

兔子大学北京校区

毕 业 论 文

基于SpringBoot + Vue的图
书馆管理系统的设计与实现

学
专
学
学

业 名
生 姓

院： 计算机学院
称： 软件工程
名： 忧伤大白兔
号： 00000001

完成日期： xxxx 年 xx 月 xx 日

摘 要

本图书馆作为一个重要的文化机构，承载着图书借阅、阅读指导、学术研究等多种功能。然而，传统的图书馆管理方式存在一些问题，如图书管理效率低下、借阅归还流程繁琐、读者信息管理不便等。为了解决这些问题，基于计算机技术的图书馆管理系统应运而生。

该系统的设计目标是实现图书馆管理服务的高效、准确和方便。为了达到这个目标，系统采用了客户端-服务器模式，将图书馆的各项功能分为客户端和服务端两部分。客户端是指读者和管理员使用的用户界面，服务器是指图书馆管理系统的后台服务。

系统的开发语言采用了 Java 编程语言，这是一种功能强大且广泛应用的编程语言。数据库选择了 MySQL，这是一种常用的关系型数据库，具有高性能和稳定性。

该系统具有多种功能，包括图书借阅、归还、预约、续借、图书信息查询、读者信息管理等。读者可以通过系统查询图书信息，预约图书并进行借阅和归还操作。管理员可以通过登录系统进行图书的分类、增删改查等管理操作。

该系统还具有安全性和可扩展性。安全性方面，系统采用了用户登录认证机制，保证只有授权的用户才能使用系统。可扩展性方面，系统的设计考虑到了未来的功能扩展和升级，可以根据需要进行系统的改进和扩展。

用户界面是系统的重要组成部分，该系统的用户界面友好、简洁明了，易于操作。读者和管理员可以通过直观的界面完成各项操作，提高了用户的体验。

在实际使用中，该系统表现出了良好的稳定性和可靠性。系统经过多次测试和优化，可以在大量读者和图书的情况下保持良好的性能。同时，用户对该系统的评价也非常好，认为系统提供了便捷的图书管理服务，大大提高了图书馆的效率和服务质量。

该系统的开发为图书馆管理提供了一种新的思路和技术手段，也具有推广应用的潜力。随着科技的不断进步，图书馆管理系统将会越来越智能化和自动化，为读者提供更好的服务体验。同时，该系统的成功开发也为其他行业的管理系统提供了借鉴和参考。

目录

摘 要.....	I
第 1 章 绪论	1
1.1 选题背景	1
1.2 选题目的	1
1.3 选题意义	2
1.4 国内外研究情况	2
1.4.1 国内研究情况	2
1.4.2 国外研究情况	3
第 2 章 关键技术介绍	5
2.1 前端技术	5
2.1.1 Vue	5
2.1.2 Axios	6
2.2 后端技术	6
2.2.1 Spring Boot	6
2.2.2 MyBatis Plus	7
2.2.3 JWT	7
第 3 章 系统设计	8
3.1 功能结构设计	8
3.2 前端设计	9
3.2.1 前端接口封装	9
3.2.2 组件封装	9
3.2.3 动态路由	9
3.3 后端设计	9
3.3.1 全局异常处理	9
3.3.2 数据传输对象	9
3.3.3 系统管理模块物理模型	9
3.3.4 读者管理模块物理模型	11
3.3.5 图书管理模块物理模型	11

3.3.6 借阅管理模块物理模型	12
3.3.7 公告管理模块物理模型	12
3.4 数据库设计	13
3.4.1 ER 图	13
3.4.2 数据库结构设计	13
第 4 章 系统实现	19
4.1 登录	19
4.2 用户信息管理	23
4.3 修改密码	27
4.4 多条件分页查询	30
4.5 菜单分配	34
第 5 章 总结	39
参考文献	41

第 1 章 绪论

1.1 选题背景

随着数字化时代的到来，人们对信息的需求越来越大。图书馆作为信息传播的中心和知识管理的场所，承担着非常重要的社会职责。然而，传统的手工管理方式已经难以满足现代图书馆管理的需要，这就需要采用计算机技术来对图书馆管理进行自动化、智能化的处理，提高图书馆管理的效率和质量。

因此，开发一套图书馆管理系统具有很高的现实意义和实用价值。该系统不仅可以方便图书馆管理者对图书的采购、管理和借还等业务进行自动化处理，还可以为读者提供更加便捷和快速的服务，同时还可以实现对图书馆的资源利用率和管理效率的提升。

在传统的图书馆管理中，由于人工操作存在繁琐、误差等问题，会导致图书管理效率低下、流程不顺畅等问题，同时也会对读者的借阅体验产生一定的影响。而采用计算机技术对图书馆管理进行自动化处理，可以避免这些问题的出现，提高图书馆管理的效率和服务质量。

除此之外，随着互联网的普及，图书馆的服务需求也发生了变化。读者可以通过网络进行图书查询和借阅，这也为图书馆管理系统的开发提供了更多的机会和需求。因此，设计和开发一套基于计算机技术的图书馆管理系统，不仅有助于提高图书馆管理的效率和质量，还可以满足现代读者对图书馆服务的需求。

1.2 选题目的

本项目的目的是设计和开发一套基于计算机技术的图书馆管理系统，以实现对图书馆管理的自动化、智能化处理，提高图书馆管理的效率和服务质量，满足现代读者对图书馆服务的需求。

具体来说，本项目的目标如下：

实现对图书的自动化管理。通过采用条码标识和管理技术，实现图书借还、查询、入库等业务的自动化处理，避免了手工管理的繁琐和误差。

实现读者的自助注册和借阅。实现读者自助注册和借阅的便捷和快速，提高了读者的借阅体验。

实现图书馆业务的智能化处理。通过并发控制技术和事务管理机制，实现图书借阅、预约等业务的智能化处理，避免了多用户同时操作所可能带来的冲突和数据不一致性问题。

提高图书馆管理的效率和服务质量。通过采用计算机技术对图书馆管理进行自动化、智能化处理，提高图书馆管理的效率和服务质量，满足现代读者对图书馆服务的需求。

总之，本项目旨在设计和开发一套功能齐全、操作简便、性能稳定的图书馆管理系统，为图书馆管理和服务提供一种全新的思路和方法，促进图书馆的现代化和信息化。

1.3 选题意义

提高图书馆管理效率和服务质量。通过采用计算机技术对图书馆管理进行自动化、智能化处理，可以提高图书馆管理的效率和服务质量。这有助于解决传统图书馆管理中存在的繁琐、误差等问题，同时也能够提高读者的借阅体验。

满足现代读者对图书馆服务的需求。随着互联网的普及，读者对图书馆服务的需求也发生了变化。读者希望能够通过网络进行图书查询和借阅，享受更加便捷、快速的服务。本项目的设计和开发可以满足这些需求，提高图书馆服务的现代化程度和用户体验。

推动图书馆信息化建设。图书馆作为信息传播和知识管理的中心，其信息化建设对于现代社会的发展具有重要意义。本项目的设计和开发可以推动图书馆信息化建设，促进图书馆的现代化和信息化。

总之，本项目的选题意义在于推动图书馆管理和服务的现代化和信息化，提高图书馆管理效率和服务质量，满足现代读者对图书馆服务的需求，促进图书馆信息化建设。

1.4 国内外研究情况

1.4.1 国内研究情况

近年来，国内对图书馆管理系统的研究得到了较为广泛的关注和探讨。许多研究者致力于开发和优化图书馆管理系统，以满足不断增长的图书馆需求和提升服务质量。

在技术方面，国内研究者主要关注系统架构和技术实现。例如，一些研究聚焦于图书馆管理系统的客户端-服务器架构，通过使用 Java 编程语言和 MySQL 数据库等技术实现系统的各项功能。同时，一些研究也探索了前后端分离技术的应用，如使用 Vue、

ElementUI 和 Axios 等技术开发用户界面，使用 Spring Boot、MyBatis Plus 和 Jwt 等技术开发后台服务。

在功能方面，国内研究者注重系统的实用性和便捷性。他们关注借阅管理、图书查询、读者信息管理等核心功能，并通过系统的用户界面设计和操作流程优化，提升用户体验。此外，一些研究也关注系统的安全性和可扩展性，如引入用户登录认证机制和考虑未来功能扩展的设计。

在应用方面，国内研究者将图书馆管理系统应用于实际图书馆中，并对系统的稳定性和可靠性进行了测试和验证。研究者通过与图书馆管理员和读者的合作，收集反馈和意见，不断改进和完善系统，以满足实际需求。

国内研究者在图书馆管理系统方面的研究取得了一定的成果，为图书馆管理提供了新的思路和技术手段。然而，与国外研究相比，国内研究在一些方面还有待进一步深入和拓展。

1.4.2 国外研究情况

国外研究者在图书馆管理系统方面的研究也取得了显著的进展。他们关注系统的创新和智能化，致力于提供更高效、智能和个性化的图书馆管理服务。

在技术方面，国外研究者不仅关注传统的客户端-服务器架构，还尝试了云计算和大数据等新技术的应用。他们通过将系统部署在云平台上，提供弹性和可扩展的服务。同时，他们利用大数据分析和机器学习等技术，对读者的阅读偏好和借阅行为进行分析，为图书馆提供个性化推荐和服务。

在功能方面，国外研究者注重系统的智能化和自动化。他们开发了自动借还书机、智能图书柜等设备，提高图书的借阅和归还效率。同时，他们关注系统的社交化功能，如用户之间的评论和分享，以促进读者之间的交流和合作。

在应用方面，国外研究者将图书馆管理系统广泛应用于各类图书馆，包括学校图书馆、公共图书馆和专业图书馆等。他们通过与图书馆界的合作和交流，不断改进和优化系统，提供更好的服务。

综上所述，国内外对图书馆管理系统的研究都取得了一定的成果。国内研究主要关注系统架构和功能实现，注重实用性和便捷性；国外研究则更加关注系统的创新和智能

化，注重云计算、大数据和智能设备的应用。这些研究成果为图书馆管理提供了丰富的思路和技术支持，为图书馆提供更高效、智能和个性化的服务。

第 2 章 关键技术介绍

2.1 前端技术

2.1.1 Vue

Vue.js 是一款轻量级的 JavaScript 框架，用于构建用户界面。它采用了 MVVM (Model-View-ViewModel) 的架构模式，能够更好地组织和管理页面的代码。Vue.js 的设计目标是尽可能简单、灵活和易于上手，同时也具备足够的扩展性和性能。

Vue.js 具有以下特点：

响应式数据绑定：Vue.js 使用了双向绑定的机制，能够将数据与 DOM 元素进行关联，当数据发生变化时，页面会自动更新。这样的特性使得开发者无需手动操作 DOM，大大简化了开发流程。

组件化开发：Vue.js 将页面划分为多个组件，每个组件负责管理自己的状态和视图。组件可以嵌套使用，形成复杂的页面结构。这种组件化开发的方式使得代码更加模块化、可复用性更高，便于团队协作和维护。

虚拟 DOM：Vue.js 采用了虚拟 DOM 的技术，通过将页面抽象成一个虚拟的 JavaScript 对象，进行 DOM 操作的计算和比对，最终只更新需要变动的部分，提高了页面渲染的效率。

插件系统：Vue.js 提供了丰富的插件系统，可以方便地扩展其功能。开发者可以根据自己的需求，选择合适的插件来增加特定的功能，如路由管理、状态管理、表单验证等。

生态系统：Vue.js 拥有庞大的生态系统，有大量的第三方库和组件可供使用。开发者可以通过官方提供的 CLI 工具或者直接引入 CDN 来快速搭建项目，并且可以方便地与其他前端技术进行集成。

总之，Vue.js 是一款非常优秀的前端框架，它简单易学，但又具备强大的功能和扩展性。无论是开发小型项目还是大型应用，Vue.js 都能够提供良好的开发体验和高效的性能。如果你想从事前端开发，Vue.js 绝对是一个值得学习和掌握的技术。

2.1.2 Axios

Axios 是一个基于 promise 的 HTTP 库，可以用在浏览器和 node.js 中，用于前端向后端发起请求，它拥有全局的请求和响应的拦截，可以非常方便的处理请求异常的问题。

2.2 后端技术

2.2.1 Spring Boot

Spring Boot 是一个基于 Spring 框架的快速开发框架，旨在简化 Java 应用程序的配置和部署。它提供了一种约定大于配置的方式，让开发者可以更专注于业务逻辑的实现，而不必花费太多时间和精力在繁琐的配置上。

Spring Boot 具有以下特点：

简化配置：Spring Boot 通过自动配置的方式，根据应用程序的依赖和环境自动配置相关的设置，大大减少了繁琐的配置工作。开发者只需要提供少量的配置，即可快速搭建一个可运行的应用程序。

内嵌服务器：Spring Boot 内置了多种常用的服务器，如 Tomcat、Jetty 等，开发者无需单独部署服务器，可以直接运行应用程序。这样可以简化部署过程，提高开发效率。

自动化依赖管理：Spring Boot 通过提供一个强大的依赖管理系统，可以自动解析和导入所需的依赖库。开发者只需要在配置文件中声明所需的依赖，Spring Boot 会自动下载并导入相关的库文件。

健康监测：Spring Boot 提供了健康监测的功能，可以监控应用程序的运行状态和性能指标。开发者可以通过配置文件或者注解来开启健康监测，方便进行应用程序的运维和故障排查。

强大的生态系统：Spring Boot 拥有庞大的生态系统，有丰富的第三方库和插件可供选择。开发者可以根据自己的需求，选择合适的插件来增加特定的功能，如数据库访问、缓存管理、消息队列等。

总之，Spring Boot 是一个简单高效的 Java 开发框架，它通过简化配置和内嵌服务器等特性，大大提高了开发效率和部署便利性。无论是开发小型应用还是大型企业级应

用，Spring Boot 都能够提供强大的支持和灵活的扩展性。如果你想从事 Java 后端开发，Spring Boot 绝对是一个值得学习和掌握的技术。

2.2.2 MyBatis Plus

MyBatis 是一款优秀的持久层框架，通过 XML 文件或注解配置来完成实体类与数据库之间的映射，舍弃了传统的 `preparedStatement` 设置参数操作数据库和使用 `resultSet` 获取结果集的过程。

MyBatis Plus 是由苞米豆团队开发的一款 MyBatis 增强工具，为简化数据库操作，提高开发效率为生。在 MyBatis 的基础上提供了常用的 `crud` 方法，甚至不需要配置 `Mapper.xml` 文件都能对数据库进行基础的操作。除此之外，MyBatis Plus 还提供了自动分页、代码生成的功能，通过配置相应的模板，就能一键生成绝大部分的后端代码，真正做到了简化开发。

2.2.3 JWT

JWT 全称 JSON Web Token，是目前比较流行跨域验证方案。相比于 `session`，`session` 生成的用户数据都会保存在服务器端，服务器只给用户的返回一个 `sessionId`，下次访问这个网站时，通过 `cookie` 将 `sessionId` 传递给服务器，从而得到相关的用户信息。毫无疑问，在这种情况下，服务器的内存会被大大的消耗，会带来一些性能开销。若服务器突然宕机，保存在服务器的用户数据就会消失，用户再次访问服务器就会被认为是第一次登录。

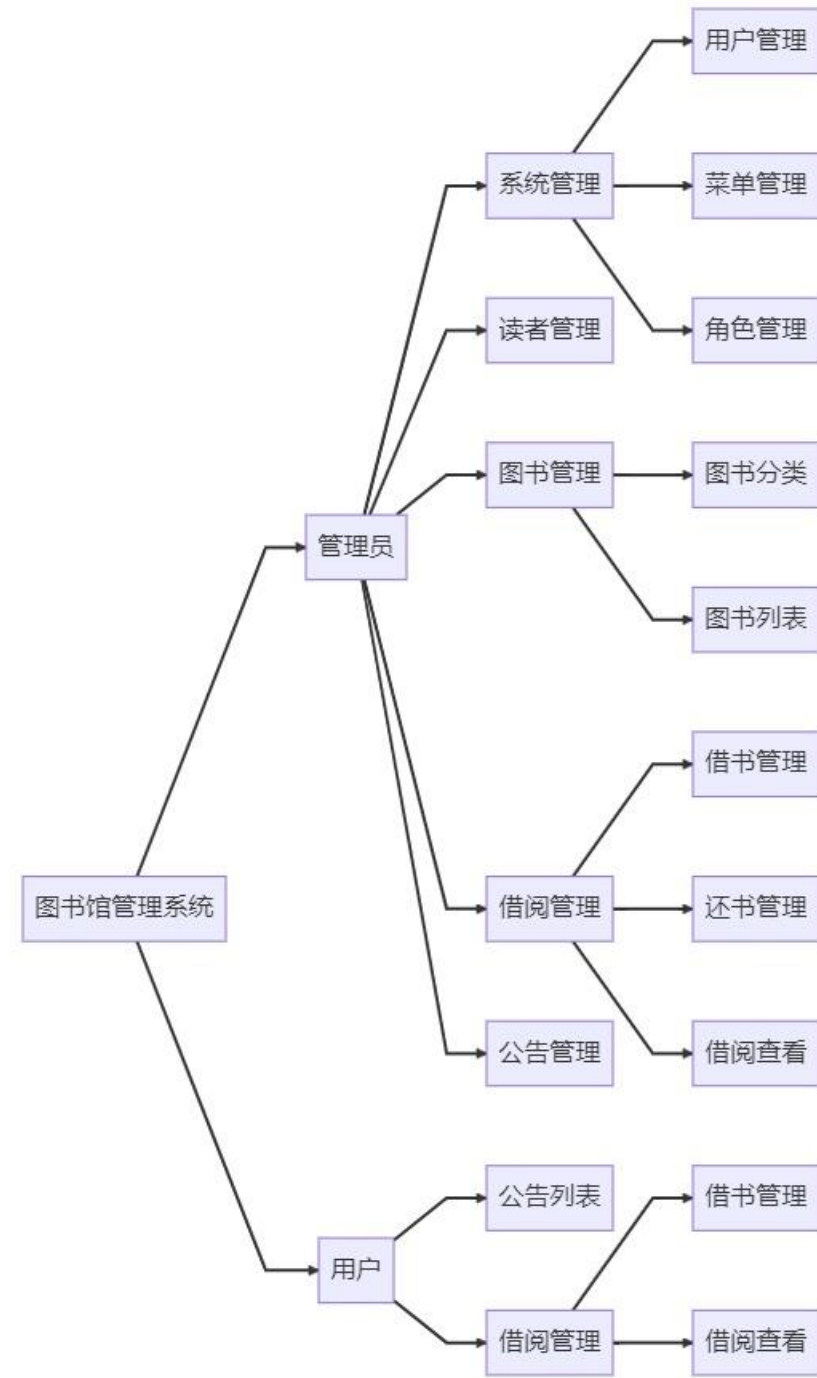
而 JWT 是保存在浏览器本地的，当用户第一次访问服务器，并且登录成功了，服务器会根据用户的唯一标识信息（比如 `id`），生成一个加密的 `token`，并返回用户信息。只要用户每次访问服务器的时候，在请求头中携带上 `token`，后端的拦截器获取 `token`，验证签证信息通过之后，就允许访问。

第 3 章 系统设计

3.1 功能结构设计

本系统：管理员主要分五个模块，分别是系统管理和读者管理、图书管理、借阅管理，公告管理。

用户主要分为两个模块借阅管理，公告列表。



系统功能结构图

3.2 前端设计

3.2.1 前端接口封装

本项目对 Axios 进行了全局的封装，对前端请求和后端响应进行了统一的拦截，并进行相应的处理。前端调用的 Api 都封装在 src/api 模块下，进行统一的管理。

3.2.2 组件封装

为了解决代码复用的问题，通过结合 Element UI。本项目对 form 表单和 table 数据表进行了进一步的组件封装。

3.2.3 动态路由

本项目采用了基于后端权限菜单的来实现动态路由，为了保证菜单数据的全局共享，

菜单数据使用 vuex 来保存。当员工访路由时，通过全局路由守卫进行拦截，并向后端请求该员工的菜单数据。

3.3 后端设计

3.3.1 全局异常处理

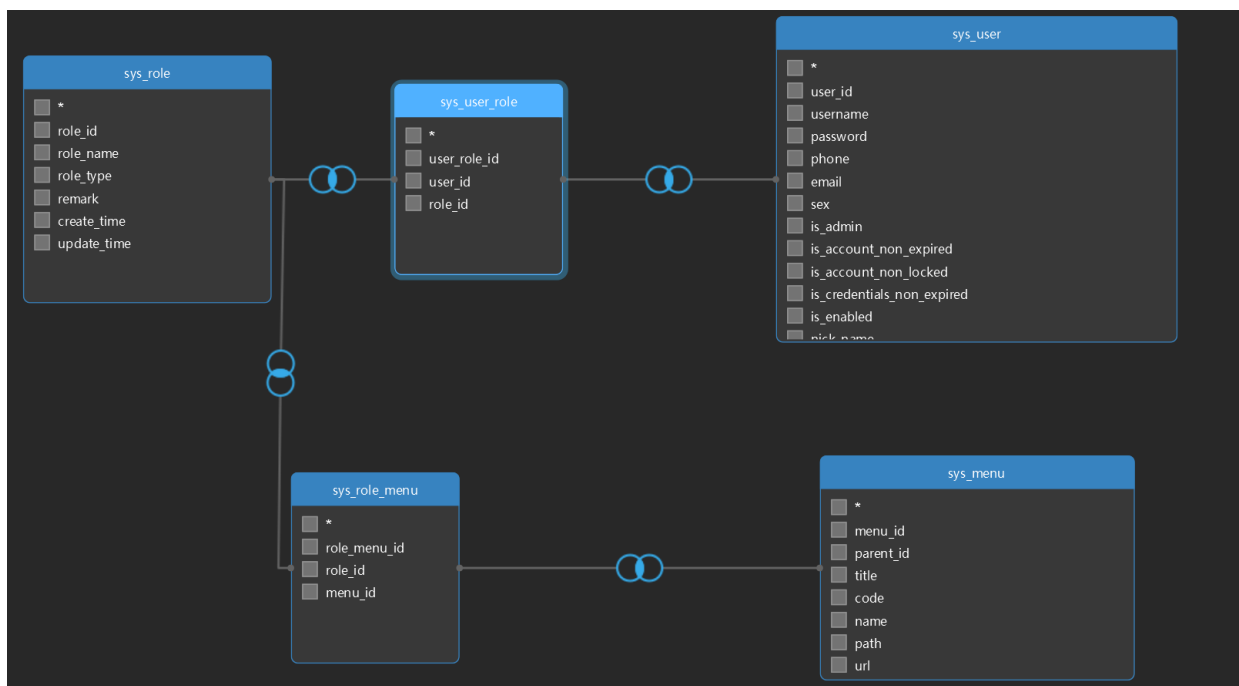
为了高效地处理异常，本项目对异常进行了全局的统一处理，使用 @GlobalExceptionHandler 声明一个全局异常处理器，在需要异常处理的地方抛出自定义的异常。

3.3.2 数据传输对象

本项目中，后端响应给前端的数据都统一封装在 ResultUtils 中，然后前端通过解析得到 ResultUtils 的 json 对象。通过定义全局的业务状态码枚举类。前端通过后端响应数据的状态码来判断业务处理是否异常。

3.3.3 系统管理模块物理模型

系统管理模块主要涉及 5 张表，负责对用户、权限、以及菜单信息进行保存。



系统管理模块物理模型

用户表：

用户表包含了用户的基本信息，如用户名、邮箱、电话等。

名	类型	长度	小数点	不是 null	虚拟	键	注释
user_id	int			<input checked="" type="checkbox"/>	<input type="checkbox"/>	1	用户id
username	varchar	36		<input type="checkbox"/>	<input type="checkbox"/>		登录账户
password	varchar	128		<input type="checkbox"/>	<input type="checkbox"/>		登录密码
phone	varchar	13		<input type="checkbox"/>	<input type="checkbox"/>		用户电话
email	varchar	64		<input type="checkbox"/>	<input type="checkbox"/>		邮箱
sex	varchar	2		<input type="checkbox"/>	<input type="checkbox"/>		0:男 1: 女
is_admin	varchar	2		<input type="checkbox"/>	<input type="checkbox"/>		是否为超级管理员 1: 是 0: 否
is_account_non_expired	tinyint			<input type="checkbox"/>	<input type="checkbox"/>		帐户是否过期(1 未过期, 0已过期)
is_account_non_locked	tinyint			<input type="checkbox"/>	<input type="checkbox"/>		帐户是否被锁定(1 未锁定, 0已锁定)
is_credentials_non_expired	tinyint			<input type="checkbox"/>	<input type="checkbox"/>		密码是否过期(1 未过期, 0已过期)
is_enabled	tinyint			<input type="checkbox"/>	<input type="checkbox"/>		帐户是否可用(1 可用, 0 删除用户)
nick_name	varchar	36		<input type="checkbox"/>	<input type="checkbox"/>		姓名
create_time	datetime			<input type="checkbox"/>	<input type="checkbox"/>		创建时间
update_time	datetime			<input type="checkbox"/>	<input type="checkbox"/>		更新时间

权限表：

用于存放用户的权限信息。

名	类型	长度	小数点	不是 null	虚拟	键	注释
role_id	int			<input checked="" type="checkbox"/>	<input type="checkbox"/>	1	角色id
role_name	varchar	36		<input type="checkbox"/>	<input type="checkbox"/>		角色名称
role_type	varchar	2		<input type="checkbox"/>	<input type="checkbox"/>		角色类型 1: 系统用户 2: 读者
remark	varchar	64		<input type="checkbox"/>	<input type="checkbox"/>		备注
create_time	datetime			<input type="checkbox"/>	<input type="checkbox"/>		创建时间
update_time	datetime			<input type="checkbox"/>	<input type="checkbox"/>		更新时间

菜单表：

用于存放用户的权限可以操作什么数据，以及赋予权限的时间。

名	类型	长度	小数点	不是 null	虚拟	键	注释
menu_id	int			<input checked="" type="checkbox"/>	<input type="checkbox"/>	1	菜单id
parent_id	int			<input type="checkbox"/>	<input type="checkbox"/>		父级菜单id
title	varchar	64		<input type="checkbox"/>	<input type="checkbox"/>		菜单名称
code	varchar	36		<input type="checkbox"/>	<input type="checkbox"/>		权限字段
name	varchar	36		<input type="checkbox"/>	<input type="checkbox"/>		路由name
path	varchar	36		<input type="checkbox"/>	<input type="checkbox"/>		路由path
url	varchar	128		<input type="checkbox"/>	<input type="checkbox"/>		组件路径
type	varchar	2		<input type="checkbox"/>	<input type="checkbox"/>		类型(0 目录 1菜单, 2按钮)
icon	varchar	36		<input type="checkbox"/>	<input type="checkbox"/>		菜单图标
parent_name	varchar	64		<input type="checkbox"/>	<input type="checkbox"/>		上级菜单名称
order_num	int			<input type="checkbox"/>	<input type="checkbox"/>		序号
create_time	datetime			<input type="checkbox"/>	<input type="checkbox"/>		创建时间
update_time	datetime			<input type="checkbox"/>	<input type="checkbox"/>		更新时间

3.3.4 读者管理模块物理模型

读者表：用于存放读者的基本信息

名	类型	长度	小数点	不是 null	虚拟	键	注释
reader_id	int			<input checked="" type="checkbox"/>	<input type="checkbox"/>	1	读者id
learn_num	varchar	36		<input type="checkbox"/>	<input type="checkbox"/>		学生证号码
username	varchar	36		<input type="checkbox"/>	<input type="checkbox"/>		姓名
id_card	varchar	20		<input type="checkbox"/>	<input type="checkbox"/>		身份证号码
sex	varchar	2		<input type="checkbox"/>	<input type="checkbox"/>		性别
phone	varchar	15		<input type="checkbox"/>	<input type="checkbox"/>		电话
password	varchar	128		<input type="checkbox"/>	<input type="checkbox"/>		密码
type	varchar	2		<input type="checkbox"/>	<input type="checkbox"/>		类型
check_status	varchar	2		<input type="checkbox"/>	<input type="checkbox"/>		审核状态 0: 未审核 1: 已审核
user_status	varchar	2		<input type="checkbox"/>	<input type="checkbox"/>		用户状态 0:停用 1: 启用
class_name	varchar	64		<input type="checkbox"/>	<input type="checkbox"/>		班级

3.3.5 图书管理模块物理模型

图书信息表：

用于保存图书的信息 如作者，图书名称，上架时间等

名	类型	长度	小数点	不是 null	虚拟	键	注释
book_id	int			<input checked="" type="checkbox"/>	<input type="checkbox"/>	1	图书id
category_id	int			<input type="checkbox"/>	<input type="checkbox"/>		分类id
book_name	varchar	128		<input type="checkbox"/>	<input type="checkbox"/>		图书名称
book_code	varchar	128		<input type="checkbox"/>	<input type="checkbox"/>		图书条码
book_place_num	varchar	128		<input type="checkbox"/>	<input type="checkbox"/>		书架号
book_auther	varchar	64		<input type="checkbox"/>	<input type="checkbox"/>		作者
book_product	varchar	128		<input type="checkbox"/>	<input type="checkbox"/>		出版社
book_price	decimal	18	2	<input type="checkbox"/>	<input type="checkbox"/>		价格
book_store	int			<input type="checkbox"/>	<input type="checkbox"/>		库存


图书分类表：

主要保存图书的分类信息以及对应的图书 id

名	类型	长度	小数点	不是 null	虚拟	键	注释
category_id	int			<input checked="" type="checkbox"/>	<input type="checkbox"/>	1	分类id
category_name	varchar	64		<input type="checkbox"/>	<input type="checkbox"/>		分类名称
order_num	int			<input type="checkbox"/>	<input type="checkbox"/>		序号

3.3.6 借阅管理模块物理模型


主要保存用户借书的时间，还书的时间，以及对应图书的标识

名	类型	长度	小数点	不是 null	虚拟	键	注释
▶ borrow_id	int			<input checked="" type="checkbox"/>	<input type="checkbox"/>	 1	主键
book_id	int			<input type="checkbox"/>	<input type="checkbox"/>		图书id
reader_id	int			<input type="checkbox"/>	<input type="checkbox"/>		读者id
borrow_time	datetime			<input type="checkbox"/>	<input type="checkbox"/>		借书时间
return_time	datetime			<input type="checkbox"/>	<input type="checkbox"/>		还书时间
apply_status	varchar	2		<input type="checkbox"/>	<input type="checkbox"/>		0: 待审核 1: 已审核 2: 拒绝
borrow_status	varchar	2		<input type="checkbox"/>	<input type="checkbox"/>		0: 审核中 1: 在借中 2: 已还 3: 拒绝
return_status	varchar	2		<input type="checkbox"/>	<input type="checkbox"/>		1: 正常还书 2: 异常还书 3: 丢失
exception_text	varchar	128		<input type="checkbox"/>	<input type="checkbox"/>		异常还书备注
apply_text	varchar	128		<input type="checkbox"/>	<input type="checkbox"/>		审核拒绝备注

3.3.7 公告管理模块物理模型

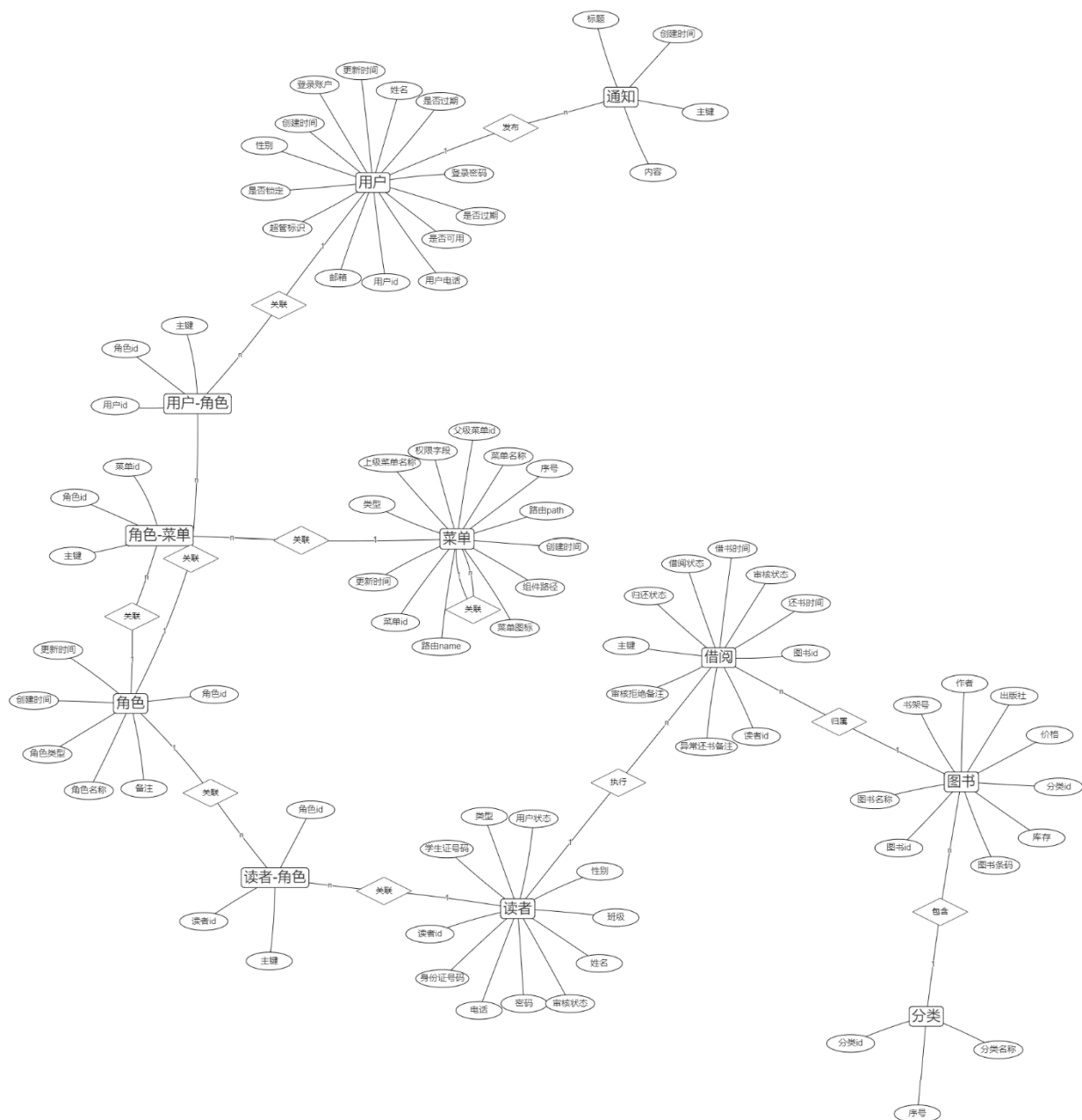
公告信息表：

主要保存公告的内容与发布的时间

名	类型	长度	小数点	不是 null	虚拟	键	注释
▶ notice_id	int			<input checked="" type="checkbox"/>	<input type="checkbox"/>	 1	主键
notice_title	varchar	128		<input type="checkbox"/>	<input type="checkbox"/>		标题
notice_content	text			<input type="checkbox"/>	<input type="checkbox"/>		内容
create_time	datetime			<input type="checkbox"/>	<input type="checkbox"/>		创建时间

3.4 数据库设计

3.4.1 ER 图



3.4.2 数据库结构设计

借阅 borrow_book

列名	数据类型	备注	长度
borrow_id	int	主键	
book_id	int	图书 id	

reader_id	int	读者 id	
borrow_time	datetime	借书时间	
return_time	datetime	还书时间	
apply_status	varchar	审核状态	2
borrow_status	varchar	借阅状态	2
return_status	varchar	归还状态	2
excepion_text	varchar	异常还书备注	128
apply_text	varchar	审核拒绝备注	128

图书 sys_books

列名	数据类型	备注	长度
book_id	int	图书 id	
category_id	int	分类 id	
book_name	varchar	图书名称	128
book_code	varchar	图书条码	128
book_place_num	varchar	书架号	128
book_auther	varchar	作者	64
book_product	varchar	出版社	128
book_price	decimal	价格	
book_store	int	库存	

分类 sys_category

列名	数据类型	备注	长度
category_id	int	分类 id	
category_name	varchar	分类名称	64
order_num	int	序号	

菜单 sys_menu

列名	数据类型	备注	长度
menu_id	int	菜单 id	
parent_id	int	父级菜单 id	
title	varchar	菜单名称	64
code	varchar	权限字段	36
name	varchar	路由 name	36
path	varchar	路由 path	36
url	varchar	组件路径	128
type	varchar	类型	2
icon	varchar	菜单图标	36
parent_name	varchar	上级菜单名称	64
order_num	int	序号	
create_time	datetime	创建时间	
update_time	datetime	更新时间	

通知 sys_notice

列名	数据类型	备注	长度
notice_id	int	主键	
notice_title	varchar	标题	128
notice_content	text	内容	65535
create_time	datetime	创建时间	

读者 sys_reader

列名	数据类型	备注	长度
reader_id	int	读者 id	
learn_num	varchar	学生证号码	36
username	varchar	姓名	36
id_card	varchar	身份证号码	20
sex	varchar	性别	2
phone	varchar	电话	15
password	varchar	密码	128
type	varchar	类型	2
check_status	varchar	审核状态	2
user_status	varchar	用户状态	2
class_name	varchar	班级	64

读者-角色 sys_reader_role

列名	数据类型	备注	长度
----	------	----	----

reader_role_id	int	主键	
reader_id	int	读者 id	
role_id	int	角色 id	

角色 sys_role

列名	数据类型	备注	长度
role_id	int	角色 id	
role_name	varchar	角色名称	36
role_type	varchar	角色类型	2
remark	varchar	备注	64
create_time	datetime	创建时间	
update_time	datetime	更新时间	

角色-菜单 sys_role_menu

列名	数据类型	备注	长度
role_menu_id	int	主键	
role_id	int	角色 id	
menu_id	int	菜单 id	

用户 sys_user

列名	数据类型	备注	长度
----	------	----	----

user_id	int	用户 id	
username	varchar	登录账户	36
password	varchar	登录密码	128
phone	varchar	用户电话	13
email	varchar	邮箱	64
sex	varchar	性别	2
is_admin	varchar	超管标识	2
is_account_non_expired	tinyint	是否过期	
is_account_non_locked	tinyint	是否锁定	
is_credentials_non_expired	tinyint	是否过期	
is_enabled	tinyint	是否可用	
nick_name	varchar	姓名	36
create_time	datetime	创建时间	
update_time	datetime	更新时间	

用户-角色 sys_user_role

列名	数据类型	备注	长度
user_role_id	int	主键	
user_id	int	用户 id	
role_id	int	角色 id	

第 4 章 系统实现

4.1 登录

此模块完成了管理员用户的登录功能，管理员用户通过用户名和密码进行登录。若用户状态异常则无法登录。



登录页面

登录页面代码：

```
<template>
  <div class="logincontainer">
    <div class="login_box">
      <div>
        <el-form ref="loginForm"
class="loginForm" :model="loginForm" :rules="rules" label-
width="80px" :inline="false"
size="normal">
          <el-form-item>
            <div>
              
              <span class="loginTitle">图书馆管理系统</span>
            </div>
          </el-form-item>
          <el-form-item prop="username">
```

```

        <el-input v-model="loginForm.username" placeholder="请输入用户名">
            <svg-icon slot="prefix" icon-class="user" class="el-input__icon
input-icon" />
        </el-input>
    </el-form-item>
    <el-form-item prop="password">
        <el-input v-model="loginForm.password" type="password" placeholder="
请输入密码">
            <svg-icon slot="prefix" icon-class="password" class="el-
input__icon input-icon" />
        </el-input>
    </el-form-item>
    <el-form-item prop="userType">
        <el-radio-group v-model="loginForm.userType">
            <el-radio :label="0">读者</el-radio>
            <el-radio :label="1">管理员</el-radio>
        </el-radio-group>
    </el-form-item>
    <el-form-item>
        <el-row :gutter="20">
            <el-col :span="24" :offset="0">
                <el-button type="primary" class="mybtn"
size="medium" :disabled="loading"
style="width:100%;background-color: #0782c2"
@click="onSubmit">
                    <span v-if="!loading">登 录</span>
                    <span v-else>登 录 中...</span>
                </el-button>
            </el-col>
        </el-row>
    </el-form-item>
    <el-form-item>
        <span style="color: #1d2123;float: right;margin-right: 15px;cursor:
pointer;"
@click="registerBtn">读者注册</span>
    </el-form-item>
</el-form>
</div>
</div>
<!-- 注册弹框 -->
<sys-
dialog :title="dialog.title" :width="dialog.width" :height="dialog.height" :visi
ble="dialog.visible"
@onClose="onClose" @onConfirm="onConfirm">
    <div slot="content">
        <el-form ref="addRef" :model="addModel" :rules="registeRules" label-
width="80px" style="margin-right: 30px">
            <el-row>

```



```

    <el-col :span="12" :offset="0">
      <el-form-item prop="learnNum" label="姓名">
        <el-input v-model="addModel.learnNum" />
      </el-form-item>
    </el-col>
    <el-col :span="12" :offset="0">
      <el-form-item prop="phone" label="电话">
        <el-input v-model="addModel.phone" maxlength="11" />
      </el-form-item>
    </el-col>
  </el-row>
  <el-row>
    <el-col :span="12" :offset="0">
      <el-form-item prop="username" label="学号">
        <el-input v-model="addModel.username" />
      </el-form-item>
    </el-col>
    <el-col :span="12" :offset="0">
      <el-form-item prop="className" label="班级">
        <el-input v-model="addModel.className" />
      </el-form-item>
    </el-col>
  </el-row>
  <el-row>
    <el-col :span="12" :offset="0">
      <el-form-item prop="idCard" label="身份证">
        <el-input v-model="addModel.idCard" maxlength="18" />
      </el-form-item>
    </el-col>
    <el-col :span="12" :offset="0">
      <el-form-item prop="password" label="密码">
        <el-input v-model="addModel.password" type="password" />
      </el-form-item>
    </el-col>
  </el-row>
  <el-row>
    <el-col :span="12" :offset="0">
      <el-form-item prop="confirmPassword" label="确认密码">
        <el-input v-model="addModel.confirmPassword" type="password" />
      </el-form-item>
    </el-col>
    <el-col :span="12" :offset="0">
      <el-form-item label="性别">
        <el-radio-group v-model="addModel.sex">
          <el-radio :label="'0'">男</el-radio>
          <el-radio :label="'1'">女</el-radio>
        </el-radio-group>
      </el-form-item>
    </el-col>
  </el-row>

```

```

        </el-col>
    </el-row>
</el-form>
</div>
</sys-dialog>
</div>
</template>

```

登录接口:

```

@PostMapping("/login")
public ResultVo login(@RequestBody LoginParm loginParm) {

```

接口说明:

当管理员用户填写好登录账号和密码之后, 前端会提交登录表单。后端接收到数据之后, 会进行数据库查询, 并将最终的查询结果以状态码的形式封装在 ResultUtils 之中, 并返回到前端。

核心代码:

```

//判断是读者还是管理员
if (loginParm.getUserType().equals("0")) { // 0:读者
    //根据读者的用户名和密码查询
    QueryWrapper<SysReader> query = new QueryWrapper<>();
    query.lambda().eq(SysReader::getUsername, loginParm.getUsername())
        .eq(SysReader::getPassword, DigestUtils.md5DigestAsHex(loginParm.getPassword().getBytes()));
    SysReader one = sysReaderService.getOne(query);
    if (one == null) {
        return ResultUtils.error(msg: "用户名或密码错误!");
    }
    //返回数据给前端
    LoginResult result = new LoginResult();
    result.setToken(jwtUtils.generateToken(one.getUsername(), loginParm.getUserType()));
    result.setUserId(one.getReaderId());
    return ResultUtils.success(msg: "登录成功", result);
}

```

登录流程:

首先通过前端传递的账号和密码进行用户查询, 之后对查询到的员工进行状态判断, 如果状态正常, 则将用户的信息和生成的 token 返回给前端, 前端将 token 保存在 localStorage 中。

4.2 用户信息管理

此模块实现了用户管理员个人信息的查看与修改



用户管理列表界面代码:

```
<template>
  <el-main>
    <el-form
      ref="searchRef"
      :model="listParm"
      label-width="80px"
      :inline="true"

      :rules="rules"
    >
      <el-form-item>
        <el-input
          v-model="listParm.nickName"
          placeholder="请输入姓名"
        />
      </el-form-item>
      <el-form-item>
        <el-input v-model="listParm.phone" placeholder="请输入电话" />
      </el-form-item>
      <el-form-item>
        <el-button @click="searchBtn"> 搜索</el-button>
        <el-button
          style="color: #ff7670"
          @click="resetBtn"
        > 重置</el-button>
      </el-form-item>
    </el-form>
  </el-main>
</template>
```

```
      v-permission="['sys:user:add']"
      type="primary"

      @click="addBtn"
    > 新增</el-button>
  </el-form-item>
</el-form>
<!-- 表格
  data: 表格的数据
  -->
<el-table :height="tableHeight" :data="tableData" border stripe>
  <el-table-column prop="nickName" label="姓名" />
  <el-table-column prop="phone" label="电话" />
  <el-table-column prop="email" label="邮箱" />
  <el-table-column label="操作" fixed="right" width="460">
    <template slot-scope="scope">
      <el-button
        v-permission="['sys:user:edit']"
        type="primary"
        @click="editBtn(scope.row)"
      > 编辑</el-button>
      <el-button
        v-permission="['sys:user:resetpassword']"
        type="warning"
        @click="resetPasswordBtn(scope.row)"
      > 重置密码</el-button>
      <el-button
        v-permission="['sys:user:delete']"
        type="danger"
        @click="deleteBtn(scope.row)"
      > 删除</el-button>
    </template>
  </el-table-column>
</el-table>
<!-- 分页 -->
<el-pagination
  :current-page.sync="listParm.currentPage"
  :page-sizes="[10, 20, 40, 80, 100]"
  :page-size="listParm.pageSize"
  layout="total, sizes, prev, pager, next, jumper"
  :total="listParm.total"
  background
  @size-change="sizeChange"
  @current-change="currentChange"
/>
<!-- 新增或编辑弹框 -->
<sys-dialog
  :title="dialog.title"
```

```
:visible="dialog.visible"
:width="dialog.width"
:height="dialog.height"
@onClose="onClose"
@onConfirm="onConfirm"
>
<div slot="content">
  <!-- el-form : 当做一个普通的 form 标签
    model : 表单绑定的数据对象
    ref : 相当于 div 的 id , 唯一的
    rules : 表单验证规则
    label-width : 表单域标签的宽度
    inline : 是否在同一行展示
    size : 尺寸
  -->
  <el-form
    ref="addRef"
    :model="addModel"
    :rules="rules"
    label-width="80px"
    :inline="false"

    style="margin-right: 40px"
  >
    <!-- el-row : 代表一行, 分为 24 等分
      el-col : 代表列
    -->
    <el-row>
      <el-col :span="12" :offset="0">
        <el-form-item prop="nickName" label="姓名">
          <el-input v-model="addModel.nickName" />
        </el-form-item>
      </el-col>
      <el-col :span="12" :offset="0">
        <el-form-item prop="phone" label="电话">
          <el-input v-model="addModel.phone" />
        </el-form-item>
      </el-col>
    </el-row>
    <el-row>
      <el-col :span="12" :offset="0">
        <el-form-item label="邮箱">
          <el-input v-model="addModel.email" />
        </el-form-item>
      </el-col>
      <el-col :span="12" :offset="0">
        <el-form-item prop="sex" label="性别">
          <el-radio-group v-model="addModel.sex">
```

```

        <el-radio :label="'0'">男</el-radio>
        <el-radio :label="'1'">女</el-radio>
    </el-radio-group>
</el-form-item>
</el-col>
</el-row>
<el-row>
    <el-col :span="12" :offset="0">
        <el-form-item prop="username" label="账户">
            <el-input v-model="addModel.username" />
        </el-form-item>
    </el-col>
    <el-col v-if="addModel.type == '0'" :span="12" :offset="0">
        <el-form-item prop="password" label="密码">
            <el-input v-model="addModel.password" />
        </el-form-item>
    </el-col>
</el-row>
<el-row>
    <el-col :span="12" :offset="0">
        <el-form-item prop="username" label="角色">
            <el-select v-model="addModel.roleId" placeholder="请选择">
                <el-option
                    v-for="item in options"
                    :key="item.roleId"
                    :label="item.roleName"
                    :value="item.roleId"
                />
            </el-select>
        </el-form-item>
    </el-col>
</el-row>
</el-form>
</div>
</sys-dialog>
</el-main>
</template>

```

```

@Auth
@PutMapping
public ResultVo editUser(@RequestBody SysUser user)

```

接口说明:

员工填写的信息会通过 el-form 组件的 data 属性，将数据绑定到一个 json 对象中，并通过 put 提交，最后后端接收数据，并完成相应员工信息的更新。

核心代码:

```
public ResultVo editUser(@RequestBody SysUser user){
    //判断账户是否被占用
    QueryWrapper<SysUser> query = new QueryWrapper<>();
    query.lambda().eq(SysUser::getUsername, user.getUsername());
    SysUser one = sysUserService.getOne(query);
    if(one != null && one.getUserId() != user.getUserId()){
        return ResultUtils.error(msg: "账户被占用!");
    }
    //密码加密
    user.setPassword(DigestUtils.md5DigestAsHex(user.getPassword().getBytes()));
    user.setUpdateTime(new Date());
    //更新
    sysUserService.editUser(user);
    return ResultUtils.success(msg: "编辑用户成功!");
}
```

流程:

用户填写的信息会通过 el-form 组件的 data 属性, 将数据绑定到一个 json 对象中, 并通过 put 提交, 最后后端接收数据, 并完成相应用户信息的更新。

4.3 修改密码

此模块完成了的员工个人密码的修改, 若员工修改的密码与上一次密码项目, 则提示修改失败。



图 5.3 密码修改

密码修改界面代码:

```
<template>
  <div class="navbar">
```

```
<hamburger
  :is-active="sidebar.opened"
  class="hamburger-container"
  @toggleClick="toggleSideBar"
/>

<breadcrumb class="breadcrumb-container"/>

<div class="right-menu">
  <el-dropdown class="avatar-container" trigger="click">
    <div class="avatar-wrapper">
      
      <i class="el-icon-caret-bottom"/>
    </div>
    <el-dropdown-menu slot="dropdown" class="user-dropdown">
      <router-link to="/">
        <el-dropdown-item> 首页</el-dropdown-item>
      </router-link>
      <el-dropdown-item divided @click.native="updatePwd">
        <span style="display: block">密码修改</span>
      </el-dropdown-item>
      <el-dropdown-item divided @click.native="logout">
        <span style="display: block">退出登录</span>
      </el-dropdown-item>
    </el-dropdown-menu>
  </el-dropdown>
</div>
<!-- 修改密码弹框 -->
<sys-dialog
  :title="dialog.title"
  :height="dialog.height"
  :width="dialog.width"
  :visible="dialog.visible"
  @onClose="onClose"
  @onConfirm="onConfirm"
>
  <div slot="content">
    <el-form
      :model="addModel"
      ref="addModel"
      :rules="rules"
      label-width="80px"
      :inline="true"

    >
      <el-form-item prop="oldPassword" label="原密码">
        <el-input v-model="addModel.oldPassword" type="password"></el-input>
```



```

        </el-form-item>
        <el-form-item prop="password" label="新密码">
            <el-input v-model="addModel.password" type="password"></el-input>
        </el-form-item>
    </el-form>
</div>
</sys-dialog>
</div>
</template>

```

密码修改接口:

```

@Auth
@PostMapping("/updatePassword")
public ResultVo updatePassword(@RequestBody
UpdatePasswordParm parm, HttpServletRequest request)

```

接口说明: 为了保证密码的安全性, 使用 put 提交。核心代码:

```

if(userType.equals("0")){ //0 : 读者
    SysReader reader = sysReaderService.getById(parm.getUserId());
    //密码对比
    if(!old.equals(reader.getPassword())){
        return ResultUtils.error(msg: "原密码错误!");
    }
    SysReader sysReader = new SysReader();
    sysReader.setPassword(DigestUtils.md5DigestAsHex(parm.getPassword().getBytes()));
    sysReader.setReaderId(parm.getUserId());
    boolean b = sysReaderService.updateById(sysReader);
    if(b){
        return ResultUtils.success(msg: "密码修改成功!");
    }
}

}else{ // 管理员
    SysUser user = sysUserService.getById(parm.getUserId());
    if(!user.getPassword().equals(old)){
        return ResultUtils.error(msg: "原密码错误!");
    }
    SysUser sysReader = new SysUser();
    sysReader.setPassword(DigestUtils.md5DigestAsHex(parm.getPassword().getBytes()));
    sysReader.setUserId(parm.getUserId());
    boolean b = sysUserService.updateById(sysReader);
    if(b){
        return ResultUtils.success(msg: "密码修改成功!");
    }
}
return ResultUtils.error(msg: "密码修改失败!");

```

当管理员用户打开修改密码的对话框，用户或管理员需要先正确填写原来的密码，然后填入将要修改的密码。当你修改的密码与原来的密码相同时，将会提示不能使用原来的密码，并且修改密码失败。密码修改成功之后，会自动退出登录，需要重新登录。

4.4 多条件分页查询

选择不同条件，进行多条件分页查询

请输入学号	请输入姓名	请输入电话号码	请输入身份证号码	搜索	重置	新增
-------	-------	---------	----------	----	----	----

姓名	学号	班级	身份证号	电话	操作
刘兰	2024004	软件4班	780456482313134343	13477777777	编辑 重置密码 删除

分页列表界面代码：

```
<template>
  <el-main>
    <!-- 搜索栏 -->
    <el-form :model="listParm" :inline="true">
      <el-form-item>
        <el-input v-model="listParm.username" placeholder="请输入学号" />
      </el-form-item>
      <el-form-item>
        <el-input v-model="listParm.learnNum" placeholder="请输入姓名" />
      </el-form-item>
      <el-form-item>
        <el-input v-model="listParm.phone" placeholder="请输入电话号码" />
      </el-form-item>
      <el-form-item>
        <el-input v-model="listParm.idCard" placeholder="请输入身份证号码" />
      </el-form-item>
      <el-form-item>
        <el-button @click="searchBtn"> 搜索</el-button>
        <el-button style="color: #ff7670" @click="resetBtn"> 重置
        </el-button>
        <el-button v-permission="['sys:reader:add']" type="primary"
@click="addBtn"> 新增
        </el-button>
      </el-form-item>
    </el-form>
    <!-- 表格 -->
    <el-table :height="tableHeight" :data="tableData" border stripe>
      <el-table-column prop="learnNum" label="姓名" />
      <el-table-column prop="username" label="学号" width="100" />
      <el-table-column prop="className" label="班级" width="120" />
      <el-table-column prop="idCard" label="身份证号" width="165" />
      <el-table-column prop="phone" label="电话" width="120" />
      <el-table-column prop="sex" label="性别" width="70">
```

```

    <template slot-scope="scope">
      <span v-if="scope.row.sex == '0'">男</span>
      <span v-if="scope.row.sex == '1'">女</span>
    </template>
  </el-table-column>
  <el-table-column prop="checkStatus" label="审核状态">
    <template slot-scope="scope">
      <el-tag v-if="scope.row.checkStatus == '0'" type="danger">未审核</el-tag>
      <el-tag v-if="scope.row.checkStatus == '1'">已审核</el-tag>
    </template>
  </el-table-column>
  <el-table-column prop="userStatus" label="用户状态">
    <template slot-scope="scope">
      <el-tag v-if="scope.row.userStatus == '0'">停用</el-tag>
      <el-tag v-if="scope.row.userStatus == '1'">启用</el-tag>
    </template>
  </el-table-column>
  <el-table-column label="操作" fixed="right" width="550">
    <template slot-scope="scope">
      <el-button v-permission="['sys:reader:edit']" type="primary"
@click="editBtn(scope.row)"> 编辑
    </el-button>
      <el-button v-permission="['sys:reader:apply']" type="warning" v-
if="scope.row.checkStatus == '0'" @click="applyBtn(scope.row)"> 同意
    </el-button>
      <el-button v-permission="['sys:reader:resetpassword']" type="danger"
@click="resetPasswordBtn(scope.row)">
        重置密码
    </el-button>
      <el-button v-permission="['sys:reader:delete']" type="danger"
@click="deleteBtn(scope.row)"> 删除
    </el-button>
    </template>
  </el-table-column>
</el-table>
<!-- 分页 -->
<el-pagination :current-page.sync="listParm.currentPage" :page-sizes="[15,
20, 40, 80, 100]"
:page-size="listParm.pageSize" layout="total, sizes, prev, pager, next,
jumper" :total="listParm.total" background
@size-change="sizeChange" @current-change="currentChange" />
<!-- 新增弹框 -->
<sys-
dialog :title="dialog.title" :width="dialog.width" :height="dialog.height" :visi
ble="dialog.visible"
@onClose="onClose" @onConfirm="onConfirm">
  <div slot="content">

```

```
<el-form ref="addRef" :model="addModel" :rules="rules" label-
width="80px" style="margin-right: 30px">
  <el-row>
    <el-col :span="12" :offset="0">
      <el-form-item prop="learnNum" label="姓名">
        <el-input v-model="addModel.learnNum" />
      </el-form-item>
    </el-col>
    <el-col :span="12" :offset="0">
      <el-form-item prop="phone" label="电话">
        <el-input v-model="addModel.phone" maxlength="11" />
      </el-form-item>
    </el-col>
  </el-row>
  <el-row>
    <el-col :span="12" :offset="0">
      <el-form-item prop="username" label="学号">
        <el-input v-model="addModel.username" />
      </el-form-item>
    </el-col>
    <el-col :span="12" :offset="0">
      <el-form-item prop="className" label="班级">
        <el-input v-model="addModel.className" />
      </el-form-item>
    </el-col>
  </el-row>
  <el-row>
    <el-col :span="12" :offset="0">
      <el-form-item prop="idCard" label="身份证">
        <el-input v-model="addModel.idCard" maxlength="18" />
      </el-form-item>
    </el-col>
    <el-col v-if="addModel.type == '0'" :span="12" :offset="0">
      <el-form-item prop="password" label="密码">
        <el-input v-model="addModel.password" />
      </el-form-item>
    </el-col>
  </el-row>
  <el-row>
    <el-col :span="12" :offset="0">
      <el-form-item label="性别">
        <el-radio-group v-model="addModel.sex">
          <el-radio :label="'0'">男</el-radio>
          <el-radio :label="'1'">女</el-radio>
        </el-radio-group>
      </el-form-item>
    </el-col>
  </el-row>
</el-form>
```

```
        </el-form>
      </div>
    </sys-dialog>
  </el-main>
</template>
```

分页接口:

```
@Auth
@GetMapping("/list")
public ResultVo getList(ReaderParm parm)
```

接口说明:

current 代表第几页, size 每次页面所展示的数据的个数, staff 包含了多条件查询的条件。

核心代码:

```
@Override
public IPage<SysReader> getList(ReaderParm parm) {
    //构造查询条件
    QueryWrapper<SysReader> query = new QueryWrapper<>();
    if (StringUtils.isEmpty(parm.getIdCard())) {
        query.lambda().like(SysReader::getIdCard, parm.getIdCard());
    }
    if (StringUtils.isEmpty(parm.getLearnNum())) {
        query.lambda().like(SysReader::getLearnNum, parm.getLearnNum());
    }
    if (StringUtils.isEmpty(parm.getPhone())) {
        query.lambda().like(SysReader::getPhone, parm.getPhone());
    }
    if (StringUtils.isEmpty(parm.getUsername())) {
        query.lambda().like(SysReader::getUsername, parm.getUsername());
    }
    //构造分页对象
    IPage<SysReader> page = new Page<>();
    page.setCurrent(parm.getCurrentPage());
    page.setSize(parm.getPageSize());
    return this.baseMapper.selectPage(page, query);
}
```

流程:

后端根据前端提交的查询数据，然后使用 MyBatis Plus 提供的分页方法进行分页，然后将满足条件的数据返回给前端。

4.5 菜单分配

此模块实现了为角色分配菜单，一个角色可以选择多个菜单。



分配菜单

角色列表权限分配页面代码：

```
<template>
  <el-main>
    <!-- 搜索栏 -->
    <el-form :model="listParm" label-width="80px" :inline="true" >
      <el-form-item>
        <el-input
          v-model="listParm.roleName"
          placeholder="请输入角色名称"
        />
      </el-form-item>
      <el-form-item>
        <el-button @click="searchBtn"> 搜索</el-button>
        <el-button
          style="color: #ff7670"
          @click="resetBtn"
        > 重置</el-button>
        <el-button
          v-permission="['sys:role:add']"
          type="primary"
          @click="addBtn"
        > 添加</el-button>
      </el-form-item>
    </el-form>
  </el-main>
</template>
```

```

        > 新增</el-button>
    </el-form-item>
</el-form>
<!-- 表格 -->
<el-table :height="tableHeight" :data="tableData" border stripe>
  <el-table-column prop="roleName" label="角色名称" />
  <el-table-column prop="roleType" label="角色类型">
    <template slot-scope="scope">
      <el-tag v-if="scope.row.roleType == '1'">系统角色</el-tag>
      <el-tag
        v-if="scope.row.roleType == '2'"
        type="success"
      >读者角色</el-tag>
    </template>
  </el-table-column>
  <el-table-column prop="remark" label="角色备注" />
  <el-table-column label="操作" width="460">
    <template slot-scope="scope">
      <el-button
        v-permission="['sys:role:edit']"
        type="primary"

        @click="editBtn(scope.row)"
      > 编辑</el-button>
      <el-button
        type="warning"
        @click="assignBtn(scope.row)"
      > 分配权限</el-button>
      <el-button
        v-if="scope.row.roleType == '2'"
        v-permission="['sys:role:delete']"
        type="danger"

        @click="deleteBtn(scope.row)"
      > 删除</el-button>
    </template>
  </el-table-column>
</el-table>
<!-- 分页 -->
<el-pagination
  :current-page.sync="listParm.currentPage"
  :page-sizes="[10, 20, 40, 80, 100]"
  :page-size="listParm.pageSize"
  layout="total, sizes, prev, pager, next, jumper"
  :total="listParm.total"
  background

```

```
@size-change="sizeChange"
@current-change="currentChange"
/>
<!-- 新增编辑弹框 -->
<sys-dialog
  :title="dialog.title"
  :height="dialog.height"
  :visible="dialog.visible"
  @onClose="onClose"
  @onConfirm="onConfirm"
>
  <div slot="content">
    <el-form
      ref="addRef"
      :model="addModel"
      :rules="rules"
      label-width="80px"

    >
      <el-row>
        <el-col :span="12" :offset="0">
          <el-form-item prop="roleName" label="角色名称">
            <el-input v-model="addModel.roleName" />
          </el-form-item>
        </el-col>
        <el-col :span="12" :offset="0">
          <el-form-item label="角色类型">
            <el-select v-model="addModel.roleType" placeholder="请选择">
              <el-option
                v-for="item in options"
                :key="item.value"
                :label="item.label"
                :value="item.value"
              />
            </el-select>
          </el-form-item>
        </el-col>
      </el-row>
      <el-row>
        <el-col :span="12" :offset="0">
          <el-form-item label="角色备注">
            <el-input v-model="addModel.remark" />
          </el-form-item>
        </el-col>
      </el-row>
    </el-form>
  </div>
</sys-dialog>
```



```
<!-- 分配权限的弹框 -->
<sys-dialog
  :title="assignDialog.title"
  :visible="assignDialog.visible"
  :width="assignDialog.width"
  :height="assignDialog.height"
  @onConfirm="assignConfirm"
  @onClose="assignClose"
>
  <div slot="content">
    <el-tree
      ref="assignTree"
      :data="assignTreeData"
      node-key="menuId"
      :props="defaultProps"
      empty-text="暂无数据"
      :show-checkbox="true"
      default-expand-all
      :default-checked-keys="assignTreeChecked"
    />
  </div>
</sys-dialog>
</el-main>
</template>
```

菜单接口:

```
@Auth
@GetMapping("/getAssingShow")
public ResultVo getAssingShow(AssignParm parm)
```

接口说明:

获取所有的菜单数据。

角色菜单接口:

```
@Auth
@PostMapping("/assignSave")
public ResultVo assignSave(@RequestBody SaveAssign parm)
```

接口说明:

根据角色 id, 和选择的菜单 id, 为角色设置菜单。

核心代码:

```
public AssignVo getAssignShow(AssignParm parm) {  
    //查询当前用户的信息  
    SysUser user = sysUserService.getById(parm.getUserId());  
    //菜单数据  
    List<SysMenu> list = null;  
    if(user.getIsAdmin().equals("1")){ //如果是超级管理员，拥有所有的权限  
        QueryWrapper<SysMenu> query = new QueryWrapper<>();  
        query.lambda().orderByAsc(SysMenu::getOrderNum);  
        list = sysMenuService.list(query);  
    }else{  
        list = sysMenuService.getMenuByUserId(user.getUserId());  
    }  
    //组装树  
    List<SysMenu> menuList = MakeTree.makeMenuTree(list, pid: 0L);  
    //查询角色原来的菜单  
    List<SysMenu> roleList = sysMenuService.getMenuByRoleId(parm.getRoleId());  
    List<Long> ids = new ArrayList<>();  
    Optional.ofNullable(roleList).orElse(new ArrayList<>()).stream().filter(item -> item != null).forEach(item -> {  
        ids.add(item.getMenuId());  
    });  
}
```

流程:

当点击分配菜单按钮，前端会向后端请求所有的菜单数据，这里只返回了父级菜单，因为子菜单都作为父级菜单的 children 属性被携带。当菜单被渲染好之后，紧接着获取当前角色的所有拥有的菜单，并将对应项勾选。提交之后，后端先将不需要的菜单禁用，然后再重新更新或设置菜单。

第5章 总结

图书馆管理系统是一种用于管理图书馆馆藏、读者信息、借阅归还等操作的计算机化系统。该系统可以帮助图书馆工作人员更加高效地管理馆藏和读者信息，提高工作效率，也可以为读者提供更加方便快捷的借阅服务。

在学习和开发图书馆管理系统的过程中，我收获了以下几点：

1, 系统分析与设计能力的提升

在开发图书馆管理系统的过程中，我学习了系统分析与设计的基本知识，并通过实践提升了自己的分析与设计能力。通过对用户需求的调研和需求分析，我能够准确地把握用户的需求，提出有效的解决方案。在系统设计方面，我能够运用 UML 建模工具进行系统建模和设计，提高了自己的系统设计能力。

2, 编程技能的提升

开发图书馆管理系统需要涉及到多种编程语言和技术，例如 Java、MySQL 等。通过学习和实践，我掌握了这些技术的基本知识和应用方法，并能够灵活运用它们进行系统开发。

3, 团队协作能力的提升

图书馆管理系统是一个比较复杂的系统，需要多人合作完成。在开发过程中，我学习了如何与团队成员进行有效的沟通和协作，如何协调各方面的需求和利益，提高了自己的团队协作能力。

总之，通过开发图书馆管理系统，我提高了自己的系统分析与设计能力、编程技能和团队协作能力。这些经验和技能不仅可以在图书馆管理系统的开发中得到应用，也可以在其他领域的系统开发和团队协作中发挥重要作用。

此外，开发图书馆管理系统还让我认识到了一些重要的管理原则和实践：

1, 用户为中心

图书馆管理系统的核心是服务读者，因此在系统开发过程中，我们必须以用户为中心，充分考虑用户需求和用户体验，确保系统能够为用户提供便捷、高效、舒适的服务。

2, 数据管理的重要性

图书馆管理系统涉及到大量的数据管理工作，如图书馆藏、读者信息、借阅记录等。因此，数据管理非常重要，必须建立规范、有效的数据管理机制，确保数据的安全性、完整性和可靠性。

3, 技术更新与升级

随着科技的发展和社会的变化，图书馆管理系统也需要不断进行技术更新和升级，以满足用户需求和时代发展的要求。因此，我们必须密切关注技术的发展趋势和变化，及时进行技术升级和改进。

4, 管理与服务的并重

图书馆管理系统不仅是一种管理工具，更是一种服务平台。因此，在开发和运营过程中，必须兼顾管理与服务的需要，既要实现对图书馆资源和读者信息的管理，也要提供高效、优质的借阅服务。

总之，图书馆管理系统的开发和实践让我收获了很多经验和技能，同时也让我认识到了管理原则和实践的重要性。这些经验和技能将对我未来的工作和生活产生重要的影响和帮助。

参考文献

- [1]杜洋。图书馆图书管理系统的设计与实现[D]. 电子科技大学, 2013.
- [2]李彤。应用于图书馆管理的图书管理系统的开发[D]. 电子科技大学, 2009.
- [3]宫昌利。图书管理系统的设计与实现[D]. 山东大学, 2009.
- [4]顾俐。图书馆图书管理系统的设计[J]. 中国科技信息, 2007 (11): 175-177.
- [5]孙磊。医院图书管理系统的应用及实现[J]. 自动化与仪器仪表, 2017 (7): 217-218.
- [6]李岩。医院图书信息化管理问题的研究与探讨[J]. 世界最新医学信息文摘, 2017 (11): 144.
- [7]陈凌云。探究图书馆图书管理系统的设计与实现实践应用[J]. 图书档案管理, 2016 (9): 16.
- [8]孙微微。浅析图书馆图书管理系统的设计[J]. 决策与信息, 2016 (23): 239.
- [9]王丽明, 郝俊勤。部队医院图书馆“一卡通”使用情况评析[J]. 中华医学图书情报杂志, 2011, 20 (2): 63-64.
- [10]吴爱平, 张吴佳。医院图书馆在局域网环境下的业务拓展[J]. 中医药管理杂志, 2008, 16 (5): 392-393.
- [11]黄京艳, 白萍, 张鲁梅, 等。网络环境下医院图书馆流通服务创新的实践[J]. 中国美容医学, 2010, 19 (z2): 406.