UserServiceImpl extends ServiceImpl<UserMapper, User>
implements UserService {

    private final UserConverter userConverter;

    @Override
    public UserVO login(AuthForm form) {
        User user = lambdaQuery()
                .eq(User::getStudentNumber, form.getUsername())
                .eq(User::getPassword,
PasswordUtil.encryptPassword(form.getPassword()))
                .one();
        if (Objects.isNull(user)) {
            throw new RuntimeException("账号不存在或账号密码错误！");
        }
        return userConverter.converter(user);
    }
}
```

- 管理员登陆

  区分超级管理员与普通管理员

```
@Slf4j
@Service
@AllArgsConstructor
public class AdminServiceImpl extends ServiceImpl<AdminMapper, Admin>
implements AdminService {

    private final UserConverter userConverter;

    @Override
    public AdminVO login(AuthForm form) {
        LambdaQueryChainWrapper<Admin> queryChainWrapper =
lambdaQuery()
                .eq(Admin::getUsername, form.getUsername())
                .eq(Admin::getPassword,
PasswordUtil.encryptPassword(form.getPassword()));
```

```
        if (RoleEnum.ROLE_SUPER_ADMIN.equals(form.getRole())) {
            queryChainWrapper.eq(Admin::getRole, 0);
        } else {
            queryChainWrapper.ne(Admin::getRole, 0);
        }
        Admin admin = queryChainWrapper.one();
        if (Objects.isNull(admin)) {
            throw new RuntimeException("账号不存在或账号密码错误！");
        }
        return userConverter.converter(admin);
    }
}
```

- Token 储存

前端每次获取数据的时候都携带 token，而后端每次接收到请求的时候都过来验证这个 token 的有效性。如果有效就给数据，反之就拒之门外。

本系统将 token 存在 MySQL 中。

```
CREATE TABLE `sys_token` (
 `id` bigint(20) NOT NULL AUTO_INCREMENT,
 `token` varchar(64) COLLATE utf8mb4_german2_ci NOT NULL COMMENT 'token 字符串',
 `auth_info` varchar(1000) COLLATE utf8mb4_german2_ci NOT NULL COMMENT '认证信息，json 内容',
 `expired_time` datetime NOT NULL COMMENT 'token 的过期时间',
 PRIMARY KEY (`id`),
 UNIQUE KEY `un` (`token`)
) ENGINE=InnoDB AUTO_INCREMENT=17 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_german2_ci
```

后端储存用户 token

```
@Slf4j
@Service
public class SysTokenServiceImpl extends ServiceImpl<SysTokenMapper,
SysToken>
implements SysTokenService {

    @Override
    public void saveToken(AuthVO authVO) {
        SysToken sysToken = new SysToken()
```

```
        .setToken(authVO.getToken())
        .setAuthInfo(JSON.toJSONString(authVO))
        .setExpiredTime(SysUtil.getTokenExpiredTime());
        try {
            save(sysToken);
        } catch (Exception e) {
            log.error("内部保存 token 失败！ token:{}",
    JSON.toJSONString(sysToken), e);
            throw new SysException("登录失败！");
        }
    }
}
```

5.2.3token 拦截认证

- 验证流程

在拦截器的 preHandle 方法中获取 token 信息后，通过 sysTokenService.authToken(token)方法进行验证：

验证方式：将请求中的 token 与 sys_token 表中存储的 token 进行比对

验证结果：

如果 token 不存在或已失效，判定为非法请求并拒绝

如果 token 有效，允许请求继续执行

- 功能说明

使用 ThreadLocal 存储用户认证信息，确保线程安全

提供多种信息获取方式：

完整认证信息(AuthVO)

用户信息(UserVO)

用户 ID

管理员信息(AdminVO)

包含清除认证信息的方法

- 拦截器配置

拦截路径：所有请求(/**)

排除路径：

静态资源(/static/**)

开放 API(/**/open-api/**/)

登录接口(/**/auth/login)

用户注册接口(/**/user/register)

- 过期 token 清理逻辑

执行频率：每 4 小时执行一次(在每小时的 15 分执行)

清理条件：删除 expired_time 小于当前时间的 token 记录

异常处理：记录清理过程中的异常信息

5.2.4 用户相关功能分析

功能实现分析

Controller 层：cn.j3code.booksys.api.controller 包

Service 层：cn.j3code.booksys.service.impl 包中的 UserServiceImpl 类

5.3.1 用户登录：

使用学号和加密密码进行验证

密码采用加密存储和验证

登录失败抛出明确异常信息

```
@Override
public UserVO login(AuthForm form) {
    User user = lambdaQuery()
        .eq(User::getStudentNumber, form.getUsername())
        .eq(User::getPassword,
PasswordUtil.encryptPassword(form.getPassword()))
        .one();
    if (Objects.isNull(user)) {
        throw new RuntimeException("账号不存在或账号密码错误！");
    }
    return userConverter.converter(user);
}
```

5.2.5 用户注册

多重数据校验机制

学号唯一性检查

密码加密存储

```
@Override
public void register(UserRegisterForm form) {
    // 学号唯一性校验
    if (lambdaQuery().eq(User::getStudentNumber,
form.getStudentNumber()).count() > 0) {
        throw new SysException("学号已存在！");
    }

    // 密码非空校验
    if (StringUtils.isBlank(form.getPassword())) {
        throw new SysException("密码不能为空！");
    }

    // 手机号格式校验
    if (Boolean.FALSE.equals(SysUtil.checkPhone(form.getPhone()))) {
        throw new SysException("请输入正确电话号码格式！");
    }

    User user = userConverter.converter(form);
    user.setPassword(PasswordUtil.encryptPassword(user.getPassword()));
    save(user);
}
```

5.2.6 用户分页查询：

支持多条件模糊查询

分页返回结果

使用对象转换器将 Entity 转为 VO

```
@Override
public IPage<UserVO> page(UserQuery query) {
    Page<User> userPage = lambdaQuery()
        .like(StringUtils.isNotBlank(query.getStudentNumber()),
User::getStudentNumber, query.getStudentNumber())
        .like(StringUtils.isNotBlank(query.getName()), User::getName,
query.getName())
        .like(StringUtils.isNotBlank(query.getCollege()), User::getCollege,
query.getCollege())
        .like(StringUtils.isNotBlank(query.getGrade()), User::getGrade,
```

```
query.getGrade())
        .like(StringUtils.isNotBlank(query.getPhone()), User::getPhone,
query.getPhone())
        .like(StringUtils.isNotBlank(query.getDormitory()), User::getDormitory,
query.getDormitory())
        .eq(Objects.nonNull(query.getSex()), User::getSex, query.getSex())
        .page(query.getPage());
    return userPage.convert(userConverter::converter);
}
```

5.2.7 用户信息更新：

更新前进行数据校验

学号唯一性检查排除当前用户

密码加密更新

```
@Override
public void update(UserRegisterForm form) {
    // ID 非空校验
    if (Objects.isNull(form.getId())) {
        throw new SysException("用户 ID 不能为空！");
    }

    // 学号唯一性校验(排除自身)
    if (lambdaQuery()
        .eq(User::getStudentNumber, form.getStudentNumber())
        .ne(User::getId, form.getId())
        .count() > 0) {
        throw new SysException("学号已存在！");
    }

    // 手机号格式校验
    if (Boolean.FALSE.equals(SysUtil.checkPhone(form.getPhone()))) {
        throw new SysException("请输入正确电话号码格式！");
    }

    User user = userConverter.converter(form);
    user.setPassword(PasswordUtil.encryptPassword(user.getPassword()));
    updateById(user);
}
```

5.2.6 用户删除：

级联删除相关数据

先删除关联的借阅和归还记录

最后删除用户主体信息

```
@Override
public void delete(Long id) {
   // 删除用户借阅信息
   bookBorrowInfoService.lambdaUpdate().eq(BookBorrowInfo::getUserId,
id).remove();
   // 删除用户归还信息
   userReturnBookInfoService.lambdaUpdate().eq(UserReturnBookInfo::getUserId,
id).remove();
   // 删除用户信息
   removeById(id);
}
```

- **图书相关功能分析与实现**

通用功能：获取图书列表（管理员和普通用户均可使用）

管理员功能：添加图书、修改图书信息

用户功能：图书借阅

5.2.8 图书功能实现：

- Controller 层

```
@Slf4j
@ResponseResult
@AllArgsConstructor
@RestController
@RequestMapping("/book")
public class BookController {
   private final BookService bookService;

   // 分页查询图书
   @GetMapping("/page")
   public IPage<BookVO> page(BookQuery query) {
      return bookService.page(query);
   }

   // 获取单本图书详情
```

```
    @GetMapping("/one")
    public BookVO one(@RequestParam(name = "id", required = true) Long id) {
        return bookService.one(id);
    }

    // 保存或更新图书信息
    @PostMapping("/saveOrUpdate")
    public void saveOrUpdate(@Validated @RequestBody BookSaveOrUpdateForm form) {
        bookService.saveOrUpdate(form);
    }
}
```

- Service 层

```
@Slf4j
@AllArgsConstructor
@Service
public class BookServiceImpl extends ServiceImpl<BookMapper, Book>
        implements BookService {

    private final BookConverter bookConverter;

    // 分页查询
    @Override
    public IPage<BookVO> page(BookQuery query) {
        return getBaseMapper().page(query.getPage(), query);
    }

    // 保存或更新图书
    @Override
    public void saveOrUpdate(BookSaveOrUpdateForm form) {
        // 检查图书编号或名称是否已存在
        List<Book> bookList = getBaseMapper().bookListBy(form.getBookNumber(),
form.getBookName());
        long count = bookList.stream()
                .filter(item -> Boolean.FALSE.equals(item.getId().equals(form.getId())))
                .count();
        if (count > 0) {
            throw new SysException("图书编号或名称已存在！");
        }
```

```
        Book book = bookConverter.converter(form);
        if (Objects.isNull(book.getId())) {
            // 新增图书
            book.setExtantTotal(book.getTotal());
            save(book);
        } else {
            // 更新图书
            Book loadBook = getById(form.getId());
            // 计算可用库存 = 新库存 - (原库存 - 原可用库存)
            book.setExtantTotal(form.getTotal() - (loadBook.getTotal() -
    loadBook.getExtantTotal()));
            if (book.getExtantTotal() < 0) {
                throw new SysException("已借出去的图书大于此库存数，请正确填写！");
            }
            updateById(book);
        }
    }

    // 获取单本图书详情
    @Override
    public BookVO one(Long id) {
        BookVO vo = getBaseMapper().one(id);
        if (Objects.isNull(vo)) {
            throw new SysException("图书信息不存在！");
        }
        return vo;
    }
}
```

· **功能特点分析**

图书管理功能

分页查询：

支持多条件查询

返回分页结果

图书新增/更新：

图书编号和名称唯一性校验

自动计算可用库存

库存变更时的业务校验

图书详情查询：

返回完整的图书信息

包含图书封面图片 URL

5.2.9 图书借阅与归还

功能列表

- 借阅：

查看借阅列表（管理员查看所有，用户查看自己的）

用户借阅图书

借阅记录合并处理

- 归还：

查看归还记录（管理员查看所有，用户查看自己的）

用户归还图书

归还数量校验

借阅功能实现

事务处理：

使用@Transactional 确保借阅操作的原子性

异常时自动回滚

智能合并：

同一用户重复借阅同一图书时自动合并记录

避免产生冗余数据

库存管理：

借阅前严格校验库存

借阅成功后实时更新可用库存

权限控制：

通过 SecurityUtil 获取当前用户 ID

确保用户只能操作自己的借阅记录

归还功能实现

权限控制：

管理员可查看所有归还记录

普通用户只能查看自己的记录

通过 SecurityUtil 实现自动过滤

参数校验：

强制要求借阅记录 ID 和归还数量参数

确保接口调用的完整性

RESTful 设计：

合理的 URL 设计(/userReturnBookInfo)

清晰的 HTTP 方法使用(GET 用于查询)

5.3 前端实现

5.3.1 前端结构

```
src/
├── api/                # 接口封装
├── assets/             # 静态资源
├── components/         # 公共组件
├── router/             # 路由配置
├── store/              # 状态管理
├── utils/              # 工具类
├── views/              # 页面组件
└── App.vue             # 根组件
```

5.3.2 页面开发流程

典型管理页面元素

查询表单区域（使用 ElementUI 表单组件）

数据表格展示（ElementUI Table 组件）

分页控件（ElementUI Pagination 组件）

功能按钮组（新增、编辑、删除等操作）

5.3.3 请求封装实现

- **axios 基础封装**

utils/request.js 核心配置

```
import axios from 'axios'
import { getToken } from '@/utils/auth'
```

```
// 创建 axios 实例
const service = axios.create({
  baseURL: process.env.VUE_APP_BASE_API, // 后端 API 地址
  timeout: 5000 // 请求超时时间
})

// 请求拦截器
service.interceptors.request.use(
  config => {
    // 在请求头中添加 token
    if (getToken()) {
      config.headers['Authorization'] = 'Bearer ' + getToken()
    }
    return config
  },
  error => {
    return Promise.reject(error)
  }
)
// 响应拦截器
service.interceptors.response.use(
  response => {
    const res = response.data
    // 处理业务逻辑错误
    if (res.code !== 200) {
      // 错误处理
      return Promise.reject(new Error(res.message || 'Error'))
    } else {
      return res
    }
  },
  error => {
    // HTTP 错误处理
    return Promise.reject(error)
  }
)

export default service
```

utils/http.js 业务封装

```
import request from './request'

export function get(url, params) {
 return request({
  url,
  method: 'get',
  params
 })
}

export function post(url, data) {
 return request({
  url,
  method: 'post',
  data
 })
}
```

- 业务 API 封装示例

api/book.js 图书接口

```
import { get, post } from '@/utils/http'

// 获取图书分页列表
export function fetchBookList(query) {
  return get('/book/page', query)
}

// 新增或更新图书
export function saveOrUpdateBook(data) {
  return post('/book/saveOrUpdate', data)
}
```

5.3.4 数据绑定与渲染

定义数据变量

```
  data() {
    return {
      bookList: [],     // 图书列表数据
      total: 0,        // 总条数
```

```
    listQuery: {      // 查询条件
      page: 1,
      limit: 10,
      bookName: undefined
    },
    dialogVisible: false // 对话框显示状态
  }
 }
```

组件数据绑定

```
<template>
 <el-table :data="bookList" style="width: 100%">
  <el-table-column prop="bookName" label="图书名称" />
  <el-table-column prop="author" label="作者" />
  <!-- 其他列 -->
 </el-table>

 <el-pagination
  @size-change="handleSizeChange"
  @current-change="handleCurrentChange"
  :current-page="listQuery.page"
  :page-sizes="[10, 20, 30]"
  :page-size="listQuery.limit"
  layout="total, sizes, prev, pager, next, jumper"
  :total="total">
 </el-pagination>
</template>
```

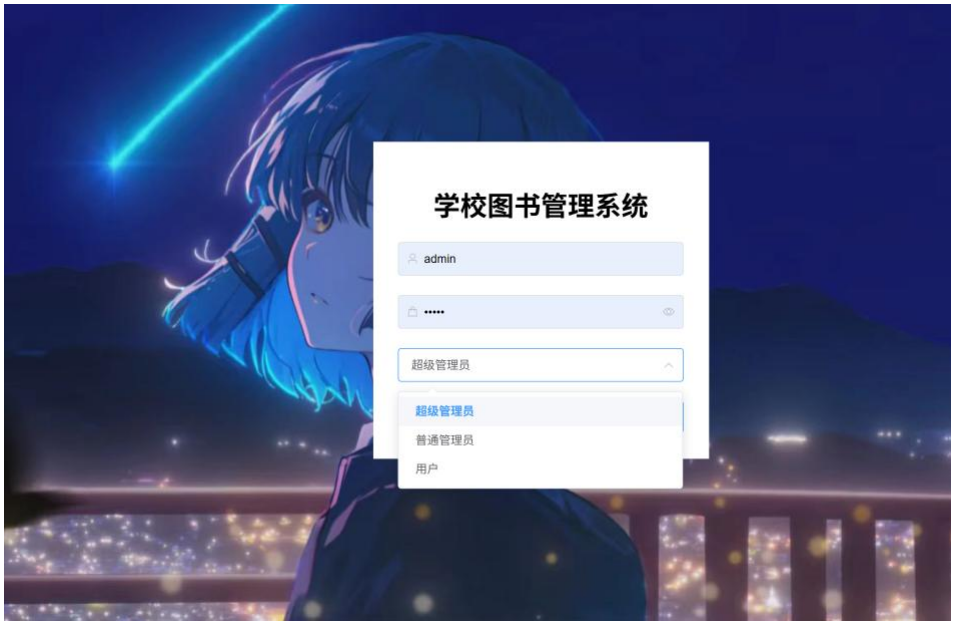
- 获取并渲染数据

```
import { fetchBookList } from '@/api/book'

methods: {
 getList() {
  fetchBookList(this.listQuery).then(response => {
   this.bookList = response.data.items
   this.total = response.data.total
  })
 },
 handleSizeChange(val) {
```

```
    this.listQuery.limit = val
    this.getList()
  },
  handleCurrentChange(val) {
    this.listQuery.page = val
    this.getList()
  }
},
created() {
  this.getList()
}
```

5.4 效果展示



六、总结

6.1 优化

6.1.1 结果集

单一封装更为简单，但企业级的写法是拆分成多个响应类。

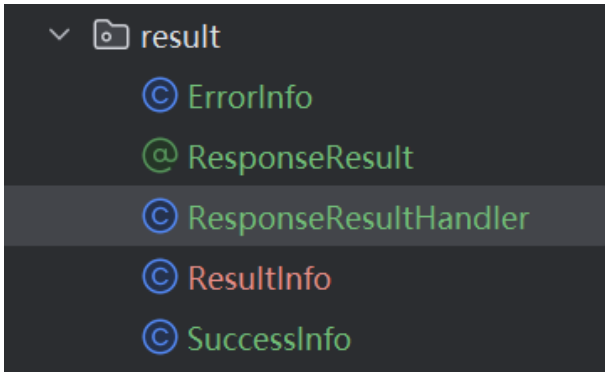
如我现在优化后的写法：

ResultInfo ： 响应结果基类

SuccessInfo, ErrorInfo ： 继承 ResultInfo，响应成功/失败

ResponseResult：注解

ResponseResultHandler：实现 ResponseBodyAdvice，统一封装响应结构



6. 1. 2Token 拦截认证

通过内置的拦截器接口来实现

admin. setPassword(passwordEncoder. encode(password)); // 替换原先的 PasswordUtil. encrypt

```java
@Override  1 个用法
public void saveSuperAdmin(String username, String password) {
    Long count = lambdaQuery()
            .eq(Admin::getRole,RoleEnum.ROLE_SUPER_ADMIN)
            .count();
    if(count > 0){
        log.info("超级管理员已存在");
        return;
    }

    Admin admin = new Admin();
    admin.setRole(RoleEnum.ROLE_SUPER_ADMIN);
    admin.setUsername(username);
    admin.setPassword(passwordEncoder.encode(password));
    admin.setNickname("超级管理员");


    this.save(admin);
}
```

6.2 后续可优化

6.2.1 图书管理功能

图片存储：考虑未来扩展为云存储方案、添加图片压缩功能

库存管理：添加库存变更历史记录、现库存预警功能

查询功能：添加更多查询条件（如按分类、状态等）、实现高级搜索功能

性能优化：对高频访问的图书信息添加缓存、考虑实现图书封面图的 CDN 加速

6.2.2 图书借阅归还

强校验：添加最大借阅数量限制、实现借阅期限控制

扩展功能：添加借阅逾期处理、实现图书预约功能

性能优化：高频查询接口添加缓存、考虑批量操作支持

用户体验：添加借阅/归还通知功能、提供借阅历史统计

6.3 心得体会

本次图书管理系统的设计与实现，是一次综合性较强的软件开发实践。通过该项目，我不仅深入掌握了 Java 编程语言及其相关技术框架的使用方法，还全面体验了一个完整软件系统从需求分析、架构设计、功能开发到测试部署的全过程。

首先，在后端开发过程中，我通过 Spring Boot 构建了项目基础框架，利用 Spring Security 与 JWT 实现了用户权限控制与安全认证机制，确保系统的数据

访问安全。MyBatis-Plus 的引入有效简化了数据库操作逻辑，提高了代码开发效率。Redis 技术为后续的高并发支持与缓存优化提供了可能性。

在前端部分，我采用 Vue3 + Element Plus 搭建了响应式页面，实现了登录、注册、图书展示、借阅管理、用户中心等主要功能模块。Axios 封装的请求逻辑，使得前后端通信更加高效清晰，路由守卫的设置也保证了访问权限的有效控制。

整个系统采用前后端分离架构，不仅增强了项目的可维护性和可扩展性，也符合现代 Web 应用的主流开发趋势。在实际开发过程中，我也通过 Swagger 文档工具对接口进行了有效管理，并编写了部分单元测试代码，保障系统稳定性。

通过本次实验，我加深了对信息管理系统开发流程的理解，掌握了多个主流框架与工具的使用方法，增强了实际项目开发的能力。