

深圳大学考试答题纸

(以论文、报告等形式考核专用)

二〇二四 ~ 二〇二五 学年度第一学期

课程编号 150470001 课序号 01 课程名称 岗位实践 主讲教师 孙智达 评分
学 号 2020191228 姓名 林科贤 专业年级 2020 级数学与应用数学 (师范)

教师评语:

题目:

图书管理系统

一、序言

该部分为项目的基本介绍，项目的有关背景（类似的项目的实现情况，包括现在市面上的项目），以及项目的特色之处。

1. 基本介绍

图书管理系统就是利用计算机，结合互联网对图书进行结构化、自动化管理的一种软件，来提高对图书的管理效率。本项目采用当前较为主流的框架进行实现，主要采用 **SpringBoot+Vue+Mybatis** 的方式（其中前端主要采用 Element-UI 组件）实现基于 Web 的图书管理系统。

2. 基本功能：

- 1) 对用户、管理员进行管理，能够实现用户、管理员的增删改查工作
- 2) 对图书、图书分类进行管理，能够实现图书和图书分类的增删改查
- 3) 对图书借阅以及归还进行管理，能够实现借还图书的增删改查
- 4) 基本的登录实现
- 5) 前端界面包括主页、图书管理、管理员管理等六个选项。

3. 附加功能（特色之处）：

- 1) **图书分类**（主键为 cid）可以在一级分类的基础上实现**二级分类**，通过图书分类中的 pid 属性来实现：当分类 A 的 pid 为空时，证明分类 A 无父类，是一级分类；当分类 B 的 pid 非空的时候，比如当 pid=1 的时候，该分类的父类为 cid=1 对应的分类。实现一个树形的逐层分类。
- 2) **数据安全(登录安全)**: Web 前端通过 **Cookies** 记录登录情况(记录用户名和密码)，当 Cookies 检测到登录数据（用户名及密码）非空的时候，证明当前处于已登录状态，当输入端口号 8080/home 时，会直接定向到主页；反之，如果检测到为非登录状态，当输入端口号并尝试定位到 8080/home 的时候，无法进入主页，退回登录页面 8080/login，通过这种方式确保了数据安全，避免账户在未登录的时候就可以直接进入主页看到后台数据。
- 3) **数据安全（密码加密）**: 在新增管理员的时候，会初始化一个 123456 的密码，管理员可以在后续修改密码。但是该密码是通过加密（**md5 加密**）后才存在数据库中的，也就是说，保存在数据库中的密码不是实际密码，确保了密码的安全性，后台无法看到各个管理员独立的密码，只有管理员本人知道。与此同时，在更改密码的功能中，当当前用户更改自己的密码时候，更改成功会退出到登录界面，需要重新登录。
- 4) **登录图片验证码**: 登录的时候有一个图片验证码功能。
- 5) **首页**: 美观的前端页面，实现了卡片效果和手风琴效果的设计，可以通过首页跳转到各大图书馆的主页，也可以通过首页看到开发者的个人信息卡片，跳转到项目的 github 或者文档。
- 6) **字段检查**: 如果填写信息时，出现错误的输入信息，例如未填写新增图书数量、手机号码

不合规或者无管理员姓名等的情况，会出现自动提示，提示正确的填写方式。

- 7) 项目托管到 **Github**: 代码上传到 Github 并且实时进行更新。
- 8) 共享开发文档: 开发文档也进行实时更新，可以通过飞书文档了解项目开发情况

二、项目设计

该部分介绍项目是如何分解的，由哪些部分构成，各部分之间的关系是怎样的（建议用结构图说明）。本人会在这部分大体介绍整个项目的所有功能，展示大部分的界面。（代码实现见第三部分）

项目主要分为前后端，前端主要采用 **Vue 框架**，结合 ElementUI 组件进行实现。后端主要采用 **SpringBoot 框架** 进行实现。并通过 SQL 对数据库进行增删改查。



图 1 项目整体目录

2.1 界面（前端结构）：

前端通过 router 进行网页切换和重定向，主要分为登录（Login）和主页两个大部分，主页下面有多个子页，包括各个管理的页面。

其中登录界面如下图 2 所示，对应图 6 左 views/login 下面的 vue 文件；



图 2 登录界面

登录界面确认登录之后，会弹出验证码窗口，如果图片验证码滑块滑到正确的位置，可以成功登录，出现如图 3 右的蓝色正确；如果图片验证码滑块没有匹配到正确的位置，会出现如图 3 左所示的红色报错。

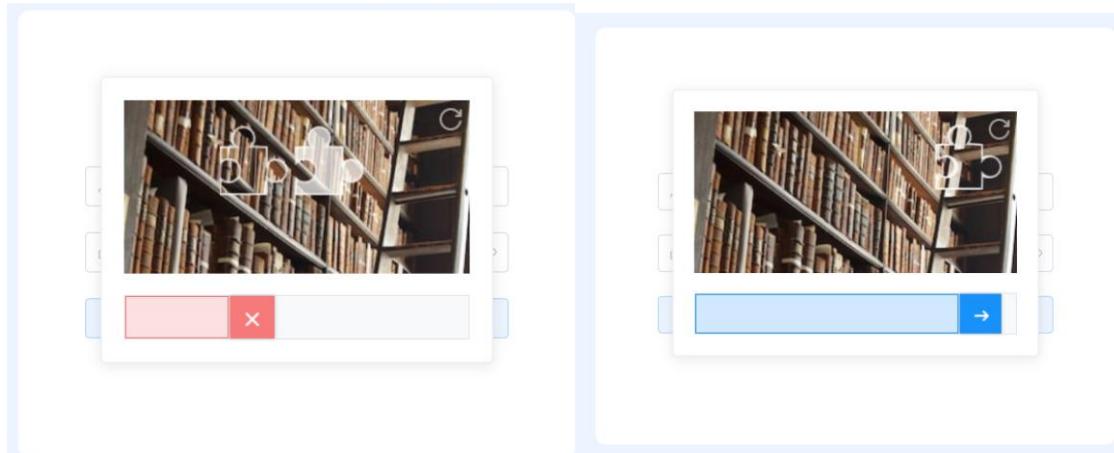


图 3 滑块验证码

如果用户名和密码匹配，那么登录成功，页面右上角会弹出如下图 4 所示提示；登录不成功，页面右上角会弹出如下图 5 所示提示，提示用户名或密码错误。

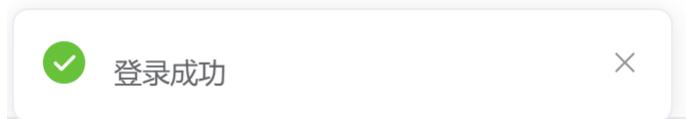


图 4 用户登录成功



图 5 用户名或密码错误

其中主页整体页面如下图 6 右所示；如下图 6 左所示目录结构 views 中的 Layout.vue 部分。具体实现见第三部分。

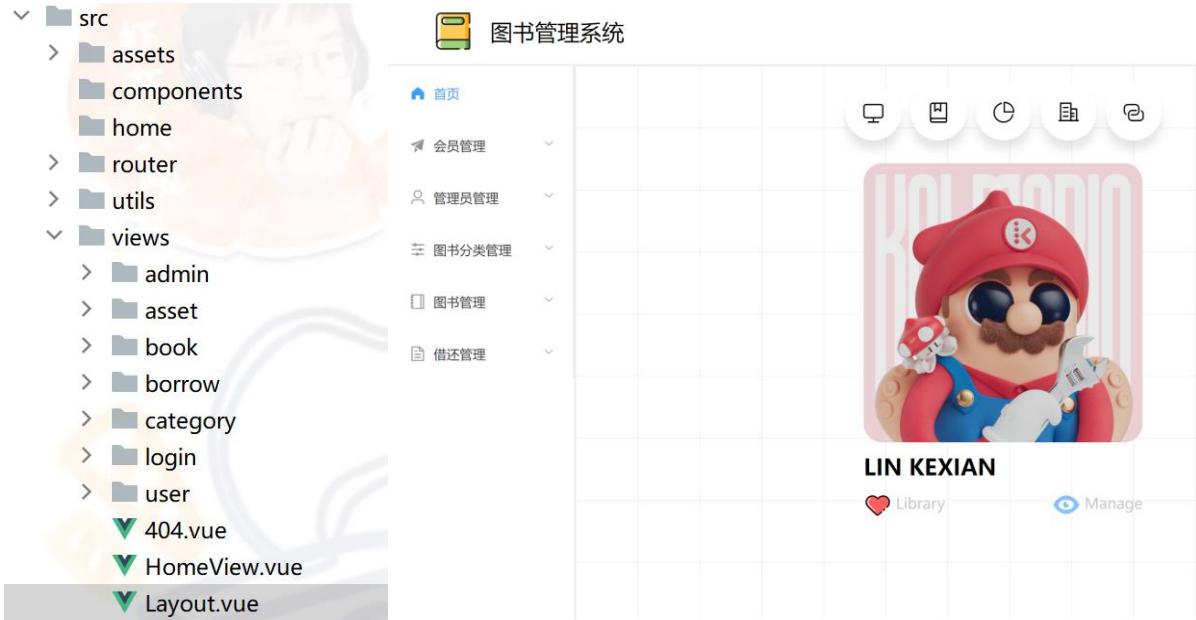


图 6 主页

以下为主页的分块介绍，首先页面结构如下图 7 所示，分为头部，目录栏，核心页面：

编号	图书名称	ISBN	用户id	用户名	用户联系方式	借书时间	操作
27	西游记	182381923891283	48	林科贤	14332333222	2023-12-17	归还图书
26	eneness	sysaaa	46	asd	13333333333	2023-10-04	归还图书
25	sys	sysab	48	林科贤	14332333222	2023-10-04	归还图书
24	sys	sysab	46	asd	13333333333	2023-10-04	归还图书

 The page also features navigation arrows and a page number indicator '1'.

图 7 主页结构

由主页可以看到目录包含首页、会员管理、管理员管理、图书分类管理、图书管理和借还管理多个部分，其中首页对应图 6 左的 views/Homeview.vue：

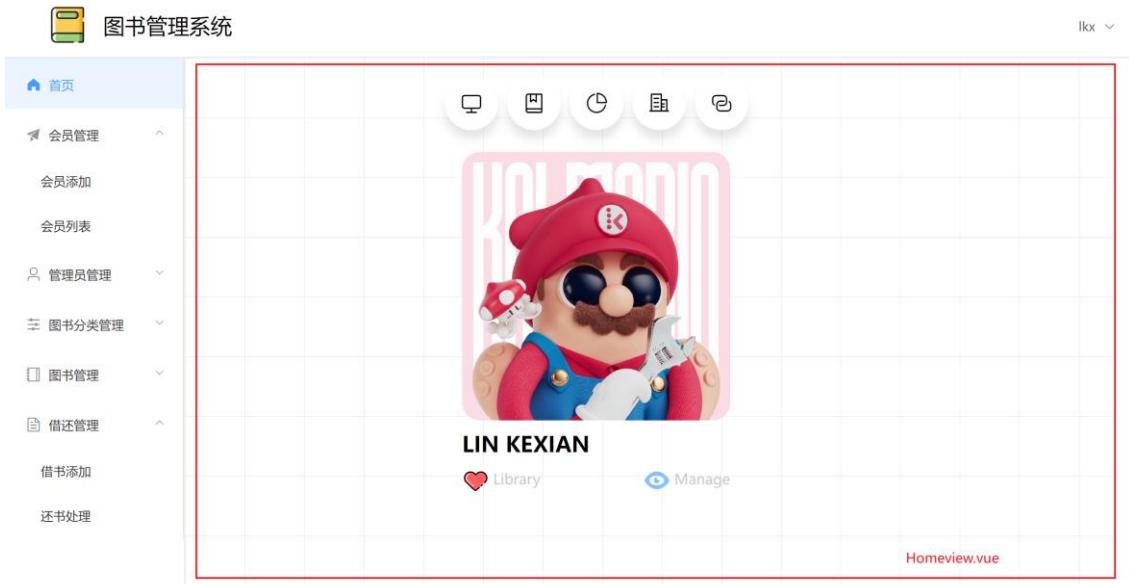


图 8 Homeview.vue

会员管理、管理员管理、图书分类管理、图书管理和借还管理分别对应 views 下面的 user 目录、admin 目录、category 目录、book 目录、borrow 目录，如图 9 所示。

我们以 user 目录为例，AddUser.vue 为会员添加页面，如图 10 所示；User 为用户列表页面，如图 11 所示；EditUser 为编辑用户页面，如图 12 所示：

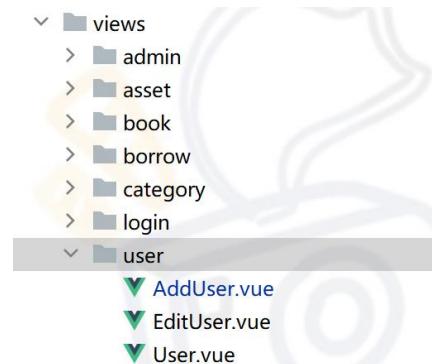


图 9 子目录结构

会员管理																																																																																										
会员添加		会员列表																																																																																								
会员添加		<table border="1"> <thead> <tr> <th>编号</th><th>会员卡号</th><th>名称</th><th>年龄</th><th>性别</th><th>地址</th><th>联系方式</th><th>创建时间</th><th>更新时间</th></tr> </thead> <tbody> <tr> <td>59</td><td>23122527587</td><td>林科贤</td><td>22</td><td>女</td><td>Division of Mathematics</td><td>13232323222</td><td>2023-12-25</td><td>2023-12-25</td></tr> <tr> <td>58</td><td>23122516341</td><td>林科贤2</td><td>22</td><td>女</td><td>szu</td><td>13232323232</td><td>2023-12-25</td><td>2023-12-25</td></tr> <tr> <td>57</td><td>23122582336</td><td>林科贤</td><td>25</td><td>女</td><td>Division of Mathematics</td><td>13239437988</td><td>2023-12-25</td><td>2023-12-25</td></tr> <tr> <td>56</td><td>23122586425</td><td>user5</td><td>28</td><td>男</td><td>Division of Mathematics</td><td>13523943798</td><td>2023-12-25</td><td>2023-12-25</td></tr> <tr> <td>53</td><td>23110855004</td><td>hhh</td><td>14</td><td>男</td><td></td><td>13432342342</td><td>2023-11-08</td><td>2023-12-18</td></tr> <tr> <td>52</td><td>2310042994</td><td>哈哈</td><td>12</td><td>男</td><td>33</td><td>14423234122</td><td>2023-10-04</td><td>2023-10-04</td></tr> <tr> <td>51</td><td>23100330243</td><td>panzhanle</td><td>18</td><td>男</td><td>粤海街道深圳 大学滨海校区 菜鸟驿站</td><td>18328383828</td><td>2023-10-03</td><td>2023-10-03</td></tr> <tr> <td>50</td><td>23100214531</td><td>xxxxaa</td><td>12</td><td>男</td><td>粤海街道深圳 大学滨海校区 菜鸟驿站</td><td>13342342342</td><td>2023-10-02</td><td>2023-10-02</td></tr> </tbody> </table>								编号	会员卡号	名称	年龄	性别	地址	联系方式	创建时间	更新时间	59	23122527587	林科贤	22	女	Division of Mathematics	13232323222	2023-12-25	2023-12-25	58	23122516341	林科贤2	22	女	szu	13232323232	2023-12-25	2023-12-25	57	23122582336	林科贤	25	女	Division of Mathematics	13239437988	2023-12-25	2023-12-25	56	23122586425	user5	28	男	Division of Mathematics	13523943798	2023-12-25	2023-12-25	53	23110855004	hhh	14	男		13432342342	2023-11-08	2023-12-18	52	2310042994	哈哈	12	男	33	14423234122	2023-10-04	2023-10-04	51	23100330243	panzhanle	18	男	粤海街道深圳 大学滨海校区 菜鸟驿站	18328383828	2023-10-03	2023-10-03	50	23100214531	xxxxaa	12	男	粤海街道深圳 大学滨海校区 菜鸟驿站	13342342342	2023-10-02	2023-10-02
编号	会员卡号	名称	年龄	性别	地址	联系方式	创建时间	更新时间																																																																																		
59	23122527587	林科贤	22	女	Division of Mathematics	13232323222	2023-12-25	2023-12-25																																																																																		
58	23122516341	林科贤2	22	女	szu	13232323232	2023-12-25	2023-12-25																																																																																		
57	23122582336	林科贤	25	女	Division of Mathematics	13239437988	2023-12-25	2023-12-25																																																																																		
56	23122586425	user5	28	男	Division of Mathematics	13523943798	2023-12-25	2023-12-25																																																																																		
53	23110855004	hhh	14	男		13432342342	2023-11-08	2023-12-18																																																																																		
52	2310042994	哈哈	12	男	33	14423234122	2023-10-04	2023-10-04																																																																																		
51	23100330243	panzhanle	18	男	粤海街道深圳 大学滨海校区 菜鸟驿站	18328383828	2023-10-03	2023-10-03																																																																																		
50	23100214531	xxxxaa	12	男	粤海街道深圳 大学滨海校区 菜鸟驿站	13342342342	2023-10-02	2023-10-02																																																																																		
借书添加																																																																																										
还书处理																																																																																										

图 10 User 界面

The screenshot shows the 'User' interface. On the left is a sidebar with the following menu items:

- 首页
- 会员管理
 - 会员添加
 - 会员列表
- 管理员管理
- 图书分类管理
- 图书管理
- 借还管理

The main content area is titled '新增用户' (Add User). It contains the following fields:

* 姓名	请输入姓名
年龄	请输入年龄
性别	请选择性别
联系方式	请输入电话
地址	请输入地址

At the bottom right are two buttons: '提交' (Submit) and '重置' (Reset).

图 11 AddUser 界面

The screenshot shows the 'EditUser' interface. The sidebar is identical to the one in Figure 11.

The main content area is titled '编辑用户' (Edit User). It displays the following user information:

会员卡号	23122527587
姓名	林科贤
年龄	22
性别	女
联系方式	13232323222
地址	Division of Mathematics

At the bottom right are two buttons: '修改' (Modify) and '取消' (Cancel).

图 12 EditUser 界面

其他的管理界面也是类似如此。

2.2项目逻辑（功能实现）：

这部分介绍系统的各个部分功能，代码实现见提交代码，代码实现大致介绍见第三部分。

2.2.1 管理员管理模块（Book Management Module）：

- 添加管理员

首先进入“管理员添加”的部分，如图 13 所示：

图 13 管理员添加

此处如果出现手机号码不合规或者无管理员姓名的情况，会出现如下所示报错；与此同时，如果点击重置，该输入表格会自动清空，可以重新填写需要新增管理员的信息：

图 14 新增提示

如果正确填写信息并且点击提交，提交成功后页面右上角也会出现如下图 15 所示的弹出提示：

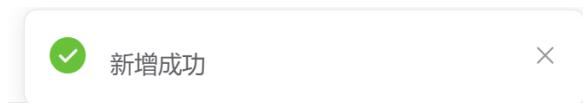


图 15 新增成功

系统会为管理员数据库新增一条数据信息，如下图 16 所示，刚刚添加的数据显示在最上层：

编号	管理员	联系方式	邮箱	创建时间	更新时间	操作
73	admin123	13232232222	ksdasda@!!!	2024-01-08	2024-01-08	<button>编辑</button> <button>删除</button> <button>修改密码</button>
71	admin8	15439949594	kkkkk@	2023-10-04	2023-12-25	<button>编辑</button> <button>删除</button> <button>修改密码</button>
63	lkx	13423412341	admin@164.com	2023-10-02	2023-12-25	<button>编辑</button> <button>删除</button> <button>修改密码</button>
62	admin1	15213423422		2023-10-01	2023-10-03	<button>编辑</button> <button>删除</button> <button>修改密码</button>
61	admin7	13423422222		2023-10-01	2023-10-02	<button>编辑</button> <button>删除</button> <button>修改密码</button>

图 16 新增数据

● 删除管理员

点击 **删除** 删除刚刚新增的信息，删除时会出现弹窗提示进行确认，避免误删数据库信息。



图 17 删除这行数据

删除成功后会出现如下右上角所示的提示，同时当前信息从管理员数据库中删除：

编号	管理员	联系方式	邮箱	创建时间	更新时间	操作
71	admin8	15439949594	kkkkk@	2023-10-04	2023-12-25	<button>编辑</button> <button>删除</button> <button>修改密码</button>
63	lkx	13423412341	admin@164.com	2023-10-02	2023-12-25	<button>编辑</button> <button>删除</button> <button>修改密码</button>
62	admin1	15213423422		2023-10-01	2023-10-03	<button>编辑</button> <button>删除</button> <button>修改密码</button>
61	admin7	13423422222		2023-10-01	2023-10-02	<button>编辑</button> <button>删除</button> <button>修改密码</button>

图 18 删除数据成功

● 更新管理员信息

点击 **编辑** 可以进入编辑页面，更改 id 为 71 的数据，比如邮箱更改为 kkkkk@163.com，点击修改则 update 数据库；点击取消则直接返回管理员列表，这里点击修改：

编辑管理员

管理员	admin8
联系方式	15439949594
邮箱	kkkkk@163.com

修改 取消

图 19 编辑管理员

右上角弹出提示更新成功，同时可以看到邮箱相比图 18 发生了变化：

The screenshot shows the 'Administrator Management' section of the library management system. A success message '更新成功' (Update successful) is displayed at the top right. The table lists administrators with columns: 编号 (ID), 管理员 (Administrator), 联系方式 (Contact), 邮箱 (Email), 创建时间 (Creation Time), 更新时间 (Last Update), and 操作 (Operations). The row for ID 71 has its email field 'kkkkk@163.com' highlighted with a red box, indicating it has been updated.

管理员管理						
编号		管理员	联系方式	邮箱	创建时间	更新时间
71	admin8	15439949594	kkkkk@163.com	2023-10-04	2024-01-08	编辑 删除 修改密码
63	lkx	13423412341	admin@164.com	2023-10-02	2023-12-25	编辑 删除 修改密码
62	admin1	15213423422		2023-10-01	2023-10-03	编辑 删除 修改密码

图 20 更新成功

除了编辑数据之外，管理员部分还包括了修改密码，也是更改数据库密码数据，点击黄色按钮“修改密码”，会弹出如下所示弹窗，修改密码为新密码，例如 admin8 密码更改为 123（新增用户的时候密码如果没有设置，则默认为 123456），修改成功：



图 21 修改密码

密码可设置显示与否：



图 22 新密码显示与否

此处有一个附加的功能，修改成功后，如果不是修改当前用户登录的密码，不会退出；如果修改的是当前用户登录的密码，会退出返回到登录界面，要求重新登录。（由于此功能是动态的，无法直接在报告中显示，这部分在验收项目的时候演示过）

● 查找管理员

在管理员页面的上方可以看到一个搜索框，此处可以对管理员姓名，联系方式或者是邮箱进行搜索，点击搜索后会筛选出相对应的信息，例如这里搜索姓名中包含“admin”的所有管理员：

admin		请输入联系方式	请输入邮箱	搜索	重置	
编号	管理员	联系方式	邮箱	创建时间	更新时间	操作
71	admin8	15439949594	kkkkk@163.com	2023-10-04	2024-01-08	<button>编辑</button> <button>删除</button> <button>修改密码</button>
62	admin1	15213423422		2023-10-01	2023-10-03	<button>编辑</button> <button>删除</button> <button>修改密码</button>
61	admin7	13423422222		2023-10-01	2023-10-02	<button>编辑</button> <button>删除</button> <button>修改密码</button>

图 23 管理员页面

2.2.2 用户管理模块 (User Management Module):

和管理员非常类似的用户增删改查功能实现，增删改查功能不作赘述，界面图片请参照图 9~图 11，以下为 9-11 未展示部分：当用户出现为输入姓名，年龄不合法（必须为非负整数），联系方式不合法的时候，会出现图 24 所示的提示信息；

新增用户

* 姓名	<input type="text" value="请输入姓名"/>
年龄	<input type="text" value="请输入年龄"/>
性别	<input type="text" value="女"/>
联系方式	<input type="text" value="请输入电话"/> 合法的手机号:11位,非“11”或“12”开头
地址	<input type="text" value="请输入地址"/>

图 24 用户列表

查找信息:

The screenshot shows the 'User Management' section of the library management system. On the left, there is a sidebar with navigation links: 首页 (Home), 会员管理 (Member Management) with sub-links 会员添加 (Add Member) and 会员列表 (List Members), 管理员管理 (Administrator Management) with sub-links 管理员添加 (Add Administrator) and 管理员列表 (List Administrators), and 图书分类管理 (Book Category Management). The main area has a search bar with placeholder '请输入名称' (Enter name) and a search button. Below it is a table listing users with columns: 编号 (ID), 会员卡号 (Member Card Number), 名称 (Name), 年龄 (Age), 性别 (Gender), 地址 (Address), 联系方式 (Contact), 创建时间 (Creation Time), 更新时间 (Last Update), and 操作 (Operations). There are five rows of data, each with edit and delete buttons. The current page is page 1 of 1.

编号	会员卡号	名称	年龄	性别	地址	联系方式	创建时间	更新时间	操作
59	23122527587	林科贤	22	女	Division of M athematics	13232323222	2023-12-25	2023-12-25	<input type="button" value="编辑"/> <input type="button" value="删除"/>
58	23122516341	林科贤2	22	女	szu	13232323232	2023-12-25		<input type="button" value="编辑"/> <input type="button" value="删除"/>
57	23122582336	林科贤	25	女	Division of M athematics	13239437988	2023-12-25	2023-12-25	<input type="button" value="编辑"/> <input type="button" value="删除"/>
49	23093098204	xxxxx	13	女	aaaaaaaaaaaaa	13242342341	2023-09-30	2023-10-02	<input type="button" value="编辑"/> <input type="button" value="删除"/>

图 24 查找用户信息

2.2.3 图书分类管理部分

图书分类管理也包含了基本的增删改查的基本功能，但是在这个基础上增加了二级分类的功能，实现是通过数据库的 pid 实现的，父级分类的 pid 为 null；子级分类的 pid 非空，它的 pid 对应的分类即为它的父级分类。

The screenshot shows the 'Book Category Management' section. It lists categories with columns: 名称 (Name), 备注 (Remarks), 创建时间 (Creation Time), 更新时间 (Last Update), and 操作 (Operations). There are seven rows of data, each with edit and delete buttons. A green '添加二级分类' (Add Secondary Category) button is visible next to some entries. The current page is page 1 of 2.

名称	备注	创建时间	更新时间	操作
课本1	123	2023-12-25	2023-12-25	<input type="button" value="编辑"/> <input type="button" value="删除"/> <input type="button" value="添加二级分类"/>
名著		2023-10-04		<input type="button" value="编辑"/> <input type="button" value="删除"/> <input type="button" value="添加二级分类"/>
欧洲名著		2023-12-17		<input type="button" value="编辑"/> <input type="button" value="删除"/>
中国名著		2023-12-17		<input type="button" value="编辑"/> <input type="button" value="删除"/>
世界名著	红楼梦等	2023-12-17		<input type="button" value="编辑"/> <input type="button" value="删除"/>
文学		2023-10-03		<input type="button" value="编辑"/> <input type="button" value="删除"/> <input type="button" value="添加二级分类"/>

图 25 图书分类管理列表

点击“添加二级分类”可以添加二级分类，可以在相应位置下面添加二级分类，比如我们在图25的基础上，在课本1下面添加二级分类：数学课本。



图 26 添加二级分类

新增二级分类成功，可以看到课本1下面有一个子分类“数学课本”：

系统				
✓ 新增二级分类成功				
请输入分类名称		○ 搜索	○ 重置	
名称	备注	创建时间	更新时间	操作
课本1	123	2023-12-25	2023-12-25	编辑 删除 添加二级分类
数学课本	math	2024-01-08		编辑 删除

图 27 添加二级分类成功

新增图书分类，备注这个框框可以下拉，还挺好玩的组件，可以拖到无限长，也可以缩小到只有一行：

新增分类

* 名称

备注

提交 重置

图 28 新增分类

主要是介绍差异的部分其他的增删改查功能全都实现了，一是和管理员增删改查类似，二是在演示的时候展示过，不作赘述。

2.2.4 图书管理部分

一样主要介绍差异部分，新增图书里面，部分信息如果不填会出现提示。

新增图书

* 名称 描述 出版日期

作者 出版社 分类

* ISBN * 图书数量 - +

封面

图 29 新增图书

这部分用的组件比较多，包括分类里面的下拉框，出版日期的日历（如图 30 右所示），图书数量的计数器，都是 ElementUI 的组件；其中分类是根据图书分类管理中一一匹配的，如图 30 左所示：

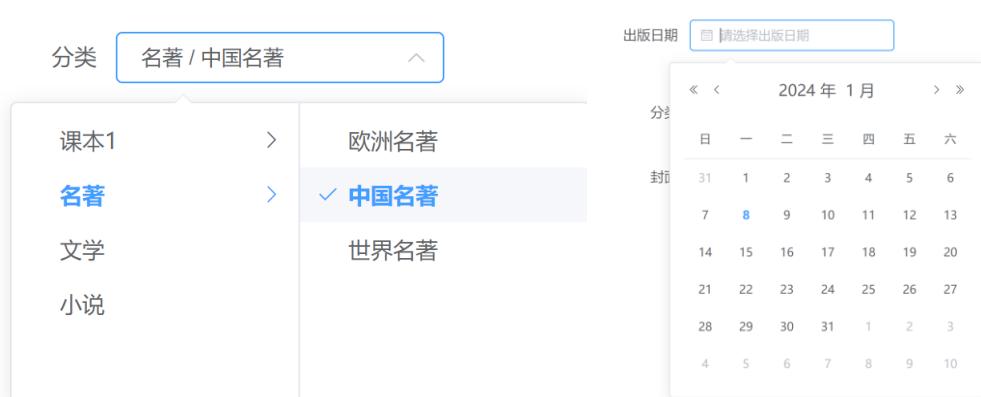


图 30 分类/日历

图书列表，有相应图片信息：

编号	图书名称	描述	发布日期	作者	出版社	分类	出版社	ISBN	剩余数量	封面	操作
20	西游记	来个正常一点的信息	2023-10-03	林科贤	林科贤出版社	文学	林科贤出版社	182381923891283	119		<input type="button" value="编辑"/> <input type="button" value="删除"/>
18	12	aa	2023-12-13	k	szu	文学	szu	123	1		<input type="button" value="编辑"/> <input type="button" value="删除"/>

图 31 图书列表

其他的就是增删改查，类似前面的功能，验收的时候演示过。

2.2.5 借还管理部分

功能还是一样的增删查，不过多赘述，“归还图书”按钮即对应删，借书添加即对应增。

The screenshot shows the 'Borrowing and Returning Management' section of the library management system. On the left, there is a sidebar with navigation links: 首页 (Home), 会员管理 (Member Management), 管理员管理 (Administrator Management), 图书分类管理 (Book Category Management), 图书管理 (Book Management), 借还管理 (Borrowing and Returning Management), 借书添加 (Add Borrowing), and 还书处理 (Handle Returns). The main area has three search input fields: '请输入图书名称' (Enter Book Name), '请输入ISBN' (Enter ISBN), and '请输入用户名' (Enter User Name), followed by a '搜索' (Search) button and a '重置' (Reset) button. Below these is a table listing borrowing records:

编号	图书名称	ISBN	用户id	用户名	用户联系方式	借书时间	操作
27	西游记	182381923891283	48	林科贤	14332333222	2023-12-17	<button>归还图书</button>
26	eneness	sysaa	46	asd	13333333333	2023-10-04	<button>归还图书</button>
25	sys	sysab	48	林科贤	14332333222	2023-10-04	<button>归还图书</button>
24	sys	sysab	46	asd	13333333333	2023-10-04	<button>归还图书</button>

Pagination controls (Previous, Next, Page 1) are located below the table.

图 32 借还管理

不一样的地方在于这里如果选择 ISBN 和用户 id，会自动查询对应的图书信息和用户信息，和实际的图书借书系统很像。

This screenshot shows the 'Add Borrowing Record' page. It contains several input fields: ISBN (182381923891283), 图书名称 (Book Name: 西游记), 剩余图书数量 (Remaining Book Quantity: 119), 用户id (User ID: 23093046952), 用户名 (User Name: asd), and 用户联系方式 (User Contact: 13333333333). At the bottom are two buttons: 提交 (Submit) and 重置 (Reset).

图 33 新增借书记录

2.2.6 首页

首页如下所示，包括一个开发者卡片（有浮动效果）和五个按钮（按钮实现手风琴效果，点击后可以分别跳转到五个对应的图书馆）

The screenshot shows the homepage of the library management system. On the left is a sidebar with the same navigation links as the previous page. In the center is a floating developer card featuring a cartoon character of Mario holding a wrench, with the text 'LIN KEXIAN' and two buttons: 'Library' (with a heart icon) and 'Manage' (with a gear icon). Above the card are five circular icons representing different library sections.

图 34 首页

手风琴效果：浮动到对应按钮上面的时候会滑动出来图书馆提示，五个按钮五个图书馆。



图 35 手风琴效果

点击图书馆名字后可以跳转到对应的图书馆。这样就方便了使用者通过系统跳转到其他图书馆，了解其他图书馆信息。

卡片：浮动后会有变化，变成下面这样，点击 Code 会跳转到这个项目的 GitHub 页面，看到项目的代码；点击 Doc 会跳转到项目的飞书开发文档。点击 Other Questions 会跳转到哔哩哔哩。

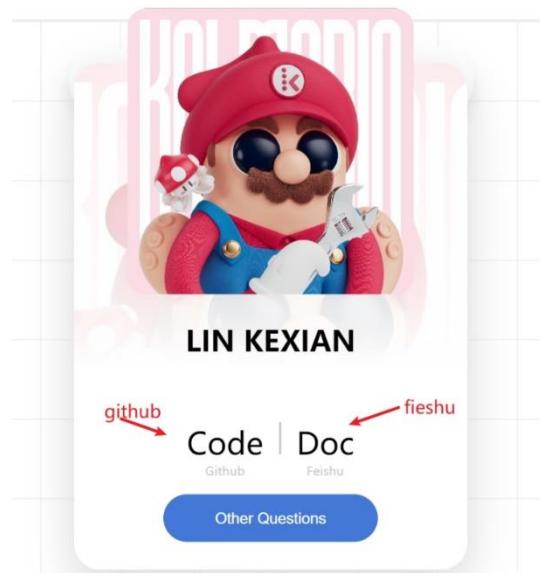


图 36 卡片

2.3 数据存储（数据存储）：

数据库文件见 **libray_management.sql**，包括了数据结构和数据。

2.3.1 静态数据

- 读者（会员）信息:** 读者姓名 name, 会员卡号 username (UNIQUE, 由后端生成), 读者编号 id (主键), 性别 sex, 年龄 age, 地址 address, 联系方式 phone, 创建时间 createtime, 更新时间 updatetime。

具体数据库字段情况见下图 (user 数据库):

名	类型	长度	小数点	不是 null	虚拟	键	注释
id	int	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	PK1	
name	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>		姓名
username	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>		用户名
age	int	0	0	<input type="checkbox"/>	<input type="checkbox"/>		年龄
sex	varchar	1	0	<input type="checkbox"/>	<input type="checkbox"/>		性别
phone	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>		联系方式
address	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>		地址
createtime	datetime	0	0	<input type="checkbox"/>	<input type="checkbox"/>		
updatetime	datetime	0	0	<input type="checkbox"/>	<input type="checkbox"/>		

图 37 user 数据库

- 管理员信息:** 管理员编号 id (主键), 管理员姓名 username, 联系方式 phone, 邮箱 email, 密码 password(经过加密处理后存储至数据库), 创建时间以及更新时间。

具体数据库字段情况见下图 (admin 数据库):

名	类型	长度	小数点	不是 null	虚拟	键	注释
id	int	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	PK1	
username	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>		用户名
phone	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>		联系方式
createtime	datetime	0	0	<input type="checkbox"/>	<input type="checkbox"/>		
updatetime	datetime	0	0	<input type="checkbox"/>	<input type="checkbox"/>		
email	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>		邮箱
password	varchar	100	0	<input type="checkbox"/>	<input type="checkbox"/>		密码

图 38 admin 数据库

- 借阅信息:** 借阅编号 id, 数据名字 bookname, 书籍编号 bookno, 用户名称 username, 用户 id userid, 用户电话 userphone, 创建时间 createtime。

具体数据库字段情况见下图 (borrow 数据库):

名	类型	长度	小数点	不是 null	虚拟	键
id	int	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	PK1
bookname	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>	
bookno	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>	
userid	int	0	0	<input type="checkbox"/>	<input type="checkbox"/>	
username	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>	
userphone	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>	
createtime	datetime	0	0	<input type="checkbox"/>	<input type="checkbox"/>	

图 39 borrow 数据库

- 图书类别:** 分类名称 name, 分类编号 id, 备注 remark, pid 父级分类 (实现多层分类哈)

具体数据库字段情况见下图 (category 数据库):

名	类型	长度	小数点	不是 null	虚拟	键	注释
id	int	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	 1	
name	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>		名称
remark	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>		备注
pid	int	0	0	<input type="checkbox"/>	<input type="checkbox"/>		父级id
createtime	datetime	0	0	<input type="checkbox"/>	<input type="checkbox"/>		
updatetime	datetime	0	0	<input type="checkbox"/>	<input type="checkbox"/>		

图 40 图书类别数据库

- **书籍信息:** 书籍编号 id, 书名 name, 书籍描述 description, 类别 category, 作者 author, 出版社 publisher, 出版日期 publishdate, 库存数量 nums, 封面 cover 等。

具体数据库字段情况见下图 (book 数据库):

名	类型	长度	小数点	不是 null	虚拟	键	注释
id	int	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	 1	
name	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>		
description	varchar	500	0	<input type="checkbox"/>	<input type="checkbox"/>		
publishdate	varchar	100	0	<input type="checkbox"/>	<input type="checkbox"/>		
author	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>		作者
publisher	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>		出版社
category	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>		
bookno	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>		标准码
createtime	datetime	0	0	<input type="checkbox"/>	<input type="checkbox"/>		
updatetime	datetime	0	0	<input type="checkbox"/>	<input type="checkbox"/>		
cover	varchar	600	0	<input type="checkbox"/>	<input type="checkbox"/>		
nums	int	0	0	<input type="checkbox"/>	<input type="checkbox"/>		图书剩余数量

图 41 图书数据库

2.3.2 动态数据

- **输入数据:** 鼠标对按钮的点击, 查询方式, 查询关键字, 新建图书, 读者记录修改, 借阅, 归还, 丢失。
- **输出数据:** 分页查询结果, 查询关键字确定的数据库记录, 统计结果, 信息录入, 删除结果, 图书借阅, 返还, 丢失。
- **内部数据:** 查询操作建立的索引。

2.3.3 E-R 关系图

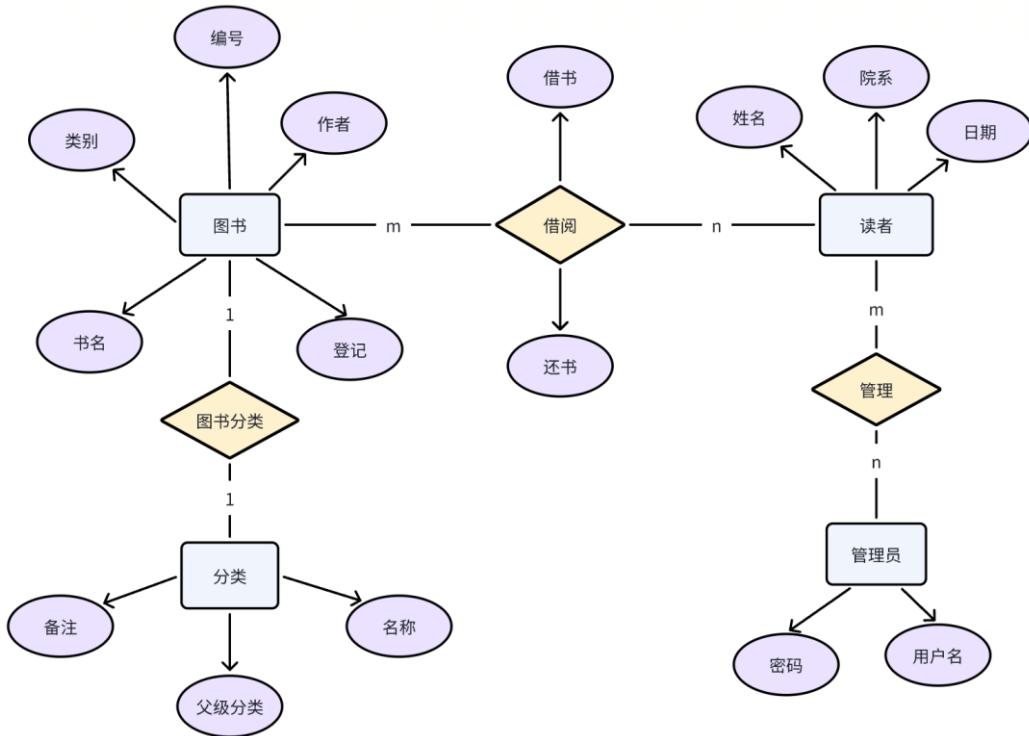


图 42 E-R 关系图

2 外部接口（外部接口）：

- 图片验证码 API (User Authentication API): 处理用户登录和注册的图片认证请求。
- 图书信息: 与外部图书信息数据库进行交互, 以获取最新的图书信息。链接到其他图书馆的主页。

3 系统管理（系统管理）:

- 日志记录 (Logging): 记录系统活动和错误, 以便进行故障排除和监控。
- 性能监控 (Performance Monitoring): 监控系统性能, 确保系统运行超出。

三、方法与实现

该部分介绍项目的具体实现, 包括采用的语言、框架、工具, 具体的实施方法。

3.1 实现分页模糊查询

在 SpringBoot+Vue 搭建的基本框架条件下, 往会员数据库中添加部分数据, 在显示数据库的基础上实现了分页模糊查询, 具体步骤如下, 导入 pagehelper 依赖, 实现列表的分页查询:

```
<dependency>
    <groupId>com.github.pagehelper</groupId>
    <artifactId>pagehelper</artifactId>
    <version>5.3.0</version>
</dependency>
```

图 42 分页模糊查询

导入 pagehelper 的时候出现 pagehelper 无效的问题，通过如下方法进行手动分页，该方法在 UserService 类下面，查询的时候调用该方法：

```
@Override
public Object page(UserPageRequest userPageRequest) {
    List<User> users = userMapper.listByCondition(userPageRequest);

    int page=userPageRequest.getPageNum();
    int pageSize=userPageRequest.getPageSize();

    //计算总记录数
    int total=users.size();
    //pageHelper无效, 手动分页, 流操作: sorted排序、skip跳记录和limit限制显示记录数
    List<User> collect=users
        .stream()
        .skip((page-1)*pageSize)
        .limit(pageSize)
        .collect(Collectors.toList());

    //计算总页数
    int pageSum = total % pageSize == 0 ? total / pageSize : total / pageSize + 1;
    PageHelper.startPage(page, pageSize);

    PageInfo<User> PageInfo = new PageInfo<User>(collect);
    //总记录数
    PageInfo.setTotal(total);
    //总页数
    PageInfo.setPages(pageSum);
    //清除分页缓存
    PageHelper.clearPage();
    return PageInfo;

}
```

图 44 实现手动分页查询

具体实现效果如下（会员管理是设置了 8 个为一页），成功实现了简单的分页模糊查询：

The screenshot shows a Vue.js application interface. On the left is a sidebar with navigation items: 首页, 会员管理 (with 会员添加, 会员列表 selected, and 管理员管理), 图书分类管理, 图书管理, and 借还管理. The main area has a search bar with '请输入名称' and '请输入联系方式' fields, and buttons for '搜索' and '重置'. Below is a table with columns: 编号, 会员卡号, 名称, 年龄, 性别, 地址, 联系方式, 创建时间, 更新时间, and 操作 (Edit, Delete). The table contains 10 rows of member data. At the bottom is a pagination bar with buttons for <, 1, 2, >.

编号	会员卡号	名称	年龄	性别	地址	联系方式	创建时间	更新时间	操作
54	23110897238	223	22			13523595555	2023-11-08		<button>编辑</button> <button>删除</button>
53	23110855004	hhhh	14			13432342342	2023-11-08		<button>编辑</button> <button>删除</button>
52	2310042994	哈哈	12	男	33	14423234122	2023-10-04		<button>编辑</button> <button>删除</button>
51	23100330243	panzhanle	18	男	粤海街道深圳 大学滨海校区 菜鸟驿站	18328383828	2023-10-03		<button>编辑</button> <button>删除</button>
50	23100214531	xxxxaa	12	男	粤海街道深圳 大学滨海校区 菜鸟驿站	13342342342	2023-10-02	2023-10-02	<button>编辑</button> <button>删除</button>
49	23093098204	xxxxxx	13	女	aaaaaaaaaaaa a	13242342341	2023-09-30	2023-10-02	<button>编辑</button> <button>删除</button>
48	23093068254	林科贤	12	女	12	14332333222	2023-09-30	2023-10-01	<button>编辑</button> <button>删除</button>
46	23093046952	asd	12	男	粤海街道深圳 大学滨海校区 菜鸟驿站	13333333333	2023-09-30	2023-09-30	<button>编辑</button> <button>删除</button>

图 45 分页模糊查询效果

3.2 实现基本的增删改查功能

以会员管理为例子，介绍增删改查功能的代码：

3.2.1 增加会员信息

- 前端 **AddUser.vue**: 实现一个表单，包括该数据库的各个子项，姓名，年龄，性别，联系方式和地址等的输入框；以及提交和重置两个按钮

```

<div style="...>
  <div style="...>
    <h3 style="...>新增用户</h3>
    <div>
      <el-form :inline="true" :model="form" :rules="rules" ref="ruleForm" label-width="100px">
        <el-form-item label="姓名" prop="name">
          <el-input v-model="form.name" placeholder="请输入姓名" style="width: 100px;"></el-input>
        </el-form-item>
        <el-form-item label="年龄" prop="age">
          <el-input v-model="form.age" placeholder="请输入年龄" style="width: 100px;"></el-input>
        </el-form-item>
        <el-form-item label="性别" prop="gender">
          <el-select v-model="form.gender" style="width: 100px;">
            <el-option value="男">男</el-option>
            <el-option value="女">女</el-option>
          </el-select>
        </el-form-item>
        <el-form-item label="联系方式" prop="phone" style="margin-top: 10px;">
          <el-input v-model="form.phone" placeholder="请输入联系方式" style="width: 100px;"></el-input>
        </el-form-item>
        <el-form-item label="地址" style="margin-top: 10px;">
          <el-input v-model="form.address" placeholder="请输入地址" style="width: 100px;"></el-input>
        </el-form-item>
      </el-form>
    </div>
  </div>
</div>

<div style="text-align: right; margin-top: 20px; margin-bottom: 10px; border: 1px solid #ccc; padding: 5px; background-color: #f0f0f0; display: flex; justify-content: space-between; align-items: center; gap: 10px; width: fit-content; margin-left: auto; margin-right: auto; font-size: 14px; font-weight: bold; border-radius: 5px; box-sizing: border-box; position: relative; z-index: 1; >
  <el-button type="primary" plain @click="save" style="font-size: 14px; font-weight: bold; border: none; border-radius: 5px; padding: 5px 15px; background-color: #fff; color: #007bff; transition: all 0.3s ease; >提交</el-button>
  <el-button type="info" plain @click="reset" style="font-size: 14px; font-weight: bold; border: none; border-radius: 5px; padding: 5px 15px; background-color: #fff; color: #007bff; transition: all 0.3s ease; >重置</el-button>
</div>
</div>

```

图 46 增加前端

具体页面效果如下，和第二部分对应：

The screenshot shows a sidebar navigation menu on the left with the following items:

- 首页
- 会员管理
 - 会员添加
 - 会员列表
- 管理员管理
- 图书分类管理
- 图书管理
- 借还管理

The main content area is titled "新增用户" (Add User) and contains the following fields:

* 姓名	请输入姓名
年龄	请输入年龄
性别	请选择性别
联系方式	请输入电话
地址	请输入地址

At the bottom right are two buttons: "提交" (Submit) and "重置" (Reset).

图 47 新增用户

- **字段检查功能实现：**实现页面的检查功能，比如姓名为必填项，年龄为 1-100 范围进行检查，联系方式限制号码长度为 11 位且不能 11 或者 12 开头，具体代码实现如下，设置 rules：

```
rules: {  
    name: [  
        {required: true, message: '请输入姓名', trigger: 'blur'},  
    ],  
    phone: [  
        {validator: checkPhone, trigger: 'blur'}  
    ],  
    age: [  
        {validator: checkAge, trigger: 'blur'}  
    ]  
}
```

图 48 字段检查功能

其中 checkPhone 和 checkAge 如下：

```
const checkPhone= (rule, value, callback) => {  
    if (!/[1][3,4,5,6,7,8,9][0-9]{9}$/, test(value)) {  
        callback(new Error('合法的手机号:11位,非"11"或"12"开头'));  
    }  
    callback()  
};  
  
const checkAge= (rule, value, callback) => {  
    if (!/[0-9]+$/, test(value)) {  
        callback(new Error('请输入数字值'));  
    }  
    callback()  
};
```

图 49 check 函数

实现效果如下，当输入框内的字符串非法的时候，会出现如下报错信息：

图 50 展示了一个表单验证示例。包含以下字段及其错误信息：

- * 姓名：请输入姓名
- 年龄：请输入年龄
- 性别：请选择性别
- 联系方式：
 - 请输入电话
 - 合法的手机号:11位,非"11"或"12"开头

图 50 字段检查效果

- **下拉框选择功能：**前端表单初始化为空的表单 form:{}，sexoption 设置为选择框，包括男和女两个选项，设置方法如下左图，在 return 里面设置，具体效果如下右图：

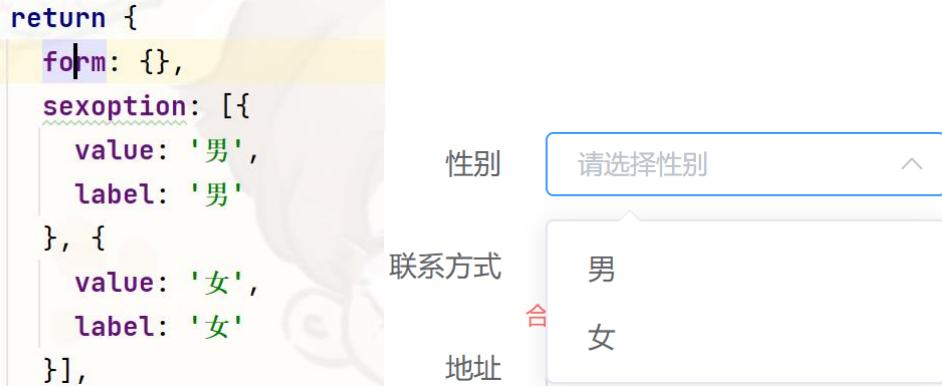


图 51 下拉框

然后是与后端关联了，实现方法就是**表单提交**与后端关联，另外**重置**按钮可以实现表单的刷新，分别对应 **reset 方法** 和 **save 方法**，**save 方法**在成功添加的时候在页面右上角弹出新增成功的窗口，具体代码如下：

```
<el-button type="primary" plain @click="save" style="...">提交</el-button>
<el-button type="info" plain @click="reset" style="...">重置</el-button>
```

```
methods: {
  save() {
    this.$refs['ruleForm'].validate((valid) => {
      if (valid) {
        request.post({ url: '/user/save', this.form }).then(res => {
          if (res.code === '200') {
            this.$notify.success('新增成功');
            this.form = {};
          } else {
            this.$notify.error(res.msg)
          }
        })
      }
    })
  },
  reset() {
    this.form = {}
  }
}
```

通过post实现表单提交

图 52 post 表单提交

- **Request.js:**

request 方法需要配置拦截请求，具体代码如下 js 文件

```
import axios from 'axios'
import router from "@/router";
import Cookies from "js-cookie";

const request=axios.create({
  baseURL:'http://localhost:9090', //后台地址
  timeout:5000
})

//request拦截器
//拦截请求
request.interceptors.request.use(
  onFulfilled: config=>{
    config.headers['Content-Type']='application/json;charset=utf-8';
    // const admin=Cookies.get('admin')
    // console.log(admin)
    // if(!admin){
    //   router.push('/login')
    // }

    //config.headers['token']=user.token;
    return config
  },
  onRejected: error => {
    return Promise.reject(error);
  }
);

```

图 53 request.js

```
//response拦截器
request.interceptors.response.use(
  onFulfilled: response=>{
    let res=response.data;
    if(typeof res === 'string'){
      res = res? JSON.parse(res) : res
    }
    return res;
  },
  onRejected: error => {
    console.log('err' + error)
    return Promise.reject(error)
  }
)

export default request
```

图 54 response 拦截器

- 后端：接下来是后端的部分

====首先是 user 类====

对应数据库定义实体的各个属性:

```
//@Data替换getter setter代码
@Data
public class User {
    private Integer id;
    private String name;
    private String username;
    private Integer age;
    private String sex;
    private String phone;
    private String address;
    @JsonFormat(pattern = "yyyy-MM-dd",timezone = "GMT+8")
    private Date createtime;
    @JsonFormat(pattern = "yyyy-MM-dd",timezone = "GMT+8")
    private Date updatetime;
}
```

图 55 user 类

====然后是 UserController 类====

前端通过 post 方法实现 form 表单数据提交到后端，对应后端的 `@PostMapping` 注解，具体代码如下，该方法调用 `UserService` 类（实现了 `IUserService` 接口，代码里用接口通用性更强一点）的 `save` 方法：

（后端实现调用关系 `usercontroller>UserService [实现 IUserService 接口] >UesrService`

```
//对外--前端 传输数据
@RestController
@RequestMapping("/user")
public class UserController {

    @Autowired
    IUserService userService;
}

@PostMapping("/save")
public Result save(@RequestBody User user){
    userService.save(user);
    return Result.success();
}
```

图 56 userController 类

====然后是 IUserService 接口====

```
//接口
public interface IUserService {
    List<User>list();

    PageInfo<User> page(BaseRequest baseRequest);

    void save(User user);

    User getById(Integer id);

    void update(User user);

    void deleteById(Integer id);
}
```

图 57 IUserService 接口

====然后是 **UserService** 类====

这里只截取了 save 方法，其他方法在其他部分介绍



```
@Override
public void save(User user) {
    //username是卡号哈
    Date date=new Date();
    user.setUsername(DateUtil.format(date, format: "yyMMdd")
        +Math.abs(IdUtil.fastSimpleUUID().hashCode()&Integer.MAX_VALUE)%100000);
    userMapper.save(user);
}
```

图 58 UserService 类

====然后是 **UserMapper** 类====

```
@Mapper
public interface UserMapper {

    List<User> list();
    List<User> listByCondition(BaseRequest userPageRequest);

    void save(User user);

    User getById(Integer id);

    void updateById(User user);

    void deleteById(Integer id);
}
```

图 59 UserMapper 类

UserMapper 类下面调用 User.xml 文件下的 save 方法, SQL 语句添加数据库信息, 用 insert 标签。

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
    PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
    "https://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.example.springboot.mapper.UserMapper">
    SQL
    <insert id="save">
        insert into user(name,username,age,sex,phone,address)
        values(#{$name},#${username},#${age},#${sex},#${phone},#${address})
    </insert>

```

图 60 UserMapper 类

● 效果

实现效果就是前端输入数据表单, 点击提交后, 后端数据新增, 具体效果如下, 和第二部分对应:

新增用户

* 姓名	林科贤
年龄	21
性别	女
联系方式	13123220000
地址	林科贤2020191228

图 61 实现效果 1

The screenshot shows a user management interface. At the top, there are two input fields labeled '请输入名称' and '请输入联系方式', followed by a blue '搜索' button and an orange '重置' button. Below this is a table with columns: 编号 (ID), 会员卡号 (Membership Card Number), 名称 (Name), 年龄 (Age), 性别 (Gender), 地址 (Address), 联系方式 (Contact Method), 创建时间 (Creation Time), 更新时间 (Update Time), and 操作 (Operations). Two rows of data are visible: one row for ID 55 with name '林科贤', age 21, gender 女 (Female), address '林科贤2020191228', contact method '13123220000', creation time '2023-11-08', and operations buttons '编辑' (Edit) and '删除' (Delete); another row for ID 54 with name '223', age 22, contact method '13523595555', creation time '2023-11-08', and operations buttons '编辑' (Edit) and '删除' (Delete).

编号	会员卡号	名称	年龄	性别	地址	联系方式	创建时间	更新时间	操作
55	23110821168	林科贤	21	女	林科贤20201 91228	13123220000	2023-11-08		<button>编辑</button> <button>删除</button>
54	23110897238	223	22			13523595555	2023-11-08		<button>编辑</button> <button>删除</button>

图 62 新增效果

可以看到新增了一条刚才输入的数据，新增功能完成。

3.2.2 删除会员信息

- 前端：User.vue（User.vue 在实验一的时候介绍过，就是会员列表页面，如上图所示分页查询并展示数据库所有数据），下面删除按钮（参考上图的红色按钮）



图 63 删除按钮

用的是 `el-popconfirm` 组件，设置弹出窗口确认是否删除该行数据，避免误删。确认 `confirm` 后，调用 `del` 方法，传入行信息的 `id`，即传入编号 `id`，由于具有唯一性，可以根据 `id` 删除唯一行的数据。

```
del(id){  
    request.delete(url: "/user/delete/" + id).then(res => {  
        if(res.code === '200'){  
            this.$notify.success('删除成功')  
            this.load()  
        }  
        else{  
            this.$notify.error(res.msg)  
        }  
    })  
}
```

图 64 删除方法

Request 调用后端的方法实现数据库删除

- 后端：

调用关系：依次传递调用

UserController 下面的 delete 方法：

```
@DeleteMapping("/delete/{id}")
public Result delete(@PathVariable Integer id){
    userService.deleteById(id);
    return Result.success();
}
```

图 65 删除 Controller 层

UserService 下面 deleteById：（实现 IUserService 接口的 deleteById）

```
@Override
public void deleteById(Integer id) { userMapper.deleteById(id); }
```

图 66 deleteById 方法

UserMapper 下面的 deleteById：

```
void deleteById(Integer id);
```

User.xml 中 delete 标签，根据 id 删除唯一对应行。

```
<delete id="deleteById">
    delete from user where id = #{id}
</delete>
```

图 67 删除 SQL 语言

● 效果：

和第二部分对应，点击删除删除编号为 54 行所对应的数据：

请输入名称		请输入联系方式		操作					
编号	会员卡号	名称	年龄	性别	地址	联系方式	创建时间	更新时间	操作
55	23110821168	林科贤	21	女	林科贤2021 91228	13123220000	2023-11-08		<button>编辑</button> <button>删除</button>
54	23110897238	223	22			13523595555	2023-11-08		<button>编辑</button> <button>删除</button>
53	23110855004	hhhh	14			13432342342	2023-11-08		<button>编辑</button> <button>删除</button>

图 68 删除

删除成功：可以看到 54 行数据删除成功。

编号	会员卡号	名称	年龄	性别	地址	联系方式	创建时间	更新时间	操作
55	23110821168	林科贤	21	女	林科贤20201 91228	13123220000	2023-11-08		<button>编辑</button> <button>删除</button>
53	23110855004	hhhh	14			13432342342	2023-11-08		<button>编辑</button> <button>删除</button>
52	2310042994	哈哈	12	男	33	14423234122	2023-10-04		<button>编辑</button> <button>删除</button>

图 69 删除成功

3.2.3 修改（编辑）会员信息

- 前端: (User.vue 下)

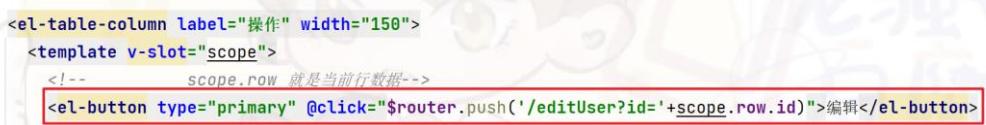


图 70 编辑按钮

前端页面效果就是上上图那个蓝色的编辑按钮。可以看到，通过传递 `scope.row.id` 作为参数进行编辑，这里用的是 `router` 来 `push`，通过路由实现也页面跳转，跳转到 `EditUser.vue` 对应的前端页面，同时传递 `scope.row.id` 参数给 `EditUser.vue`。

`EditUser.vue`, 代码和 `AddUser.vue` 差不多:

```
<div style="...">
  <h3 style="...">编辑用户</h3>
  <div>
    <el-form :inline="true" :model="form" label-width="100px">
      <el-form-item label="会员卡号">
        <el-input v-model="form.username" disabled style="..."></el-input>
      </el-form-item>
      <el-form-item label="姓名" ...>
      <el-form-item label="年龄" ...>

      <el-form-item label="性别" ...>
      <el-form-item label="联系方式" ...>
      <el-form-item label="地址" ...>

    </el-form>
  </div>
</div>

<div style="...">
  <el-button type="primary" @click="save" style="...">修改</el-button>
  <el-button @click="back" style="...">取消</el-button>
</div>
```

图 71 编辑用户

在编辑用户页面 `created` 的时候获得从 `User.vue` 传递过来的 id:

```
created() {  
    const id=this.$route.query.id
```

图 72 编辑用户获取 id

由此 id 可以查询数据库信息，显示在表单上面，也就达到了编辑的时候显示所编辑行数据的效果。点击编辑，切换前端页面到编辑用户界面：这里通过 **disabled** 属性设置了会员卡号不得更改，如下图所示，其他数据根据 id 查询并显示到输入框中。用户可在这个基础上更改信息。

会员卡号	23110821168
姓名	林科贤
年龄	21
性别	女
联系方式	13123220000
地址	林科贤2020191228

修改 **取消**

图 73 编辑用户

● 后端

UserService 等类和新增/删除差不多，这里不作赘述，就是修改数据库的 SQL 语句不一样：

```
<update id="updateById">  
    update user set name= #{name}, age=#{age}, sex=#{sex},  
    phone=#{phone}, address=#{address}, updatetime=#{updatetime}  
    where id=#{id}  
</update>
```

图 74 update SQL 语言

● 实现效果：

点击编辑：

编号	会员卡号	名称	年龄	性别	地址	联系方式	创建时间	更新时间	操作
55	23110821168	林科贤	21	女	林科贤2020191228	13123220000	2023-11-08		编辑 删除

图 75 编辑效果

进入编辑用户界面（通过路由切换），如图 73，点击取消则不更改数据，返回列表页面。这里修改一下数据，年龄改 18！弹出更新成功提示并且年龄成功更改：

The screenshot shows a user management interface. At the top right, there is a green checkmark icon and the text '更新成功' (Update Successful). Below the header, there are search fields for '请输入名称' (Enter Name) and '请输入联系方式' (Enter Contact Information), followed by a blue '搜索' (Search) button and an orange '重置' (Reset) button. The main area contains a table with columns: 编号 (ID), 会员卡号 (Member Card Number), 名称 (Name), 年龄 (Age), 性别 (Gender), 地址 (Address), 联系方式 (Contact Method), 创建时间 (Creation Time), 更新时间 (Last Update), and 操作 (Operations). Two rows of data are shown:

编号	会员卡号	名称	年龄	性别	地址	联系方式	创建时间	更新时间	操作
55	23110821168	林科贤	18	女	林科贤2020191228	13123220000	2023-11-08	2023-11-10	<button>编辑</button> <button>删除</button>
53	23110855004	hhhh	14			13432342342	2023-11-08		<button>编辑</button> <button>删除</button>

图 76 更新成功

3.2.4 查找会员信息

实现筛选数据：

```
public interface UserMapper {
    List<User> list();
    List<User> listByCondition(BaseRequest userPageRequest);
```

图 77 筛选数据方法

ListByCondition 方法也实现了分页查询。

- 前端：

```
<el-input style="..." placeholder="请输入名称" v-model="params.name"></el-input>
<el-input style="..." placeholder="请输入联系方式" v-model="params.phone"></el-input>
<el-button style="..." type="primary" @click="load">
  <i class="el-icon-search"></i>
  <span>搜索</span>
</el-button>
<el-button style="..." type="warning" @click="reset">
  <i class="el-icon-refresh"></i>
  <span>重置</span>
```

图 78 分页 ElementUI

- 后端：

调用关系也是和增删改差不多的，不作赘述（具体见 [github 代码](#)），SQL 语句有区别：筛选名字和电话符合查询条件的信息

```

<select id="listByCondition" resultType="com.example.springboot.entity.User">
    select * from user
    <where>
        <if test="name !=null and name!=''">
            name like concat('%',#{name},'%')
        </if>
        <if test="phone !=null and phone!=''">
            and phone like concat('%',#{phone},'%')
        </if>
    </where>
    order by id desc
</select>

```

图 79 listByCondition 的 SQL 语句

- 实现效果：

比如查询电话号码包含“13”的用户：

编号	会员卡号	名称	年龄	性别	地址	联系方式	创建时间	更新时间	操作
55	23110821168	林科贤	18	女	林科贤20201 91228	13123220000	2023-11-08	2023-11-10	<button>编辑</button> <button>删除</button>
53	23110855004	hhhh	14			13432342342	2023-11-08		<button>编辑</button> <button>删除</button>
50	23100214531	xxxxaa	12	男	粤海街道深圳 大学沧海校区 菜鸟驿站	13342342342	2023-10-02	2023-10-02	<button>编辑</button> <button>删除</button>
49	23093098204	xxxxxx	13	女	aaaaaaaaaaaa a	13242342341	2023-09-30	2023-10-02	<button>编辑</button> <button>删除</button>
46	23093046952	asd	12	男	粤海街道深圳 大学沧海校区 菜鸟驿站	13333333333	2023-09-30	2023-09-30	<button>编辑</button> <button>删除</button>

图 80 查询效果

3.3附加功能

除了会员列表，还有管理员列表，图书分类管理，图书管理，分别对应 **admin,book,borrow,category** 列表。它们都实现了增删改查的功能（这部分功能相似，不再赘述，可以看 GitHub 上面代码），但是数据库属性，列表属性都存在一些区别，但是增删改查功能是一样的。这里先介绍管理员管理的差异；

3.3.1 管理员管理修改密码功能：

- 前端：

```

<el-button style="margin-left: 10px" type="warning" @click="handlechangePass(scope.row)">修改密码</el-button>

```

点击“修改密码”后实现 handlechangePass 方法：

```
handlechangePass(row){  
    this.form=JSON.parse(JSON.stringify(row))  
    this.dialogFormVisible=true  
},
```

图 81 修改密码 1

dialogForm 从本来的不可见变为可见 Visible，弹出修改密码窗口：



The screenshot shows a Vue.js application interface. At the top, there is a navigation bar with several items. Below the navigation bar, there is a search input field and a button labeled '搜索' (Search). The main content area contains a table with columns: '操作' (Operation), '用户名' (Username), '昵称' (Nickname), '状态' (Status), and '最后登录' (Last Login). A modal dialog box is centered on the screen, titled '修改密码' (Change Password). It contains a form with a single input field labeled '新密码' (New Password) and a placeholder '请输入新密码' (Please enter new password). At the bottom of the dialog are two buttons: '取消' (Cancel) and '确定' (Confirm). A red arrow points from the text '弹出修改密码窗口' (Pop up password change window) to the 'handleCurrentChange' method call in the code.

```
<div style="margin-top: 20px">  
  <el-pagination  
    background  
    :current-page="params.pageNum"  
    :page-size="params.pageSize"  
    layout="prev, pager, next"  
    @current-change="handleCurrentChange"  
    :total="total">  
  </el-pagination>  
</div>  
  
<el-dialog title="修改密码" :visible.sync="dialogFormVisible" width="30%">  
  <el-form :model="form" :inline="true" label-width="100px" :rules="rules" ref="formRef">  
    <el-form-item label="新密码" prop="newPass">  
      <el-input v-model="form.newPass" autocomplete="off" show-password></el-input>  
    </el-form-item>  
  </el-form>  
  <div slot="footer" class="dialog-footer">  
    <el-button @click="dialogFormVisible = false">取 消</el-button>  
    <el-button type="primary" @click="savePass">确 定</el-button>  
  </div>  
</el-dialog>
```

图 82 修改密码 2

点击确定后调用 savePass 方法：

```

savePass(){
    this.$refs['formRef'].validate((valid)=>{
        if(valid){
            request.put( url: '/admin/password',this.form).then(res=>{
                if(res.code === '200'){
                    if(this.form.id==this.admin.id){
                        //当前修改的用户id等于当前登录的管理员id，修改成功之后重新登录
                        this.$notify.success("修改成功,请重新登录")
                        Cookies.remove('admin')
                        this.$router.push('/login')
                    }
                    else{
                        this.$notify.success("修改成功")
                        this.load()
                        this.dialogFormVisible=false
                    }
                }
                else{
                    this.$notify.error("修改失败")
                }
            })
        }
    })
}

```

图 83 修改密码 3

如果修改的密码为当前账户的密码（即使上图代码中特判 this.form.id==this.admin.id）的时候，会退出回到登录界面（登录界面下一个实验再写），通过路由切换。如果修改的不是当前登录账户的密码，就不会退出，只是关闭修改密码窗口，提示修改成功。

- 效果：

点击 **修改密码**，弹出如下界面：



图 84 修改密码弹窗

要求密码不为空，点击确认后更改密码成功。点击眼睛可以显示/隐藏密码：

* New Password	123456		* New Password	
----------------	--------	--	----------------	-------	--

图 85 密码是否可见

3.3.2 登录功能后台基本逻辑

除了后台数据库与前端展现数据“增删改查”基础功能外，也有后台的登录功能，系统需从主页登录后才进入到后台管理界面。首先明确，系统是基于 admin（管理员数据库）做的登录。

在 adminController 创建一个 login 的 post 接口：

```
@PostMapping("/login")
public Result login(@RequestBody LoginRequest request){
    LoginDTO login=adminService.login(request);
    return Result.success(login);
}
```

图 86 adminController

写一个 LoginRequest 类：

```
@Data
@AllArgsConstructor
@NoArgsConstructor
public class LoginRequest {
    private String username;
    private String password;
}
```

图 87 LoginRequest 类

LoginDTO：用于登陆的时候返回基本数据，登录接口

```
@Data
public class LoginDTO {
    private Integer id;
    private String username;
    private String phone;
    private String email;
}
```

图 87 LoginDTO 类

登录 login 方法：

```

@Override
public LoginDTO login(LoginRequest request) {
    request.setPassword(securePass(request.getPassword()));
    Admin admin = adminMapper.getByUsernameAndPassword(request.getUsername(), request.getPassword());
    if(admin==null){
        throw new ServiceException("用户名或密码错误");
    }
    LoginDTO loginDTO = new LoginDTO();
    BeanUtils.copyProperties(admin, loginDTO);
    return loginDTO;
}

```

图 88 Login 方法

Login 方法根据参数 loginrequest，获得登陆时的用户名和密码，然后调用 adminMapper 的 getUsernameAndPassword 方法，验证是否能够登录，如果能够登录，返回的 admin 非空；如果不能登录，返回的 admin 为空值。

如果 admin 对象是空的时候，证明登录账号或者密码不匹配，我们抛出一个 ServiceException 异常。如果 admin 对象非空的时候，BeanUtils 把 admin 对象，相同的属性赋值给 loginDTO 对象。然后返回 loginDTO（包含所登录账户基本数据），成功登录。

adminMapper 的 getUsernameAndPassword 方法：

```
Admin getByUsernameAndPassword(@Param("username") String username,@Param("password")String password);
```

实现：基于管理员数据库，当后台账号和前台输入账号相同且密码也对应的时候，可以登录成功，查询到唯一的管理员并返回为 admin 对象。

```

<select id="getByUsernameAndPassword" resultType="com.example.springboot.entity.Admin">
    select * from admin where username = #{username} and password=#{password}
</select>

```

图 89 Login 对应 SQL

主页和登录的区分依赖于路由，具体见代码。

```

const routes = [
    //-----1_Login-----
    {
        path: '/login',
        name: 'Login',
        component: ()=>import('@/views/login/Login.vue'),
    },
    //-----1_Layout-----
    {
        path: '/',
        name: 'Layout',
        component: Layout,
        redirect:'/home',
        children:[
            //-----2_HOME-----
            {path: 'home', name: 'Home', component: HomeView},
            // -----2_User-----
            {path: 'userList', name: 'UserList', component: () => import('@/views/user/User.vue')},
            {path: 'addUser', name: 'AddUser', component: () => import('@/views/user/AddUser.vue')}
            {path: 'editUser', name: 'EditUser', component: () => import('@/views/user/EditUser.vue')}

            // -----2_Admin-----
        ]
    }
]

```

图 90 Router 定向

3.3.3 登录界面的代码实现

亮点：在基础登录表单的基础上增加了一个验证码功能，稍后介绍



图 91 登录前端

与后端链接：

```
methods: {
  login(){
    this.$refs['loginForm'].validate((valid) => {

      if(valid){
        request.post( url: '/admin/login',this.admin).then(res =>{
          if(res.code==='200'){
            this.loginAdmin=res.data
          }
          else{
            this.$notify.error(res.msg)
          }
        })
      }
    })
  }
}
```

图 92 login 方法

点击登陆后，调用 `login` 方法，函数内调用 `post` 请求，请求后端（前面 1 介绍的），实现登录。账户密码错误：注意这里密码有一个眼睛，可以保护密码。点击隐藏，点击显示。登陆成功，就会进入主页；失败则提示，效果参见第二部分设计部分：

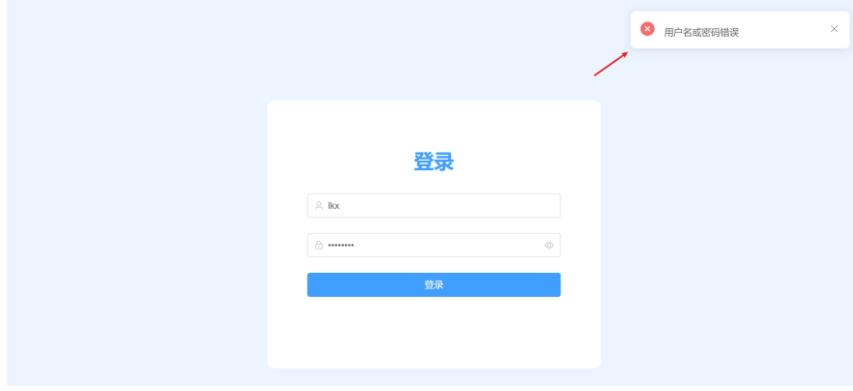


图 93 登录效果

3.3.4 数据安全——页面保护

前端的登录界面和后台的基本逻辑完成了。但是，现在有个问题。账户应该有登录/未登录的状态，未登录的时候应该无法进入主页，但现在是输入 `http://localhost:xxxx/home` 网址可以直接看到主页，也就是说可以跳过 `login` 这个功能，所以现在需要保证数据安全。保证记录了状态是否登录，确保未登录的时候无法进入主页，确保数据安全。

首先，如果登录成功的话用浏览器的 Cookies 保存用户信息哈：

```
//----- 滑块 -----
onSuccess(){
    this.$notify.success("登录成功")
    if(this.loginAdmin!==null){
        Cookies.set('admin',JSON.stringify(this.loginAdmin))
    }
    this.$router.push('/')
},
onFail(){
    this.msg = ''
},
onRefresh(){
    console.log('refresh')
}
```

图 94 登录成功代码

`Logout()` 方法清除浏览器数据并且将主页推回 `login` 界面。



```

export default {
  name: "Layout",
  data(){
    return{
      admin:Cookies.get('admin') ? JSON.parse(Cookies.get('admin')):{}
    }
  },
  methods:{
    logout(){
      //清除浏览器用户数据
      Cookies.remove('admin')
      this.$router.push('/login')
    }
  }
}

```

图 95 Logout 代码

在主页需要退出到 login 界面的时候，就调用 logout 方法，前端见 github 代码。效果：主页右上角有个退出按钮，点击后会 logout，返回到 login 界面。

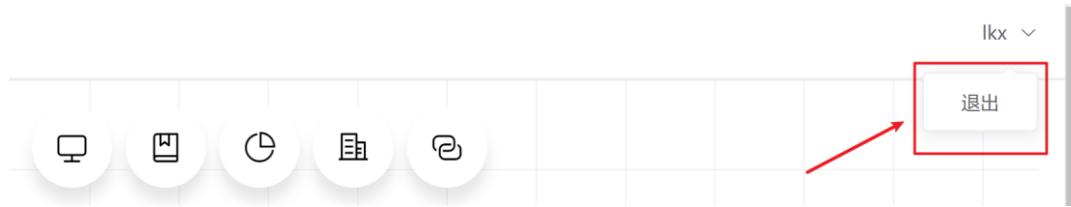


图 96 退出效果

基于 cookie 和上面 lagout 的思路方法，我们有了数据安全保护的方法，那就是判断 Cookies 里面的 admin 是否未空，如果是空的，那么证明当前是未登录状态，push 到/login 路径；如果不是空的，说明已经登录，就不用强制退回登录界面了，确保了数据安全。以下代码在 request.js 下：

```

router.beforeEach( guard: (to : Route ,from : Route ,next : NavigationGuardNext<Vue> )=>{
  if(to.path=='/login') next()
  const admin=Cookies.get("admin")
  if(!admin && to.path!=='/login') return next( to: "/login") //强制退回到登录页面
  next()
})

```

图 97 数据安全 2

- 效果：

就是 localhost: 8080，未登录的话网址定向会定向到下面这里，无法看到主页：

```
) localhost:8080/login
```

图 98 网页保护

3.3.5 数据安全——密码加密

可以看到，我们现在用 username, password 直接判断密码是否对应，但实际应用上，后台密码往往不能直接保存外界信息。所以还需要数据安全之密码加密功能。

具体的就是，save 的时候加盐：

SecurePass: (md5)

```
public String securePass(String password){  
    return SecureUtil.md5( data: password+PASS_SALT);  
}
```

图 99 SecurePass 方法

保存密码的时候以加盐后的密码存入数据库:

```
@Override  
public void save(Admin obj) {  
    //默认密码123  
    if(StrUtil.isBlank(obj.getPassword())){  
        obj.setPassword(DEFAULT_PASS);  
    }  
    //设置md5加密- 加盐  
    obj.setPassword(securePass(obj.getPassword()));  
    try {  
        adminMapper.save(obj);  
    }  
    catch(DuplicateKeyException e){  
        log.error("数据插入失败,username:{}" ,obj.getUsername());  
        throw new ServiceException("username已存在");  
    }  
}
```

图 100 save 方法

后台数据库看到的密码是这样的:

对象	admin @library_managem...					
开始事务	文本	筛选	排序	导入	导出	
id	username	phone	createtime	updatetime	email	password
61	admin7	1342342222	2023-10-01 22:02:01	2023-10-02 1(Null)		36b1a13d6e0ea39b8ee9bd53ed46ee32
62	admin1	1521342342	2023-10-01 22:03:01	2023-10-03 1(Null)		6f293138afbddd9f9db787826be6513a9
63	lkx	1342341234	2023-10-02 12:03:02	2023-10-02 1admin@8e8c1b3bee3fa14057f31c6c87834db0		8e8c1b3bee3fa14057f31c6c87834db0
71	admin8	1543994959	2023-10-04 12:03:04	2023-10-04 1kkkkk@8e8c1b3bee3fa14057f31c6c87834db0		8e8c1b3bee3fa14057f31c6c87834db0

图 101 后台数据库密码

比如后面两个管理员，原密码是“123456”，保存在数据库的就是加密的另一个密码，如上所示。Login 的时候也加盐:

```
@Override  
public LoginDTO login(LoginRequest request) {  
    request.setPassword(securePass(request.getPassword()));  
    Admin admin = adminMapper.getByUsernameAndPassword(request.getUsername(), request.getPassword());  
    if(admin==null){  
        throw new ServiceException("用户名或密码错误");  
    }  
    LoginDTO loginDTO = new LoginDTO();  
    BeanUtils.copyProperties(admin, loginDTO);  
    return loginDTO;  
}
```

图 102 LoginDTO 加盐

加盐之后的密码和后台保存的加密过的密码验证是否一样，如果一样，证明密码是对的。这样就可以保证了后台保存的数据具有保密性，数据库无法直接看到用户的密码。

注意，在修改密码的时候也一样需要加盐：

```
@Override  
public void changePass(PasswordRequest request) {  
    request.setNewPass(securePass(request.getNewPass()));  
    int count=adminMapper.updatePassword(request);  
    if(count<=0){  
        throw new ServiceException("修改密码失败");  
    }  
}
```

图 103 changePass 方法

3.3.6 滑块验证码

前面提到了登录时候的滑块验证码的功能，这里来看看效果。具体的，我用的是这个验证码，个人觉得挺好用的就用了：

<https://gitee.com/monoplasty/vue-monoplasty-slide-verify>

导入：

```
import 'element-ui/lib/theme-chalk/index.css';  
import '@/assets/global.css'  
import SlideVerify from 'vue-monoplasty-slide-verify';
```

```
Vue.config.productionTip = false  
Vue.use(ElementUI, options: {size:'small'});  
Vue.use(SlideVerify);
```

图 104 SlideVerify

前端: slide-verify 组件



图 104 SlideVerify 组件

实现效果: 点击登录之后, 如果账号密码匹配, 会弹出这样一个对话框, 如果位置贴合, 那就成功登录, 滑条为蓝色; 否则的话提示红色报错, 具体效果可以看第二部分设计实现。

3.3.7 404 页面

界面效果: 当网址输入为错误网址的时候, 会跳转到 404 界面, 比如随便输入几个错误的网址, 会出现如下所示的网址, 点击返回主页会跳转回 home 界面 (已登录状态/Cookies 检查到登录信息非空), 或者跳转到 login 界面 (未登录状态/Cookies 检查到登录信息为空):

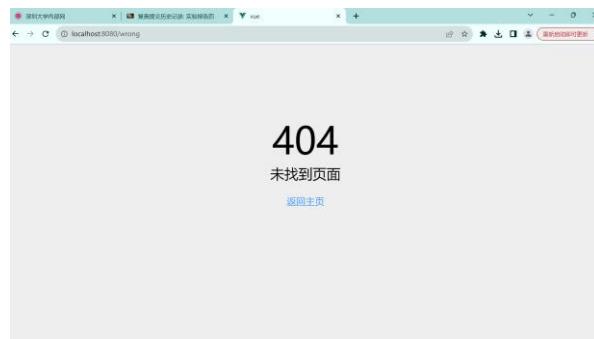


图 105 404 页面

代码:

```

<template>
  <div style="text-align: center">
    <div style="margin-top: 150px;font-size: 80px;">
      404
    </div>
    <div style="font-size: 30px;">
      未找到页面
    </div>
    <el-button type="text" style="margin-top: 20px;font-size: 20px" @click="$router.push('/')">
      返回主页
    </el-button>
  </div>
</template>

```

图 106 404 前端代码

3.3.8 图书分类管理——二级分类

图书分类管理在一级分类增删改查的基础上还有二级分类（如下绿色按钮），在一级分类的基础上可以添加二级分类。界面效果如下所示：

名称	备注	创建时间	更新时间	操作
名著		2023-10-04		编辑 删除 添加二级分类
文学		2023-10-03		编辑 删除 添加二级分类
小说		2023-10-03		编辑 删除 添加二级分类

图 107 添加二级分类

二级分类也可以完成增删改查的功能，在一级分类的基础上更加完善了。前端：（绿色按钮）

```

<el-dialog title="添加二级分类" :visible.sync="dialogFormVisible" width="30%">
  <el-form :model="form" :inline="true" label-width="100px" :rules="rules" ref="formRef">
    <el-form-item label="分类名称" prop="name">
      <el-input v-model="form.name" autocomplete="off"></el-input>
    </el-form-item>
    <el-form-item label="备注" prop="remark">
      <el-input v-model="form.remark" autocomplete="off"></el-input>
    </el-form-item>
  </el-form>
  <div slot="footer" class="dialog-footer">
    <el-button @click="dialogFormVisible = false">取消</el-button>
    <el-button type="primary" @click="save">确定</el-button>
  </div>
</el-dialog>

```

图 108 添加二级按钮

后端：该数据库为 category 数据库，数据库属性包含 pid 属性，也就是父级 id，当 pid 为空的时候，该类别为一级分类；当 pid 非空的时候，该类别是二级分类，显示在对应父级分类下。

```

handleAdd(row) {
    // 将当前行的id作为二级分类的pid
    this.pid=row.id
    this.dialogFormVisible=true
},
save() {
    this.$refs['formRef'].validate((valid) => {
        if (valid) {
            // 给二级分类赋值pid
            this.form.pid=this.pid
            request.post( url: '/category/save' , this.form).then(res => {
                if (res.code === '200') {
                    this.$notify.success('新增二级分类成功');
                    this.form={}
                    this.dialogFormVisible=false
                    this.load()
                } else {
                    this.$notify.error(res.msg)
                }
            })
        }
    })
}

```

图 109 save 方法 pid

数据库：例如这里，刚刚添加的世界名著、中国名著和欧洲名著三个二级分类，他们的 pid 就是名著，对应 pid 为 21。

id	name	remark	pid	createtime
19	小说	(Null)	(Null)	2023-10-03 16:12:06
20	文学	(Null)	(Null)	2023-10-03 16:12:26
21	名著	(Null)	(Null)	2023-10-04 16:33:46
22	世界名著	红楼梦等	21	2023-12-17 23:06:12
23	中国名著	(Null)	21	2023-12-17 23:11:39
24	欧洲名著	(Null)	21	2023-12-17 23:11:47

图 110 父级 id

3.3.9 借还管理——自动赋值

也就是面向用户的借书系统，可以新增借书，也可以归还图书。在原本的增删改查基本功能实现的基础上，增加了如下功能：

在 ISBN 号输入后，图书名称和剩余图书数量会自动显示在灰色框内，也就是说，ISBN 号输入后，后台通过 ISBN 号作为主键去查找了图书数据库，获得了名称和数量的信息，显示在前端上。其次是用户 id，也是同理，userid 作为主键去查找用户数据库对应的用户名和用户联系方式，然后显示在前端上：

新增借书记录

ISBN	182361923891283	图书名称	西游记	剩余图书数量	120
用户id	23093068254	用户名	林科贤	用户联系方式	14332333222
			提交	重置	

图 111 自动获取

在借书完成后，后台会自动更新剩余图书数量，图书数据库中剩余图书数量-1，例如上面的图

书，剩余数量 update 后变成了 119。

新增借书记录

ISBN	182381923891283	图书名称	西游记	剩余图书数量	120
用户id	23093068254	用户名	林科贤	用户联系方式	14332333222
			<input type="button" value="提交"/>	<input type="button" value="重置"/>	

图 112 借书数量变动

代码：

```
handleChange(val){  
    console.log(val)  
},  
selBook(){  
    const book = this.books.find(v => v.bookno === this.form.bookno)  
    this.form.bookname=book.name  
    this.form.nums=book.nums      找到对应的书名和数量，显示在前端  
},  
selUser(){  
    const user = this.users.find(v => v.id === this.form.userid)  
    this.form.username=user.name  
    this.form.userphone=user.phone  
}  
}
```

图 113 findbook

3.3.9 首页——美化前端（外部接口）

● 卡片

这里是在 `github` 上面找的一个比较好看的效果，具有动态效果，具体代码见 `/user/Homeview.vue` 文件。



图 114 卡片

● 图书馆跳转

包括广东图书馆/广州图书馆等五个图书馆，前端实现手风琴效果，点击图书馆名称后会跳转到对应的图书馆。



图 115 其他图书馆

以上为项目的最后开发阶段，对各个部分进行了完善并且逐步优化。整合后的系统已经进行了完整的初步测试，无明显错误。项目各个预定功能也都在各个实验中完成了，系统功能正确实现。最后完整的项目已经整合好并上传到 `github` 了，链接如下：

https://github.com/hellokx/library_management

四、总结与展望

4.1 项目展望

- **利用最新技术提升系统性能：**随着科技的迅猛发展，图书管理系统应当不断采纳最新的技术，以提高系统的性能和响应速度。引入先进的数据库管理系统、云计算技术和人工智能算法，能够使系统更加高效、智能化，提升用户体验。
- **强化用户体验与界面设计：**未来的图书管理系统应该注重用户体验和界面设计，使得用户能够更加轻松、愉快地使用系统。引入响应式设计，使系统能够在不同设备上都有良好的表现，并优化交互流程，简化操作步骤。
- **拓展多功能模块：**图书管理系统不仅仅是一个简单的图书检索工具，还应该成为一个多功能的学术资源平台。引入在线阅读、学术论文检索、多媒体资源管理等功能，满足用户多样化的需求，提升系统的综合服务水平。
- **引入社交化元素：**图书管理系统可以引入社交化元素，促进用户之间的交流与合作。例如，用户可以分享自己的阅读心得、评论图书，系统可以推荐与用户口味相近的读者或图书，从而构建一个更加丰富的学术社交圈。
- **强化数据安全与隐私保护：**随着信息安全问题日益突出，图书管理系统需要加强对用户数据的保护，包括数据加密、权限管理等措施，确保用户信息安全。同时，要遵循相关法规，保护用户的隐私权益。
- **跨平台兼容性：**未来图书管理系统需要具备良好的跨平台兼容性，能够在不同的操作系统和浏览器上正常运行。这有助于扩大系统的用户群体，提高系统的普适性。

4.2 总结：

本课程完整完成了一个项目的开发，从需求分析，到实现到最后测试，本人在各个环节中都收获颇多，对软件工程/开发更为熟悉，受益匪浅。

图书管理系统是一个为图书馆、学校、企业等提供图书管理服务的重要工具。通过不断引入新技术、强化用户体验、拓展功能模块等手段，可以使系统更好地适应不断变化的需求和环境。在系统的设计中，要着重考虑用户的需求，通过人性化的界面设计和智能化的功能，提高用户的使用体验。同时，系统的安全性和隐私保护也是至关重要的，必须采取有效的措施来保护用户的数据和隐私。未来的图书管理系统应当致力于打破传统的图书管理模式，通过引入社交化元素，促进用户之间的交流与合作，构建一个更加丰富的学术社交平台。同时，跨平台兼容性的提高可以使系统更具灵活性，更好地适应不同用户的使用习惯。综上所述，图书管理系统的未来发展方向应当是技术的不断创新、用户体验的不断提升以及功能模块的不断拓展，以更好地服务于用户群体。