**安徽财经大学字**

**本科毕业设计**

|  |  |
| --- | --- |
| **题 目** | 基于SpringBoot的学生考勤系统的设计与实现 |
| **学 院** | 管理科学与工程学院 |
| **专 业** | 物联网工程 |
| **班 级** |  |
| **学 号** |  |
| **姓 名** | 王志 |
| **指导老师** |  |

**2022 年 5 月**

安徽财经大学管理科学与工程学院

本科生毕业论文（设计）诚信承诺书

本人承诺：

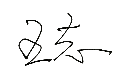
1.所呈交的毕业论文（设计）《 基于SpringBoot的学生考勤系统的设计与实现》，是在认真学习理解《安徽财经大学学位论文作假行为处理办法》和《管理科学与工程学院本科毕业论文（设计）工作管理办法》后，保质保量独立完成的，没有弄虚作假，没有抄袭别人的内容；

2.毕业论文（设计）所使用的相关资料、数据、观点等均真实可靠，文中所有引用的他人观点、材料、数据、图表均已注释说明来源；

3.毕业论文（设计）中无抄袭、剽窃或不正当引用他人学术观点、思想和学术成果，伪造、篡改数据的情况；

4.本人已被告知并清楚：学院对毕业论文（设计）中的抄袭、剽窃、弄虚作假等违反学术规范的行为将严肃处理，并可能导致毕业论文（设计）成绩不合格，无法正常毕业、取消学士学位资格或注销并追回已发放的毕业证书、学士学位证书等严重后果；

5.若在省教育厅、学校、学院组织的毕业论文（设计）检查中，被发现有抄袭、剽窃、弄虚作假等违反学术规范的行为，本人愿意接受学院按有关规定给予的处理，并承担相应责任。

学生（签名）： 

2022年 05 月 23日

指导老师（签名）： 

2022年 05 月 23日

**基于SpringBoot的学生考勤系统的设计与实现**

**摘 要**

传统的学校考勤基本上采用假条请假的模式，对学生进行考勤和请假管理。但是因为每个学生的信息量和学生数量较多，导致单纯的人工考勤模式不能更好的满足校园的实际需求。基于SpringBoot的学生考勤系统（以下简称为“学生考勤系统”）能更高效得集中管理学生的个人信息，从而提高学校考勤的效率，减少对纸质假条的浪费，节约了学校日常开销的成本，所以设计一个在线学生考勤系统是很有价值的。本系统的讲解流程遵循了软件开发的生命周期。学生考勤系统采用B/S模式，能对学生的个人信息进行管理，可以实时掌握学生的请假情况，教师可以在线批阅学生假条，在需要时可以迅速导出学生信息。

开发此系统所使用的是面向对象工程方法学，用这种技术开发出的软件系统由对象组成。按照人类习惯的思维方法建立问题域的模型，开发出尽可能直观、自然地表现求解方法的软件系统。系统采用前后端分离的模式，在统一的标准下，两端分离开发，极大的提高了开发效率。前端采用的是Vue.js语言编写，编程环境是WebStorm，后端采用的是Java语言编写，编程环境是IDEA。本系统在线考勤的功能满足系统的需求。由于系统避免了软件开发的缺点，使得开发出来的学生考勤系统有优秀的交互界面，同时也可以免去用户一些繁琐的操作。在交互设计的过程中本系统没有涉及到太多专业知识，所以使用者不必具有高深的计算机知识也可以使用这个系统来完成各种任务，系统数据和后端设计对不同的角色加载不同菜单，后台也有身份验证，确保用户的数据的安全性。

**关键词：**学生考勤；B/S架构；管理系统；前后端分离

**Design and Implementation of Student Attendance System based on Spring-boot**

**Abstract**

Body The traditional school attendance management basically adopts the mode of leave note to ask for leave to manage students' attendance and leave. However, due to the large amount of information and number of individual students, the simple artificial leave note attendance mode can not better meet the needs of the school. Student attendance system based on SpringBoot (hereinafter referred to as "student attendance system") can more effectively manage students' personal information, improve the efficiency of school attendance, reduce the waste of paper leave, save the cost of school daily expenses, so this system is very worth developing. This article strictly according to the software life cycle to explain the development process of student attendance system. Student attendance system adopts B/S mode, can manage the personal information of students, can understand the students' leave in real time, teachers can review students' leave notes online, can quickly export student information when needed.

The development of this system is based on object-oriented engineering methodology. The software system developed by this technology is composed of objects. According to the human habit of thinking method to establish the problem domain model, developed as intuitive as possible, natural expression of the solution of the software system. The system adopts the mode of front and back end separation. Under the unified standard, both ends are developed separately, which greatly improves the development efficiency. Vue.js framework is used to write the front-end, the programming environment is WebStorm, Java language is used to write the back-end, the programming environment is IDEA. Basically meet the function of student attendance, and meet the data needs of the system. The system absorbs the historical experience of software development, which makes the student attendance system developed have friendly man-machine interaction interface and system error information processing, and provides necessary help to users in time. Considering the users of the system of computer professional knowledge don't understand, so try to avoid in the process of user interaction design involves much professional knowledge, even if they don't have too much computer knowledge can be normal use this system to complete various tasks, at the same time, in order to make sure the system data will not be any view and abuse, The system grants different front-end working interfaces to different categories of users, and the back-end is designed to prevent malicious requests for data.

**Key words： S**tudents' Attendance；B/S Architecture；Management System；Front end separation；

**目 录**

[1 引言 - 1 -](#_Toc104835234)

[2 业务流程阐述 - 2 -](#_Toc104835235)

[2.1 宏观业务阐述 - 2 -](#_Toc104835236)

[2.2 数据流图阐述 - 2 -](#_Toc104835237)

[3 功能需求分析 - 3 -](#_Toc104835238)

[3.1 系统功能需求 - 3 -](#_Toc104835239)

[3.2 系统数据需求 - 5 -](#_Toc104835240)

[3.2.1 ER（实体关系图） - 5 -](#_Toc104835241)

[3.2.2数据库设计 - 5 -](#_Toc104835242)

[3.2.3 IPO（输入加工输出图） - 8 -](#_Toc104835243)

[3.3系统其他需求 - 10 -](#_Toc104835244)

[4 宏观设计 - 10 -](#_Toc104835245)

[4.1 系统功能结构分析 - 10 -](#_Toc104835246)

[4.2 架构模式 - 10 -](#_Toc104835247)

[4.3数据库设计环境配置说明 - 10 -](#_Toc104835248)

[5 具体设计 - 11 -](#_Toc104835249)

[5.1 软件宏观结构 - 11 -](#_Toc104835250)

[5.2 请/批假组件 - 12 -](#_Toc104835251)

[5.3 用户信息组件 - 15 -](#_Toc104835252)

[5.4 角色信息组件 - 19 -](#_Toc104835253)

[5.5 权限信息组件 - 21 -](#_Toc104835254)

[5.5 其他重要组件 - 22 -](#_Toc104835255)

[6 系统使用说明 - 23 -](#_Toc104835256)

[6.1 用户模组 - 23 -](#_Toc104835257)

[6.2 学生模组 - 24 -](#_Toc104835258)

[6.3 教师模组 - 25 -](#_Toc104835259)

[6.4管理用户模组 - 25 -](#_Toc104835260)

[6.5 管理角色模组 - 29 -](#_Toc104835261)

[6.6 管理权限模组 - 29 -](#_Toc104835262)

[7 总结 - 29 -](#_Toc104835263)

[参考文献 - 31 -](#_Toc104835264)

[致谢 - 32 -](#_Toc104835265)

# 1 引言

习近平主席提出了后疫情时代推动世界经济绿色复苏的设想[1]。这两年，企业、政府以及学校办公的模式从传统线下办公，纸质办公变成了现在的线上办公，无纸化办公，在线办公在生活中越来越占据主导地位。这种在线办公不仅提高了学校的办公效率，也极大的方便了学生和教师。

学生考勤作为学校的多种办公工作里的一种，快速严格的考勤方式，可以降低辅导员管理的难度，也可以规范学校的考勤工作，避免在考勤过程中出现的效率低下的现象。传统的考勤模式不仅费时费力，而且经常出现效率不高、不规范、信息不透明等问题，考勤工作变成了一种表面的、可暗箱操作的无效考勤。

因此，针对传统的考勤模式，本文设计了一款结合学生请假以及教师批假功能的学生考勤平台，该平台可以减少纸质的浪费，提高教师的办公效率，同时提供了丰富的后台管理功能。本系统运用MVC模式的设计思想，数据库使用Mysql数据库[2]。

学生考勤系统在MVC的模式基础上，将项目结构分为下面这四层：

视图层：负责展示用户所看到的交互页面，采用Vue.js 技术,它是一套构建用户界面的渐进式框架，采用自底向上增量开发的设计方式[3]。单页面开发，通过Vue路由和Axios进行请求和响应处理[4]，其实质还是DOM操作[5]。运用了Element UI对页面进行设计,数据可视化方面采用了Echarts，它有显示直观、交互、自定义、可视化数据等优点[6],提升了交互体验。

控制层：负责接收视图层数据，传给业务逻辑层。采用SpringBoot技术，作为Spring项目组的微服务产品[7]。相比于Spring配置文件大大减少了，内嵌了servlet容器降低了环境要求。使用JSON模式进行响应。对网络硬件环境要求较低，因此适应范围更广[8]。相对于JSP做到了真正的解耦，让前端不再痛苦于页面开发，页减轻了服务器压力。

业务逻辑层：主要实现具体到点的功能，接收Controller传入的数据，用于访问数据持久层。并将结果集返回给控制层。

数据持久层：主要实现对数据库的访问[9]，使用了Mybatis-plus框架，摒弃了Mybatis的XML方式，采用注解的方式进行开发，提高了可读性和可维护性，降低了工作难度。

通过前后端分离的技术，实现学生考勤系统的开发。

# 2 业务流程阐述

## 2.1 宏观业务阐述

学生考勤系统的数据来源是学生和教师以及管理员，学生提交请假申请，教师负责批假，管理员负责管理用户角色和权限信息。学生考勤系统的终点是，学生查看请假记录。学生考勤系统主要有“首页”，“用户列表”，“角色列表”，“权限列表”，“个人信息”，“我要请假”，“请假记录”，“请假名单”，“批假记录”的功能，对于此系统的数据库，用户、角色、权限、请假日志都是必不可少的。另外后端增加权限验证来阻止未许可的用户访问用户重要数据。

## 2.2 数据流图阐述



图2.1 学生考勤系统的顶层数据流图



图2.2 学生考勤系统的一层数据流图

# 3 功能需求分析

## 3.1 系统功能需求

根据业务粗略分析，本系统主要完成：

1. 学生功能：请假，查看请假记录；
2. 教师功能：批假，查看批假记录；
3. 管理员功能：用户管理、角色管理、权限管理。

图3.1为学生考勤系统用例图：



图3.1 学生考勤系统用例

## 3.2 系统数据需求

### 3.2.1 ER（实体关系图）

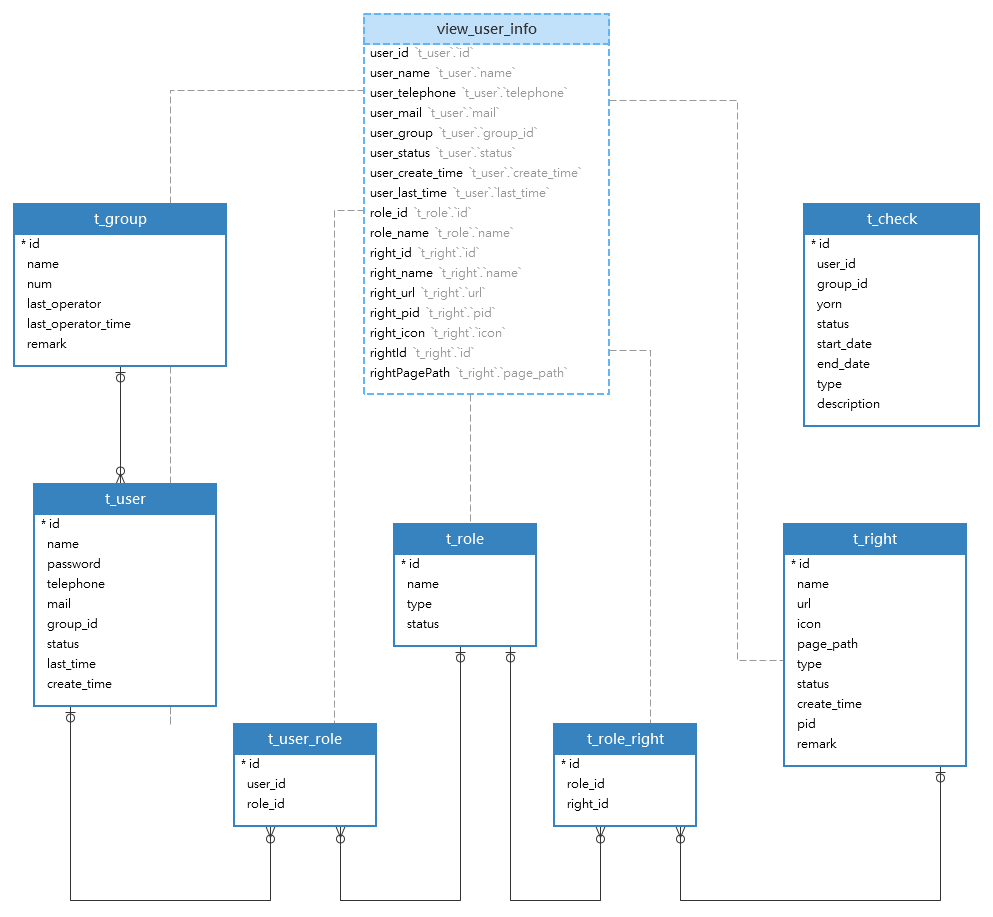


图3.2 ER图

### 3.2.2数据库设计

表3.1 用户信息表（表名：t\_user）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用途 | 存放用户的基本信息 | | | | |
| 字段名 | 类型 | 长度 | 键 | 中文名 | 备注 |
| id | int | 0 | 主键 | 用户编号 | 用户编号 |
| name | varchar | 16 |  | 用户名 | 用户名 |
| password | varchar | 32 |  | 用户密码 | 用户密码 |
| telephone | varchar | 12 |  | 用户手机号 | 用户手机号 |
| mail | varchar | 16 |  | 用户邮箱 | 用户邮箱 |
| group\_id | varchar | 0 | 外键 | 所属班级编号 | 所属班级编号 |
| status | tinyint | 0 |  | 账户状态 | 账户状态 |
| last\_time | datetime | 0 |  | 上次登陆时间 | 上次登陆时间 |
| create\_time | timestamp | 0 |  | 用户创建时间 | 用户创建时间 |
| 相关表 | 班级信息表t\_group，用户角色表t\_role | | | | |

表3.2 用户角色绑定表（表名：t\_user\_role）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用途 | 存放用户编号对应的角色编号 | | | | |
| 字段名 | 类型 | 长度 | 键 | 中文名 | 备注 |
| id | int | 0 | 主键 | 用户角色编号 | 用户角色编号 |
| user\_id | int | 0 | 外键 | 用户编号 | 用户编号 |
| role\_id | int | 0 | 外键 | 角色编号 | 角色编号 |
| 相关表 | 用户信息表t\_user，用户角色表t\_role | | | | |

表3.3 角色表（表名：t\_role）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用途 | 存放角色的基本信息 | | | | |
| 字段名 | 类型 | 长度 | 键 | 中文名 | 备注 |
| id | int | 0 | 主键 | 角色编号 | 用户编号 |
| name | varchar | 16 |  | 角色名 | 角色名 |
| type | tinyint | 0 |  | 角色类型 | 角色类型 |
| status | tinyint | 0 |  | 角色状态 | 角色状态 |
| 相关表 | 角色权限表t\_role\_right | | | | |

表3.4 角色权限绑定表（表名：t\_role\_right）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用途 | 存放用户编号对应的角色编号的表 | | | | |
| 字段名 | 类型 | 长度 | 键 | 中文名 | 备注 |
| id | int | 0 | 主键 | 角色权限编号 | 用户角色编号 |
| role\_id | int | 0 | 外键 | 角色编号 | 角色编号 |
| right\_id | int | 0 | 外键 | 权限编号 | 权限编号 |
| 相关表 | 角色表t\_role，权限表t\_right | | | | |

表3.5 权限表（表名：t\_right）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用途 | 存放权限信息的表 | | | | |
| 字段名 | 类型 | 长度 | 键 | 中文名 | 备注 |
| id | int | 0 | 主键 | 权限编号 | 权限编号 |
| name | varchar | 16 |  | 权限名 | 权限名 |
| url | varchar | 255 |  | 权限类型 | 角色类型 |
| iocn | varchar | 16 |  | 权限状态 | 角色状态 |
| page\_path | varchar | 32 |  | 路由地址 | 路由地址 |
| type | int | 0 |  | 权限类型 | 权限类型 |
| status | int | 0 |  | 权限状态 | 权限状态 |
| create\_time | timestamp | 0 |  | 权限创建时间 | 权限创建时间 |
| pid | int | 0 |  | 父级权限编号 | 父级权限编号 |
| remark | varchar | 255 |  | 权限备注 | 权限备注 |
| 相关表 | 角色权限表t\_role\_right | | | | |

表3.6 班级表（表名：t\_group）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用途 | 存班级信息的表 | | | | |
| 字段名 | 类型 | 长度 | 键 | 中文名 | 备注 |
| id | int | 0 | 主键 | 班级编号 | 班级编号 |
| name | varchar | 32 |  | 班级名 | 班级名 |
| num | int | 0 |  | 班级人数 | 班级人数 |
| last\_operator | varchar | 16 |  | 最后操作的人 | 最后操作的人 |
| last\_operator\_time | datetime | 0 |  | 最后操作的时间 | 最后操作的时间 |
| remark | varchar | 255 |  | 备注 | 备注 |
| 相关表 | 用户表t\_user | | | | |

表3.7 考勤表（表名：t\_check）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用途 | 存放角色的基本信息 | | | | |
| 字段名 | 类型 | 长度 | 键 | 中文名 | 备注 |
| id | int | 0 | 主键 | 考勤ID | 考勤ID |
| user\_id | int | 0 |  | 用户编号 | 用户编号 |
| group\_id | int | 0 |  | 班级编号 | 班级编号 |
| yorn | int | 0 |  | 是否同意批假 | 是否同意批假 |
| status | tinyint | 0 |  | 请假状态 | 请假状态 |
| start\_date | datetime | 0 |  | 请假时间 | 请假时间 |
| end\_date | datetime | 0 |  | 销假时间 | 销假时间 |
| type | varchar | 255 |  | 类型 | 类型 |
| description | varchar |  |  | 描述 | 描述 |
| 相关表 | 用户表t\_user | | | | |

### 3.2.3 IPO（输入加工输出图）

系统：查询用户子系统

作者：王志

模块：分页查询模块

日期：2022.04.20

编号：001

被调用：

调用：数据库

输入：

当前页，页面条数。

输出：

用户信息的集合

处理：

将网页上接受到的数据包装成分页查询的SQL语句，从数据库查出相应的结果，包装成JSON对象返回给前端。

局部数据元素：用户信息

注释：

图3.3 用户分页查询的IPO图

系统：用户管理子系统

作者：王志

模块：用户操作模块

日期：2022.04.20

编号：002

被调用：

调用：数据库

输入：

用户信息。

输出：

成功输出成功，

失败输出相应的理由。

处理：

根据前端传入的用户信息，查询该用户是否存在，存在就修改，不存在就添加。修改时判断用户名是否重复。

局部数据元素：用户信息

注释：

图3.4 添加或修改用户的IPO图

系统：学生考勤系统

作者：王志

模块：设置用户权限模块

日期：2022.04.20

编号：003

被调用：

调用：数据库

输入：

角色编号，权限列表

输出：

是否修改成功。

处理：

根据前端传入的参数，开启事务，先将数据库中该角色所有权限删除，在设置为传入的权限。失败时回滚。

局部数据元素：角色列表

注释：

图3.5 添加或修改用户的IPO图

## 3.3系统其他需求

可恢复性：数据库应定期备份。

安全性：系统运用了Shiro安全框架[10]，对身份的验证、授权、密码和会话进行管理。

扩展性[11]：低耦合，高内聚的代码设计，公用方法尽量抽出来写。

# 4 宏观设计

本章中的设计充分考虑了软件的易理解性、可拓展性、可重用性与可维护性，坚持面向对象的设计原则[12]

## 4.1 系统功能结构分析

图4.1 学生考勤系统的软件结构

## 4.2 架构模式

本系统采用Java技术进行网站开发，B/S模式，使用Chrome浏览器运行

## 4.3数据库设计环境配置说明

使用当前社区活跃度最高的免费的关系型数据库MySQL，数据模型采用RBAC模型[13] 。数据库的管理工具是Navicat。

# 5 详细设计

## 5.1 软件宏观结构



图5.1 学生考勤系统流程图

由图5.1可以看出，本平台使用时，会先进入主页，进入主页发现用户未登录，会跳转至用户登录页面。如果已登录会根据用户的角色信息加载其所拥有的权限菜单。

当登陆用户为学生时，可以进行请假和查看历史请假的功能。当学生请假时，系统会在考勤表上添加一条数据，初始状态为待批阅。使用请假历史功能时系统会从数据库查询出所有当前用户的请假历史并在前端展示。

当登录用户为教师时，可以进行批假和查看历史批假功能。当教师批假时，系统会把根据当前假条的编号修改考勤表中的请假状态。使用批假历史功能时系统会从数据查询当前教师所在班级的所有请假记录。

当登陆的用户是管理员的情况下，有用户组件，角色组件，权限组件，这些组件可以对上面提到的三个实体进行基本的增加、修改、删除和查询操作。

## 5.2 请/批假组件



图5.2 请假批假流程图

由图5.2可以看出，学生请假需要填写一些必要的信息（“病假”或者“事假”，请假开始的时间、销假的时间以及所需要的理由）这些信息都填写后点击申请按钮，系统会在数据库中插入一条请假信息，初始状态为待批阅。教师在学生提交申请后可以在批假管理看到学生的请假信息，可以选择同意和驳回。教师操作后，学生可在请假历史记录里看到这次请假的批阅状态。

表5.1 请假模块主要类

|  |  |
| --- | --- |
| 分类 | 包含的主要类 |
| 控制类 | CheckController.java |
| 业务逻辑类 | CheckServiceImpl.java |
| 持久对象类 | Check.java |
| 数据持久类 | CheckMapper.java |

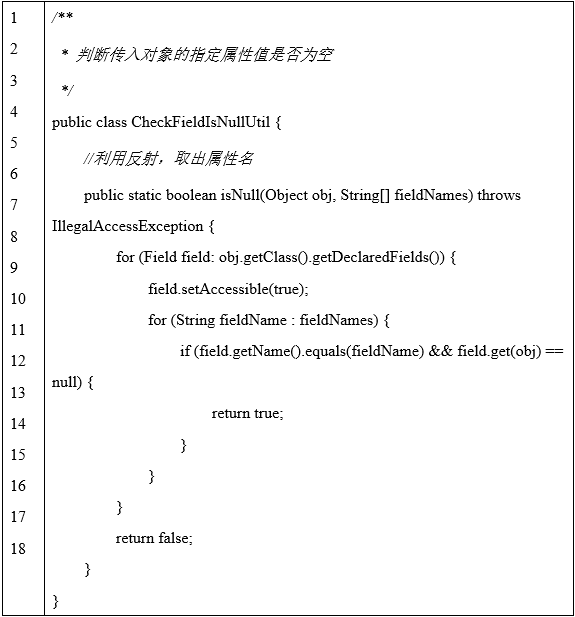
传入请假数据时，不能依赖前端的数据安全校验，因此通过前后端双重校验的方式，验证数据是否可靠，具体代码如下：

CheckController.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | @PostMapping("/create") public ResponseFormat createCheck(@RequestBody Check check) {  *//需要校验的属性名* String[] fieldNames = {"userId", "groupId", "startDate", "endDate", "type"};  try {  if (CheckFieldIsNullUtil.*isNull*(check, fieldNames)) {  return ResponseFormat.*fail*("请填写具体的表单内容");  }  } catch (IllegalAccessException e) {  return ResponseFormat.*fail*("表单不符合标准");  }  checkMapper.insert(check);  return ResponseFormat.*successful*("已通知辅导员，请不用重复提交申请"); } |

在判断对象的某个属性值是否为空时，本系统利用自定义的CheckFieldIsNullUtil类中的isNull方法来实现。该方法需要两个参数，参数一：需要判断的对象，参数二：需要判断的属性名。原理是利用反射技术，获取传入对象的属性，当属性名与参数二中的属性名相同时，循环取出参数二中所有属性值，如果参数二中有任一个属性值不存在的话，返回个布尔值true，反之为false。

CheckFieldIsNullUtil.java



## 5.3 用户信息组件



图5.3 用户管理流程图

由图5.3可以看到，进入用户管理模块时，有添加用户功能、修改用户、删除/批量删除用户、按条件查找、导入导出用户和角色设置。本页面在后端添加了角色认证，需要管理员权限才可以访问。

表5.2 用户模块主要类

|  |  |
| --- | --- |
| 分类 | 包含的主要类 |
| 控制类 | AccountController.java  UserController.java |
| 业务逻辑类 | AccountServiceImpl.java  UserServiceImpl.java |
| 持久对象类 | Account.java  User.java  LoginDTO.Java  UserDTO.java |
| 数据持久类 | AccountMapper.java  UserMapper.java |

对于用户的登录登出本系统单独使用AccountController来进行业务逻辑管理，其他业务逻辑则放在UserController。

下图是登录方法，该方法需要两个参数，参数一：LoginDTO，参数二：HttpServletResponse。当调用该方法时，系统前端传入LoginDTO对象，Web容器传入HttpServletResponse对象，login()方法通过业务逻辑层到数据持久层查询这个用户是否在数据库中，如果存在，则通过本系统自定义的JWTUtil类，生成JWT,设置响应头的Authorization的值为生成的Token并加上数据库的查出的用户信息返回给前端。

AccountController.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | @PostMapping("/login") public ResponseFormat login(@Validated @RequestBody(required = false) LoginDTO dto, HttpServletResponse response) {  *//查询数据库是否存在该用户* User user = service.getOne(new QueryWrapper<User>().eq("name", dto.getName()));   *//省略部分代码*  *//用户名密码正确 token不存在或者失效 生成Token返回给前端* String jwt = jwtUtil.generateToken(user);  response.setHeader("Authorization", jwt);    *//设置返回的header* 把*Authorization对象暴露*  response.setHeader("Access-Control-Expose-Headers", "Authorization");  *//执行登录* SecurityUtils.*getSubject*().login(new JwtToken(jwt));  UserDTO userInfo = service.getUserInfo(user.getId());  return ResponseFormat.*successful*(userInfo); } |

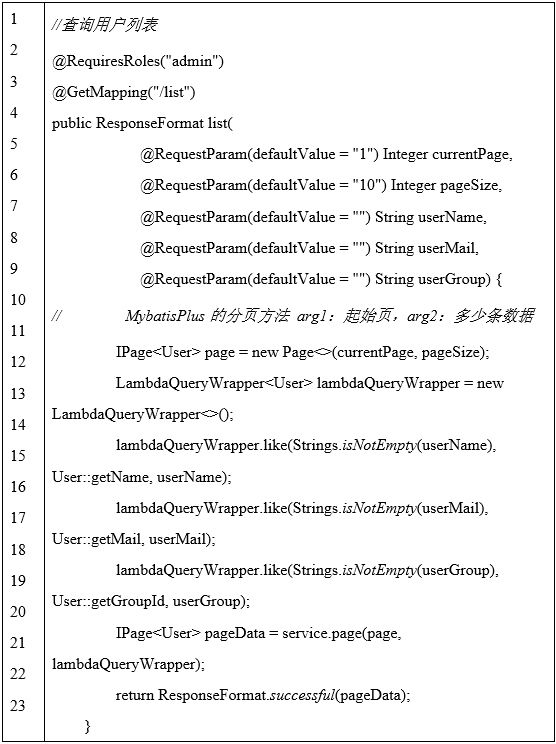
当执行SecurityUtils.getSubject().login()方法时，登录用户的信息会被传递给Realm进行认证和授权。下图为登陆的认证过程，首先，本系统会调用getUserInfo()方法，获取用户的角色和权限信息。调用SimpleAuthorizationInfo ()类中的addRole()方法。为User添加角色。添加好角色后，添加权限需要再调用addStringPermissions()方法传入权限列表。最后返回已经被授权了的用户信息。

AccountRealm.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | *// 授权* @Override  protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection principalCollection) {  SimpleAuthorizationInfo info = new SimpleAuthorizationInfo();  *//获取token对应的用户* User user = (User) principalCollection.getPrimaryPrincipal();  try {  *//查询出用户相关角色和权限，添加到Subject中*  UserDTO userInfo = userService.getUserInfo(user.getId());  info.addRole(userInfo.getRoleName());  List<String> url = new ArrayList<>();  userInfo.getRights().forEach(right -> url.add(right.getUrl()));  info.addStringPermissions(url);} catch (Exception e) {  ResponseFormat.*fail*("权限查询失败");  }  return info;  } |

下图为用户管理页的分页查询代码。参数一：当前页数，参数二：页面的条数，参数三：模糊查询时的用户名，参数四：模糊查询时的邮箱，参数五：模糊查询时的用户班级。调用该方法时，系统会调用Mybatis-plus中的分页查询方法，需要传入当前页码和页面大小。使用LambdaQueryWrapper类可以添加一些参数不为空的值，进行模糊查询，最后返回分页查询的结果集。

UserCotroller.java



## 5.4 角色信息组件



图5.4 角色管理流程图

由图5.4可以看出，进入角色管理模块时，有角色信息的添加组件、有删除角色组件，还有多选框选中使用批量删除角色的组件、修改角色信息、按条件查找和权限设置。角色表通过用户角色表关联，后端操作数据库需要前端传入相应的角色信息。

表5.3 角色管理模块主要类

|  |  |
| --- | --- |
| 分类 | 包含的主要类 |
| 控制类 | RoleController.java  RoleRightController.java |
| 业务逻辑类 | RoleServiceImpl.java  RoleRightServiceImpl.java |
| 持久对象类 | Role.java  RoleRight.java |
| 数据持久类 | RoleMapper.java |

下图是设置权限的代码，参数是角色权限对象，当前端调用该接口时本系统会调用同步权限的方法。

RoleRightController.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | @PostMapping("/setRoleRight") public ResponseFormat setRoleRight(@RequestBody RoleRightDTO dto) {  *//省略部分代码*  service.syncRight(dto);  return ResponseFormat.*operate*(200, "权限设置成功", null); } |

下图是同步设置权限的方法，参数是RoleRightDTO对象，调用该发方法首先会删除当前角色的所有权限，再根据dto中的权限列表设置该角色所拥有的权限。使用@Transactional注解，因为设置权限前需要删除所有权限，要开启事务保证设置权限失败时回滚。经测试，当插入权限方法执行失败时可以回滚。

RoleRightServiceImpl.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | *//开启事务，失败时回滚* @Transactional public void syncRight(RoleRightDTO dto) {  QueryWrapper<RoleRight> wrapper = new QueryWrapper<>();  wrapper.eq("role\_id", dto.getRoleId());  mapper.delete(wrapper); *//先删除当前用户的所有权限* List<Integer> rightIds = dto.getRightIds();  rightIds.forEach(i -> {  RoleRight roleRight = new RoleRight();  roleRight.setRoleId(dto.getRoleId());  roleRight.setRightId(i);  mapper.insert(roleRight);  }); } |

## 5.5 权限信息组件



图5.5 角色管理流程图

表5.4 角色管理模块主要类

|  |  |
| --- | --- |
| 分类 | 包含的主要类 |
| 控制类 | RightController.java |
| 业务逻辑类 | RightServiceImpl.java |
| 持久对象类 | Right.java |
| 数据持久类 | RightMapper.java |

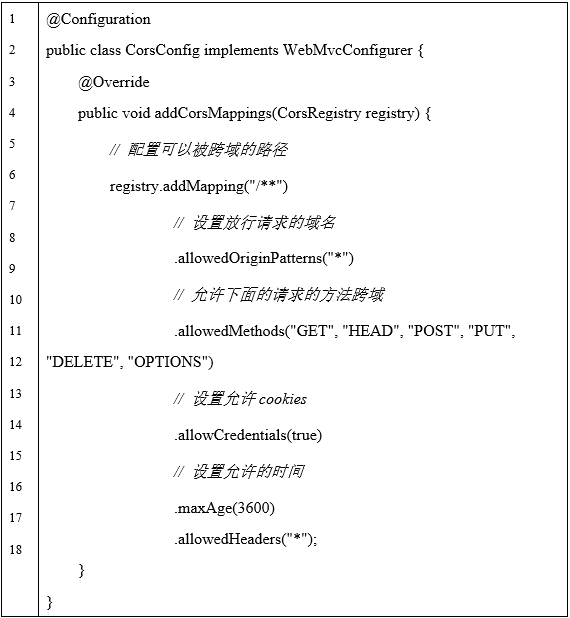
下图是添加和修改权限信息的代码，参数：权限信息。当调用该方法时，首先会判断权限信息中是否包含了权限编号，如果包含权限编号，根据权限编号修改权限信息。修改权限名称时，需要判断当权限色表中除了该权限叫此名称，是否还有其他权限名称与待修改权限名称相同。如果有其他权限名称和待修改的名称相同，则提示权限名已存在。如果不存在则调用saveOrUpdate()方法，进行保存，最后返回结果集。

RightController.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | @PostMapping("/add") public ResponseFormat add(@RequestBody Right right) {  if (right.getId() != null) {  Right temp = mapper.selectOne(new QueryWrapper<Right>().eq("name", right.getName()));  if (temp == null || right.getId().equals(temp.getId())) {  return ResponseFormat.*successful*(service.saveOrUpdate(right));  }  return ResponseFormat.*fail*("权限名已存在");  }  if (mapper.selectCount(new QueryWrapper<Right>().eq("name", right.getName())) > 0) {  return ResponseFormat.*fail*("权限名已存在");  }  return ResponseFormat.*successful*(service.saveOrUpdate(right)); } |

## 5.5 其他重要组件

后端处理跨域问题。有本项目是前后端分离的项目，所以处理跨域是必不可少的[14]。



# 6 系统使用说明

## 6.1 用户模组

用户登录页面用来验证用户账户和身份，验证成功后进入项目主页。点击登录按钮，如果数据库中这个User没有查询结果，弹出相关信息“用户不存在”以此提示用户，若User有查询结果，比对UserNmae和UserPassword 是否和前端传入的值相同，如果不同弹出相关信息“密码不正确”以此提示用户，校验通过会进入系统主页，达到优化用户体验的目的。如图6.1。



图6.1 用户登录页面

## 6.2 学生模组

当系统验证为学生时展示此页面（图6.2），左侧导航栏有主页，我要请假和请假记录三个模块，需要请假时要填写请假的信息。如果填写不完整系统会弹出请填写具体的表单内容。如果填写的内容合法，弹出已通知辅导员，请不要重复申请。请假完成时，可以在请假记录模块中看到历史请假记录（图6.3），如果教师未批阅会显示待批阅。已批阅会显示教师的意见。

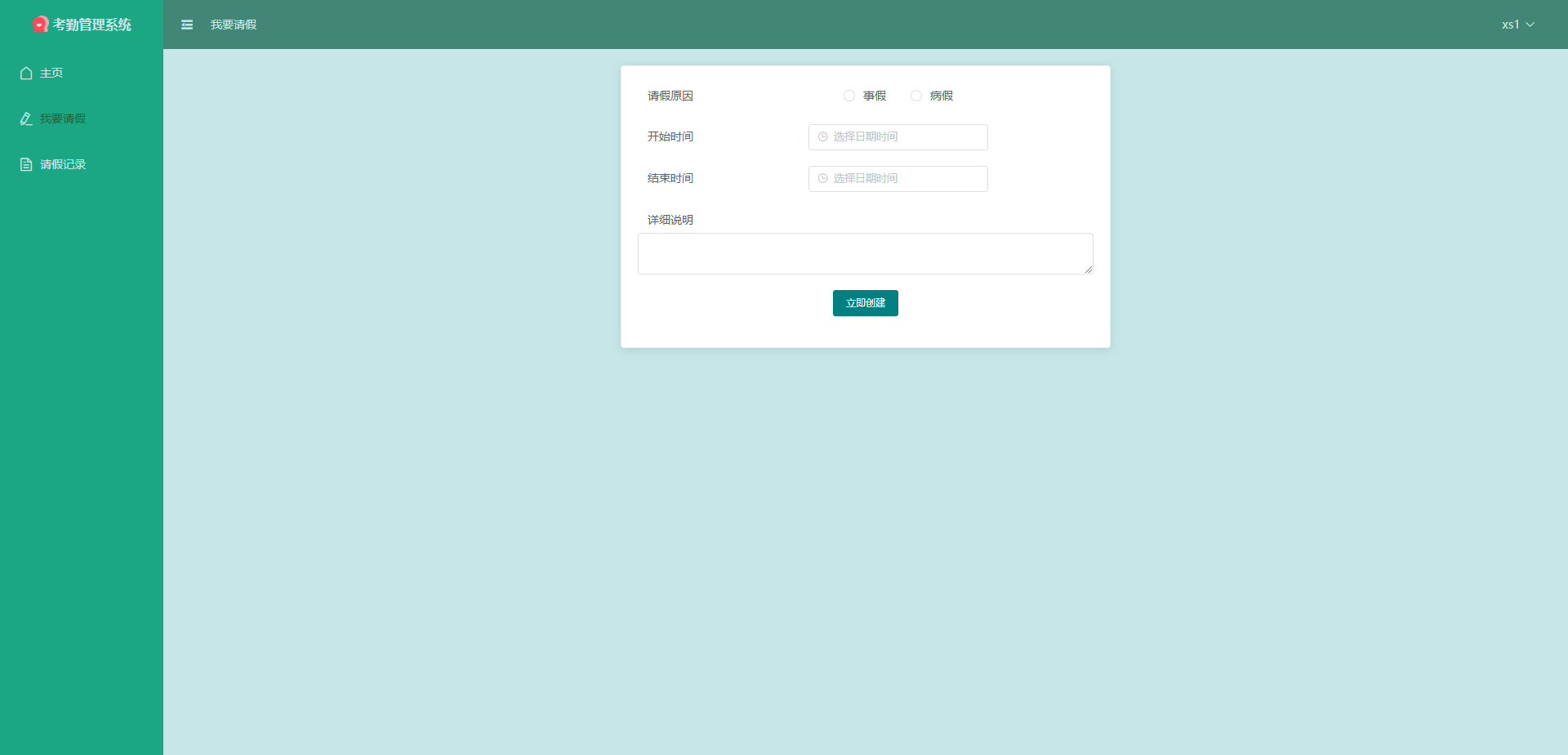


图6.2 我要请假模块

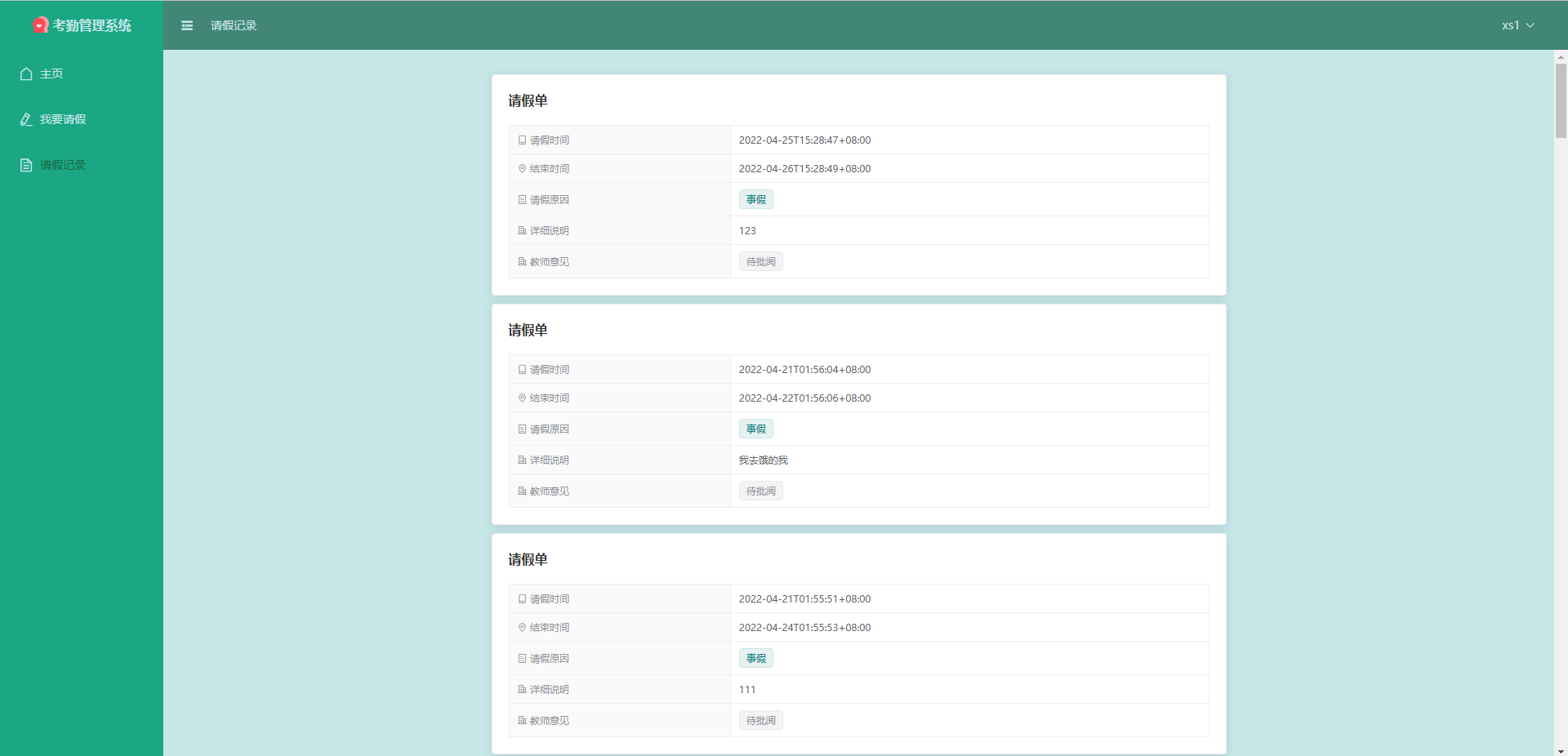


图6.3 请假记录模块

## 6.3 教师模组

当系统验证为教师时展示此页面（图6.4），左侧导航栏有主页，请假名单和批假记录三个模块，需要批假时可以点击请假单右上角同意或者拒绝。批假完成时，可以在批假记录模块中看到历史批假记录（图6.5），如果教师未批阅则不会展示在批假记录模块中。

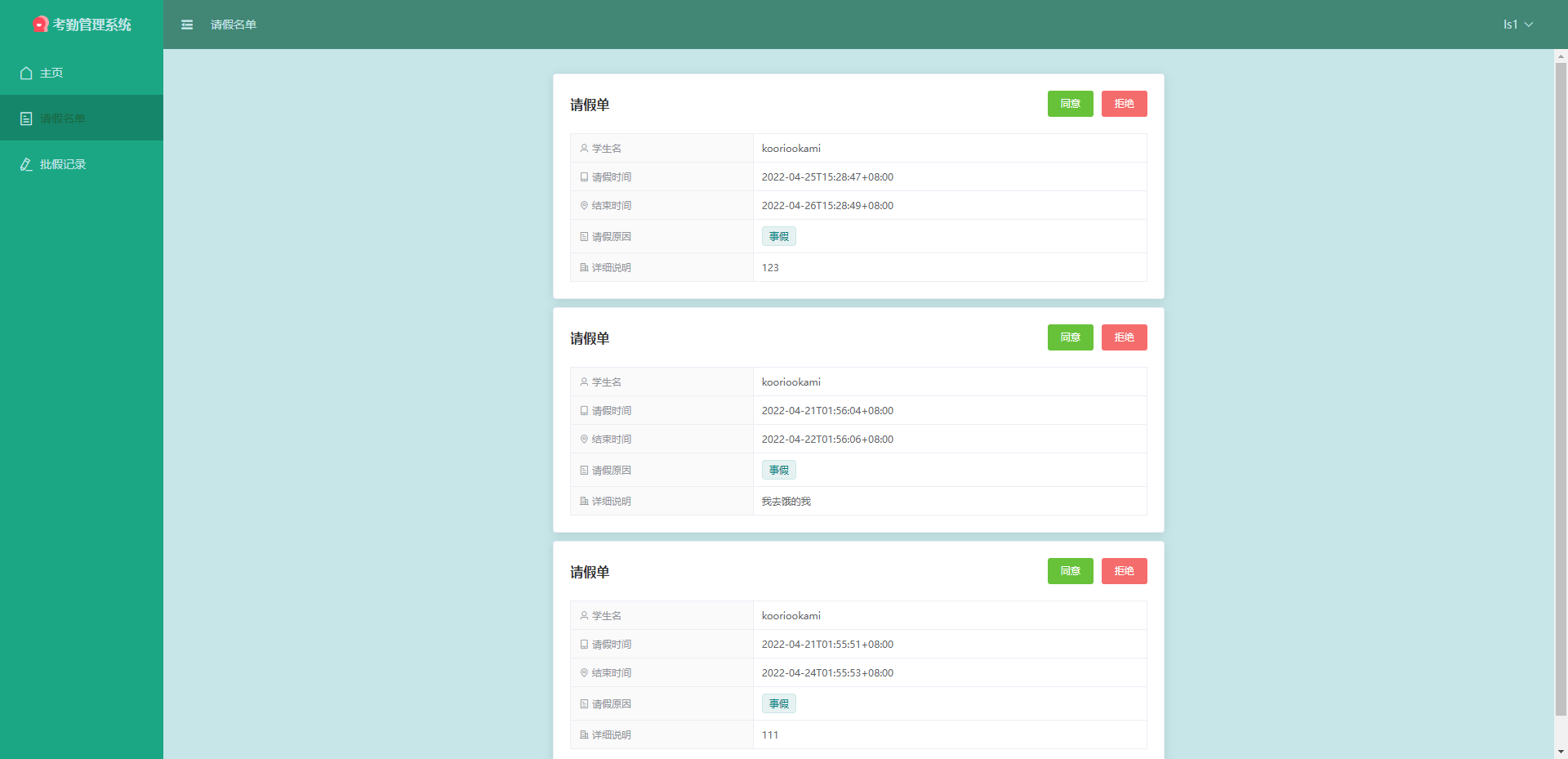


图6.4 请假名单模块

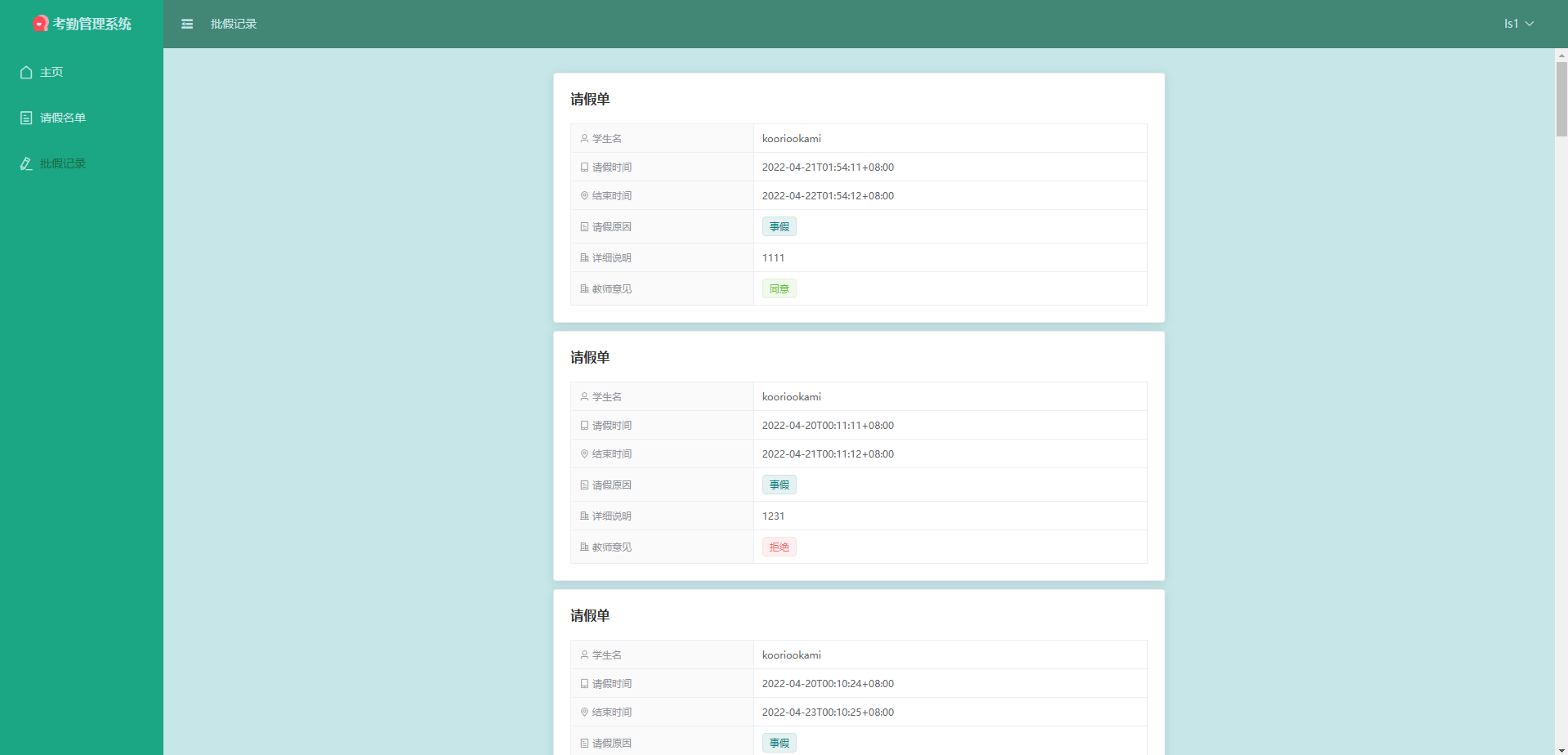


图6.5 批假记录模块

## 6.4管理用户模组

当系统验证为管理员时，展示管理员这个角色的页面（图6.6）。点击用户管理可以看到屏幕上出现所有用户信息列表，可以通过屏幕上特定的组件对用户进行相关操作，也可以设置用户角色。

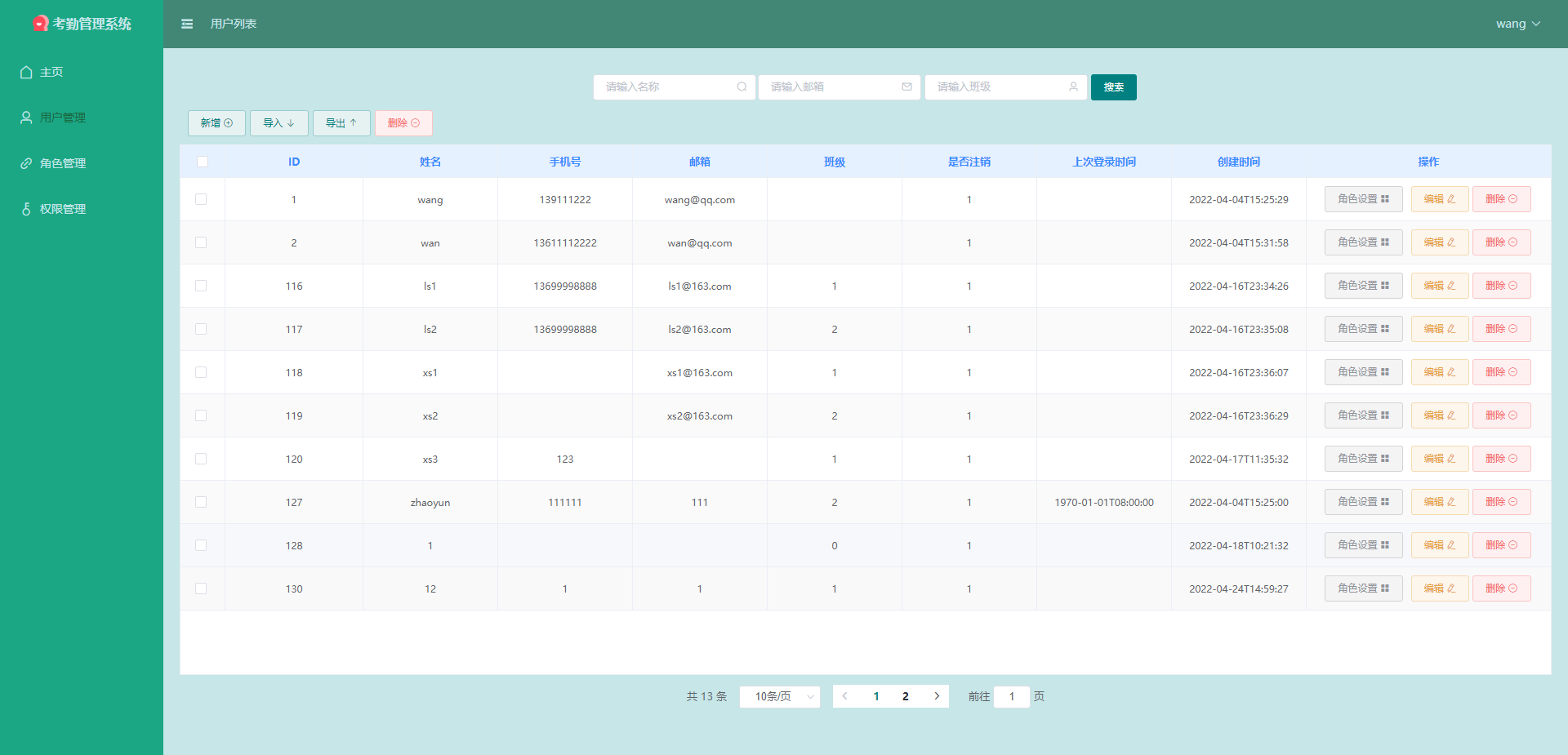


图6.6 用户管理模块

点击左上角新增按钮，屏幕上出现“用户信息”弹窗（图6.7），管理员填写完整用户信息后，点击确定系统会验证用户名是否已存在，存在弹出提示“用户名重复”，不存在提示“保存成功”，之后会刷新用户列表。



图6.7 新增用户

点击导入按钮会弹出Windows的资源管理器（图6.8），选择需要导入的Excel进行用户的导入。

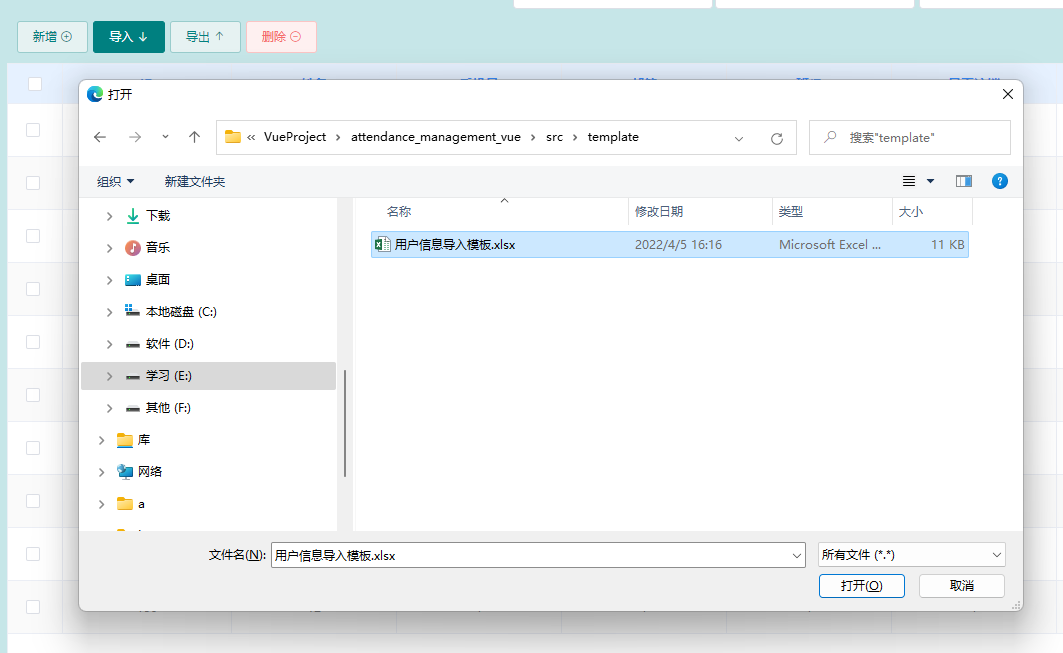


图6.8 导入用户

点击导出组件，本系统右上角会弹出下载框（图6.9）用户可以在Office电子表格打开查看导出的用户数据。

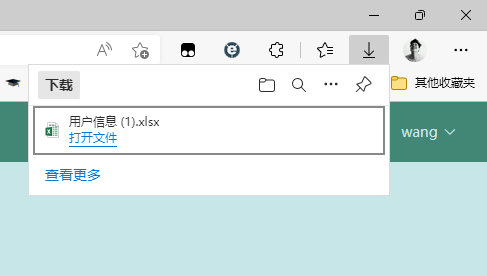


图6.9 导出用户

点击编辑按钮会弹出用户信息弹窗（图6.10），弹窗中显示当前用户的信息。用户点击保存按钮保存修改完的记录。如果用户名已存在，则修改不成功，系统提示用户名已存在。

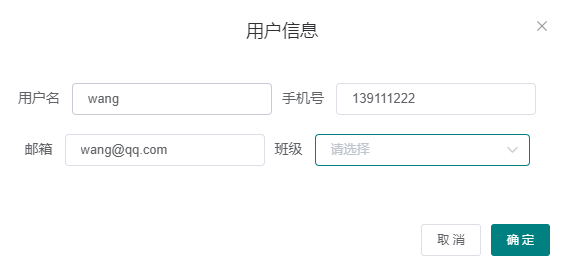


图6.10 编辑用户信息

点击某个用户的删除按钮，系统会弹出确定删除的提示，再次点击确定，会删除该用户。



图6.11 删除用户

选中多选框，点击顶部的删除按钮，系统会弹出确定删除的提示，再次点击确定，会删除选中的所有用户。

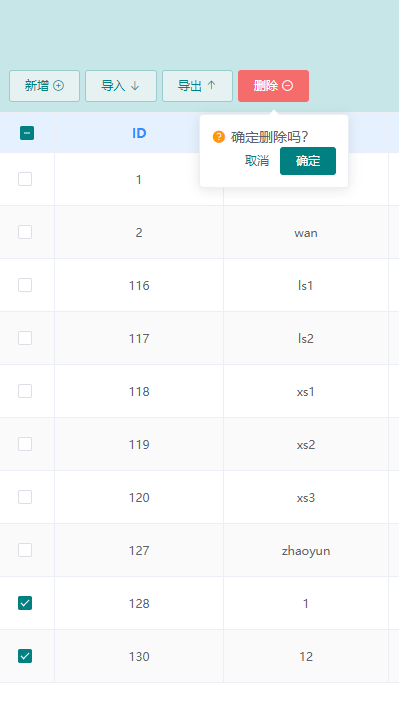


图6.12 批量删除用户

按下角色设置按钮，屏幕上会出现角色设置的组件（图6.12），在这个组件上可以对User的Role进行设置，设置Role是为了集中式管理，当User需要一些权限时，不需要在为用户一个一个添加权限了，只需要给用户设置一个角色就可以。角色的添加可在角色管理组件中使用新增按钮进行添加。

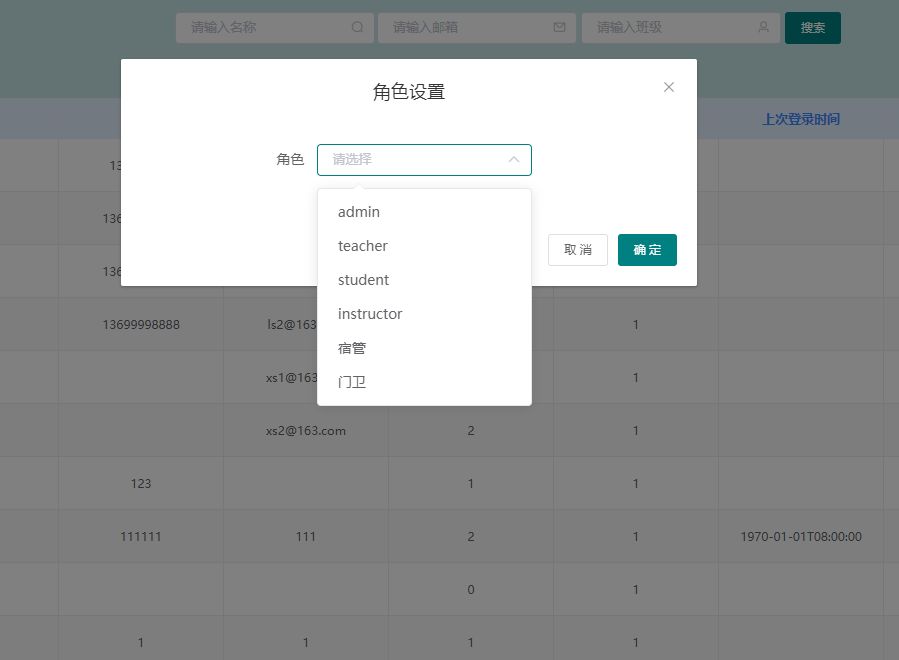


图6.13 角色设置

## 6.5 管理角色模组

基本操作和用户类似，主要展示不同点。

下图为角色管理中的权限设置（图6.13），作用是为当前Role设置不同的权限，用户登录后会加载用户所属角色的所有权限，并根据角色的权限加载不同的导航栏。例如一个用户的角色为学生，学生的权限只有我要请假，则只要是学生的登录上本平台之后在导航栏只能看到我要请假模块。

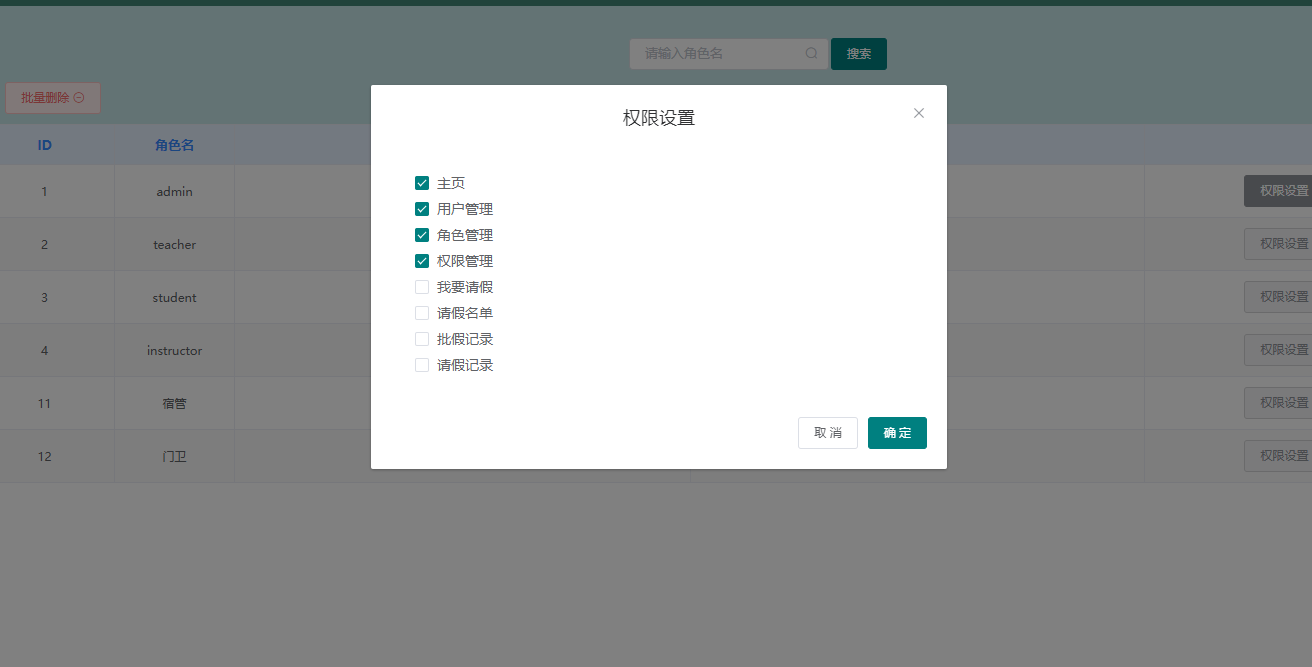


图6.13 权限设置

## 6.6 管理权限模组

操作逻辑同用户管理和角色管理。

# 7 总结

时间过得很快，我的毕业论文就快要写完了。前前后后用了差不多两个月的时间，从最初的我和我的导师李峰老师沟通确定选题，到选题之后的技术选型，我之前是做过SSM项目的，想在这次毕业设计的时候多学一点新技术。就确定了这次的技术栈SpringBoot+Vue+MySQL，安全框架选择Shiro，认证方案用的是JWT。由于我的前端技术薄弱，所以界面采用了Element UI，请求框架采用了Axios，数据可视化用的是Echarts，好的工具大大提高了了我的开发效率。

在这次的毕业设计开发中，我学到了很多东西，基本掌握了前后端分离开发的流程，积累了很多软件开发的经验，锻炼了独立解决问题的能力。也走了不少弯路，由于最初的想法没那么成熟，导致后来在开发过程中有很多设计不合理的地方，功能也较为单一。有很多好的想法没能实现，如教师课堂点名功能，学生指定时间段请假功能，任课教师查看本节课请假名单功能。虽然未能实现上述功能，但本系统还是有很多可取之处的，如基于RBAC模型[15]的数据库的设计，可以更好地管理用户，为多表关联查询建立视图简化查询操作，后端加入权限认证，双重验证防止用户信息泄漏。

对于毕业设计付出的努力是值得的，在今后的学习中，我还将继续奋斗。

# 参考文献

1. 胡鞍钢. 中国实现2030年前碳达峰目标及主要途径[J]. 北京工业大学学报(社会科学版),2021.5
2. 黄缙华. MySQL入门很简单[M]. 北京:清华大学出版社,2011.1.
3. 朱二华. 基于Vue.js的Web前端应用研究[J]. 科技与创新, 2017(20):119-121.
4. 贺紫珺. 基于SpringBoot和Vue框架的第三方医疗器械供应链平台的设计与实现[D]. 上海:华东大学,2019
5. 麦冬,陈涛,梁宗湾. 轻量级响应式框架Vue.js应用分析[J]. 信息与电脑(理论版),2017(07):58-59.
6. Deqing Li, Honghui Mei, Yi Shen, Shuang Su, Wenli Zhang, Junting Wang, Ming Zu, Wei Chen. ECharts: A Declarative Framework for Rapid Construction of Wed-based Visualization[Z]. Visual Informatics, 2018.
7. 王振宇. 基于 SpringBoot 的整车出库管理系统设计与开发[D]. 南京邮电大学,2018.
8. 蒋黎旺. 物流公司仓储信息管理系统的设计与实现[D]. 成都:电子科技大学,2002.
9. 胡亚捷. 基于消息队列的账单聚合系统的设计与实现[D]. 南京:东南大学,2020
10. Apache Shiro. Understand Permissions in Apache Shiro [Z]. <http://shiro.apache.org/permissions.html>.
11. 焦鹏珲. 基于SpringBoot和Vue框架的电子招投标系统的设计与实现[D]. 南京大学,2018.
12. 黎炳权. 基于Web和移动应用的高校学生信息管理平台[D]. 东华大学,2017.
13. Sandhu R S, Coyne E J, Feinstein H L, Youman C E．Role-based access control models[Z]．IEEE Computer，1996，29 （2）：38～47
14. 袁满. 多模式匹配算法在电子病历系统中的应用[D]. 武汉:华中师范大学,2020
15. 刘建圻, 曾碧, 郑秀璋. 基于 RBAC 权限管理模型的改进与应用[J]. 计算机应用, 2008,28(9):2449-2451.

# 致谢

如果没有XX老师和XXX老师的悉心指导，本论文就不可能完成。从提纲的起草到论文的成稿、修改、定稿。多次从XX老师那里得到了的宝贵意见。论文才得以顺利完成。给我留下了深刻的印象的是导师渊博的知识和严谨的治学态度，指导期间让我受益匪浅！借此机会，对导师表达崇高的敬意和由衷的感谢！

最后，感谢各位评审委员和答辩委员在百忙之中抽出时间阅读本文，敬请指导。