



基于SpringBoot框架的毕业论文管理系统

|  |  |
| --- | --- |
| 学院 | 计算机 |
| 专业 | 计算机科学与技术 |
| 年级班别 | 计算机科学与技术2班 |
| 学号 | 3118004130 |
| 学生姓名 | 吴海韬 |
| 指导教师 | 陈平华 |

2022年4月 20

摘要

近年来，随着高等院校的不断扩招，本科毕业人数不断增加，学校处理毕业论文的压力也不断增大。为了解决的高等院校毕业论文管理流程繁杂，交流困难，文件信息繁多的问题，本文设计了基于SpringBoot框架的毕业论文管理系统，提供了用户管理、学生管理、导师管理、课题管理、毕业选题、过程文档管理、成绩管理等功能。同时使用simhash算法实现了查重功能，有效减少了管理员的工作量，提高高校对论文管理与论文质量把控的工作效率，便利学生与导师之间交流，对推动高等院校全面电子化、信息化具有重要意义。

关键词：SpringBoot，simhash，毕业论文管理

Abstract

In recent years, With the continuous expansion of colleges and universities, the number of undergraduate graduates continues to increase, and the pressure on schools to deal with graduation thesis is also increasing . In order to solve the problems of complicated dissertation management process, difficult communication and large amount of document information in colleges and universities, this paper designs a graduation thesis management system based on SpringBoot framework, which provides user management, student management, tutor management, topic management, graduation topic selection, process document management, grade management and other functions. At the same time, the simhash algorithm is used to realize the function of checking weights, which effectively reduces the workload of administrators, improves the work efficiency of colleges and universities on paper management and paper quality control, facilitates communication between students and tutors, and is of great significance to promoting the comprehensive electronic and informatization of colleges and universities

Key words: SpringBoot , simhash , graduation thesis management

目录

[1 输入您的章标题 1](#_Toc94839844)

[1.1 输入您的节标题 1](#_Toc94839845)

[1.1.1 输入您的条标题 1](#_Toc94839846)

[结论 2](#_Toc94839847)

[参考文献 3](#_Toc94839848)

[致谢 4](#_Toc94839849)

[附录 A 输入您的附录标题 5](#_Toc94839850)

# 绪论

* 1. 研究背景

撰写毕业论文是检验学生在校学习成果的重要措施，也是提高教育质量的重要环节。大学生在毕业前都必须完成毕业论文的撰写任务。申请学位必须提交相应的学术论文，经答辩通过后，方可取得学士学位。虽然毕业论文不一定能对社会直接带来巨大的效益，对专业产生开拓性的影响，但撰写毕业论文是提高教学质量的重要环节，是保证出好人才的重要措施。而对毕业论文流程的把控和对质量的检测是保证学生顺利撰写毕业论文的重要举措，也是高等院校保证论文质量的重要前提

* 1. 研究意义

高等院校毕业论文管理流程繁杂，交流困难，文件信息繁多。一般要经历：分配导师、导师出题、学生选题、课题开题、论文设计、提交论文、导师评阅、论处查重、综合评审、论文答辩、成绩评定等环节，需要导师、学生、教务管理员三方协作交流。随着中国高等院校的进一步扩招，以往的微信交流传送文件、人工线下搜集资料的方法已经难以满足日渐庞大论文管理工作。而如近高等院校信息化程度不断提高，互联网软件发展迅猛。结合当今互联网软件技术，急需设计出一个供学生、导师、管理员使用的毕业论文管理系统

* 1. 研究现状

为了更加方便地对毕业生毕业论文的指导，高等院校或企业开发毕业论文管理系统， 多数毕业论文系统的设计采用的是 B/S 架构，不需要下载客户端方便操作。大多数的论文管理系统使用的是 Spring+SpringMVC 为后端服务技术，但是当前 Spring 技术已经不再流行，更多的人更想使用 SpringBoot 为后端技术，相较于 Spring 框架SpringBoot 更简单，减少了很多的样板代码[6]。

# 相关技术介绍

* 1. SpringBoot框架

Spring Boot 是由Pivotal团队2013年提供的在JavaEE编程领域一个全新的轻量级框架。是基于Spring的4.0版本设计的，承载了原有Spring框架的优秀基因，为简化Spring项目配置而生，能轻松地创建独立的、可生产的、基于Spring的且能直接运行的应用程序[7]

* 1. Vue框架

Vue.js是一款MWM前端框架，通过双向绑定数据，数据变化实时反映并映射到虚拟DOM上，简化了前端开发流程，提升了开发效率。同时，Vue也是一款用于构建前端用户界面的渐进式框架自底向上逐层应是Vue框架 最大的特点。Vue框架更多是关注视图层，并且Vue能够非 常 便利的与第三方项目整合。通过与现代化的工具链和 各种类库结合使用 Vue可为复杂的单页面用程序提供驱动[8]。

* 1. MySQL 数据库

MySQL 数据库是当下最受欢迎的数据库，采用标准化SQL 语言搭建，具有开源代码，有运行速度快、安全性高、支持多种平台、支持多种开发语言等特点。项目中将博客、用户、管理员等信息都保存在 MySQL 数据库的表中[10]。

# 系统分析

* 1. 可行性分析
     1. 技术可行性

在开发技术方面，本项目使用的主流的Java编程语言、java具有提供了丰富的类库，很方便开发程序，Java虚拟机技术保证了可移植性和安全性，支持继承，重载，多态等面向对象的特性便于网站维护。idea程序开发工具，具有智能代码助手、代码自动提示、重构等功能，便于系统开发

在数据库技术方面，本项目采用Mysql，由于 MySQL 数据库体积小、速度快、总体拥有成本低、开放源代码，其有着广泛的应用在中小心型网站的开发中占据很大的市场

* + 1. 操作可行性

在使用人群方面，管理员一般要求掌握计算机技术，会使用基础的管理软件，熟悉IT产品，有丰富的管理系统使用经验。在学生和导师方面，现代互联网普及程度较高，高等院校人员对互联网软件操作已十分熟悉，只需要提供操作文档，即可快速上手使用

* 1. 系统需求分析
     1. 功能需求

为了更好的进行系统功能分析，应从系统用户的角度出发，分析用户的需求，了解用户使用方式和思维习惯，来帮助我们更好的分析系统设计。毕业论文管理系统有三个用户主体，分别为：管理员、学生、导师

管理员:

1. 公告管理：公告管理分为发布公告、管理公告两个模块，发布公告可以输入公告的标题和和公告内容，完成公告的发布，管理公告可以显示目前的所以的公告信息，并且可以对其进行删除、更改和模糊查询
2. 菜单管理：显示目前系统中各菜单的属性，方便管理员进行后期维护的初步修改

c) 权限管理： 显示当前系统中的角色，可以增加修改删除角色，也可以为角色设置不同的菜单权限，只要拥有相应角色权限的用户的才能使用相应的菜单，实现登录权限控制

d) 人员管理：人员管理分为三个模块：学生管理、导师管理、用户管理。学生、导师管理页面显示相应的学生、导师的全部信息，同时可以实现对学生、导师的删除、修改和模糊查询。用户管理负责对用户信息进行管理，包括用户名、密码、角色、绑定的学号或工号、可以新增不同角色的用户、删除、修改和模糊查询

e) 文件管理：文件管理实现对不同的文件进行分类管理、本系统包括对开题报告和论文正文的管理，页面中可对文件删除、模糊查询，还可下载文件

导师：

1. 修改个人信息：显示个人信息，并且可以修改提交到后台
2. 课题管理：课题管理分为两个模块：上传课题和审核课题，上传课题部分需要填写课题名称、课题难度、课题类型、课题来源和课题摘要。如果学生选择自己发布的课题会在审核课题中显示，导师可决定是否通过选题，若选题未通过，学生不能进行下一流程
3. 查看学生信息：显示导师指导的学生各项信息
4. 过程文档管理：d) 过程文档管理分为两个模块:审核开题报告和审核毕业论文，学生提交的文档的会在该页面显示，导师需下载文档对文档进行评阅并填写评阅意见 ，再决定是否通过审核，毕论论文需对文档进行院内查重
5. 成绩管理：学生的毕业论文文档通过审核，方可进行成绩评定，该部分分为两个模块，添加成绩部分可以选择学生姓名、输入对应学生成绩进行成绩提交，查看成绩模块显示指导的学生的成绩，也可以对成绩结果进行修改

学生：

1. 毕业选题：毕业选题分为选择课题和课题结果两个模块，选择课题模块显示导师发布的课题，学生可以进行选择操作，在课题结果模块显示已选课题的信息，可以进行退选操作
2. 过程文档管理：过程文档管理分为提交开题报告和提交毕业论文两个模块，两个模块页面相似，学生可以提交报告到系统，同时显示最新已提交报告的的信息和审核情况信息。
3. 修改个人信息：显示个人信息，并且可以修改提交到后台

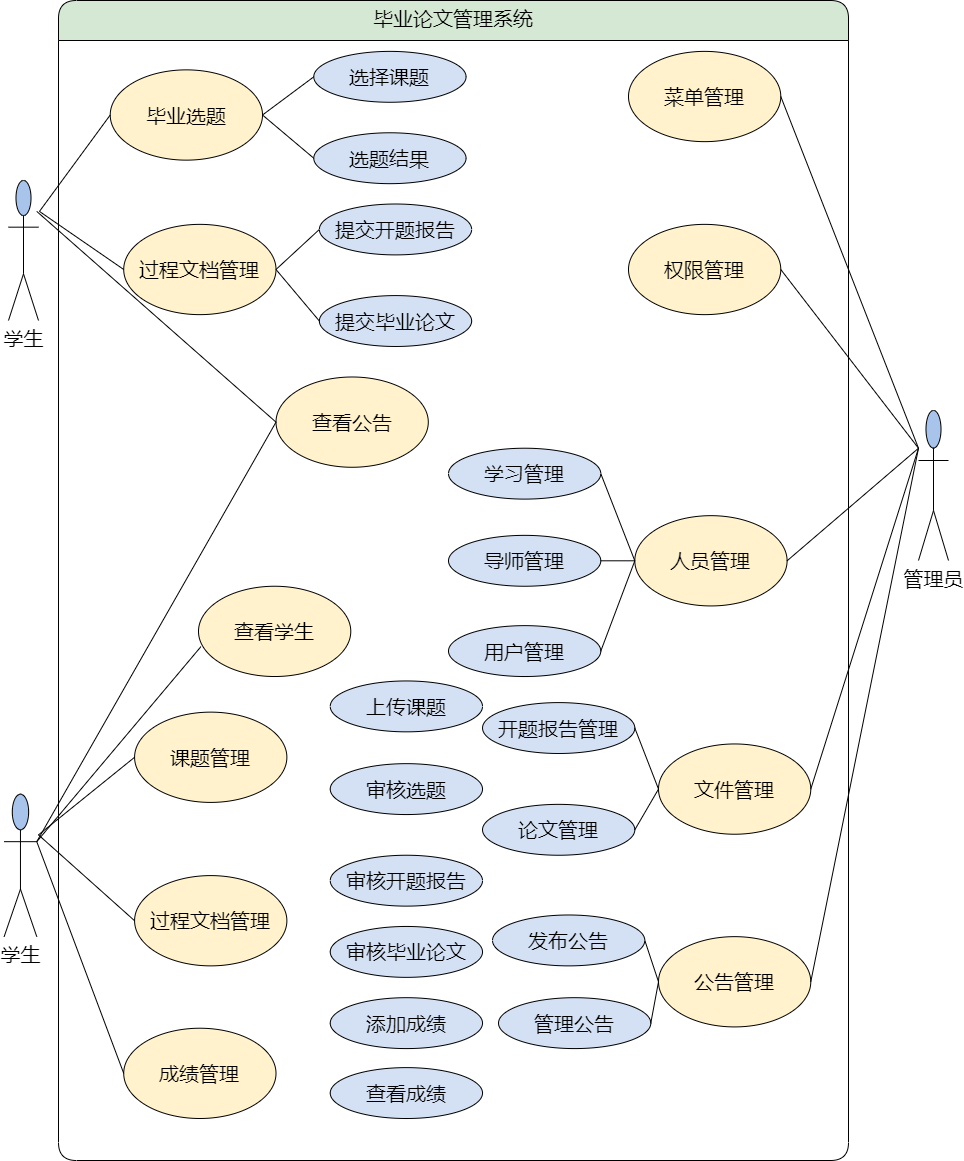
整个系统需要进行

* + 1. 安全性需求

建立合理的用户登录注册体系，完善的权限管理，对访问后台各个页面数据的请求药进行权限认证，保护数据库数据安全

* + 1. 用例图

用例图，是用户和与之相关的用例的来表示的用户与系统交互的最简形式，开发者可以与其他图表文档配合，以更高的层次观察整体系统的功能与细节，也是开发者相互交流的好形式

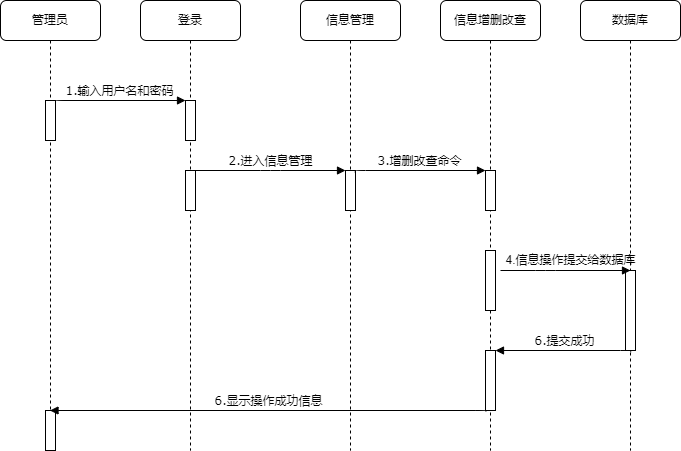


* + 1. 时序图

时序图（Sequence Diagram）是显示对象之间交互的图，这些对象是按时间顺序排列的。时序图中显示的是参与交互的对象及其对象之间消息交互的顺序。时序图包括的建模元素主要有：对象（Actor）、生命线（Lifeline）、控制焦点（Focus of control）、消息（Message）等等。通过建立时序图，可以帮助程序员更好的理解对象直接的交互关系、交互顺序。

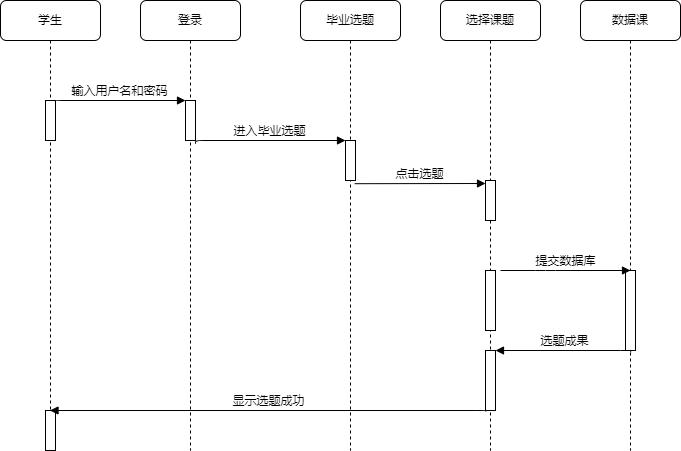
1. 用户、学生、导师管理操作

此项为管理员对用户、学生、导师管理操作，管理员首先要在系统登录，登录进系统后进入学生管理、导师管理、用户管理页面，通过增加、删除、修改、查询对用户、学生、导师信息进行操作，然后将信息操作提交到数据库。时序图如图



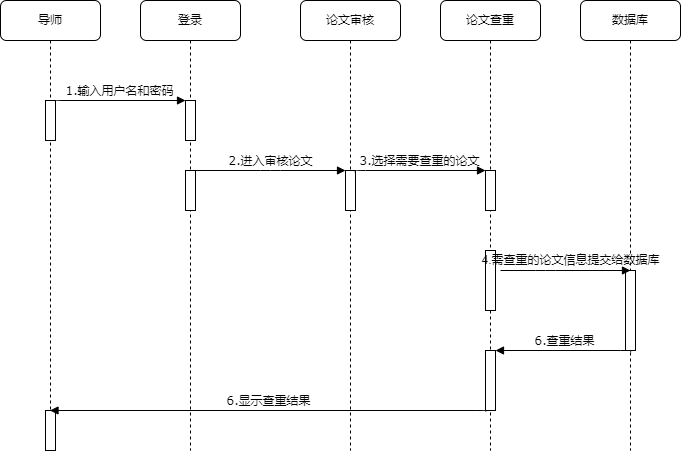
1. 学生选择课题流程操作

学生需要输入用户名和密码登录系统后，进入毕业选题页面，在毕业选题页面会显示当前导师发布的课题内容，点击选题操作后，系统会将选题信息提交到数据库、时序图如图



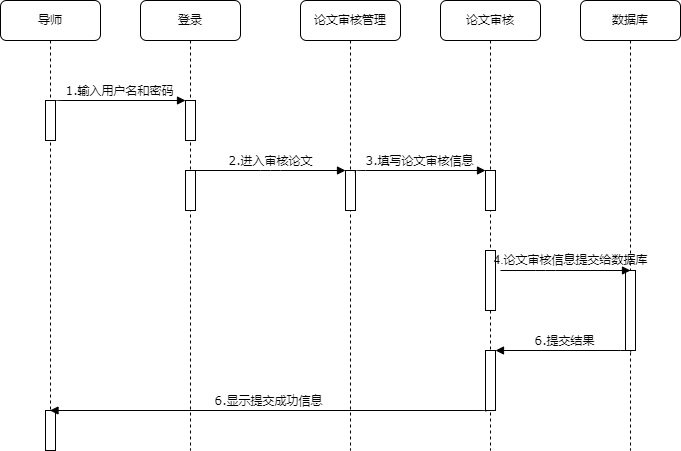
1. 导师查重论文

导师需要输入用户名和密码登录系统后，进入论文审核页面，在论文审核页面会显示当前学生已提交的论文的信息，点击所需查重的论文，提交到数据库进行比较。时序如图



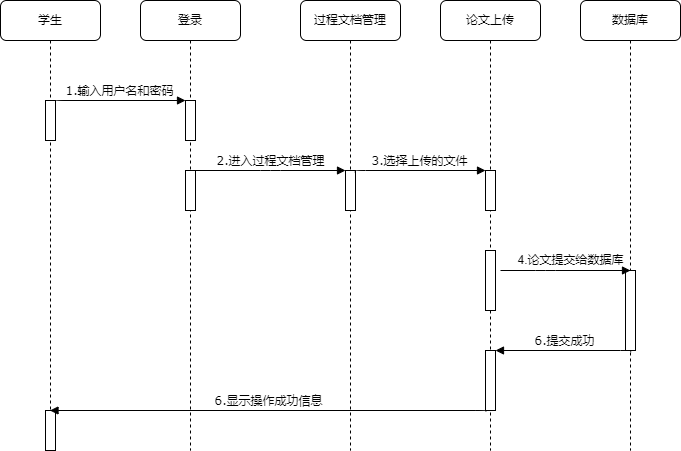
1. 导师审核论文

导师需要输入用户名和密码登录系统后，进入论文审核页面，在论文审核页面会显示当前学生已提交的论文的信息，在论文查重通过后点击审核会弹出审核页面、输入相应的审核信息提交到数据库，时序图如图



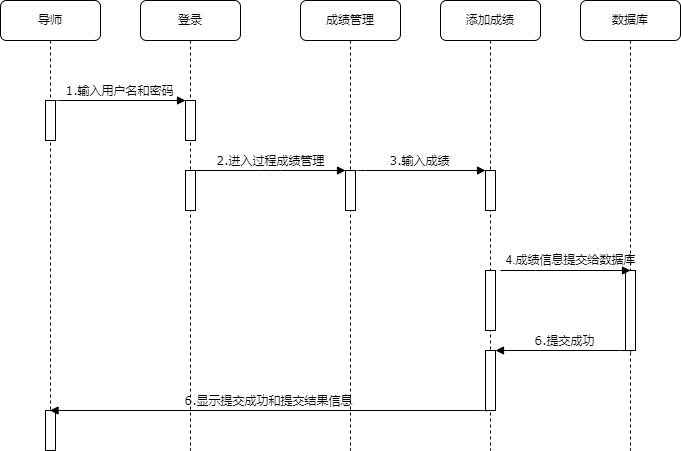
1. 学生提交论文

学生需要输入用户名和密码登录系统后，进入过程文档管理页面，选择将要上传的论文，然后回提交到数据库，数据库返回提交成功，并且将当前论文的信息显示在过程文档管理页面上，时序图如图



1. 导师添加成绩

导师需要输入用户名和密码登录系统后，进入成绩管理页面，选择学生，填写成绩，然后回提交到数据库，数据库返回提交成功，时序图如图

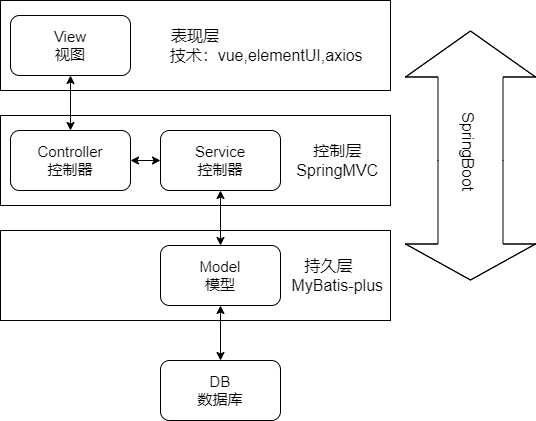


# 系统概要设计

* 1. 系统框架图

SpringMVC框架是基于Servlet的MVC模型，主要解决WEB开发的问题，MVC 模式代表 Model-View-Controller（模型-视图-控制器） 模式。这种模式用于应用程序的分层开发。

* Model（模型） - 模型代表一个存取数据的对象或 JAVA POJO。它也可以带有逻辑，在数据变化时更新控制器，实现数据持久化
* View（视图） - 视图代表模型包含的数据的可视化，在视图层显示管理员、用户、导师的前端页面
* Controller（控制器） - 控制器作用于模型和视图之上，接收用户请求，委托给模型进行处理，控制数据流向模型对象，把返回的模型数据给视图

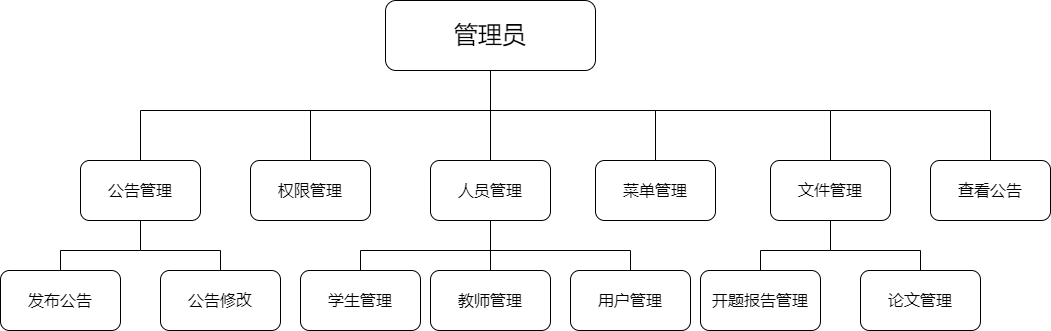


* 1. 系统功能模块图

根据需求分析，毕业论文管理系统总体分为管理员、学生、导师三个用户，三个用户分别对应了三个功能模块

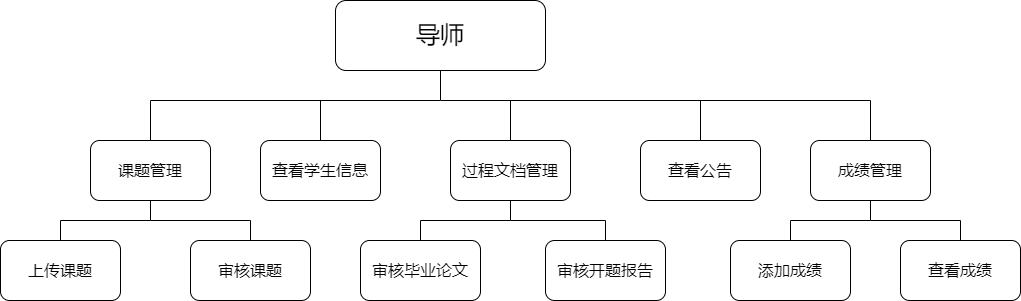
* + 1. 管理员模块

管理员模块主要对系统总体进行管理，包括权限管理：分配角色的菜单信息；员管理：对学生、教师、进行删改查操作，对用户进行增删改查操作；文件管理：对开题报告、论文进行删改查操作；公告管理包括发布公告和修改公告操作，模块图如图



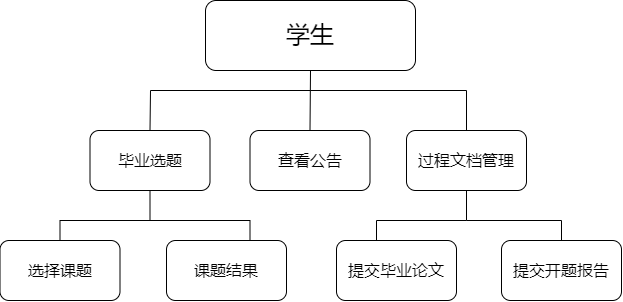
* + 1. 导师模块

导师模块包括课题管理：上传课题、审核课题；过程文档管理：审核开题报告、审核毕业论文，其中审核毕业论文要进行校内查重；成绩管理：选择学生添加成绩，在查看成绩显示学生成绩信息，也可以修改学生成绩；查看学生信息；查看公告



* + 1. 学生模块

学生模块包括毕业选题：选择课题、课题结果，如果已选择课题，则不能再选择，再课题结果模块可以退选课题；过程文档管理：提交毕业论文、提交开题报告，在提交文档页面显示当前最新提交文档的信息和文档审核的信息；查看公告

、

* 1. 类模块设计
     1. 用户实现类

UserServiceImpl类继承IUserService接口，其中login(UserDTO userDTO)方法实现登录功能；register(UserDTO userDTO)方法实现注册功能；updatePassword()方法实现更新密码功能；setStudentTeacher（StudentTeacher studentTeacher）实现设置学生老师表；getUserInfo(UserDTO userDTO)方法实现获取当前用户功能，getRoleMenus(String roleFlag)方法获取当前角色的菜单列表。

* + 1. 导师实现类

TeacherServiceImpl类继承ITeacherService接口，其中getTeaPage(Page<Teacher> page, String name, String tno)实现分页获取导师信息，同时可以实现模糊查询， checkPaper(String simHash, String stuNum)方法实现导师论文查重功能

* + 1. 学生实现类

StudentServiceImpl类继承IStudentService接口，其中getStuPage(Page<Student> page, String name, String sno) 实现分页获取学生信息，同时可以实现模糊查询，GetStuFileInfo(String teaNum, String fileType)实现根据导师获取指导的学生信息，

* + 1. 课题实现类

TitleServiceImpl类继承ITitleService接口，其中pageTitle(Page<Object> Page)实现分页获取课题信息，getResult(String stuNum, String teaNum)方法获取学生课题成绩，

Withdrawal(Integer id, String stuNum, Integer flag)方法实现改选课题

* + 1. 角色实现类

RoleServiceImpl类继承IRoleService接口，其中setRoleMenu(Integer roleId, List<Integer> menuIds)实现设置角色菜单功能；getRoleMenu(Integer roleId)方法获取当前角色菜单

* + 1. 菜单实现类

MenuServiceImpl类继承IMenuService接口，其中findMenus(String name)方法获取当前数据库所以菜单，并按照一级二级菜单形式进行存储。

* + 1. 论文实现类

PaperServiceImpl类继承IPaperService接口，其中showPaperInfo(String stuNum)方法获取学生上传的论文信息；uploadPaper(MultipartFile file, String stuNum)方法实现论文上传

# 数据库设计

* 1. 数据字典

1. 用户表（sys\_user）

|  |  |  |  |
| --- | --- | --- | --- |
| 数据项目名称 | 数据类型 | 长度(kb) | 描述 |
| id | int | 0 | id |
| username | varchar | 50 | 用户名 |
| password | varchar | 50 | 密码 |
| create\_time | timestamp | 0 | 创建时间 |
| role | varchar | 50 | 角色 |
| st\_num | varchar | 10 | 绑定的学号或者工号 |

1. 导师表（sys\_teacher）

|  |  |  |  |
| --- | --- | --- | --- |
| 数据项目名称 | 数据类型 | 长度(kb) | 描述 |
| id | int | 0 | 导师id |
| tno | varchar | 30 | 导师工号 |
| name | varchar | 30 | 导师姓名 |
| sex | varchar | 2 | 导师性别 |
| dept | varchar | 30 | 导所在院系 |
| title | varchar | 10 | 导师职称 |
| phone | varchar | 30 | 导师联系电话 |
| email | varchar | 30 | 导师邮箱 |

1. 学生表（sys\_student）

|  |  |  |  |
| --- | --- | --- | --- |
| 数据项目名称 | 数据类型 | 长度(kb) | 描述 |
| id | int | 0 | 学生id |
| sno | varchar | 20 | 学生学号 |
| name | varchar | 30 | 学生姓名 |
| sex | varchar | 3 | 学生性别 |
| dept | varchar | 20 | 学生所在院系 |
| phone | varchar | 20 | 学生联系电话 |
| email | varchar | 30 | 学生邮箱 |
| is\_select | tinyint | 1 | 是否已选课 |

1. 课题表（sys\_titile）

|  |  |  |  |
| --- | --- | --- | --- |
| 数据项目名称 | 数据类型 | 长度(kb) | 描述 |
| id | int | 0 | 课题id |
| name | varchar | 30 | 课题名称 |
| difficulty | varchar | 10 | 课题难度 |
| type | varchar | 10 | 课题类型 |
| source | varchar | 20 | 课题来源 |
| brief | varchar | 1000 | 课题简介 |
| stu\_num | varchar | 20 | 课题学生id |
| tea\_num | varchar | 20 | 课题指导老师id |
| is\_verify | tinyint | 1 | 是否审核通过 |
| score | float | 0 | 最终成绩 |

1. 审核表（sys\_verify）

|  |  |  |  |
| --- | --- | --- | --- |
| 数据项目名称 | 数据类型 | 长度(kb) | 描述 |
| id | int | 0 | 审核表id |
| tea\_num | varchar | 30 | 审核老师工号 |
| time | datetime | 0 | 审核时间 |
| state | tinyint | 1 | 审核状态 |
| comment | varchar | 500 | 审核意见 |
| file\_id | int | 0 | 文件id |

1. 论文表（sys\_paper）

|  |  |  |  |
| --- | --- | --- | --- |
| 数据项目名称 | 数据类型 | 长度(kb) | 描述 |
| id | int | 0 | 文件id |
| name | varchar | 30 | 文件名称 |
| size | bigint | 0 | 论文大小 |
| url | varchar | 255 | 下载链接 |
| md5 | varchar | 255 | 文件MD5 |
| stu\_num | varchar | 20 | 学生学号 |
| create\_time | datetime | 0 | 上传时间 |
| sim\_hash | varchar | 255 | sim哈希值 |

* 1. 概念结构设计

1. 表信息：

用户(id，用户名，密码，创建时间，角色，绑定的学号或者工号)

导师(id，工号，姓名，性别，所在院系，职称，联系电话，邮箱)

学生(id, 学号，姓名，性别，所在院系，联系电话，邮箱，是否已选课)

课题 (id，名称，难度，类型，来源，简介，学生学号，指导老师工号，是否审核通过，最终成绩)

审核(id，审核老师工号，审核时间，状态，审核意见，文件id)

论文(id, 文件名称, 论文大小, 下载链接, 文件MD5, 学生学号, 上传时间, sim哈希值)

角色(id, 名称, 描述, 唯一标识)

公告(id, 标题, 发布时间, 公告内容)

菜单(id, 名称, 路径, 图标, 描述, 父级id, 页面路径, 排序)

2. 表关系

导师与学生是一对多的关系

课题和学生是一对一关系

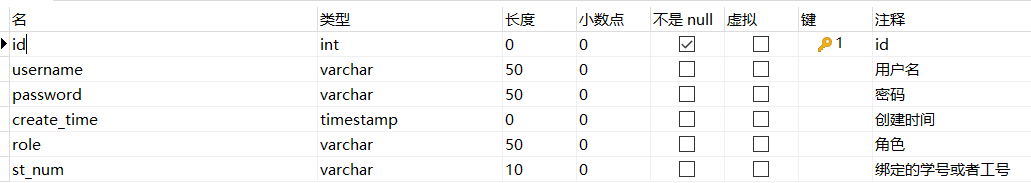
学生与论文是一对多关系

用户与学生是一对一关系

用户与导师是一对一关系

* 1. 逻辑结构设计

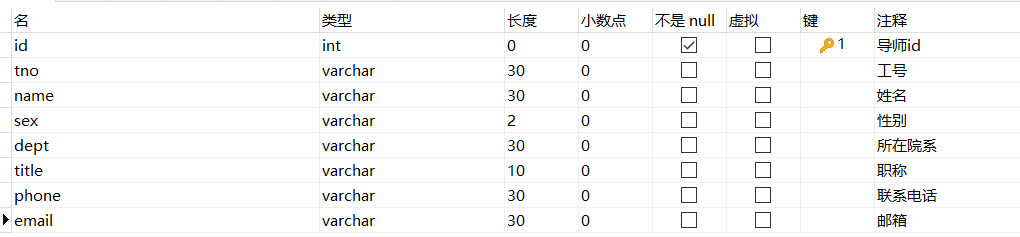
1. sys\_user



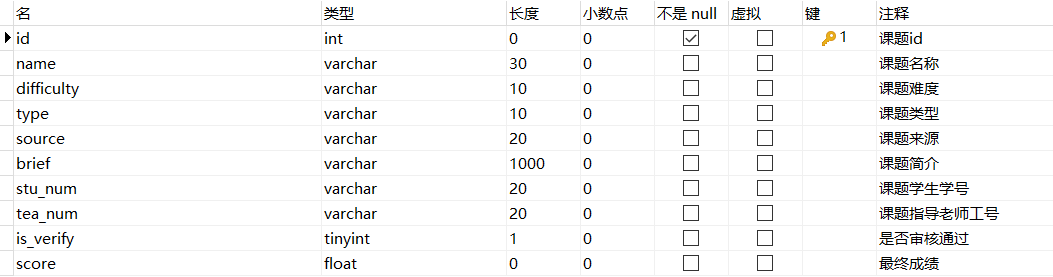
1. sys\_student



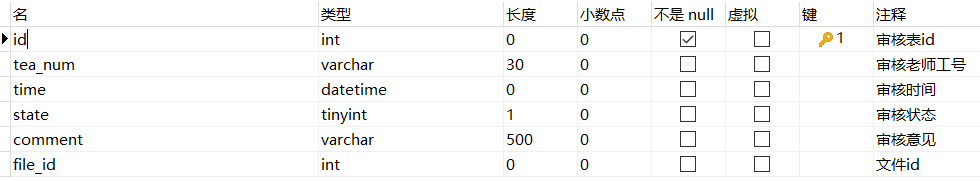
1. sys\_teacher



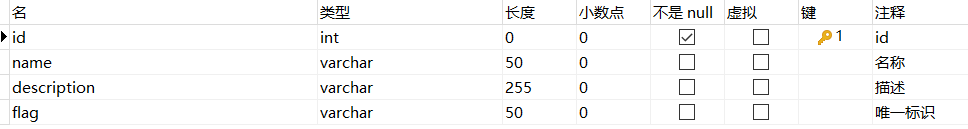
1. sys\_title



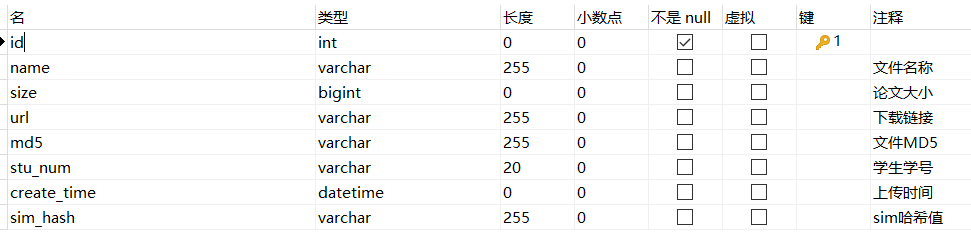
1. sys\_verify



1. sys\_role



1. sys\_paper



1. sys\_proposal



1. sys\_news

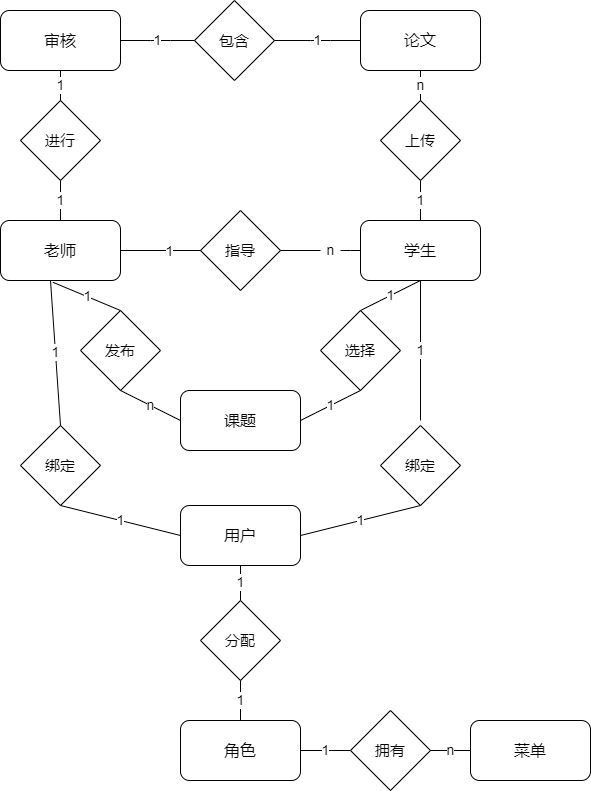


1. sys\_menu

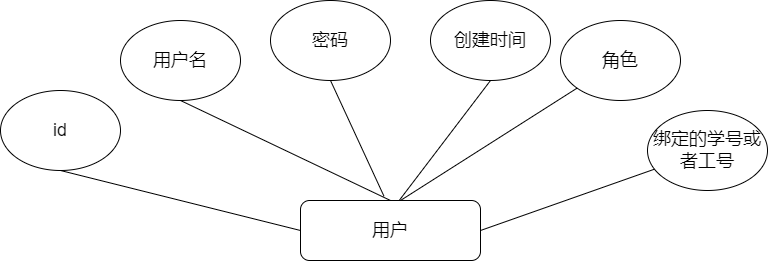


* 1. 物理结构设计

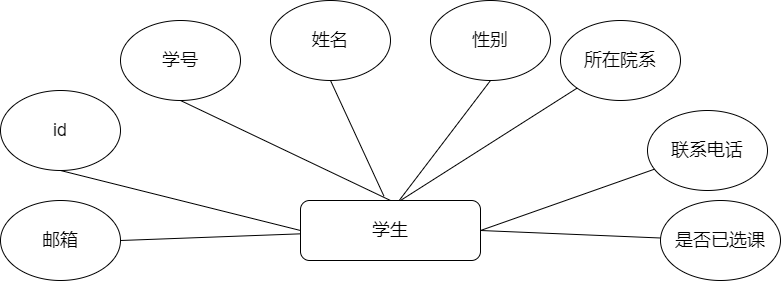
为了描述现实世界的概念模型，需要将设计的表标识成实体属性图，并分析各实体之间的关系，即ER图，总体ER图如下图所示。



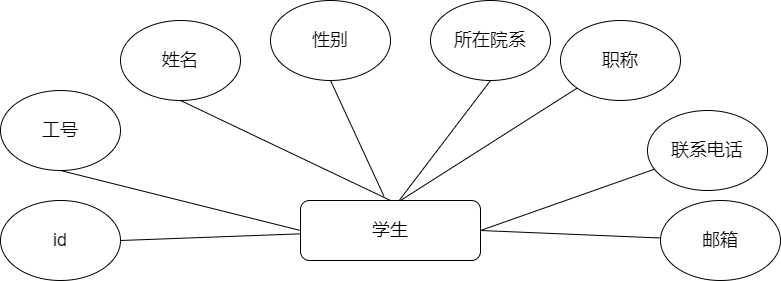
1. 用户E-R图



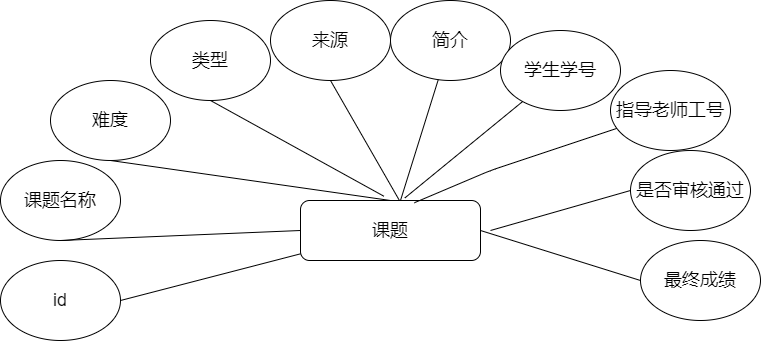
1. 学生E-R图



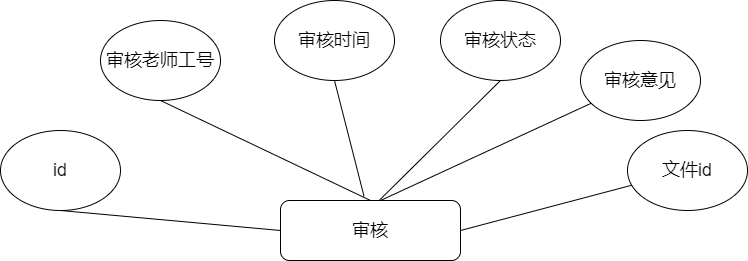
1. 导师E-R图



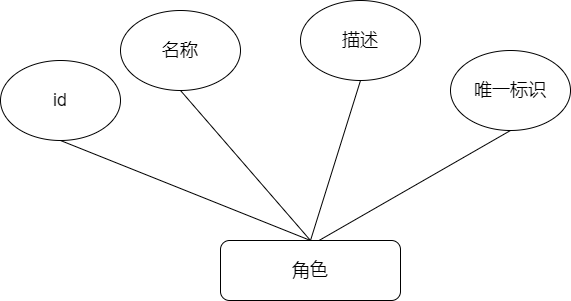
1. 课题E-R图



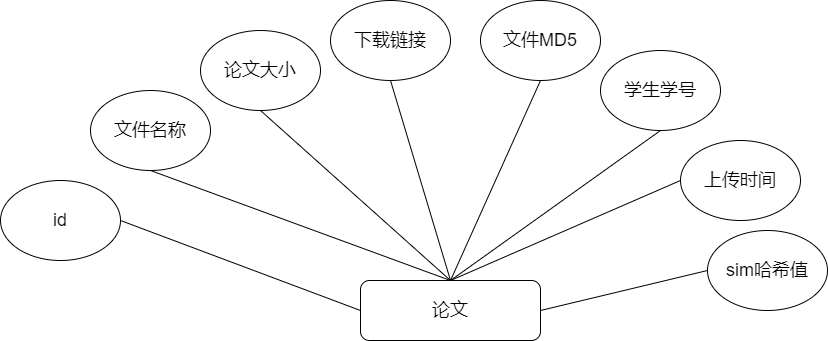
1. 审核E-R图



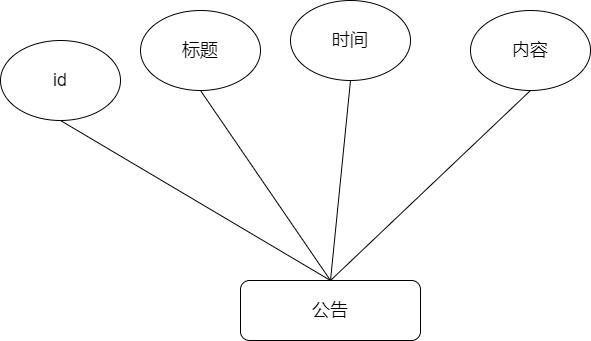
1. 角色E-R图



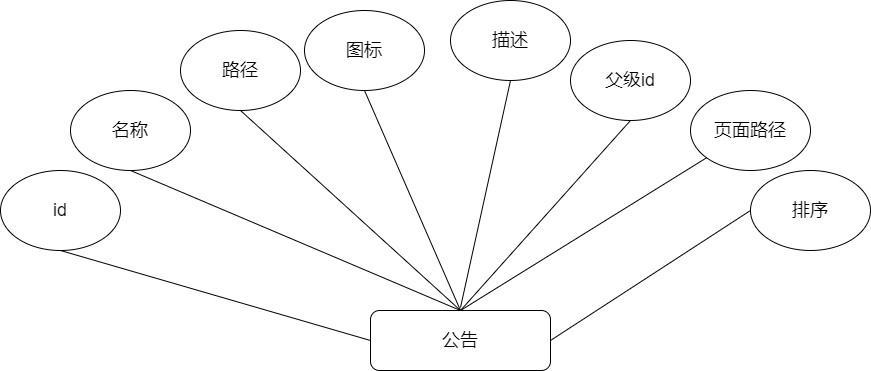
1. 论文E-R图



1. 公告E-R图



1. 菜单E-R图



# 系统创新

为了提高毕业论文质量水平，增强高等院校对论文质量的把控能力。本系统设计了论文查重功能来辅助导师鉴别论文中的抄袭作弊内容。论文查重也称为文本复制检测，就是利用计算机技术判断两篇文本的相似度，并给出一个量化的数值。相似度越高表明两个文本的内容雷同较多[3]

* 1. Simhash算法

文本相似度主要采用simhas算法、分词算法并结合海明距离计算进行实现。simhas算法的主要思想是降维，将高维的特征向量映射成一个f-bit的指纹，通过比较两个文本f-bit指纹的海明距离来确定内容是否重复并计算出两者的相似度值[4]。

* + 1. 计算simhash值

1. 分词

给定一段语句，进行分词，得到有效的特征向量，然后为每一个特征向量设置1-5等5个级别的权重，如果是给定一个文本，那么特征向量可以是文本中的词，其权重可以是这个词出现的次数。

1. 计算hash值

通过hash函数计算各个特征向量的hash值，hash值为二进制数01组成的n-bit签名。

1. 加权

在hash值的基础上，给所有特征向量进行加权，即W = Hash \* weight，且遇到1则hash值和权值正相乘，遇到0则hash值和权值负相乘。

1. 合并

将上述各个特征向量的加权结果累加，变成只有一个序列串。

1. 降维

对于n-bit签名的累加结果，如果大于0则置1，否则置0，从而得到该语句的simhash值，最后我们便可以根据不同语句simhash的海明距离来判断它们的相似度。

* + 1. 计算海明距离

每篇文档得到simhash值后，接着计算两个签名的海明距离即可。根据经验值，对64位的 simhash值，海明距离在3以内的可认为相似度比较高。海明距离的求法：异或时，只有在两个比较的位不同时其结果是1，否则结果为0，两个二进制“异或”后得到1的个数即为海明距离的大小。

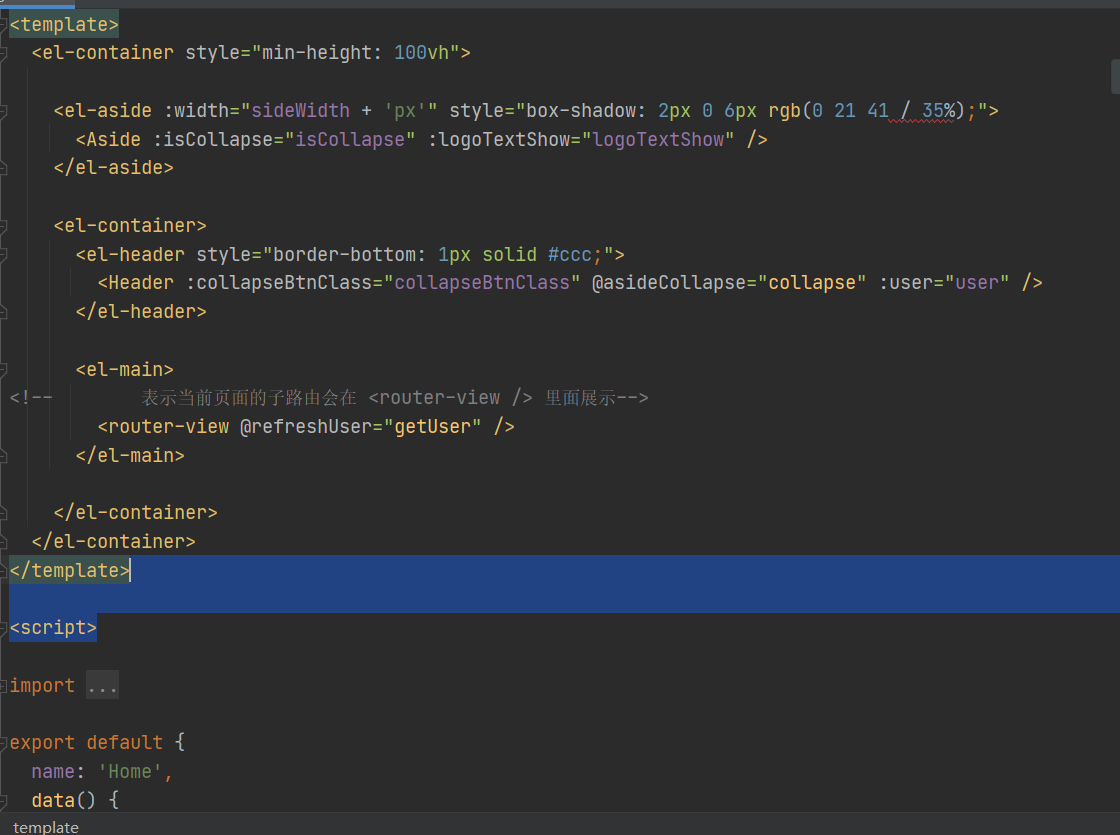
# 系统详细设计

* 1. 系统前端设计与实现

【vue原理介绍好处】

* + 1. Vue路由功能实现

1. 通过路由设定访问路径，将路径和组件映射起来。以manage.vue（如图）为管理系统主体框架，将传统的超链接实现整体页面切换跳转转变成组件切换



1. 根据用户的角色权限获取的菜单数据，对组件Mange.vue实现路由动态拼接，以实现用户权限管理控制，核心代码如下：

export const setRoutes = () => {

const storeMenus = localStorage.getItem("menus");

if (storeMenus) {

const currentRouteNames = router.getRoutes().map(v => v.name)

if (!currentRouteNames.includes('Manage')) {

const manageRoute = { path: '/', name: 'Manage', component: () => import('../views/Manage.vue'), redirect: "/home", children: [

{ path: 'studentEdit', name: '个人信息', component: () => import('../views/StudentEdit.vue')},

{ path: 'teacherEdit', name: '个人信息', component: () => import('../views/TeacherEdit.vue')},

{ path: 'password', name: '修改密码', component: () => import('../views/Password.vue')}

] }

const menus = JSON.parse(storeMenus)

menus.forEach(item => {

if (item.path) { // 当且仅当path不为空的时候才去设置路由

let itemMenu = { path: item.path.replace("/", ""), name: item.name, component: () => import('../views/' + item.pagePath + '.vue')}

manageRoute.children.push(itemMenu)

} else if(item.children.length) {

item.children.forEach(item => {

if (item.path) {

let itemMenu = { path: item.path.replace("/", ""), name: item.name, component: () => import('../views/' + item.pagePath + '.vue')}

manageRoute.children.push(itemMenu)

}

})

}

})

router.addRoute(manageRoute)

}

* + 1. axios实现ajax请求

使用axios将request请求封装到request.js文件中，并且设置好request 拦截器,对请求发送前进行统一的数据处理，比如统一加上token。设置好response 拦截器，对服务器返回的数据作统一处理，核心代码如下：

request.interceptors.request.use(config => {

config.headers['Content-Type'] = 'application/json;charset=utf-8';

let user = localStorage.getItem("user") ? JSON.parse(localStorage.getItem("user")) : null

return config

}, error => {

return Promise.reject(error)

});

request.interceptors.response.use(

response => {

let res = response.data;

if (response.config.responseType === 'blob') {

return res

}

if (typeof res === 'string') {

res = res ? JSON.parse(res) : res

}

if (res.code === '401')

return res;

},

error => {

console.log('err' + error) // for debug

return Promise.reject(error)

}

* + 1. 前端表单验证功能实现

基于element-ui使用Form 组件rules 属性传入约定的验证规则，并将 Form-Item 的 prop 属性设置为需校验的字段名即可，核心代码如下;

Form表单：

<el-form :model="user" :rules="rules" ref="userForm">

<el-form-item prop="username">

<el-input size="medium" style="margin: 10px 0" prefix-icon="el-icon-user" v-model="user.username"></el-input>

…..

</el-form>

验证rules：

rules: {

username: [

{required: true, message: '请输入用户名', trigger: 'blur'},

{min: 3, max: 10, message: '长度在 3 到 5 个字符', trigger: 'blur'}

],

password: [

{required: true, message: '请输入密码', trigger: 'blur'},

{min: 1, max: 20, message: '长度在 1 到 20 个字符', trigger: 'blur'}

],

}

* 1. 系统主要后端设计与实现
     1. 管理员登录功能实现

1. 用户在前端页面输入账号密码传递到后端，UserServiceImpl将前端传递来的信息与数据库的信息对比，若数据库存在，则生成token，同时将该用户的菜单信息传递到前端。登录功能代码实现：

public UserDTO login(UserDTO userDTO) {

User one = getUserInfo(userDTO);

if (one != null) {

BeanUtil.copyProperties(one, userDTO, true);

String token = TokenUtils.genToken(one.getId().toString(), one.getPassword());

userDTO.setToken(token);

String role = one.getRole(); // ROLE\_ADMI

List<Menu> roleMenus = getRoleMenus(role);

userDTO.setMenus(roleMenus);

return userDTO;

} else {

throw new ServiceException(Constants.CODE\_600, "用户名或密码错误");

}

}

1. 使用JWT实现用户信息认证

在登录的时候使用用户id和密码生成token，传递到前端存储，后端不保存，前端每次请求服务都要带上token,后端开启JWT拦截器，当为请求拦截资源使，解析请求中的token，查询数据库是否有符合用户数据

生成Token的代码：

public static String genToken(String userId, String sign) {

return JWT.create().withAudience(userId)

.withExpiresAt(DateUtil.offsetHour(new Date(), 2))

.sign(Algorithm.HMAC256(sign));

}

JWT拦截器代码：

public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) {

String token = request.getHeader("token");

if(!(handler instanceof HandlerMethod)){

return true;

} else {

HandlerMethod h = (HandlerMethod) handler;

AuthAccess authAccess = h.getMethodAnnotation(AuthAccess.class);

if (authAccess != null) {

return true;

}

} if (StrUtil.isBlank(token)) {

throw new ServiceException(Constants.CODE\_401, "无token，请重新登录");

} String userId;

try {

userId = JWT.decode(token).getAudience().get(0);

} catch (JWTDecodeException j) {

throw new ServiceException(Constants.CODE\_401, "token验证失败，请重新登录");

} User user = userService.getById(userId);

if (user == null) {

throw new ServiceException(Constants.CODE\_401, "用户不存在，请重新登录");

} JWTVerifier jwtVerifier = JWT.require(Algorithm.HMAC256(user.getPassword())).build();

try {

jwtVerifier.verify(token); // 验证token

} catch (JWTVerificationException e) {

throw new ServiceException(Constants.CODE\_401, "token验证失败，请重新登录");

}

return true;

}

* + 1. 管理员注册功能实现

前端发起两次post请求，第一次调用UserServiceImpl中register(UserDTO userDTO)方法，将用户名和密码存储到数据库中，第二次调用TeacherController或者StudentController中save（）方法，将个人信息保存在数据库相应的表中

注册代码：

public User register(UserDTO userDTO) {

User one = getUserInfo(userDTO);

if (one == null) {

one = new User();

BeanUtil.copyProperties(userDTO, one, true);

// 默认一个普通用户的角色

// one.setRole(RoleEnum.ROLE\_USER.toString());

save(one); // 把 copy完之后的用户对象存储到数据库

} else {

throw new ServiceException(Constants.CODE\_600, "用户已存在");

}

return one;

}

* + 1. 管理员人员管理功能实现

1. 学生管理功能

管理员删除学生信息，在后端StudentController类中调用delete方法;修改学生信息，在后端StudentController类中调用save方法;模糊查询和分页查询查找，需要经StudentController，StudentServiceImpl调用StudentMapper中getStuPage方法，最终通过StudentMapper中的sql语句向数据库查询数据; 设置学生的指导老师，调用的是UserServiceImpl中setStudentTeacher方法向学生设置指导老师

模糊查询和分页查询查找sql语句：

<select id="getStuPage" resultMap="pageStudent">

select stu.\* , tea.name as tea\_name

FROM sys\_student stu

LEFT JOIN sys\_student\_teacher st

ON st.stu\_num = stu.sno

LEFT JOIN sys\_teacher tea

ON st.tea\_num = tea.tno

<where>

<if test="name != null and name != ''">

and stu.name like concat('%', #{name} ,'%')

</if>

<if test="sno != null and sno != ''">

and stu.sno like concat('%', #{sno} ,'%')

</if>

</where>

</select>

设置学生的指导老师代码:

public void setStudentTeacher(StudentTeacher studentTeacher) {

int stu\_num = studentTeacher.getStu\_num();

int tea\_num = studentTeacher.getTea\_num();

if(null == stuTeaMapper.getTeaNum(stu\_num)){

stuTeaMapper.addStuTea(stu\_num,tea\_num);

}else {

if(stuTeaMapper.getTeaNum(stu\_num) != tea\_num)

stuTeaMapper.update(stu\_num,tea\_num);

}

}

1. 导师管理功能

管理员删除导师信息，在后端TeacherController类中调用delete方法;修改导师信息，在后端TeacherController类中调用save方法;模糊查询和分页查询查找，需要经TeacherController，TeacherServiceImpl调用StudentMapper中getTeaPage方法，最终通过TeacherMapper中的sql语句向数据库查询数据;

模糊查询和分页查询查找sql语句：

<select id="getTeaPage" resultMap="pageStudent">

select tea.\* , stu.sno as sno, stu.name as stu\_name

FROM sys\_teacher tea

LEFT JOIN sys\_student\_teacher st

ON st.tea\_num = tea.tno

LEFT JOIN sys\_student stu

ON stu.sno = st.stu\_num

<where>

<if test="name != null and name != ''">

and tea.name like concat('%', #{name} ,'%')

</if>

<if test="tno != null and tno != ''">

and tea.tno like concat('%', #{tno} ,'%')

</if>

</where>

</select>

1. 用户管理功能

管理员增加用户可分别按照三个角色添加：管理员、学生、导师，代码逻辑和注册功能一样，删除用户信息，在后端UserController类中调用delete方法;修改导师信息，在后端UserController类中调用save方法;模糊查询和分页查询查找，在后端UserController类中调用findPage方法

findPage方法代码：

@GetMapping("/page")

public Result findPage(@RequestParam Integer pageNum,

@RequestParam Integer pageSize,

@RequestParam(defaultValue = "") String username,

@RequestParam(defaultValue = "") String stNum) {

IPage<User> page = new Page<>(pageNum,pageSize);

QueryWrapper<User> queryWrapper = new QueryWrapper<>();

if(!username.equals("") && username != null){

queryWrapper.eq("username", username);

}

if(!stNum.equals("") && stNum != null){

queryWrapper.eq("st\_num", stNum);

}

return Result.success(userService.page(page,queryWrapper));

}

* + 1. 管理员权限功能实现

权限管理页面的增删改查与上述管理页面逻辑类似。管理员对角色菜单重新设置，在前端，将roleId，与menuId传递到后端RoleServiceImpl类中，然后先删除当前角色id所有的绑定关系，再把前端传过来的菜单id数组绑定到当前的这个角色id上去，由于菜单中存在父级菜单和二级菜单，则需要判断菜单是否为二级菜单，若传过来的menuId数组里面没有它的父级id，则需要补充这个父级id。

RoleServiceImpl类代码如下：

roleMenuMapper.deleteByRoleId(roleId);

List<Integer> menuIdsCopy = CollUtil.newArrayList(menuIds);

for (Integer menuId : menuIds) {

Menu menu = menuService.getById(menuId);

if (menu.getPid() != null && !menuIdsCopy.contains(menu.getPid())) {

RoleMenu roleMenu = new RoleMenu();

roleMenu.setRoleId(roleId);

roleMenu.setMenuId(menu.getPid());

roleMenuMapper.insert(roleMenu);

menuIdsCopy.add(menu.getPid());

}

RoleMenu roleMenu = new RoleMenu();

roleMenu.setRoleId(roleId);

roleMenu.setMenuId(menuId);

roleMenuMapper.insert(roleMenu);

* + 1. 学生选择课题功能实现

1. 选择课题功能

学生选择课题提交后，将学生的学号和所选课题的id提交到后端TitleServiceImpl类中，先由titleMapper中updateById方法更新数据库信息，再由studentMapper中setIsSelect方法更新学生表中is\_select字段信息

TitleServiceImpl中代码：

if(getTitleByStuNum(title.getStuNum()) == null){

titleMapper.updateById(title);

studentMapper.setIsSelect(title.getStuNum(),title.getFlag());

return Result.success();

}else return Result.error(Constants.CODE\_400,"选题失败！只能选择一个题目,如需改选,请先退选其它毕业题目!");

studentMapper中代码：

@Update("update sys\_student set is\_select = #{flag} where sno = #{stuNum}")

void setIsSelect(@Param("stuNum") String stuNum, @Param("flag") Integer flag);

1. 课题结果功能

课题结果页面首先将后台的课题信息显示再前端，需要在TitleMapper中对teacher表和student表进行联表查询。课题退选功能是将要退选的课题id和学生学号传递到TitleServiceImpl，然后在TitleServiceImpl类中由studentMapper设置学生表中中is\_select字段信息，再由titleMapper中WithdrawalTitle更改课题表学生学号字段信息。

TitleMapper中联表查询代码：

<select id="getResult" resultMap="titleResult">

select title.\* , tea.name as tea\_name, stu.name as stu\_name

FROM sys\_title title

LEFT JOIN sys\_teacher tea

ON tea.tno = title.tea\_num

LEFT JOIN sys\_student stu

ON stu.sno = title.stu\_num

<where>

<if test="stuNum != null and stuNum != ''">

title.stu\_num = #{stuNum}

</if>

<if test="teaNum != null and teaNum != ''">

title.tea\_num = #{teaNum}

and title.stu\_num != ''

</if>

</where>

</select>

TitleServiceImpl中退选部分代码：

public void Withdrawal(Integer id, String stuNum, Integer flag) {

studentMapper.setIsSelect(stuNum,flag);

titleMapper.WithdrawalTitle(id);

}

* + 1. 学生过程文档管理功能实现

1. 文件上传功能

学生上传文件要将文档信息和当前学生学号传递到后端的PaperServiceImpl类中

首先先判断学生是否选择课题，若有则再判断课题是否通过审核，若有则进入上传文档逻辑:首先初始化文件唯一的标识码，然后判断配置的文件目录是否存在，若不存在则创建一个新的文件目录，生成文件的md5，查询数据库是否存在相同的记录，若无将文件上传到磁盘，并生成新的url作为下载的链接，最后将上述的信息存储至数据库。

PaperServiceImpl上传部分代码:

QueryWrapper<Title> queryWrapper = new QueryWrapper<>();

queryWrapper.eq("stu\_num", stuNum);

Title title = titleMapper.selectOne(queryWrapper);

if(title != null){

if(title.getIsVerify() == true){

String originalFilename = file.getOriginalFilename();

String type = FileUtil.extName(originalFilename);

long size = file.getSize();

String fileUUID = IdUtil.fastSimpleUUID() + StrUtil.DOT + type;

String fileTotalPath = fileUploadPath + fileUUID;

File uploadFile = new File(fileTotalPath);

File parentFile = uploadFile.getParentFile();

if(!parentFile.exists()) {

parentFile.mkdirs();

}

String url;

String md5 = SecureUtil.md5(file.getInputStream());

Paper dbPaper = getFileByMd5(md5);

if (dbPaper != null) {

url = dbPaper.getUrl();

} else {

file.transferTo(uploadFile);

url = "http://" + serverIp + ":9090/proposal/" + fileUUID;

}

String simHash = getSimHash(fileTotalPath);

Paper savePaper = new Paper();

savePaper.setName(originalFilename);

savePaper.setSize(size/1024); // 单位 kb

savePaper.setUrl(url);

savePaper.setMd5(md5);

savePaper.setStuNum(stuNum);

savePaper.setSimHash(simHash);

paperMapper.insert(savePaper);

return Result.success(url);

}else return Result.error(Constants.CODE\_400,"课题审核未通过");

}else return Result.error(Constants.CODE\_400,"请先选择课题");

1. 显示最新论文信息和审核信息

论文信息是通过PaperServiceImpl类中showPaperInfo方法查询，先根据前端传来的学生学号查询数据库中所有的论文，再根据上传时间排序，提取时间最新的一条新整理发送到前端，前端再根据最新的论文id为参数发送get请求请求审核信息

PaperServiceImpl类相关代码：

QueryWrapper<Paper> QWPaper = new QueryWrapper<>();

QWPaper.eq("stu\_num", stuNum);

QWPaper.orderByDesc("create\_time");

if (paperMapper.selectList(QWPaper).size() != 0){

Paper paper = paperMapper.selectList(QWPaper).get(0)

QueryWrapper<Title> QWtitle = new QueryWrapper<>();

QWtitle.eq("stu\_num", stuNum);

String titleName = titleMapper.selectOne(QWtitle).getName();

if(paper != null){

paper.setTitleName(titleName);

}

return paper;

* + 1. 导师过程文档管理功能实现

1. 显示文档信息功能

显示文档信息功能先将改导师的导师工号作为参数,进行get请求，后端在TeacherServiceImpl类，对请求文档类型进行判断，最后在mapper中对数据库进行查询操作。

TeacherMapper中相关代码：

<select id="GetStuPaperInfo" resultMap="StuPaper">

select st.stu\_num as sno , stu.`name` as stu\_name , pap.\* , v.state

FROM sys\_student\_teacher st

LEFT JOIN sys\_paper pap

ON pap.stu\_num = st.stu\_num

LEFT JOIN sys\_student stu

ON stu.sno = pap.stu\_num

LEFT JOIN sys\_verify v

ON v.file\_id = pap.id

WHERE st.tea\_num = #{teaNum} AND pap.stu\_num IS NOT NULL

</select>

1. 论文查重功能

在论文上传时使用getSimHash方法计算每个文档的simHash值，并存储到数据库，导师发送查重请求，使用getHammingDistance方法计算海明距离，将数据处理传递到前端

getSimHash代码：

1. 分词：

List<String> keywordList = HanLP.extractKeyword(str, str.length());

1. 获取 hash 值

String keywordHash = getHash(keyword);

if (keywordHash.length() < 128) {

// hash值可能少于128位，在低位以0补齐

int dif = 128 - keywordHash.length();

for (int j = 0; j < dif; j++) {

keywordHash += "0";

}

}

1. 加权、合并

for (int j = 0; j < v.length; j++) {

if (keywordHash.charAt(j) == '1')

v[j] += (10 - (i / (size / 10)));

} else {

v[j] -= (10 - (i / (size / 10)));

}

}

i++;

}

1. 降维

String simHash = "";// 储存返回的simHash值

for (int j = 0; j < v.length; j++) {

// 从高位遍历到低位

if (v[j] <= 0) {

simHash += "0";

} else {

simHash += "1";

计算海明距离部分代码：

public static int getHammingDistance(String simHash1, String simHash2) {

int distance = 0;

if (simHash1.length() != simHash2.length()) {

// 出错，返回-1

distance = -1;

} else {

for (int i = 0; i < simHash1.length(); i++) {

// 每一位进行比较

if (simHash1.charAt(i) != simHash2.charAt(i)) {

distance++;

}

}

}

return distance;

}

TeacherServiceImpl类中导师查重代码：

public List<checkResult> checkPaper(String simHash, String stuNum) {

List<checkResult> simHashes = teacherMapper.getSimHashes(stuNum);

List<checkResult> checkPapers = new ArrayList<>();

for (checkResult DateSimHash : simHashes) {

if (HammingUtils.getSimilarity(DateSimHash.getSimHash(), simHash) > 0.80) {

double similarity = HammingUtils.getSimilarity(DateSimHash.getSimHash(), simHash);

DateSimHash.setSimilarity(similarity);

checkPapers.add(DateSimHash);

}

}

if (checkPapers.size() > 0) return checkPapers;

else return null;

# 系统测试

* 1. 测试用例
     1. 登录、注册模块测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试标题 | 测试步骤 | 测试数据 | 预期结果 |
| 登录测试 | 1. 入用户名和密码  2. 登录按钮 | 用户名：11111  密码：123 | 登录成功 |
| 学生注册测试 | 1. 点击学生注册按钮 2. 输入注册信息 3. 点击注册按钮 | 用户名：88888  密码：123  姓名：李四  学号: 311808 | 学生注册成功 |
| 导师注册测试 | 1. 点击导师注册按钮  2. 输入注册信息  3. 点击注册按钮 | 用户名：77777  密码：123  姓名：吴三  学号: 107 | 导师注册成功 |

* + 1. 用户管理模块测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试标题 | 测试步骤 | 测试数据 | 预期结果 |
| 新增管理员用户 | 1. 点击新增用户按钮 2. 输入管理员用户信息 3. 点击确定按钮 | 用户名：admin2  密码：admin2 | 添加管理员用户成功 |
| 新增学生用户 | 1. 增学生按钮  2. 输入学生用户信息 | 用户名：xuesheng  密码：123  姓名：黄八  学号：311810 | 添加学生用户成功 |
| 新增导师用户 | 1. 点击新增学生按钮  2. 输入导师用户信息  3. 确定按钮 | 用户名：daoshi  密码：123  姓名：陈五  工号：113 | 添加导师用户成功 |
| 删除用户 | 1. 点击删除按钮 2. 点击确定 |  | 删除用户成功 |
| 编辑用户 | 1. 点击编辑用户按钮 2. 修改用户信息   点击确定 | 密码：  admin2 ->admin3 | 编辑用户成功 |

* + 1. 学生管理模块测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试标题 | 测试步骤 | 测试数据 | 预期结果 |
| 编辑学生信息 | 1. 点击编辑学生按钮 2. 修改学生信息，点击确定 | 联系电话：  2222 -> 1780000 | 修改成功 |
| 删除学生信息 | 1. 点击删除按钮 2. 点击确定 |  | 删除成功 |
| 选择指导老师 | 1. 点击选择指导老师按钮 2. 选指导老师 3. 点击确定 |  | 设置成功 |

* + 1. 导师课题管理模块测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试标题 | 测试步骤 | 测试数据 | 预期结果 |
| 上传课题 | 1. 输入课题信息 2. 点击发布课题 | 课题名称：基于卷积层的人脸识别系统  课题难度:偏难  课题类型：计算机软件研发类  课题来源：自拟项目  课题摘要：这是…. | 添加成功 |
| 审核选题 | 1. 点击审核通过按钮 2. 确认审核 |  | 审核成功 |

* + 1. 导师审核模块测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试标题 | 测试步骤 | 测试数据 | 预期结果 |
| 查重论文 | 点击查重按钮 |  | 1. 显示查重通过 2. 显示查重结果 |
| 审核论文 | 1. 点击审核按钮 2. 输入审核信息 | 部通过  审核意见：查重率过高 | 审核通过 |

* + 1. 导师成绩管理模块测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试标题 | 测试步骤 | 测试数据 | 预期结果 |
| 添加成绩 | 1. 选择学生 2. 输入成绩信息 3. 点击添加按钮 | 成绩：80 | 添加成功 |
| 修改成绩 | 1. 进入查看成页面 2. 点击修改按钮 3. 输入修改的数据 | 成绩：  80->70 | 修改成功 |

* + 1. 学生毕业选题模块测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试标题 | 测试步骤 | 测试数据 | 预期结果 |
| 选择课题 | 1. 点击选择按钮 2. 确认选题 |  | 选题成功 |
| 退选课题 | 1. 点击退选 2. 确认退选 |  | 退选成功 |

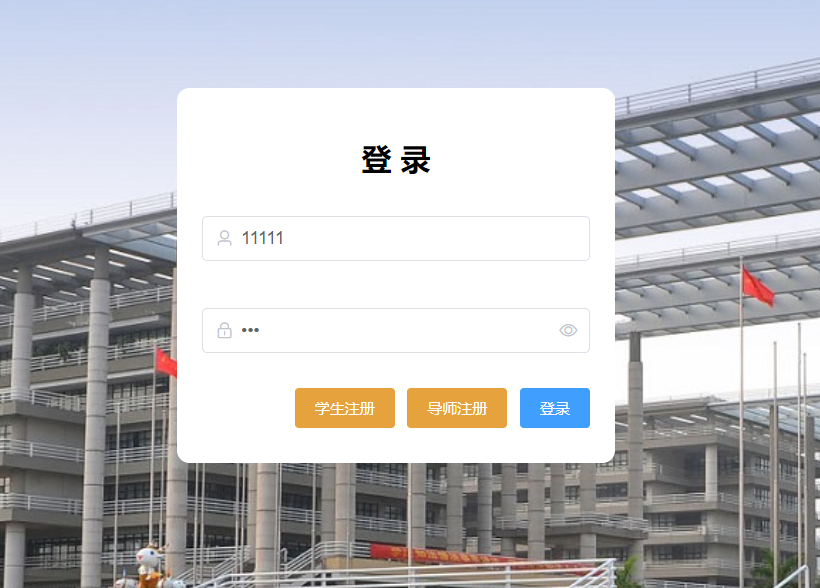
* + 1. 学生提交文档模块测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试标题 | 测试步骤 | 测试数据 | 预期结果 |
| 上传论文 | 1. 点击上传按钮 2. 选择上传文件 | 文件：本科毕业论文.txt | 上传成功 |

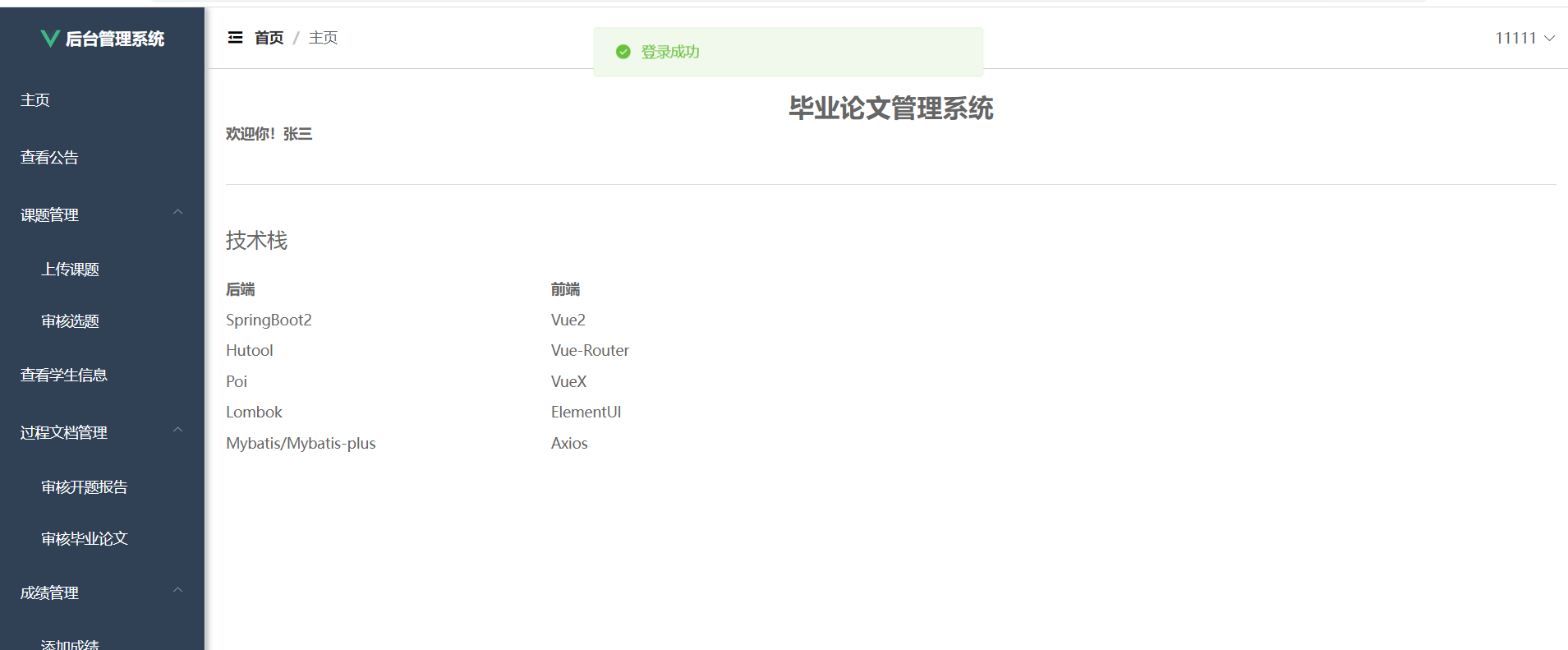
* 1. 测试结果
     1. 登录、注册模块测试结果

1. 登录测试结果

输入登录数据



登录成功：



1. 注册测试结果

输入注册数据



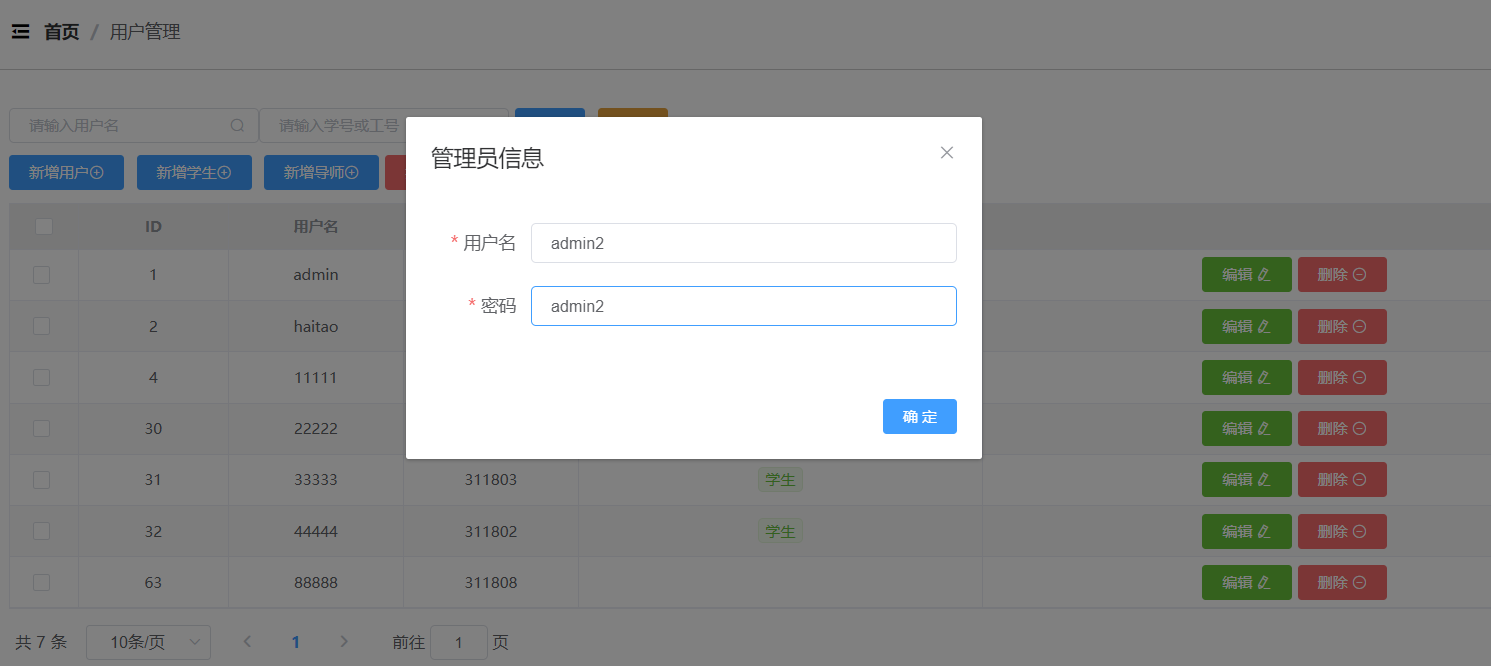
注册成功



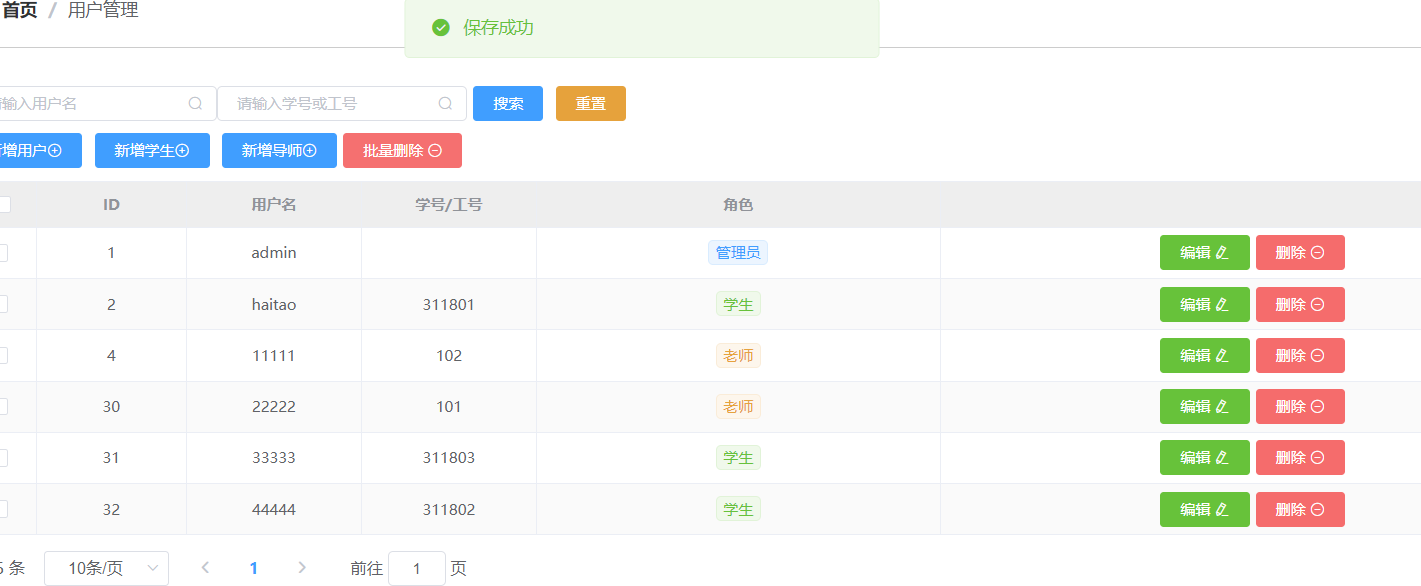
* + 1. 用户管理模块测试结果

1. 新增管理员用户

输入添加管理员用户数据：

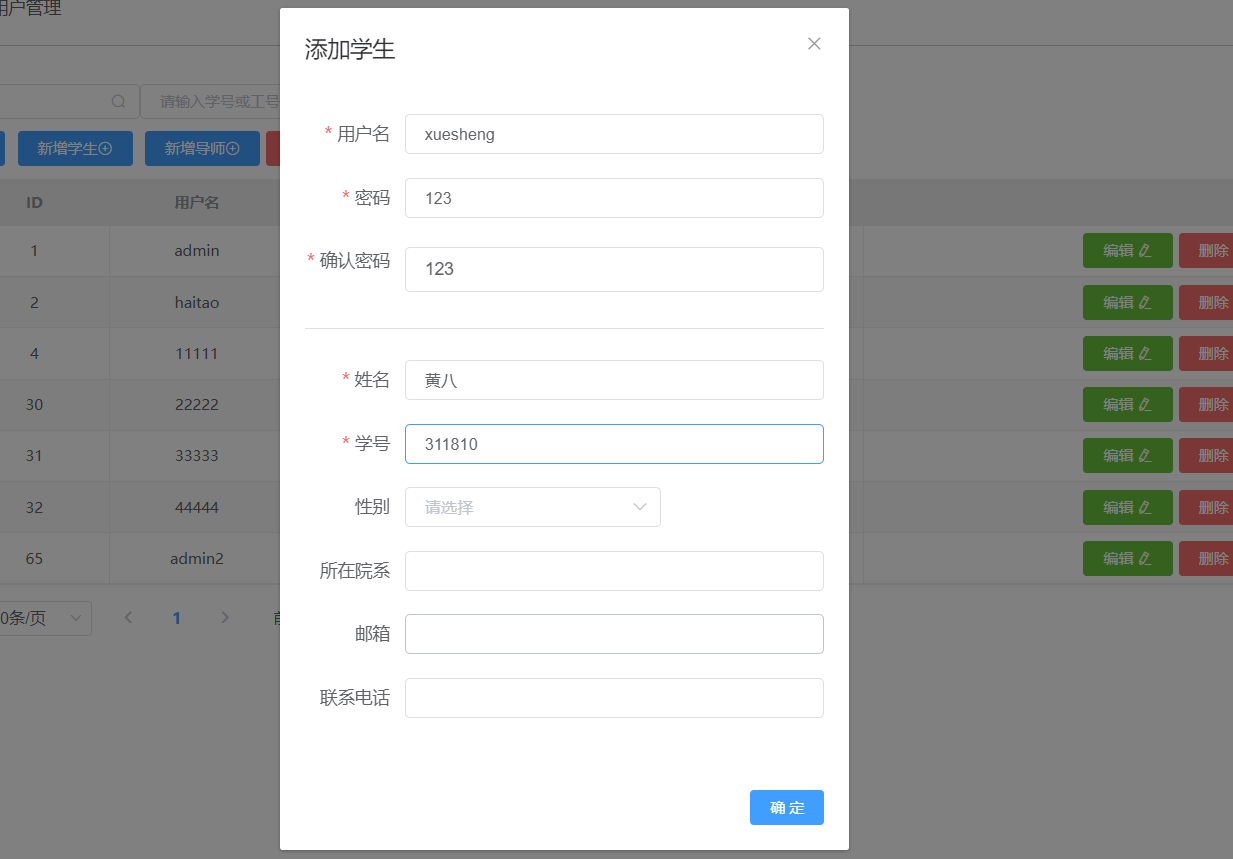


添加结果：



1. 新增学生用户

输入添加学生用户数据：

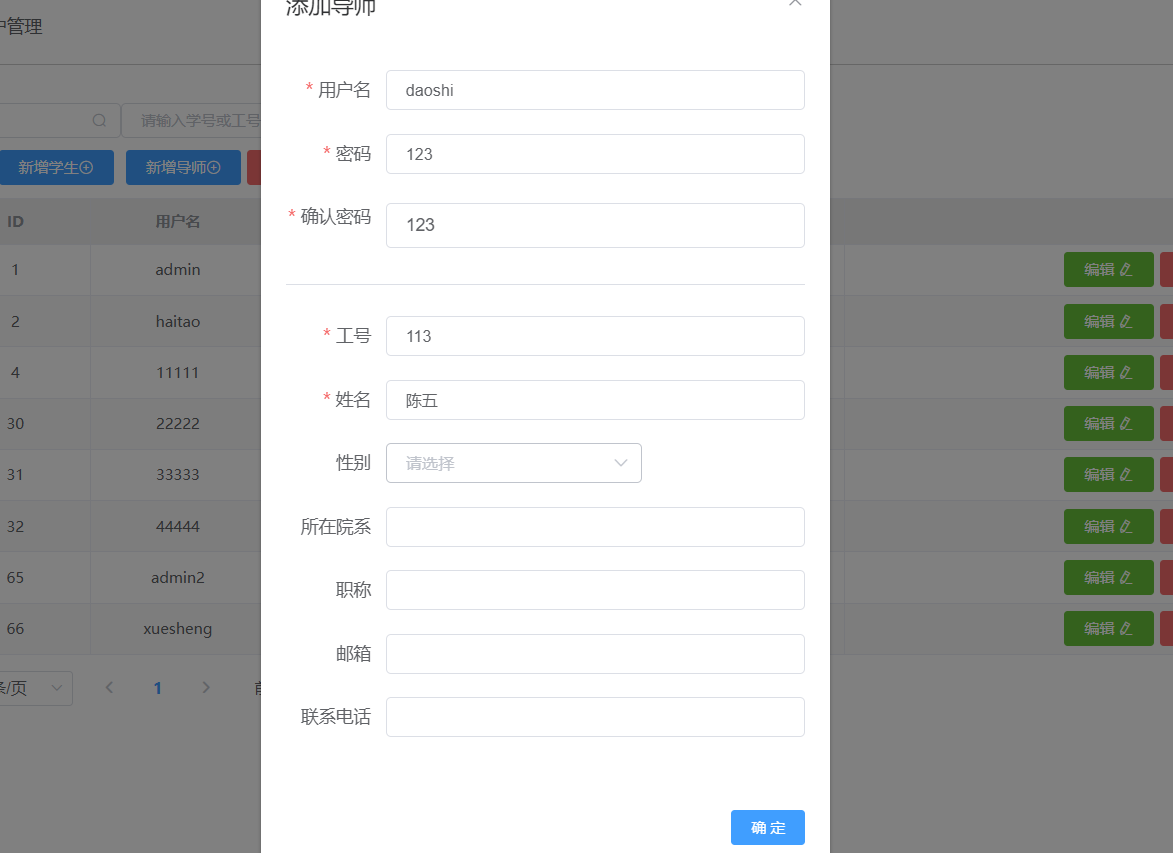


添加结果：

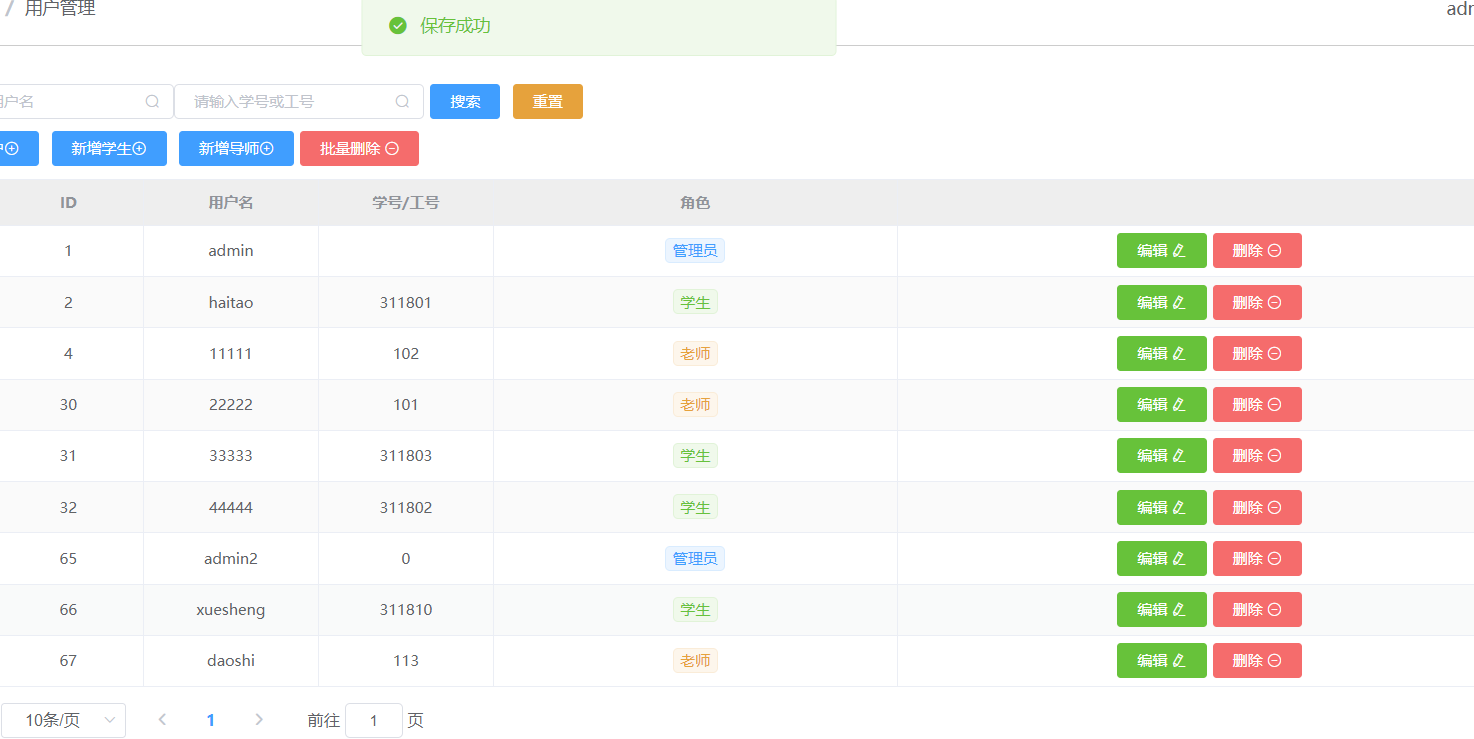


1. 新增导师用户

输入添加导师用户数据：



添加结果：

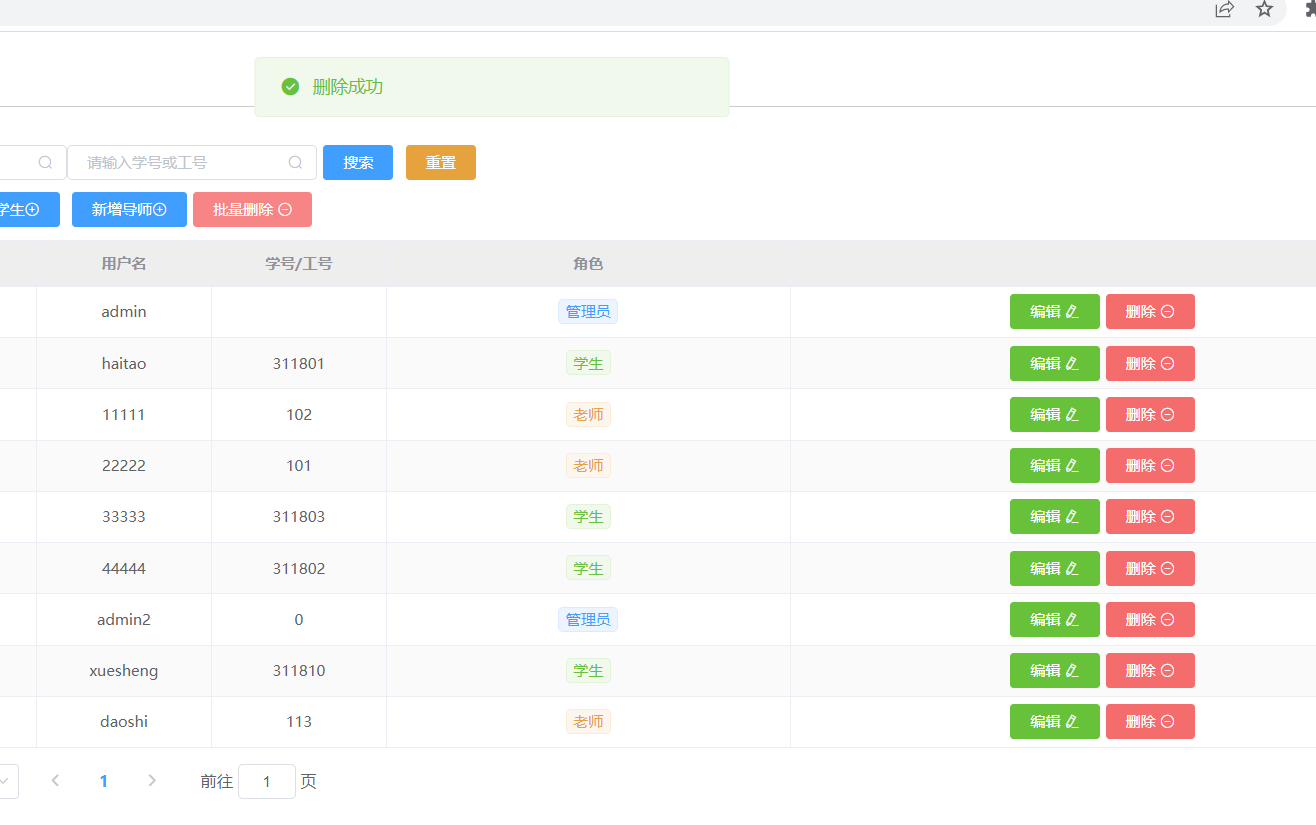


1. 删除用户

点击删除：



删除结果：

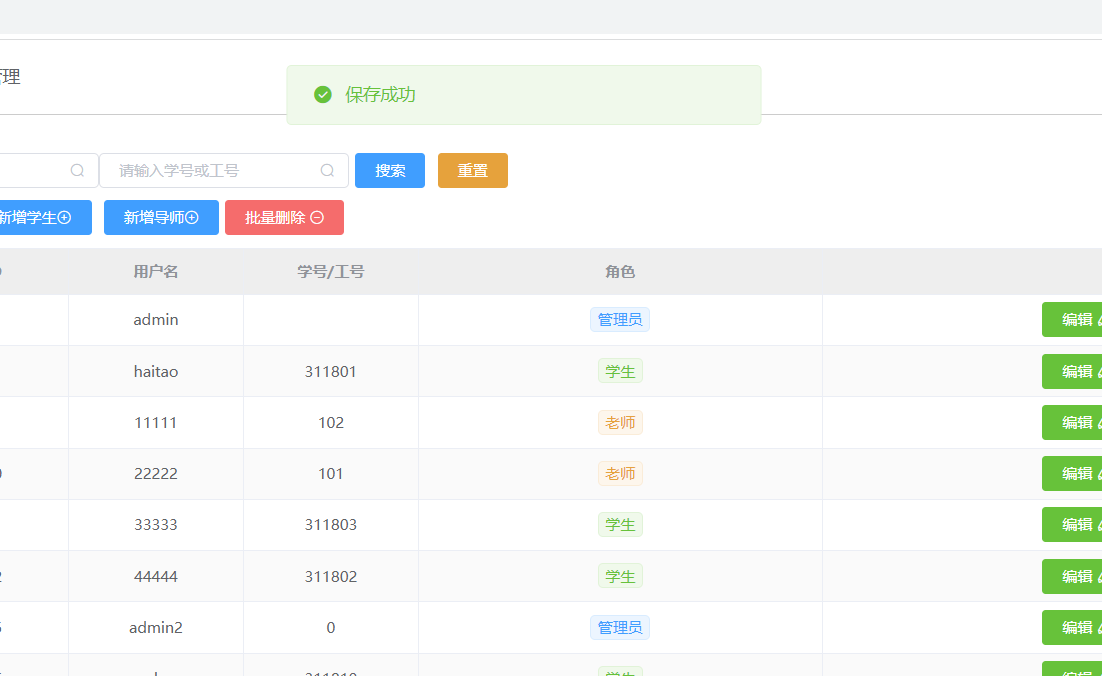


1. 编辑用户

输入编辑信息：



编辑结果：



* + 1. 学生管理模块测试结果

1. 编辑学生信息

输入编辑信息：

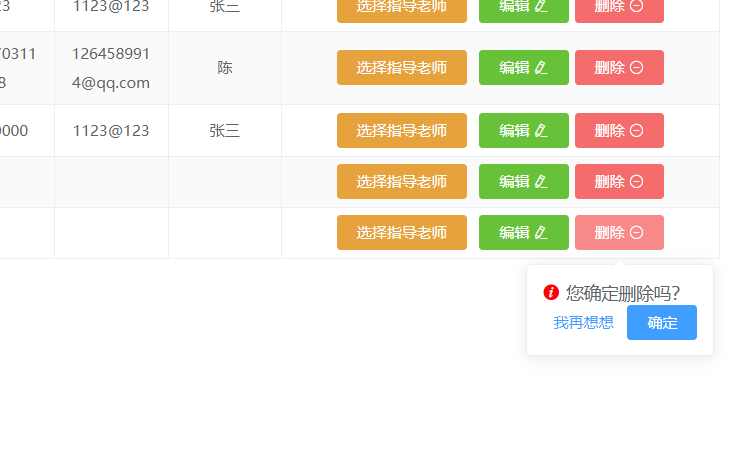


编辑结果：



1. 删除学生信息

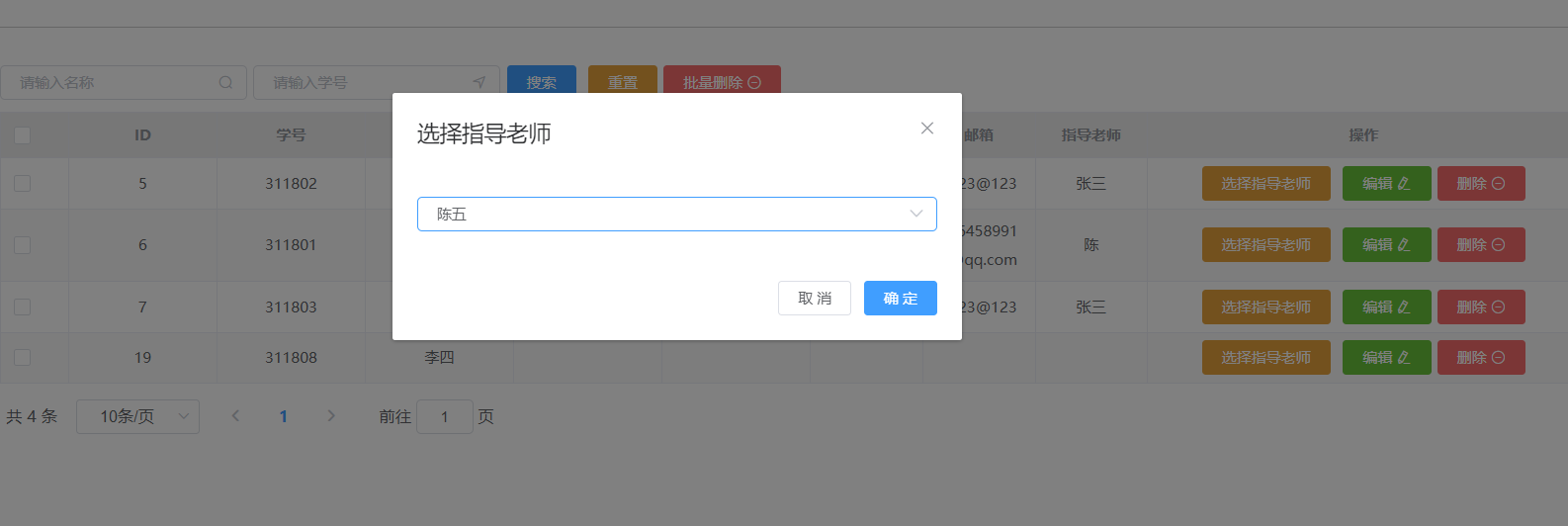
点击删除：



删除结果：



1. 选择指导老师



设置结果：



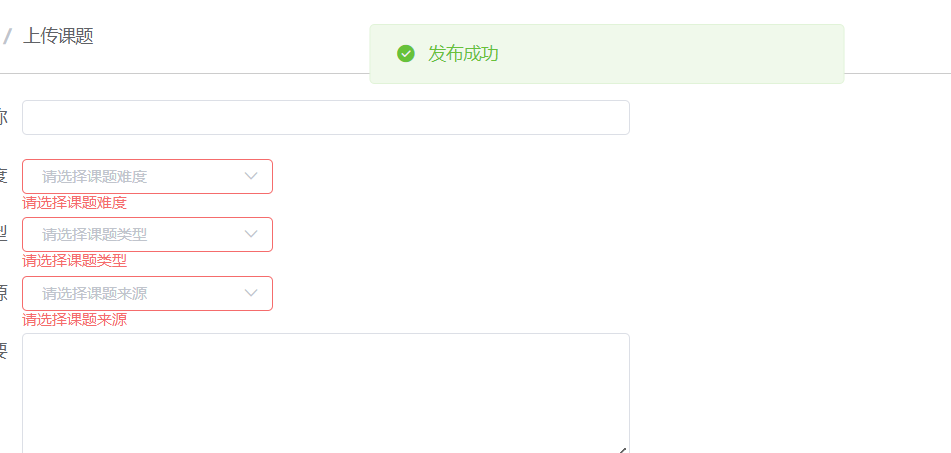
* + 1. 导师课题管理模块测试结果

1. 上传课题

输入课题信息：



上传结果：



1. 审核选题

点击审核中通过：

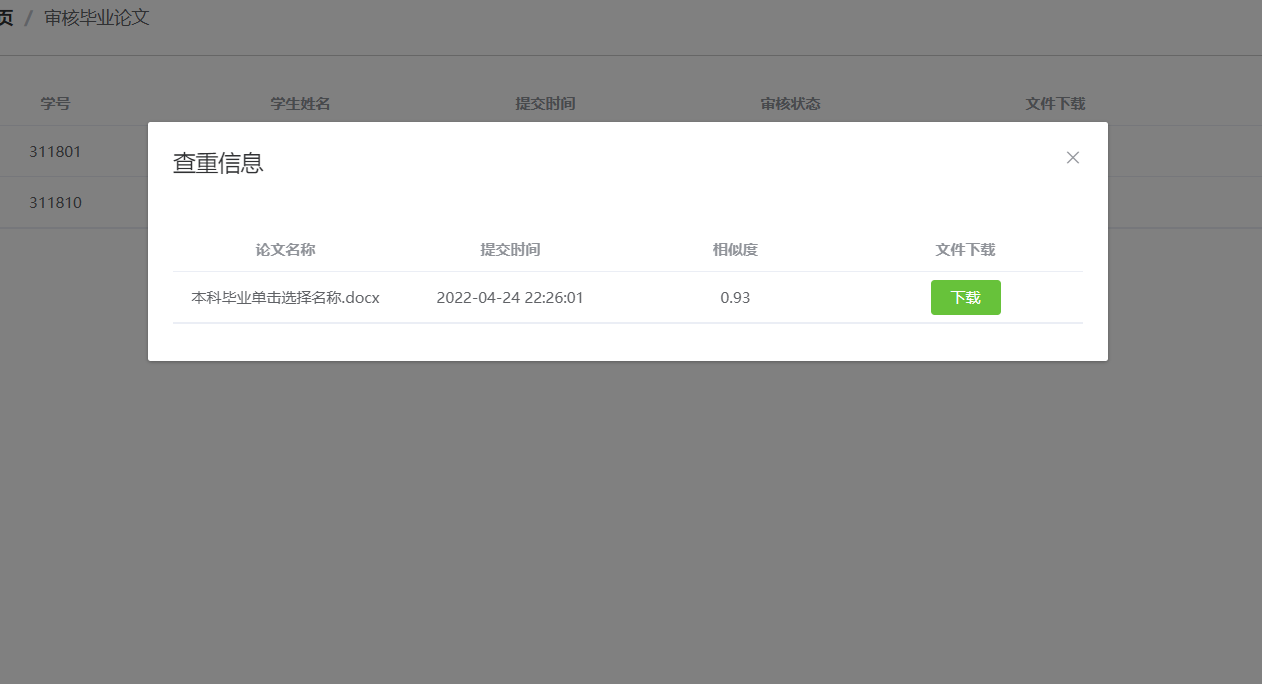


审核结果：



* + 1. 导师审核模块测试结果

1. 查重论文

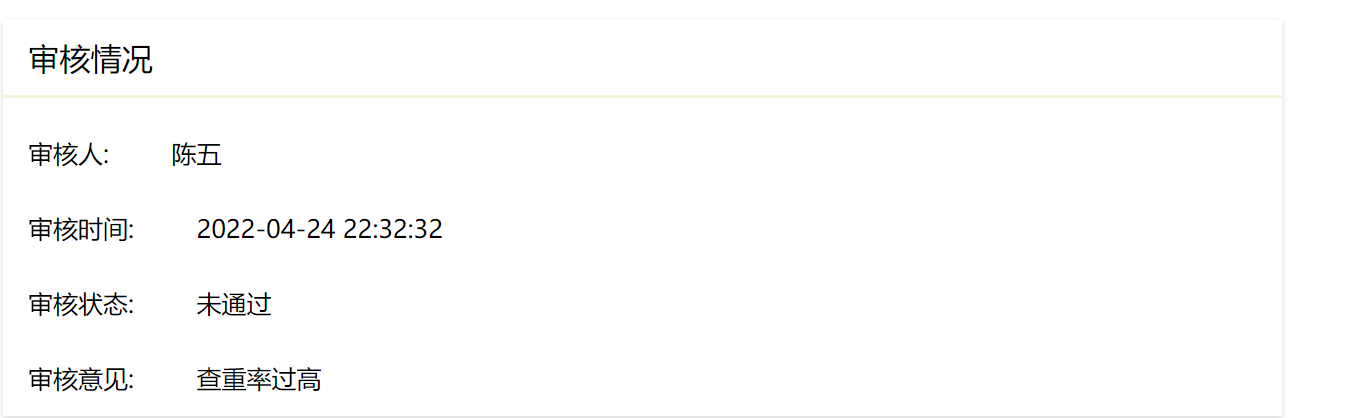


1. 审核论文

输入查重信息：



查重结果:



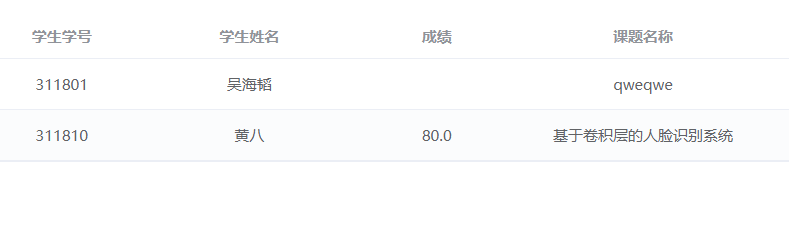
* + 1. 导师成绩管理模块测试结果

1. 添加成绩

输入成绩：



添加结果：



1. 修改成绩

输入修改成绩：



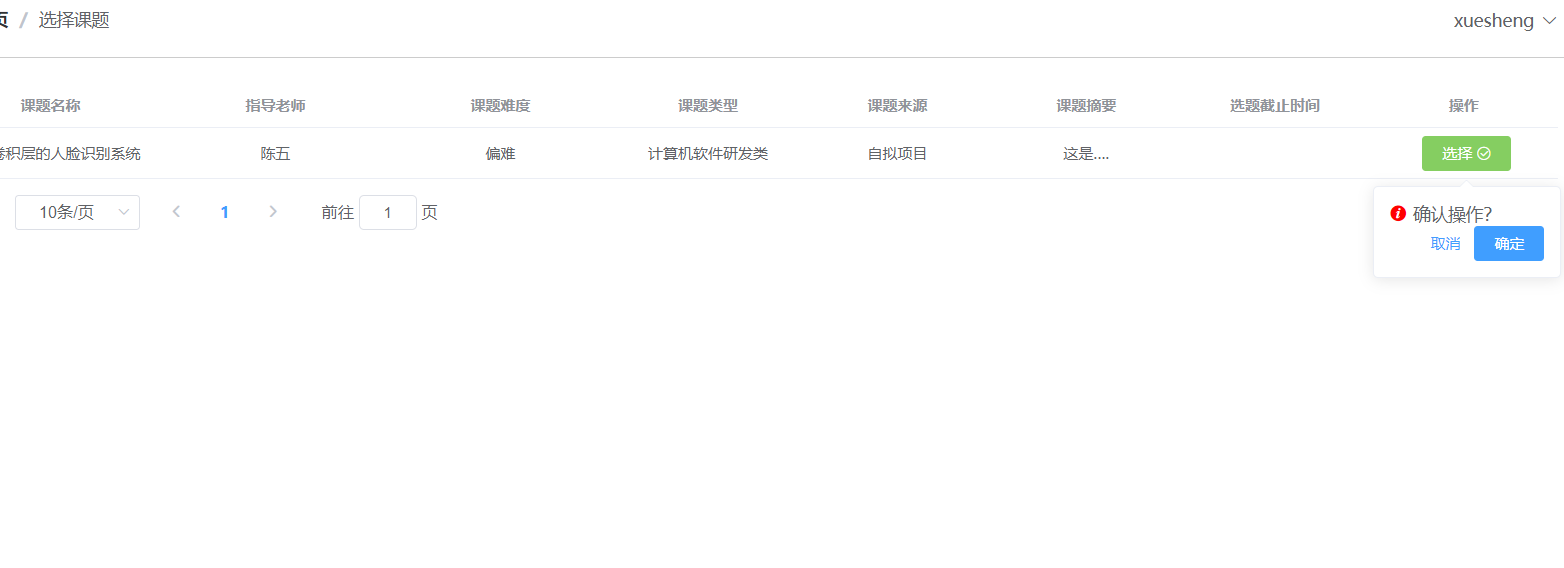
修改结果：



* + 1. 学生毕业选题模块测试结果

1. 选择课题

点击选题：



选题结果：



1. 退选课题

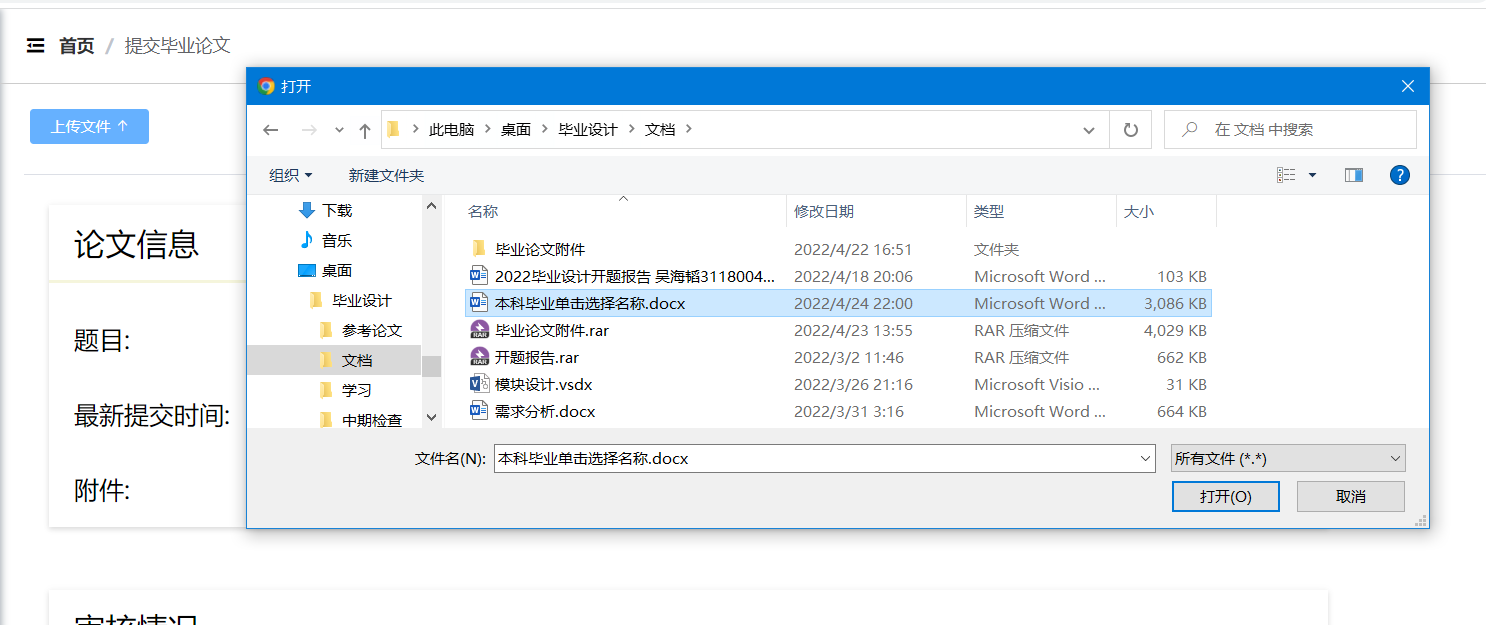
点击退选:

退选结果：

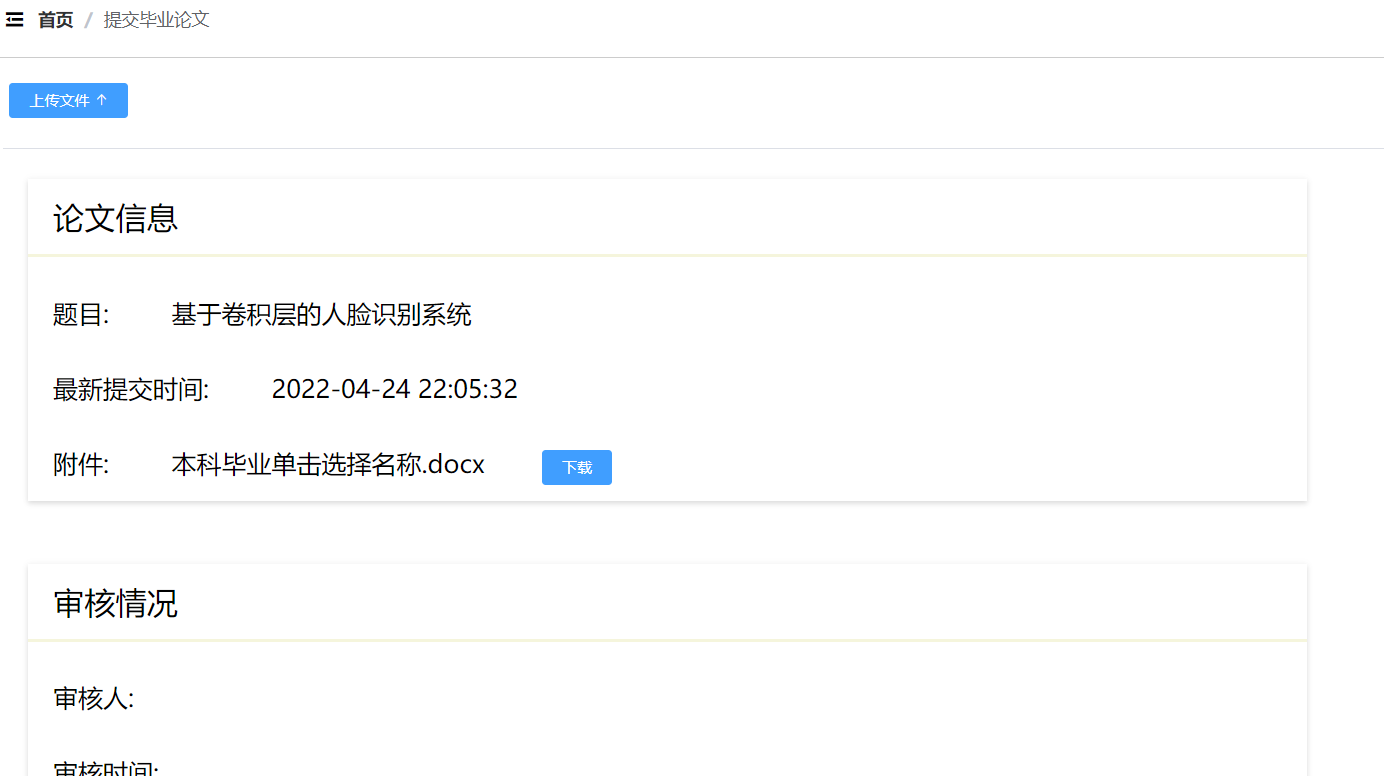


* + 1. 学生提交文档模块测试结果

选择文件上传：



上传结果：



# 结束语

本文论述了使用SpringBoot和Vue框架结合simhash算法制作出具有查重功能的毕业论文管理系统。本系统具有较强的可维护性和可移植性，提供较为简洁和优美的用户页面，缩短了高校管理员处理毕业论文的时长，减轻了教师指导毕业论文的压力，也为学生上传毕业论文，查看自身论文情况带来了便利。查重功能也能为高校做论文的初步审查，对提高毕业论文质量具有重要作用。但同时系统在非需求功能，如性能需求、数据库安全性还有待改进，在查重算法方面存在计算准确率无法保证、运算效率低的问题。

参考文献

* + - 1. Sadowski C, Levin G. Simhash: Hash-based similarity detection[J]. 2007.

[2] Sood S, Loguinov D. Probabilistic near-duplicate detection using simhash[C]//Proceedings of the 20th ACM international conference on Information and knowledge management. 2011: 1117-1126.

[3]王宇,夏丽丽,何颖刚.基于SSM框架的高校毕业论文管理系统的设计与实现[J].电子技术与软件工程,2021(22):26-29

[4]张晨阳,段国云,文春生.基于Simhash算法的文本查重系统的设计与实现[J].湖南科技学院学报,2021,42(5):51-54

[5]彭双和,图尔贡.麦提萨比尔,周巧凤.基于Simhash的中文文本去重技术研究[J].计算机技术与发展,2017,27(11):137-140145

[6]胡明,文翠,黄川.基于SpringBoot的毕业论文管理系统设计和开发[J].电脑编程技巧与维护,2020(7):92-93104

[7]吕宇琛.SpringBoot框架在web应用开发中的探讨[J].科技创新导报,2018,15(8):168-168173

[8]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.

[9]边超.基于B/S架构高校毕业论文管理系统的设计——以中国矿业大学银川学院为例[J].企业科技与发展,2020(12):64-67.

[10]钟怡旸,郭昱君.基于Springboot的博客管理系统设计与实现[J].现代信息科技,2021,5(07):18-20+24.DOI:10.19850/j.cnki.2096-4706.2021.07.005.

致谢

一晃四年过去，又来到一个人生阶段的结束的结点。四年的本科生涯经历了转专业、职业志向的转变，经历了很多困难、茫然和挫折。但幸运的遇见了很多朋友，在无助的时候，拉了我一把，在这里我想对他们表示我的感谢

感谢我的导师——陈平华教授，作为导师尽心尽责为了答疑解惑。正是导师的耐心指导，我才能顺利完成毕业论文

感谢我的舍友，与我度过了一个短暂又精彩的大学生活，也是有了他们的帮助，让我对计算机领域有了更深的认识

感谢我的父母的养育之恩，在我困难的时候给与支持，对我的任何选择都无条件支持。

感谢我的好朋友-曾彦祺，作为好朋友，给了我精神上极大的支持，为我加油打气、为了排忧解难，我很幸运能在生命中遇到这样一位挚友。

输入您的附录标题