基于SSH框架的学生信息管理系统的分析与实现

【原文对照报告-大学生版】

报告编号：c454a48bb333d317　　检测时间：2020-04-14 18:50:43　　检测字数：22,003字

作者名称：孙文阁　　所属单位：云南工商学院（教务处）

检测范围：

◎ 中文科技期刊论文全文数据库
◎ 中文主要报纸全文数据库
◎ 中国专利特色数据库
◎ 博士/硕士学位论文全文数据库
◎ 中国主要会议论文特色数据库
◎ 港澳台文献资源
◎ 外文特色文献数据全库
◎ 维普优先出版论文全文数据库
◎ 互联网数据资源/互联网文档资源
◎ 高校自建资源库
◎ 图书资源
◎ 古籍文献资源
◎ 个人自建资源库
◎ 年鉴资源
◎ IPUB原创作品

时间范围：1989-01-01至2020-04-14

检测结论：

全文总相似比		复写率		他引率		自引率		专业术语
13.31%	=	10.37%	+	2.94%	+	0.0%	+	0.0%

其他指标：

自写率：86.69%

专业术语：0.0%

高频词：信息,返回,管理,学生,成功

典型相似性： 无

指标说明：

复写率：相似或疑似重复内容占全文的比重

他引率：引用他人的部分占全文的比重，请正确标注引用

自引率：引用自己已发表部分占全文的比重，请正确标注引用

自写率：原创内容占全文的比重

专业术语：公式定理、法律条文、行业用语等占全文的比重

典型相似性：相似或疑似重复内容占互联网资源库的比重，超过30%可以访问

总相似片段：76

期刊：7 博硕：34 外文：0 综合：1 自建库：6 互联网：28

毕业论文（设计）

基于SSH框架的学生信息管理系统的分析与实现

学 院 智能科学与工程学院

专 业 计算机科学与技术

班 级 2016级【二本】计算机科学与技术1班

学 号 16612500025

姓 名 孙文阁

指 导 教 师 向金明

成 绩

云南工商学院

20 年 月

云南工商学院

毕业论文（设计）诚信声明

本人声明，所呈交的论文（设计）是本人在导师的指导下独立完成的研究成果。除了文中特别加以标注引用的内容外，本论文（设计）不包含法律意义上已属于他人的任何形式的研究成果,也不包含本人已用于其他学位申请的论文（或成果。对本文的研究作出重要贡献的个人和集体，均已在文中以明确方式表明。本人完全意识到本声明的法律后果由本人承担。本毕业论文(设计)成果归云南工商学院所有。

作者签名：

日 期：

基于SSH框架的学生信息管理系统的分析与实现

摘 要

本次课题主要应用于高校学生管理，主要为解决高校学生管理工程量大，信息储存困难，流程复杂化而开发的一个管理系统，主要介绍开发工程师在开发系统过程中所经历的需求分析，系统的功能设计与数据库设计到代码的编写与完成，和对系统进行的各项测试，本人在本次课题内担任全栈开发工程师。

依照本次课题所设计的管理系统，为解决各项难题所需要用到Spring、Spring MVC、Hibernate以及B/S框架技术和前端的Layui框架，数据库采用MySQL服务器。

该系统完成后，能够减轻高校管理人员的工作负担，高效率、规范化地处理、储存大量的学生信息，促进学校对于学生的规范管理。

本次课题能积累大量的开发经验与了解高校学生管理的复杂逻辑，熟悉系统开发的生命周期，锻炼独立解决问题能力，更多的了解系统各项错误的解决方法。

关键词：基于SSH框架的学生信息管理系统；全栈开发工程师；Spring+SpringMVC +Hibernate；B/S；J2EE；ASP；Java

ABSTRACT

This subject is mainly applied to the management of college students. It is mainly developed to solve the problems of large amount of student management engineering, difficult information storage and complicated process. It mainly introduces the requirement analysis experienced by the development engineer in the process of developing the system, the functional design and database design of the system to the compilation and completion of the code, and the various tests of the system. I am a full stack development engineer in this project.[1]

According to the management system designed in this paper, in order to solve the problems, Spring~SpringMVC, Hibernate, B / S framework technology and front-end Layui framework are used, and the database adopts MySQL server.[2]

After the completion of the system, the work burden of university managers can be reduced, a large number of student information can be processed and stored efficiently and standardized, and the standardized management of students can be promoted.

This project can accumulate a lot of development experience to understand the complex logic of college student management, familiar with the life cycle of system development, exercise the ability to solve problems independently, and more understand the solutions to the errors of the system.

Key Words: Student information management system based on SSH framework; full stack development engineer; Spring+SpringMVC+Hibernate; B/S; J2EE; ASP; Java

目 录

引言

学生信息化管理对于现在的各个高校至关重要，老式的高校学生管理存在着信息处理效率慢，管理人员工作负担大，信息储存不方便的而产生的各种问题，因此，现在的高校都急需改变学生管理方式。而采用学生信息系统，能够更加信息化、规范化的管理学生信息，能大大提高高校管理效率。

随着信息化时代的快速发展，信息化管理将会越发成熟，它能够提高各个行业的工作水平与工作效率，带动经济的快速发展。老式的传统信息管理方式必定会被计算机处理的信息化管理所取代。

该项目可发布与高校网内，用于规范，合理的管理学生信息，减轻管理人员的工作负担，增加信息交互的效率，可节约纸张资源。对于促进高校学生管理有着重大意义。该项目使我增加大量开发经验，熟悉项目开发的完整流程，增强自己的开发能力，对于复杂的业务能够更好的理解。

学生管理系统中，管理员能够实现对学校、学院、专业、班级的增删改，以及对菜单的增删改，对用户权限的开启，禁用，对任课信息的管理和公告信息的管理。

学生用户登录后能发表自己的信息，展示在前端平台上，能够根据条件查询自己各个科目的成绩，可以根据学期统一查询成绩。

教师用户登录后对学生信息按条件查询，修改。对自己授课的班级内学生进行打分与评价。

第一章、绪 论

1.1.开发背景

现如今正是信息化和科技高速发展的时代，各行各业都离不开信息化处理，信息化为人们带来的好处随处可见，帮助人们减轻工作负担，减少工作流程，尤其是对于复杂的信息管理，信息化管理能充分发挥它的优势。

信息管理系统是进行对信息的进行储存、查询、修改合理操作的系统。它是科学管理和科学技术的发展而形成的。在各大高校中，拥有一个学生信息管理系统对高校来说是必须的。它的功能对于高校的管理者起极大的促进作用，能够为高校的管理者提供更便捷且准确的管理方式。传统的学生信息管理方式为以纸张作为媒介进行信息的储存与信息的采集、传递的办公方式，运用纸张储存学生信息与办公，人为的传递信息存在着工作效率低、工作流程复杂、信息保密性差等很多缺点，在信息的存储和学生信息的查询时都会出现不少问题。

1.2.开发意义

为了提高高校管理人员的办公效率和减轻人员繁琐的工作流程，开发学生信息管理系统，运用网路信息技术对学生信息进行管理，已经形成一种趋势。这种学生信息管理方式更节省时间，更加能实现规范的学生信息管理，更加方便管理人员的维护和储存学生的复杂信息。完整的系统可在学校的管理中发挥着巨大作用。能够实现工作流程的信息化，提高工作效率和减轻工作负担。

该项目可发布与高校网内，用于规范，合理的管理学生信息，减轻管理人员的工作负担，增加信息交互的效率，可节约纸张资源。对于促进高校学生管理有着重大意义。[1]

1.3.可行性分析

1.3.1经济可行性分析

针对本系统的开发，主要的开发消耗为办公设备的消耗（电脑，服务器，数据库等），办公用品的消耗（纸张，笔等），办公人员的生活消耗（吃，水电，房租等）。项目总体预估耗费30000元。本系统非收益性质开发，重要的是为解决学校进行学生信息管理的需要。

1.3.2技术可行性分析

系统的硬件要求方面不存在特殊的要求，不需要花高价的服务器，只要能够满足我们使用的需求就好。保证其用户在使用时的稳定，顺畅。就目前的台式计算机来说，基本都能满足我们的使用要求。在此方面无须太多担心，保证系统的可运行。所以在本次开发硬件这一方面是不存在问题的。

本次开发使用的JSP，子类重用，可以把功能模块化，方便数据的调用，并进行有效的存储。使用JAVA开发语言的一个最大的特点就是可以重复调用同一个代码功能模块，代码复用。所用到的技术都已掌握，该系统只是把所学技术合理应用即可开发，所以技术上是完全可行的。

1.3.3应用价值的可行性分析

开发学生信息管理系统，使用计算机对学生信息进行管理，可提高高校管理人员办公效率，减轻高校人员的工作流程，节省时间，实现更加规范的学生信息管理，更加方便管理人员的维护和储存学生的复杂信息。完整的系统可在学校的管理中发挥巨大作用。实现工作流程的信息化，提高工作效率和减轻工作负担。

学生信息系统能够更安全的储存与处理大量的学生信息，保障学生信息的安全性，减少职工人员在办公时的精力消耗，把更多的精力放在对学生的管理上，促进学校对于学生的规范管理。

1.4.研究内容介绍

本课题研究基于SSH的学生信息管理系统的设计与实现。该系统的主要目标为简化学生信息，方便学生信息的采集与管理。主要研究学生信息的查询、储存，教师对学生信息的修改以及基础的系统管理，系统的主流程为教师对学生信息的管理统计和对学生成绩的打分与对学生的评价，在主流程中，包含多个分流程及细节操作。[2]

本课题实现网站所涉及到的技术：SSH框架、日志管理、权限管理、JSP、Layui框架和MySQL数据库。其中艰难部分为学生信息的统计

与采集，教师用户与课程信息和学生信息以及班级的映射关系。以及教师用户对课程的绑定以及对学生用户进行打分和点评的复杂逻辑关系。

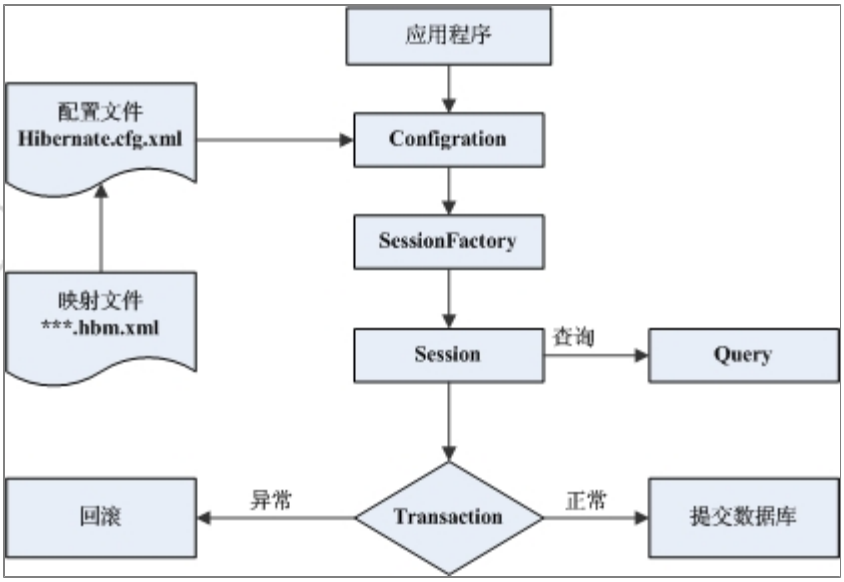
第二章、相关技术概论

2.1

2.1.JavaWeb

JavaWeb就是后台处理用java代码编写的web项目， Java开发后端技术包含spring MVC框架、Spring容器、分布式session管理等技术，Web前端中可以使用JSP、html、jQuery、vue、layui等技术。

2.2.Hibernate



Hibernate是一种ORM框架，可以自动生成数据库语句，自动执行。所又JDBC可使用的场所，Hibernate皆可以使用。使程序开发工作者更方便。Hibernate的运行过程如图所示：

2.3.MY SQL

MySQL比之SQL服务器具有更强大的性能，它最大特点就是适用于中小型企业用，并有着存储数据快特点，并且MySQL服务器的可扩展性与开放的数据引擎更让SQL黯然失色，相比与SQL减少了额外的复杂操作、磁盘处理、内存损耗等等。[3]

2.4.Apache Tomcat8.0

Apache Tomcat是由Apache开发的免费且开源的轻量级WEB应用服务器，8.0更是相比其他版本具有稳定性，是初学者开发和调试JSP程序的首选。[4]

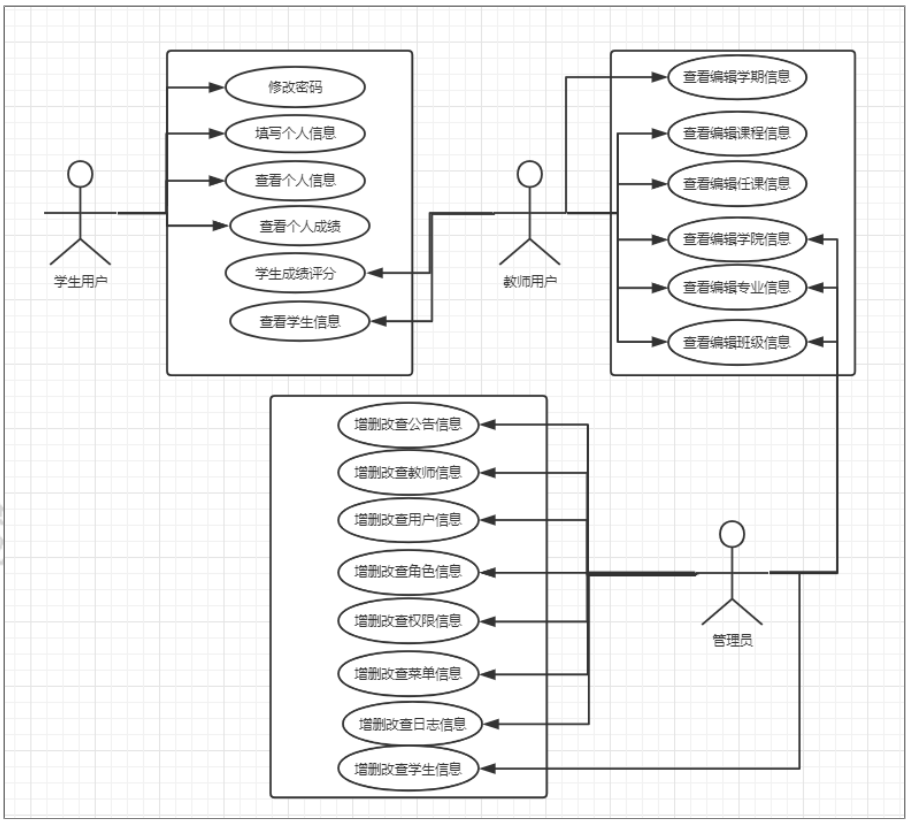
第三章、需求分析

3.1

3.1.编写目的

该项目可发布与高校网内，用于规范，合理的管理学生信息，减轻管理人员的工作负担，增加信息交互的效率，可节约纸张资源。对于促进高校学生管理有着重大意义。该项目使我增加大量开发经验，熟悉项目开发的完整流程，增强自己的开发能力，对于复杂的业务能够更好的理解。[5]

3.2.



需求描述

3.2.1学生用户需求

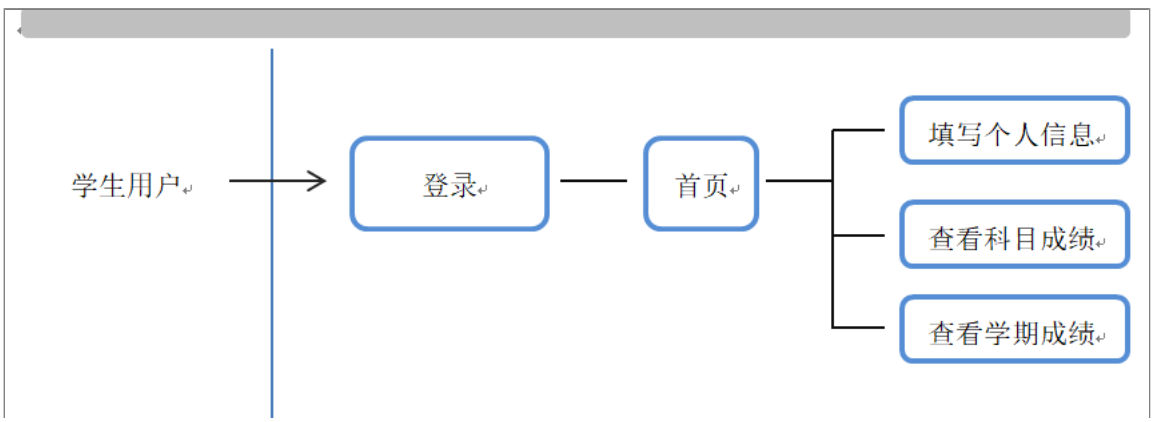
学生用户登录可查询，修改，提交自己的基本信息，提交后登录页面可查看自己个人信息并对个人进行编辑操作。并可以根据条件查看自己各个学期，各个科目的成绩及教师评价。

3.2.2教师用户需求

教师登录后能够按条件查询学生信息，显示学生信息列表，并能对学生按班级查询自己授课的科目进行学生的评价与打分，还能够对学期信息查询、修改对课程信息进行查询、修改、对已授权教师进行课程的绑定。

3.2.3管理员用户需求

管理员能够实现对学院、专业、班级的按名称查询与添加，查询后可进行修改与删除，对学期信息、课程信息、任课信息进行添加、删除、修改，以及对教师身份真实性的校验和用户状态的启用与禁用，对学校新闻信息的添加、修改、删除。以及对系统管理内的各个操作模块的查询、添加、修改、删除，还可查看系统的登录与异常日志信息并且进行修改与删除。

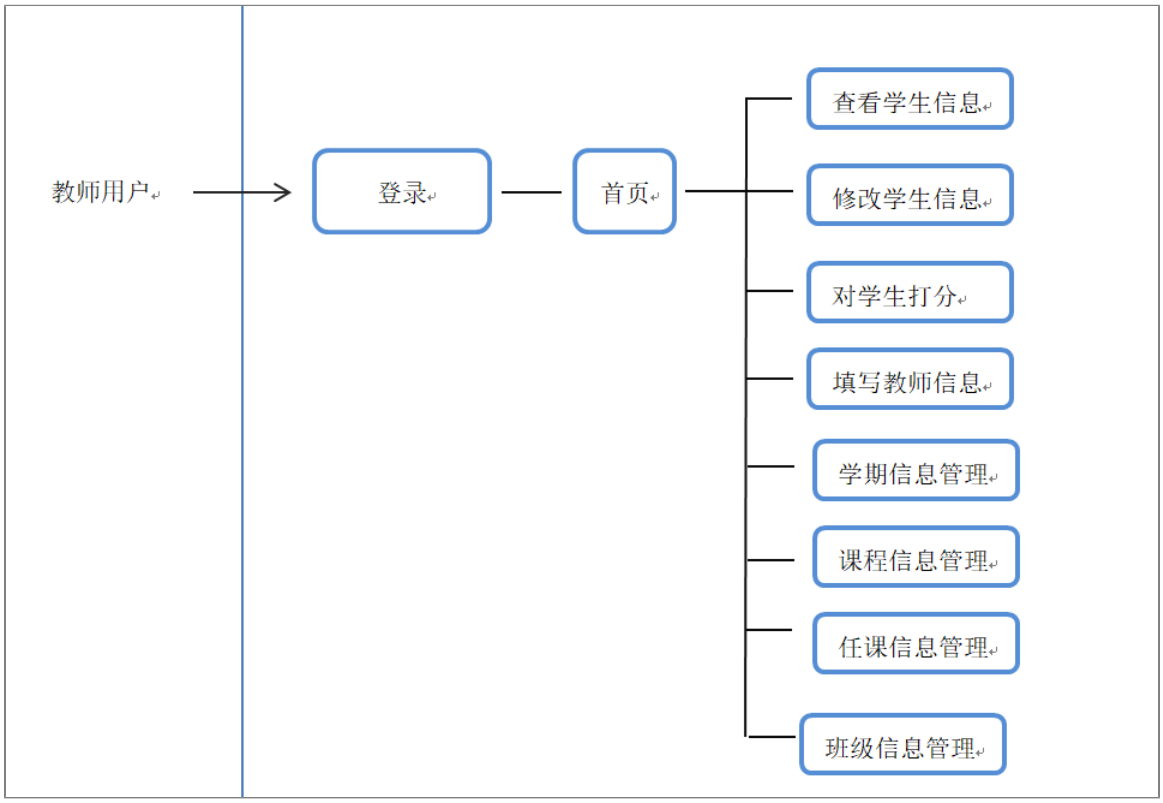


第四章、概要设计

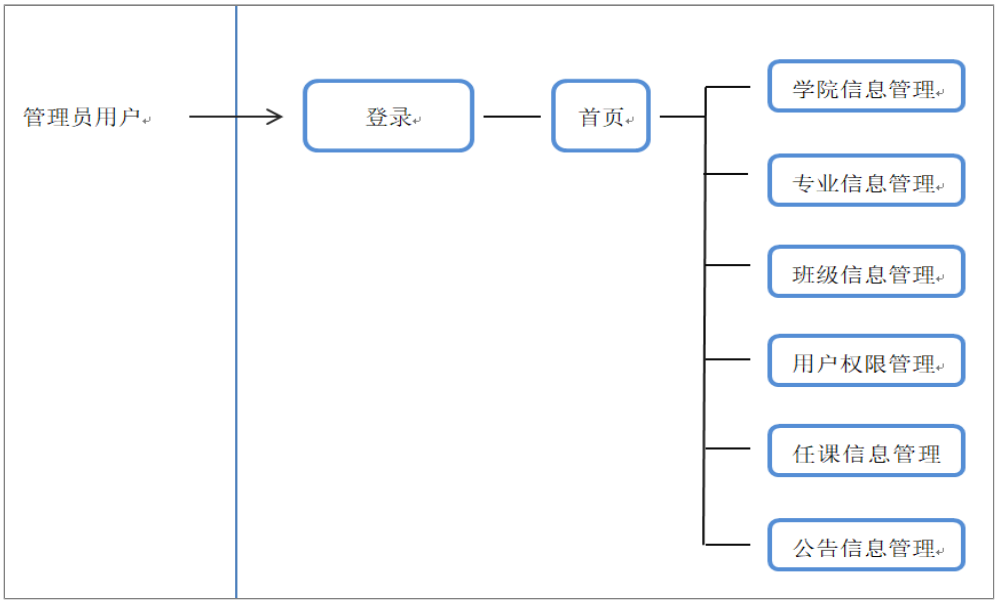
4.1

4.1.系统逻辑架构

学生用户登录，判断成功后进入首页，学生可填写，更新自己的个人信息，上传自己的个人头像，查看自己每个科目的详细成绩信息以及教师所做评价，以及按照学期查询的各个科目成绩信息。

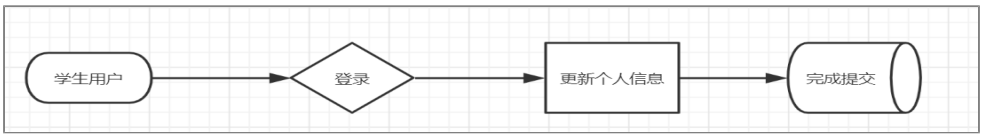


教师用户登录，判断成功后进入首页，教师用户可填写自己基本信息，根据条件查询学生信息，列出学生列表，选中学生，可修改学生信息。可根据学院、专业、班级查询，可根据自己所授课进行查询。查询后对学生成绩进行评分，对学生进行评价。教师登录后进行权限判断，可管理学期信息，课程信息，任课信息以及学院、专业、班级信息。

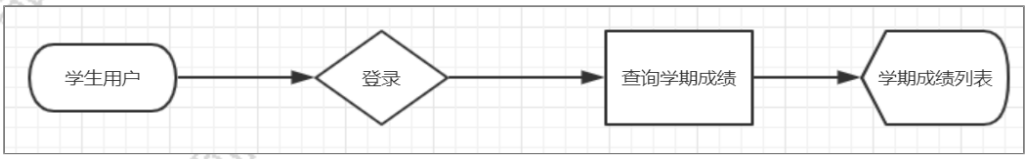


管理员用户登录，判断成功后进入首页，可对学院信息进行添加、修改、删除，查改以及对学院院长的绑定，对专业信息的添加、修改、删除，对班级信息的添加、修改、删除，对各个用户菜单权限的开启，禁用，对教师用户任课信息的添加、修改、删除，查改，对学校公告信息的发布，删除，修改。[6]

4.2.

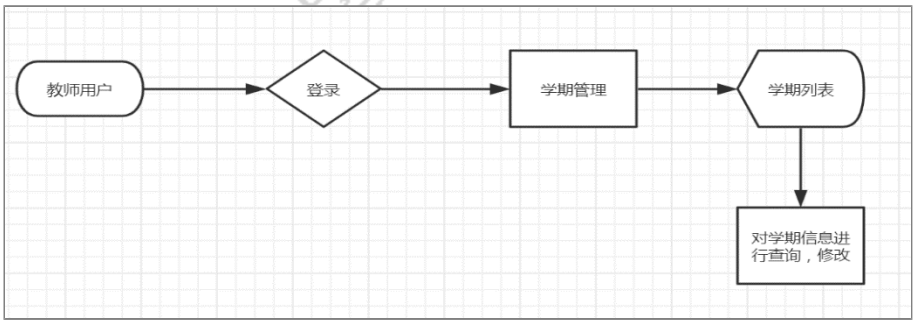


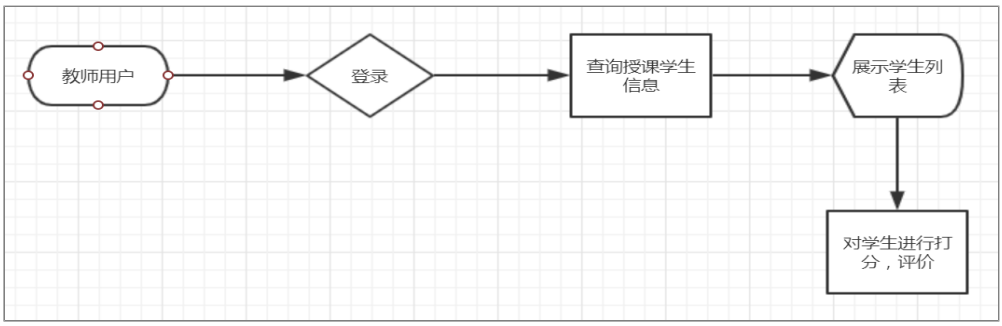
系统功能模块设计



学生用户进行登录，登录成功后可以查看和更新个人信息，并完成个人头像的上传与修改。

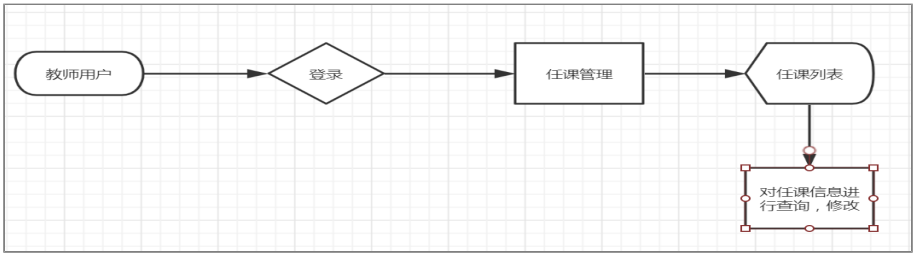
学生用户进行登录，登录成功后可按条件查询每个学期成绩，成功显示查询到的学期成绩列表。

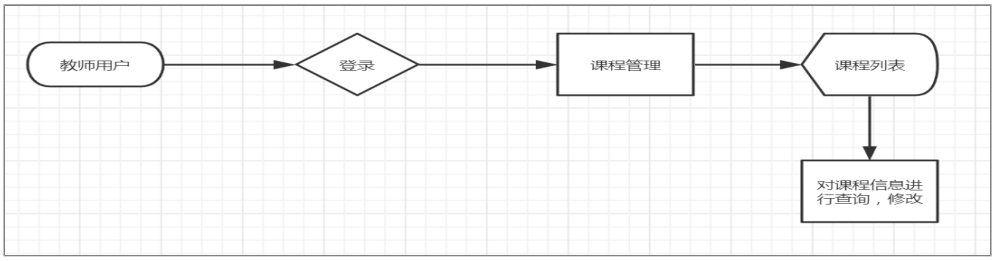




教师用户登录，判断成功后可按条件查询自己授课学生，查询后展示学生列表，可对学生成绩进行打分，对学生进行评价。

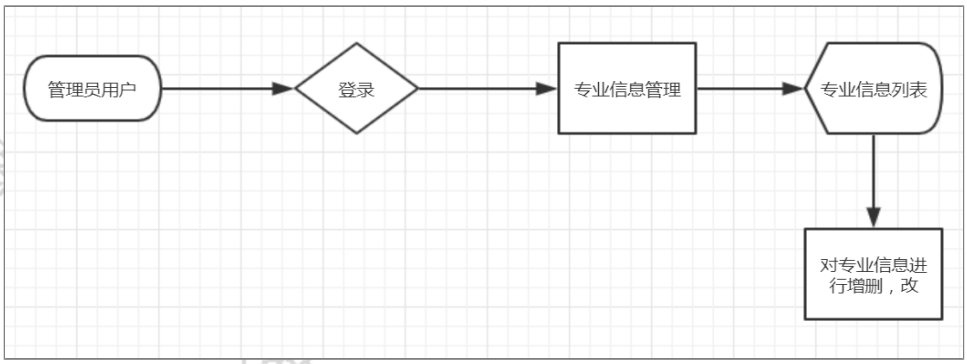
教师用户登录，判断成功后可按条件查询学期信息，查询后展示学期列表，可对学期信息进行查询，修改。



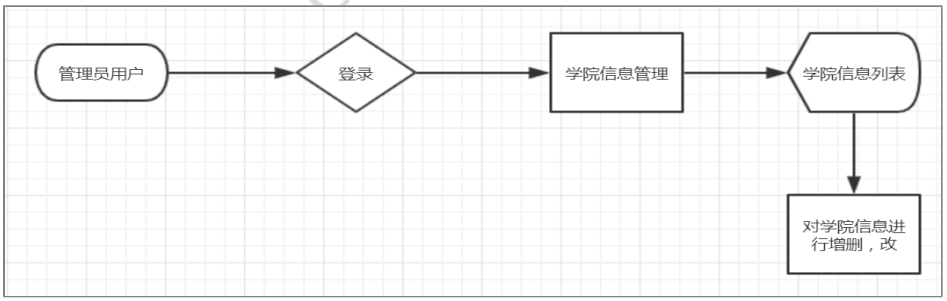


教师用户登录，判断成功后可按条件查询课程信息，查询后展示课程信息列表，可对课程信息进行添加，修改，删除。

教师用户登录，判断成功后可按条件查询任课信息，查询后展示任课信息列表，可对任课信息进行查询，修改。

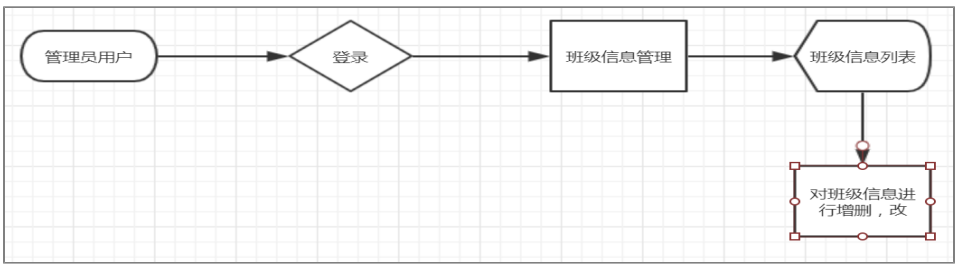


管理员用后登录，判断成功后可进入学院信息管理，获得学院信息列表，可对学院信

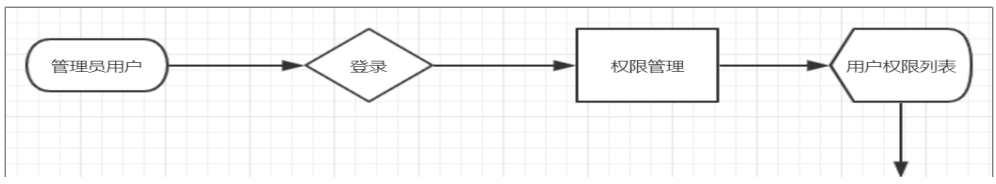


息进行添加，修改，删除，对学院院长进行修改。

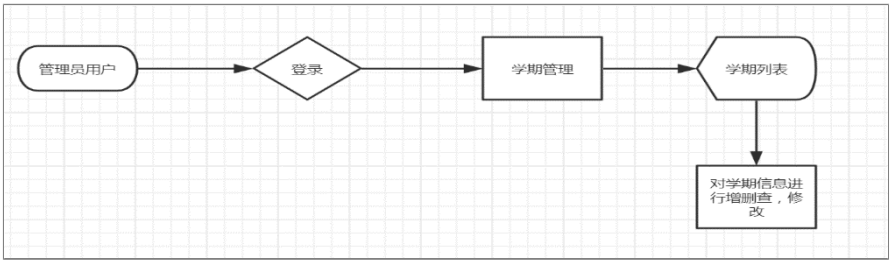
管理员用户登录，判断成功后可进入专业信息管理，获得专业信息列表，可对专业信息按名称以及学院进行查询，可添加、修改、删除专业信息。

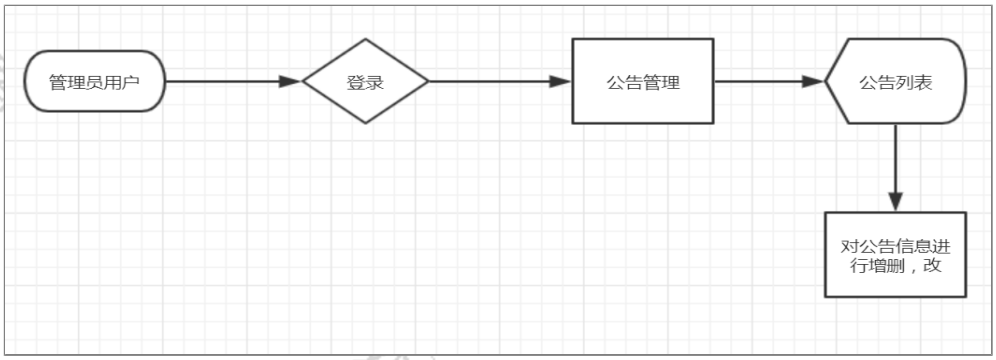


管理员用后登录，判断成功后可进入班级信息管理，获得班级信息列表，可对班级信息进行添加，修改，删除，对班级进行辅导员的修改。

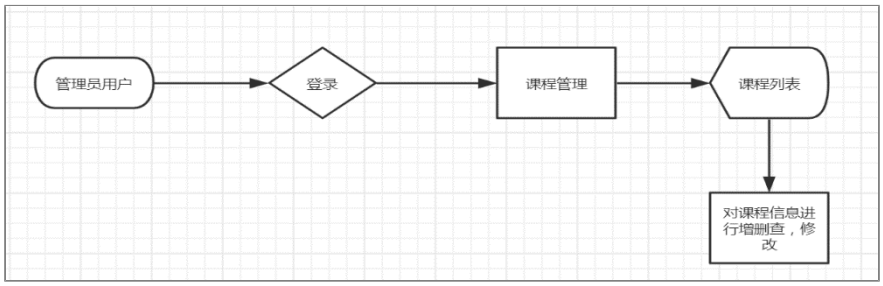


管理员用后登录，判断成功后可进入权限管理，获得用户权限列表，可按条件查询用户权限，对用户权限进行启用，禁用。

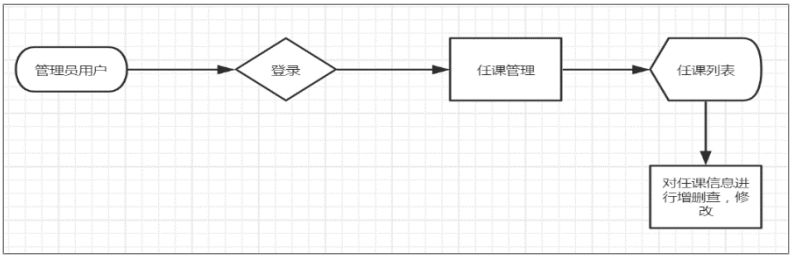




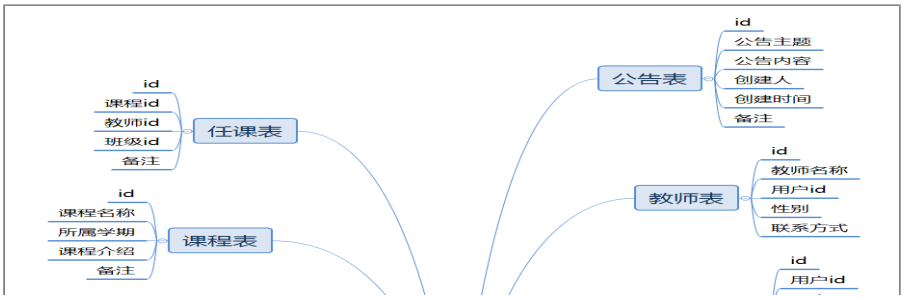
管理员用后登录，判断成功后可进入公告管理，获得公告列表，可按公告标题查询公告信息，对公告信息进行添加，修改，删除。

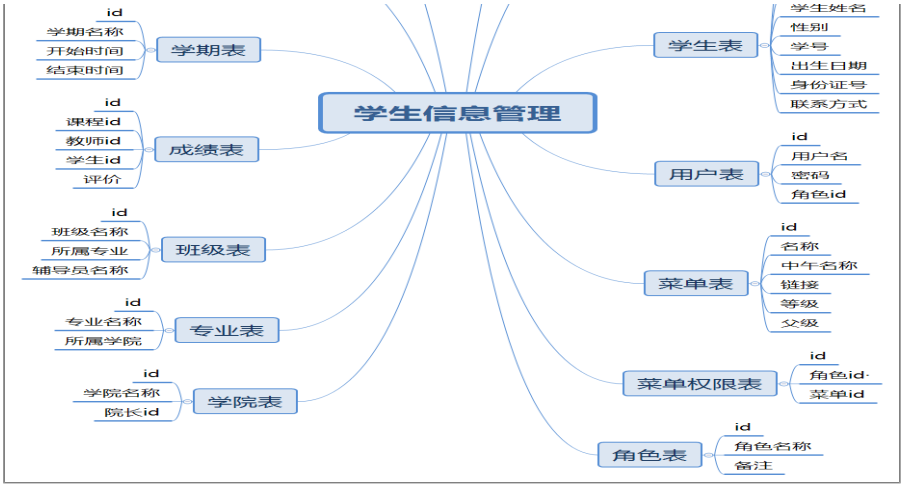


管理员用后登录，判断成功后可进入学期管理，获得学期列表，可按学期名称查询学期信息，对学期信息进行添加，修改，删除



管理员用后登录，判断成功后可进入课程管理，获得课程列表，可按课程名称查询课程信息，对课程信息进行添加，修改，删除。





管理员用后登录，判断成功后可进入任课管理，获得任课列表，可按课程名称以及教师查询任课信息，对任课信息进行添加，修改，删除。

4.3.数据库设计

4.5.1术语定义

表格“备注”列表说明：“*”表示唯一主键，要做索引，“1++”表示值为系统自增，步长为1。

T_ :表名前缀，表示table_

V_ ： 视图前缀，表示view_

4.5.2表详细设计

用户信息表T_User:

表4.1 用户信息表

字段名称	字段标识	类型	长度	备注
用户信息ID	userID	int		*
用户名	account	nvarchar	80	
密码	pwd	nvarchar	40	
用户状态	usertype	bit	1	
角色id	roleIID	int		fk
是否删除	isdel	int		

角色信息表T_RoleB:

表4.2 角色信息表

字段名称	字段标识	类型	长度	备注

角色ID	roleID	int		* 1++
角色名称	rolename	nvarchar	80	
备注	remark	ntext		
是否删除	isdel	int		

角色权限表T_RoleSystemModel:

表 4.3 角色权限表

字段名称	字段标识	类型	长度	备注
ID	id	int		* 1++
角色ID	roleID	int		fk
菜单id	sysid	int		fk
是否删除	isdel	int		

菜单表T_SystemModel:

表4.4 菜单表

字段名称	字段标识	类型	长度	备注
ID	id	int		* 1++
名称	name	nvarchar	80	
中文名称	chinesename	nvarchar	80	
链接	navurl	nvarchar	80	
等级	deepth	int		
父级	parentid	int		

序号	displayorder	int		
图标链接	imageurl	nvarchar	80	
是否删除	isdelete	int		

日志信息表T_SystemLog:

表4.5 日志信息表

字段名称	字段标识	类型	长度	备注
ID	id	int		* 1++
类型	opertype	nvarchar	80	
操作	description	nvarchar	80	
操作地址	opermethod	nvarchar	80	
请求IP	requestip	nvarchar	80	
异常代码	exceptioncole	nvarchar	80	
异常详细信息	exceptiondetail	nvarchar	80	
操作用户	params	nvarchar	80	
创建时间	createdate	datetime		
创建人	createby	nvarchar	80	

学生信息表T_Student:

表4.6 学生信息表

字段名称	字段标识	类型	长度	备注
ID	studentid	int		* 1++
用户id	userID	nvarchar		fk

学生姓名	stuName	nvarchar	80	
学生性别	stusex	nvarchar	10	
学生学号	stunumber	nvarchar	80	
出生日期	stubirthday	datetime	80	
身份证号	stucard	nvarchar	80	
邮箱	email	nvarchar	80	
联系方式	phone	nvarchar	80	
班级	classid			
是否删除	isdelete	int		

教师信息表T_ Teacher:

表4.7 教师信息表

字段名称	字段标识	类型	长度	备注
ID	teacherid	int		* 1++
用户id	userID	nvarchar		fk
教师姓名	teaName	nvarchar	80	
教师性别	teasex	nvarchar	10	
联系方式	teaphone	nvarchar	80	
确认状态	auditstatus	bit	1	
是否删除	isdelete	int		

公告表T_ Notice:

表4.8 公告表

字段名称	字段标识	类型	长度	备注
ID	noticeid	int		* 1++
公告主题	noticeName	nvarchar	80	
公告内容	noticecontent	text		
备注	remarks	text		
创建人	userID	nvarchar		fk
创建时间	createdate	datetime		
是否删除	isdelete	int		

学院表T_ College:

表4.9 学院表

字段名称	字段标识	类型	长度	备注
ID	collegeid	int		* 1++
学院名称	collegeName	nvarchar	80	
院长	userID	nvarchar		fk
是否删除	isdelete	int		

专业表T_ major:

表4.10 专业表

字段名称	字段标识	类型	长度	备注
ID	majorid	int		* 1++

班级名称	majorName	nvarchar	80	
所属学院	collegeid	int		fk
科长	userID	nvarchar		fk
是否删除	isdelete	int		

班级表T_ Class:

表4.11 班级表

字段名称	字段标识	类型	长度	备注
ID	classid	int		* 1++
班级名称	className	nvarchar	80	
所属专业	majorid	int		fk
辅导员	userID	nvarchar		fk
是否删除	isdelete	int		

学期表T_ Semester:

表4.12 学期表

字段名称	字段标识	类型	长度	备注
ID	semesterid	int		* 1++
学期名称	semesterName	nvarchar	80	
开始时间	startdate	datetime		
结束时间	enddate	datetime		
是否删除	isdelete	int		

课程表T_ Curriculum:

表4.13 课程表

字段名称	字段标识	类型	长度	备注
ID	curriculumid	int		*
课程名称	curriculumName	nvarchar	80	
所属学期	semesterid	int		fk
课程介绍	introduce	Text		
备注	remarks	text		
是否删除	isdelete	int		

任课表T_ Course:

表4.14 任课表

字段名称	字段标识	类型	长度	备注
ID	courseid	int		*
课程id	curriculumid	int		fk
教师id	teacherid	int		fk
班级id	classid	int		fk
备注	remakes	text		
是否删除	isdelete	int		

成绩表T_ Sheet:

表4.15 成绩表

字段名称	字段标识	类型	长度	备注

ID	sheetid	int		*
任课id	courseid	int		fk
学生id	studentid	int		fk
分数	fraction	doeble		
评价	curriculumName	nvarchar	80	
是否删除	isdelete	int		

4.5.3视图详细设计

用户视图V_ AdminUser:

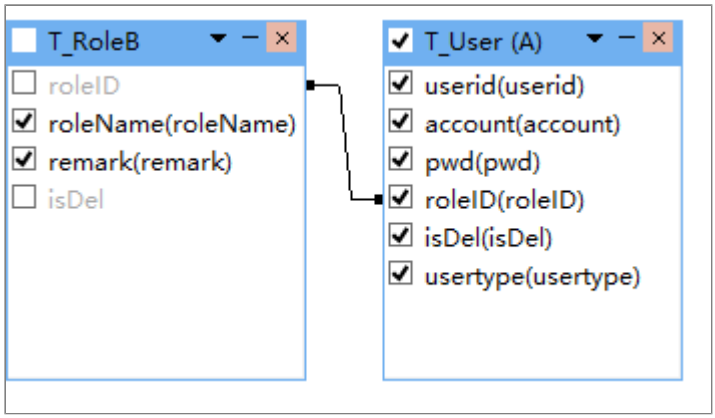
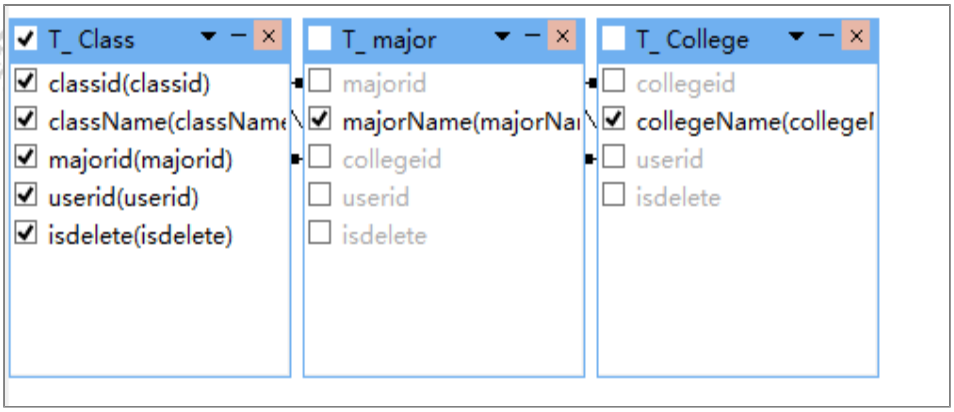


表4.16 用户视图表

字段名称	字段标识	类型	长度	备注
用户信息ID	userID	int		*
用户名	account	nvarchar	80	
密码	pwd	nvarchar	40	
用户状态	usertype	bit	1	

角色id	roleIID	int		
是否删除	isdel	int		
角色名称	rolename	nvarchar	80	
备注	remark	ntext		

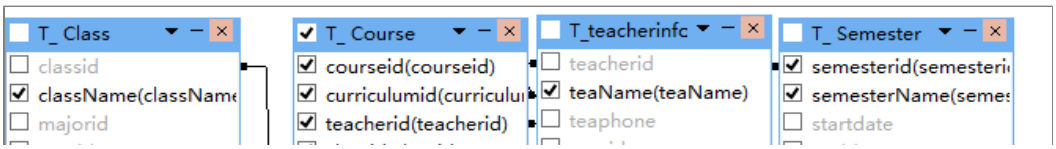


班级视图V_ class:

表4.17 班级视图表

字段名称	字段标识	类型	长度	备注
ID	classid	int		* 1++
班级名称	className	nvarchar	80	
所属专业	majorid	int		
辅导员	userID	nvarchar		
专业名称	majorName	nvarchar	80	
学院名称	collegeName	nvarchar	80	

任课视图V_Course:



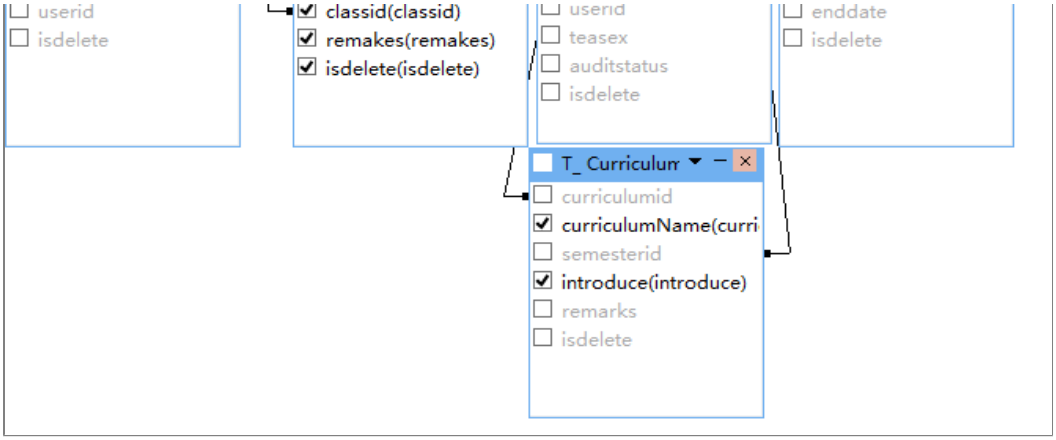
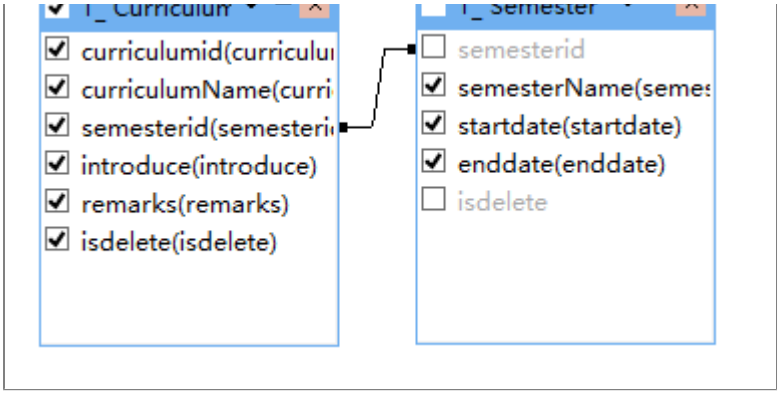


表4.18 任课视图表

字段名称	字段标识	类型	长度	备注
ID	courseid	int		*
课程id	curriculumid	int		
教师id	teacherid	int		
班级id	classid	int		
备注	remakes	text		
是否删除	isdelete	int		
班级名称	className	nvarchar	80	
教师姓名	teaName	nvarchar	80	
ID	semesterid	int		* 1++
学期名称	semesterName	nvarchar	80	
课程名称	curriculumName	nvarchar	80	
课程介绍	introduce	Text		



课程视图V_Curriculum:

表4.19 课程视图表

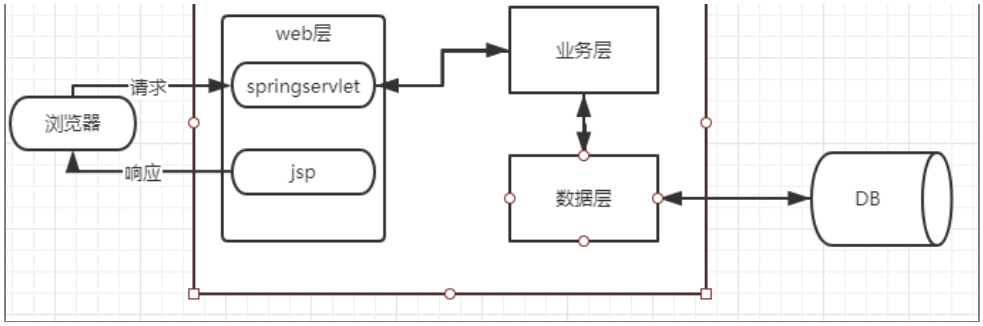
字段名称	字段标识	类型	长度	备注
ID	curriculumid	int		*
课程名称	curriculumName	nvarchar	80	
所属学期	semesterid	int		
课程介绍	introduce	Text		
备注	remarks	text		
是否删除	isdelete	int		
学期名称	semesterName	nvarchar	80	
开始时间	startdate	datetime		
结束时间	enddate	datetime		

第五章、详细设计

5.1

5.1.系统层次设计

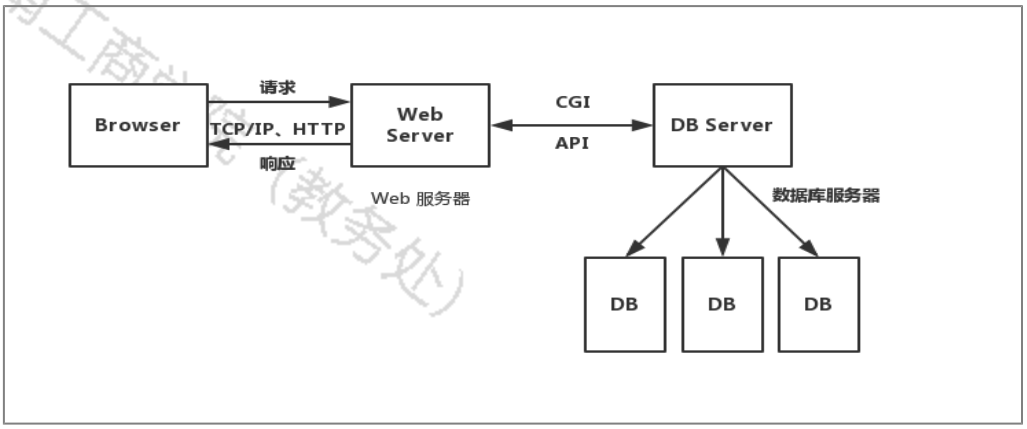
系统分层：是对系统中的各种类进行统一的分组。其优点在于，将层与层之间的依赖降到最低，可以将一层当作一个有机整体并且可以替换某一层的具体实现，更有利于标准化工作。



系统分层的常用方法：

5.1.1三层服务：用户层、业务层、数据层；

5.1.2B/S架构的模式为：表示层、业务层、服务接口层、数据链接层；



5.2.系统项目规范

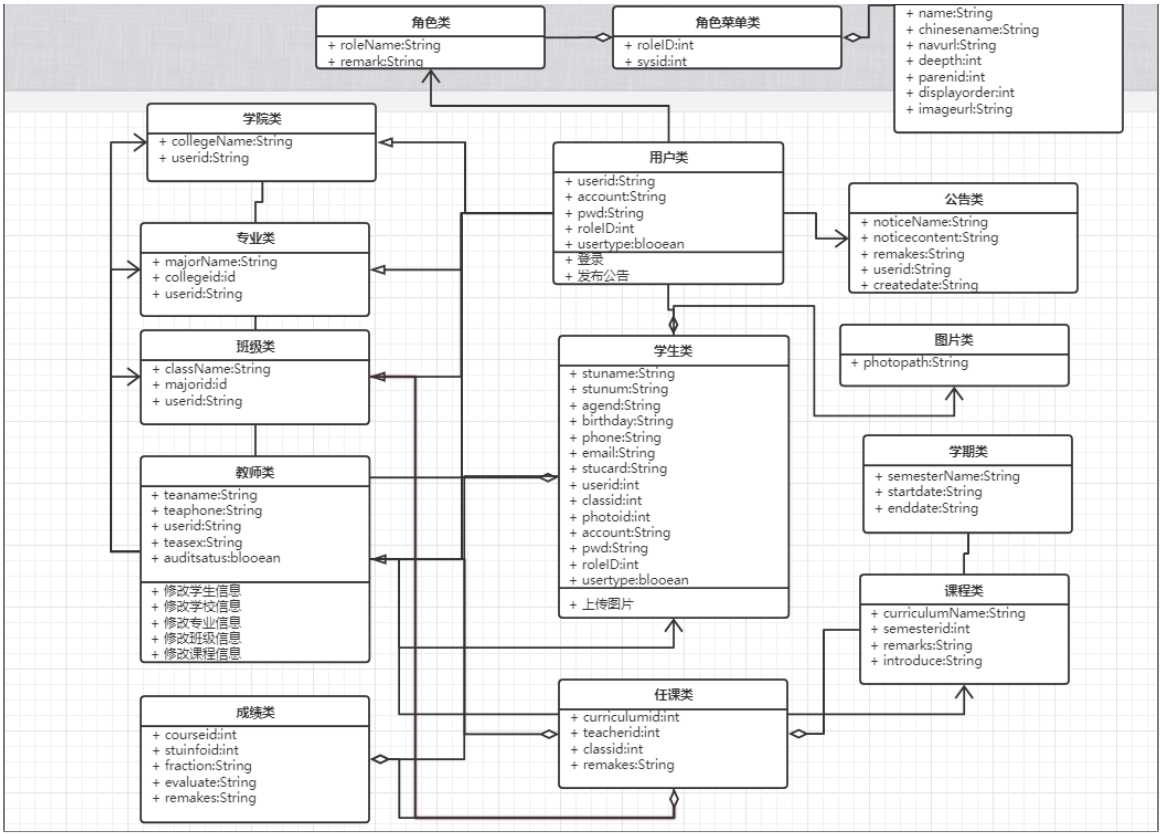
5.2.1项目文件命名规范

该系统内的项目文件命名规范如下表所示：

表5.1 项目文件命名规范表

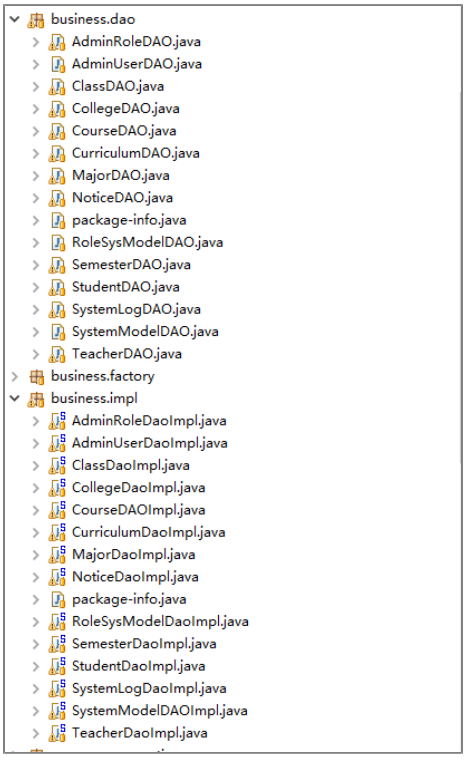
标识符	说明
T实体类名	表实体类，如T_AdminUser表的实体类为TAdminUser
V实体类名	视图实体类，如V_AdminUser视图的实体类为VAdminUser
业务逻辑类名DAO	业务逻辑接口类,如AdminUserDao
业务逻辑类名Impl	业务逻辑实现类，如AdminUserDaoImpl
控制类类名controller	业务控制类，如后台用户管理的控制类AdminUserController

5.3.系统功能类图设计



类图既UML图，是在开发系统时以面向对象的方式对各种类型的系统进行描述，是一种通用的建模语言，适用与系统的各个开发阶段，可对具有静态和动态行为的系统进行建模，从需求规格描述直到系统完成后的测试与维护。

5.4. 系统业务接口设计



系统业务接口根据功能模块进行设计，表中例举出需要实现的功能，且会详细说明接口实现所需要的参数与返回的结果。接口类设计如图所示：

AdminUserDAO（后台用户信息管理业务类接口）：

表5.2后台用户信息管理业务类接口表

接口名	参数	返回值	描述
getaAdminUserList	wherecondition、page、pageSize（查询条件、当前页、每页条数）	成功返回List<VAdminUser>对象，失败返回null	用户信息列表(带分页)
getaAdminUserList	roleID（角色id）	成功返回List<TroleB>对象，失败返回null	根据角色查询用户列表
getaAdminUserList		返回List<VAdminUser>	返回所有用户列表
getAdaminUserAmount	wherecondition（查询条件）	成功返回int，失败返回0	返回查询到的信息数量
addAdminUser	Tuser（用户列表实体类）	成功返回true对象，失败返回false	添加用户
login	VAdminUser（查询条件）	成功返回VAdminUser对象，失败则null	用户登录
delAdminUser	Tuser（用户表实体类）	成功返回true对象，失败返回false	删除用户信息
updatePwd	userid,pwd（用户名，密码）	成功返回true对象，失败返回false	修改用户密码
changeState	userid(用户名)	成功返回true对象，失败返回false	修改用户状态
getuser	userid(用户名)	成功返回Tuser对象，失败返回null	根据用户名查询信息
update	Tuser（用户表实体类）	成功返回true对象，失败返回false	修改用户信息

AdminRoleDAO（角色管理业务类接口）：

表5.3角色管理类接口表

接口名	参数	返回值	描述
getRolelist	Opreation（查询条件）	成功返回List<TroleB>，失败返回null	查询用户角色信息

delAdminRole	TroleB（角色实体类）	成功返回true，失败返回false	删除角色信息
addAdminRole	TroleB（角色实体类）	成功返回true，失败返回false	添加角色信息
edlAdminRole	TroleB（角色实体类）	成功返回true，失败返回false	修改角色信息

CollegeDAO（后台学院信息管理业务类接口）：

表5.4 后台学院信息管理业务类接口表

接口名	参数	返回值	描述
getCollegeList	wherecondition、page、pageSize（查询条件、当前页、每页条数）	成功返回List<TCollege>对象，失败返回null	学院信息列表(带分页)
getCollegeList	wherecondition（查询条件）	成功返回int，失败返回0	返回查询到的信息数量
getCollegeList		成功返回List<TCollege>对象，失败返回null	返回所有学院信息列表
addCollege	TCollege（学院实体类）	成功返回true，失败返回null	添加学院信息
delCollege	TCollege（学院实体类）	成功返回true，失败返回null	删除学院信息
update	TCollege（学院实体类）	成功返回true，失败返回null	修改学院信息

NoticeDAO（后台公告信息管理业务类接口）：

表5.5后台公告信息管理业务类接口表

接口名	参数	返回值	描述
getNoticeList	wherecondition、page、pageSize（查询条件、当前页、每页条数）	成功返回List< TNotice>对象，失败返回null	公告信息列表(带分页)
getNoticeList	wherecondition（查询条件）	成功返回int，失败返回0	返回查询到的信息数量
addNotice	TNotice（公告实体类）	成功返回true，失败返回null	添加公告信息

delNotice	TNotice（公告实体类）	成功返回true，失败返回null	删除公告信息
update	TNotice（公告实体类）	成功返回true，失败返回null	修改公告信息

CurriculumDAO（课程管理业务类接口）：

表5.6课程管理类接口表

接口名	参数	返回值	描述
getVcurrList	wherecondition、page、pageSize（查询条件、当前页、每页条数）	成功返回List< VCurriculum >对象，失败返回null	课程信息列表(带分页)
getVcurrList	wherecondition（查询条件）	成功返回int，失败返回0	返回查询到的信息数量
getVcurrList		成功返回List< VCurriculum >对象，失败返回null	返回所有课程信息列表
addTCurr	TCurriculum（课程实体类）	成功返回true，失败返回null	添加课程信息
delTCurr	TCurriculum（课程实体类）	成功返回true，失败返回null	删除课程信息
update	TCurriculum（课程实体类）	成功返回true，失败返回null	修改课程信息

CourseDAO（后台任课信息管理业务类接口）：

表5.7 后台任课信息管理业务类接口表

接口名	参数	返回值	描述
getCourseList	wherecondition、page、pageSize（查询条件、当前页、每页条数）	成功返回List< VCourse >对象，失败返回null	任课信息列表(带分页)
getCourseList	wherecondition（查询条件）	成功返回int，失败返回0	返回查询到的信息数量
getCourseList		成功返回List< VCourse >对象，失败返回null	返回所有任课信息列表
getCourseL		成功返回List< VCourse >对象，失败	根据条件返回任课信息

ist	teacherid,classid(教师id,班级id)	返回null	列表
getVsheetList	stuinfoid,ourseid(教师id,班级id)	成功返回List< Vsheet >对象，失败返回null	根据条件返回成绩信息列表
addTCourse	TCourse（任课实体类）	成功返回true，失败返回null	添加任课信息
delTCourse	TCourse（任课实体类）	成功返回true，失败返回null	删除任课信息
update	TCourse（任课实体类）	成功返回true，失败返回null	修改任课信息

StudentDAO（学生管理业务类接口）：

表5.8 学生管理类接口表

接口名	参数	返回值	描述
getVstuList	wherecondition、page、pageSize（查询条件、当前页、每页条数）	成功返回List< Vstudent >对象，失败返回null	学生信息列表(带分页)
getVstuList	wherecondition（查询条件）	成功返回int，失败返回0	返回查询到的信息数量
getVstuList	wherecondition（查询条件）	成功返回List< Vstudent >，失败返回null	返回查询到的信息数量
getsheetList	wherecondition、page、pageSize（查询条件、当前页、每页条数）	成功返回List< Vsheet >对象，失败返回null	学生成绩信息列表(带分页)
getsheetList	wherecondition（查询条件）	成功返回int，失败返回0	返回查询到的信息数量
getVstudentList	classid(班级id)	成功返回List< Vstudent >对象，失败返回null	根据班级查询学生信息
addsheet	TSheet（学生成绩实体类）	成功返回true，失败返回null	添加学生成绩信息
upsheet	TSheet（学生成绩实体类）	成功返回true，失败返回null	修改学生成绩信息
getSheet	id（学生成绩id）	成功返回TSheet，失败返回null	获取学生成绩信息

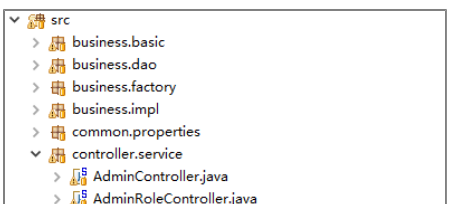
getstu	id（学生id）	成功返回TStuinfo，失败返回null	获取学生成绩信息
addStu	TStuinfo（学生实体类）	成功返回true，失败返回null	添加学生信息
delStu	TStuinfo（学生实体类）	成功返回true，失败返回null	删除学生信息
update	TStuinfo（学生实体类）	成功返回true，失败返回null	修改学生信息

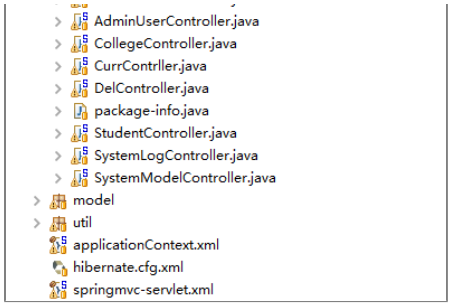
TeacherDAO（后台教师信息管理业务类接口）：

表5.9 后台教师信息管理业务类接口表

接口名	参数	返回值	描述
getteacherList	wherecondition、page、pageSize（查询条件、当前页、每页条数）	成功返回List< VTeacher >对象，失败返回null	教师信息列表(带分页)
getteacherList	wherecondition（查询条件）	成功返回int，失败返回0	返回查询到的信息数量
getteacherList		成功返回List< VTeacher >对象，失败返回null	返回所有教师信息列表
getVteacherList	userid（用户名）	成功返回List< VTeacher >对象，失败返回null	根据用户名查询教师
addTteacher	Tteacher（教师实体类）	成功返回true，失败返回null	添加教师信息
delTteacher	Tteacher（教师实体类）	成功返回true，失败返回null	删除教师信息
changeState	Teacherid（教师id）	成功返回true，失败返回null	修改教师状态
update	TCurriculum（教师实体类）	成功返回true，失败返回null	修改教师信息

5.5.系统交互动作接口设计





系统交互动作接口设计信息如下所示，在控制类说明、方法名和方法功能中都有详细的文字描述。

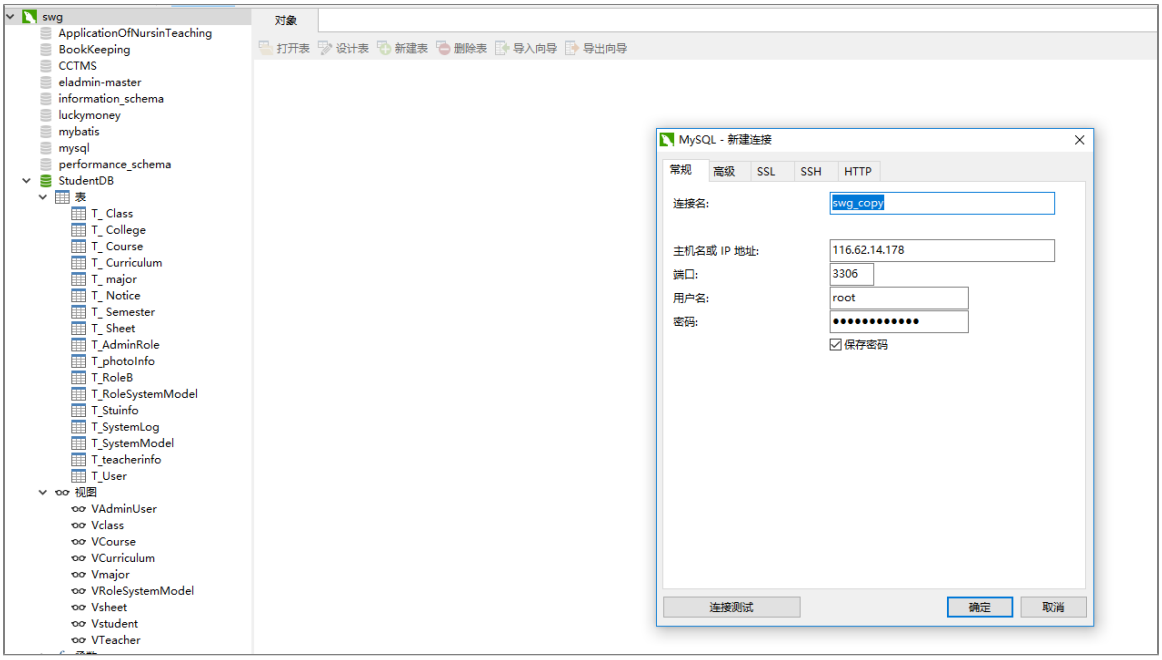
表5.10 控制类表

控制类名称	控制类说明	请求方法名	方法功能
AdminControllerr url=”/admin2”	管理员用户处理类	getNoticeList (HttpServletRequest request, intpage,intlimit,tringnoticename, HttpServletResponseresponse, Model model) url=”/getnotice”	公告信息分页
		addNotice(String noticeName,String noticecontent,String remarks,HttpServletRequest request,HttpServletResponse response, Model model) url=” deleteAdminRole”	添加公告信息
		getTeacherList (HttpServletRequest request, intpage,intlimit,tringnoticename, HttpServletResponseresponse, Model model) url=”/getnotice”	教师信息分页
		Addteacher(String teaName,String teaphone,String userid,String teasex, HttpServletRequest request, HttpServletResponse response, Model model) url=” addteacher”	添加教师信息
		changeUser(String userid, HttpServletRequest request,HttpServletResponse response, Model model) url=” changeuser”	修改用户状态
		GetLoacdtear(HttpServletRequestrequest,HttpServletResponse response, Model model) url=” loacdtea”	加载教师下拉框
		getCollegeList(HttpServletRequest request, intpage,intlimit,tringnoticename, HttpServletResponseresponse, Model model) url=”/ getcollege”	学院信息分页

CollegeController url=”/college”	学院信息处理类	addCollege(String collegeName,String userid, HttpServletRequest request, HttpServletResponse response, Model model) url=”/getcollege”	添加学院信息
		upCollege(Integer collegeid,String collegeName,String userid, HttpServletRequest request, HttpServletResponse response, Model model) url=”/upcollege”	修改学院信息
		getMajorList(HttpServletRequest request, intpage,intlimit,tringnoticename, HttpServletResponseresponse, Model model) url=”/getmajor”	专业信息分页
		addMajor(String majorName,String userid, Integer collegeid, HttpServletRequest request, HttpServletResponse response, Model model) url=”/addmajor”	添加专业信息
		getclass(HttpServletRequest request, intpage,intlimit,tringnoticename, HttpServletResponseresponse, Model model) url=”/getclass”	班级信息分页
		addcalss(String calssName,String userid, Integer majorid, HttpServletRequest request, HttpServletResponse response, Model model) url=”/addclass”	添加班级信息
		GetLoacduser(HttpServletRequestrequest,HttpServletResponse response, Model model) url=” loacduser”	加载用户下拉框
		getcurrList(HttpServletRequest request, intpage,intlimit,tringnoticename, HttpServletResponseresponse, Model model) url=”/getcurr”	课程信息分页
		addcurr(String curriName,Integer semesterid,String introduce, String remarks, HttpServletRequest request, HttpServletResponse response, Model model) url=”/addcurri”	添加课程信息

CurrContr ller url=”/ curr	课程信息 处理类	GetLoacdcurr(HttpServletRequestrequest,HttpServletResponse response, Model model) url=”/ loacdcurr”	加载课程 下拉框
		getcourseList(HttpServletRequest request, intpage,intlimit,tringnoticename, HttpServletResponseresponse, Model model) url=”/ getcourse”	任课信息 分页
		getSemesterList(HttpServletRequest request, intpage,intlimit,tringnoticename, HttpServletResponseresponse, Model model) url=”/ getsemester”	学期信息 分页
		addcourseUser(Integer curriculumid,Integer teacherid,Integer classid, Stringremakes, HttpServletRequestrequest,HttpServletResponse response, Model model) url=”/ addcourse”	添加任课 信息
		GetLoacdcurr(HttpServletRequestrequest,HttpServletResponse response, Model model) url=”/ loacdcurr”	学期信息 下拉框
		GetLoacdsemester(HttpServletRequest request,Integer classid,HttpServletResponse response, Model model) url=”/ loacdseme”	根据班级 获取课程

5.6.服务器设计与实现

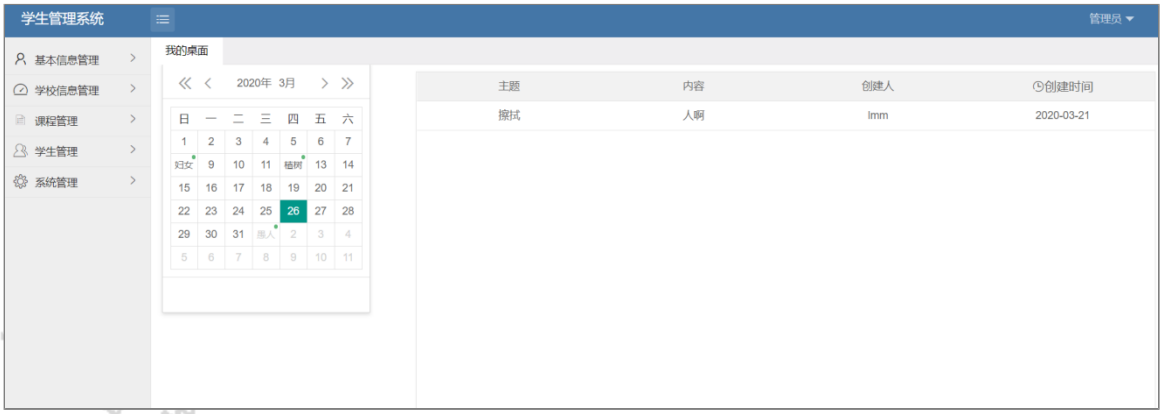


本系统服务器部署于阿里云内，在电脑内下载navicat,在navicat中新建MySql连接,在链接属性中输入连接名，IP地址，端口，用户名，密码。如图所示：

5.7.

系统UI设计

用户登录后，首页左边包含用户登录判断权限后所能操作的信息管理，页面内显示日期的备注与记录、最新发布的公告信息与待处理的信息。如下图所示：

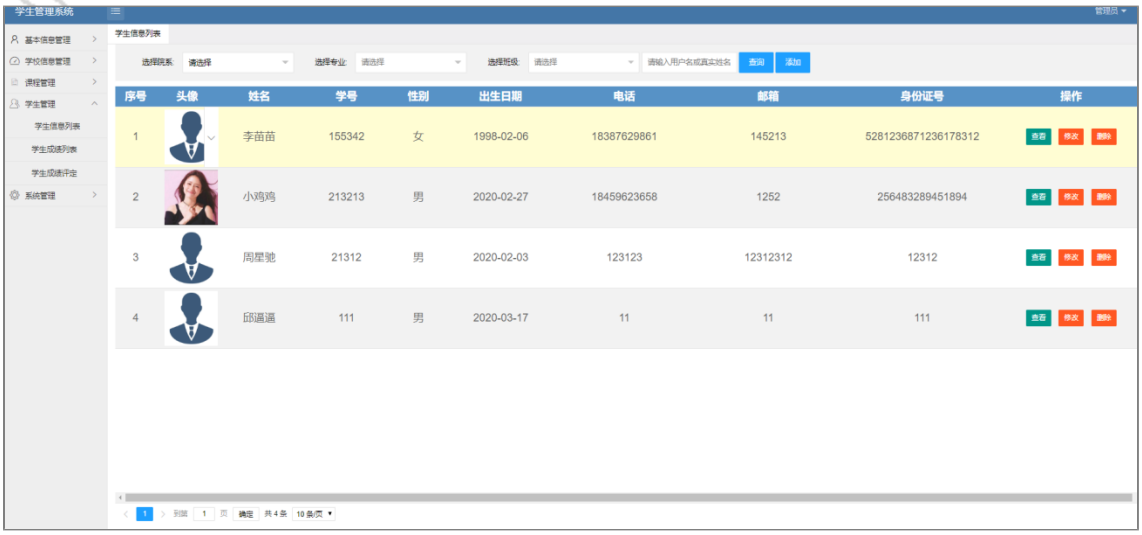


系统功能模块只允许管理员角色操作，功能包括：用户管理进行用户信息的按条件查询展示列表，对用户信息进行添加、修改、删除以及对用户状态的启用与禁用，角色管理是对角色信息添加、修改、删除，菜单管理是对菜单信息添加、修改与删除，角色权限管理是对角色进行条件查询，然后进行角色与菜单权限的授权与禁用，系统日志管理是查看系统各种日志信息以及删除。如下图所示：

基本信息管理模块允许管理员、院长、科长角色操作，功能包括公告管理与教师管理，公告信息管理进行公告信息的添加、修改与删除，教师信息管理进行教师信息、添加、修改、删除，以及对教师用户的状态进行开启与授权。如下图所示：

学校管理模块允许管理员、院长、科长角色操作，功能包括学院管理、专业管理、班级管理，学院信息管理进行学院的添加、修改和删除，专业信息管理进行专业信息的添加、修改和删除，班级信息管理进行班级信息的添加、修改和删除。如下图所示：

课程管理模块允许管理员、院长、科长、辅导员角色操作，功能包括学期管理、课程管理、任课管理，学期信息管理进行学期信息添加、修改、删除，课程信息管理进行课程信息的添加、修改、删除，任课管理是对授权教师与课程信息的绑定。如图所示：



学生管理模块允许院长、科长、辅导员、任课教师角色操作，功能包括：学生信息管理是按照学院、专业、班级进行学生个人信息的汇总与统计，查询后业务员可对学生信息进行修改与删除，学生成绩管理是以学院、专业、班级以及课程对学生成绩信息的汇总与统计，查询后可查看详细信息，学生成绩评定是任课教师对任课班级内的学生进行打分与评价。如下图所示：

学生登录系统后，可查看系统内管理员发出的公告信息，以及查看和修改自己的个人信息，能够按照学期查看个人的成绩信息。如下图所示：

第六章、系统的实现与测试

6.1

6.1.数据层的实现

本系统的数据库采用阿里云服务器上的Mysql，使用navicat进行连接，链接名：swg，IP：116.62.14.178，建立数据库：StudentDB，运行数据库SQL文件进行建表，建立基础数据。

6.2.数据连接层的实现

```
        "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<!-- Generated by MyEclipse Hibernate Tools.                   -->
<hibernate-configuration>

    <session-factory>
        <property name="hbm2ddl.auto">update</property>
        <property name="dialect">
            org.hibernate.dialect.MySQLDialect
        </property>
        <property name="connection.url">
            jdbc:mysql://116.62.14.178:3306/StudentDB?characterEncoding=UTF-8
        </property>
        <property name="connection.username">[redacted]/property>
        <property name="connection.password">[redacted]</property>
        <property name="connection.driver_class">
            com.mysql.jdbc.Driver
        </property>
        <mapping resource="model/TroleB.hbm.xml" />
        <mapping resource="model/Trolesystemmodel.hbm.xml" />
        <mapping resource="model/Tuser.hbm.xml" />
        <mapping resource="model/TSystemLog.hbm.xml" />
        <mapping resource="model/TSystemModel.hbm.xml" />
        <mapping resource="model/VRoleSystemModel.hbm.xml" />
        <mapping resource="model/TAdminRole.hbm.xml" />
        <mapping resource="model/VAdminUser.hbm.xml" />
        <mapping resource="model/TNotice.hbm.xml" />
        <mapping resource="model/TCollege.hbm.xml" />
        <mapping resource="model/TClass.hbm.xml" />
        <mapping resource="model/Tmajor.hbm.xml" />
        <mapping resource="model/Vmajor.hbm.xml" />
        <mapping resource="model/Vclass.hbm.xml" />
        <mapping resource="model/TStuinfo.hbm.xml" />
        <mapping resource="model/Tsemester.hbm.xml" />
        <mapping resource="model/TCurriculum.hbm.xml" />
        <mapping resource="model/VCurriculum.hbm.xml" />
```


该系统数据连接技术采用Hibernate，hibernate.cfg.xml为数据连接层的配置文件。

6.3.基于Hibernate的数据库

```
package business.basic;

import java.util.List;

/**
 * 基于hibernate技术实现的BaseDAO基类
 *
 * @author swg
 * @since 2019-12-19
 * @version 2019-4-20
 */
public interface iHibBaseDAO {
    /**
     * 实现单个的瞬时实体类对象的持久化操作（添加对象到数据库中）该hibernate添加方法仅用于对单个的瞬时对象进行持久化操作
     *
     * @param Object
     *            任意的Hibernate的表映射实体类对象
     * @return Object 主键id. null则表示添加失败
     */
    public Object insert(Object obj);

    /**
     * 实现批量的瞬时实体类对象的持久化操作（添加对象到数据库中）该hibernate添加方法仅用于对单个的瞬时对象进行持久化操作
     *
     * @param List
     *            <Object> 任意的Hibernate的表映射实体类对象列表
     * @return boolean true成功, false失败
     */
    public boolean insert(List<Object> list);

    /**
     * 使用无参数的hql实现查询的. 也将返回基于hql中查询对象的List数组对象
     * @param String 符合hql结构的无参数查询语句
     * @return object  失败返回null
     */
    public Object selectbyhq(String hql) ;
```

本系统采用Hibernate框架来连接数据库，如图所示：

6.4.数据操作层的实现

数据操作层采用hibernate框架，封装了对数据的增删查改的方法，可以自动生成数据库语句，自动执行，便于业务逻辑层的调用。

类名：HibBaseDAO

```
public class iHibBaseDAOImpl implements iHibBaseDAO {
    public static final int INSERT = 1;// 代表添加操作
    public static final int UPDATE = 2;// 代表更新操作
    public static final int DELETE = 3;// 代表删除操作

    // private static final Log log=LogFactory.getLog(iHibBaseDAOImpl.class);

    @Override
    public Object insert(Object obj) {// obj必须是符合hibernate的pojo对象

        Session session = HibSessionFactory.getSession();
```

```
        Transaction tx = null;
        try {
            tx = session.beginTransaction();// 开始事务
            Serializable key = session.save(obj);
            tx.commit();// 持久化操作
            session.close();
            return key;
        } catch (Exception e) {
            // TODO Auto-generated catch block
            // log.error(LogUtil.error("Basic.iHibBaseDAOImpl.insert",
            // e));//向日志输出error级别的日志信息
            e.printStackTrace();
            if (tx != null)
                tx.rollback();// 撤销
            if (session != null)
                session.close();
        }
        return null;
    }
```

示例代码：

6.5.数据操作层的调用实现

业务层根据业务功能模块先定义DAO的各种业务接口，再实现以DAO为父类的Impl类。

类名：AdminuserDaoImpl

```
@Component("adminuserdao")
public class AdminUserDaoImpl implements AdminUserDAO {
    private iHibBaseDAO hdao = null;

    public AdminUserDaoImpl() {
        this.hdao = new iHibBaseDAOImpl();
    }

    @Log(isSaveLog = false)
    @Override
    public List<VAdminUser> getaAdminUserList(String wherecondition, int page,
            int pageSize) {
        String hql = "from VAdminUser";
        if (wherecondition != null && !wherecondition.equals("")) {
            hql += wherecondition;
        }
        hql += " order by userid asc";
        List<VAdminUser> list = hdao.selectByPage(hql, page, pageSize);
        return list;
    }

    @Log(isSaveLog = false)
    @Override
    public int getAdaminUserAmount(String wherecondition) {
        String hql = "select count(userid) from VAdminUser";
        if (wherecondition != null && !wherecondition.equals("")) {
            hql += wherecondition;
        }
        return hdao.selectValue(hql);

    }
```

示例代码：

6.6.实体交换层的实现

实体层用于存储数据对象，进行数据的存储和传输，贯穿与表示层、业务层、数据层，在三层之间进行数据的传递，运营hibernate可以直接逆向生成实体对象，实体对象一般放在model层。

```
<hibernate-mapping>
    <class name="model.Tuser" table="T_User" catalog="StudentDB">
        <id name="userid" type="java.lang.String">
            <column name="userid" length="11" />
            <generator class="assigned"></generator>
        </id>
        <property name="account" type="java.lang.String">
            <column name="account" length="50" not-null="true" />
        </property>
        <property name="pwd" type="java.lang.String">
            <column name="pwd" length="50" not-null="true" />
        </property>
        <property name="roleId" type="java.lang.Integer">
            <column name="roleID" />
        </property>
        <property name="isDel" type="java.lang.Integer">
            <column name="isDel" />
        </property>
        <property name="usertype" type="java.lang.Boolean">
```

```
            <column name="usertype" />
        </property>
    </class>
</hibernate-mapping>
```

生成的实体层的.xml文件Tuser.hbm.xml实例代码：

```
public class Tuser implements java.io.Serializable {

    // Fields

    private String userid;
    private String account;
    private String pwd;
    private Integer roleId;
    private Integer isDel;
    private Boolean usertype;

    // Constructors

    /** default constructor */
    public Tuser() {
    }

    /** minimal constructor */
    public Tuser(String userid, String account, String pwd) {
        this.userid = userid;
        this.account = account;
        this.pwd = pwd;
    }

    /** full constructor */
    public Tuser(String userid, String account, String pwd, Integer roleId,
            Integer isDel, Boolean usertype) {
        this.userid = userid;
        this.account = account;
        this.pwd = pwd;
        this.roleId = roleId;
        this.isDel = isDel;
        this.usertype = usertype;
    }

    // Property accessors

    public String getUserid() {
        return this.userid;
    }

    public void setUserid(String userid) {
        this.userid = userid;
    }

    public String getAccount() {
        return this.account;
    }
```

实体层Tuser类实例代码：

6.7.系统业务层的实现

系统业务层接口调用model层。

类名：AdminUserDAO

```
/**
 * 管理端管理员用户业务接口
 *
 * @author Administrator
 *
 */
public interface AdminUserDAO {

    /**
     * 根据条件获取管理员用户列表
     *
     * @param wherecondition
     * @return List
     */
    public List<VAdminUser> getaAdminUserList(String wherecondition, int page,
            int pageSize);

    /**
     * 根据条件获取管理员用户列表
     *
     * @param wherecondition
     * @return List
     */
    public List<VAdminUser> getaAdminUserList(int roleID);


    /**
     * 根据条件获取符合条件的管理员用户的数量
     *
     * @param wherecondition
```

```
     *                如："userRole = '超级管理员' and userid = 'zhangjs'"
     * @return
     */
    public int getAdaminUserAmount(String wherecondition);

    /**
     * 实现一个管理员用户的添加
     *
     * @param user
     */

    public boolean addAdminUser(Tuser user);
```

示例代码：

6.8.信息业务层的调用实现

类名：DAOFactory

```
public class DAOFactory {
    private static ApplicationContext ctx = null;

    static {
        ctx = new ClassPathXmlApplicationContext("springmvc-servlet.xml");
    }

    /**
     * 得到一个用于管理员业务操作的AdminUserDAO实现类对象
     *
     * @return AdminUserDAO
     */
    public static AdminUserDAO getAdminUserDAO() {
        return (AdminUserDAO) ctx.getBean("adminuserdao");
    }

    /**
     * 得到一个用于管理员角色业务操作的AdminRoleDAO实现类对象
     *
     * @return AdminRoleDAO
     */
    public static AdminRoleDAO getAdminRoleDAO() {
        return (AdminRoleDAO) ctx.getBean("adminroledao");
    }

    /**
     * 得到一个用于系统菜单管理业务操作的SystemLogDAO实现类对象
     *
     * @return SystemLogDAO
     */
    public static SystemLogDAO getSystemLogDAO() {
        return (SystemLogDAO) ctx.getBean("systemlogdao");
    }

    /**
     * 得到一个用于系统菜单管理业务操作的SystemModelDAO实现类对象
     *
     * @return SystemModelDAO
     */
    public static SystemModelDAO getSystemModelDAO() {
        return (SystemModelDAO) ctx.getBean("systemmodeldao");
    }
```

业务层的的调用代码示例：

6.9.控制层的实现

控制层采SpringMVC框架技术，客户端提交请求到DispatcherServlet,再由DispatcherServlet控制器查询HandlerMapping找到出来该请求的Controller,然后将请求提交给Controller，Controller调用业务逻辑处理请求返回给ModelAndView，DispatcherServlet找到它指定的视图，视图将结果显示到客户端。

```
@Controller
@RequestMapping(value = "admin")
public class AdminUserController {
    /**
     * 获取管理员用户列表
     *
     * @param request
     * @param page
     * @param limit
     * @param realname
     * @param roleid
     * @param response
     * @param model
     */
    @RequestMapping(value = "getuser")
    public void getAdminUserList(HttpServletRequest request, int page,
            int limit, String realname, Integer roleid,
            HttpServletResponse response, Model model) {

        AdminUserDAO audao = new AdminUserDaoImpl();
        // 查询条件
        Expression exp = new Expression();

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

            exp.andLeftBraLike("realname", realname, String.class);
            exp.orRightBraLike("userid", realname, String.class);
        }
        if (roleid != null && roleid != 0) {
            exp.andEqu("roleid", roleid, Integer.class);
        }
        String opreation = exp.toString();
        // System.out.println(opreation);
        int allcount = audao.getAdaminUserAmount(opreation);

        List list = audao.getaAdminUserList(opreation, page, limit);

        response.setCharacterEncoding("utf-8");
        response.setContentType("application/json");

        LayuiData laydata = new LayuiData();
        laydata.code = LayuiData.SUCCESS;
        laydata.msg = "执行成功";
        laydata.count = allcount;
        laydata.data = list;
```

```
        Writer out;
```

类名：AdminUserController

示例代码：

6.10.控制层的调用实现

```
$.ajax({
    type : 'get',
    url : '../admin/addadminuser?userid='+addUserName+'&pwd='+pwd+'&roleid='+usertype+'&realname='+realname,
    datatype : 'json',
    success : function(data) {
        if (data.code == "0") {
            layer.confirm(data.msg, {
              btn: ['确定'],
              icon:1
            }, function(){
                table.reload("adminUserid", { //此处是上文提到的 初始化标识id
                    where: {
                        keyword:data.code=='10001'
                    }
                });
                layer.closeAll();
            });
        }else{
            layer.confirm(data.msg, {
              btn: ['确定'],
              icon:2
            });
        }
    },
    error : function() {
        layer.confirm('出现错误，请重试！', {
                icon: 6,
                  btn: ['确定']
            });
    }
});
```

在页面内用户输入想添加的数据或执行相应的操作，调用Ajax[7]，向业务层发送请求，示例代码如下：

6.11.表示层的实现

1)代码

系统表示层主要由html页面和JSP页面组成，主要采用LayUI框架[8]和jQuery技术。Web服务器上面先接到了JSP或html网页的访问请求，通过程序段到ＪＳＰ文件中的ＨＴＭＬ代码，然后把其运行结果返回请求的用户。

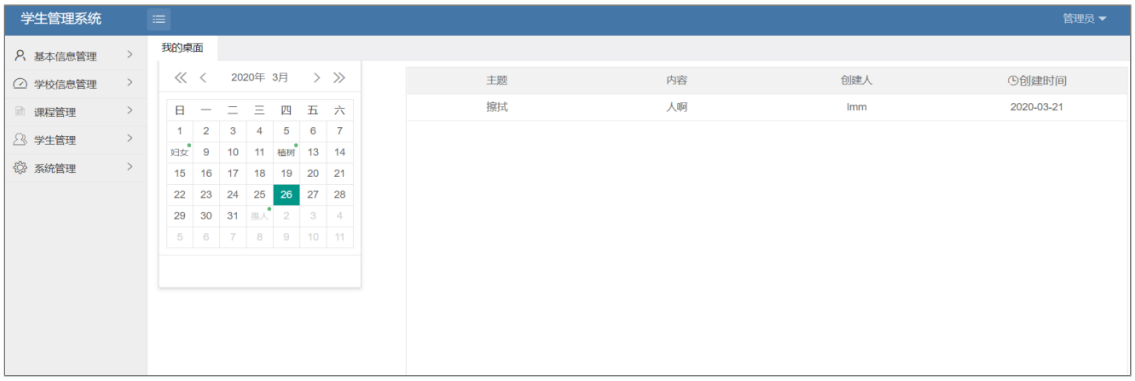
```
<body>
    <!--操作调用内容Start-->
    <div id="adminuserdetail" class="adminuserdetail">

          <div class="artTypeLayer">
                <form class="layui-form" action="">
                <div class="layui-form-item">
                      <label class="layui-form-label">用户类型:</label>
                      <div class="layui-input-block">
                        <select id="txtusertype">
                          <option value="">请选择用户类型</option>
                        </select>
                      </div>
                    </div>
                    <div class="layui-form-item">
                        <label class="layui-form-label">用户名:</label>
                        <div class="layui-input-block">
                            <input type="text" name="txtuserid" id="txtuserid"
                                lay-verify="txtuserid" autocomplete="off" placeholder="请输入用户名" class="layui-input">
                        </div>
                    </div>
                    <div class="layui-form-item">
                        <label class="layui-form-label">用户密码:</label>
                        <div class="layui-input-block">
                            <input type="password" name="txtpassword" id="txtpassword"  autocomplete="off" placeholder="请输
                        </div>
                    </div>
                    <div class="layui-form-item">
                        <label class="layui-form-label">真实姓名:</label>
                        <div class="layui-input-block">
                            <input type="text" name="realname" id="textrealname" lay-verify="textrealname" autocomplete="of
                        </div>
                    </div>

                </form>
            </div>
    </div>
    <!--操作调用内容END-->
```

代码示例：

2)



实用结果

6.12. 软件测试用例说明

1)测试目的

检查本系统的功能是否达到预期目标，各个功能是否正常运行。

检测系统运行时页面的美观程度与协调性，是否符合用户的操作习惯与良好的页面效果，是否符合大部分用户需求的页面部署效果，通过测试发现系统的缺陷与不足，然后进一步完善。

2)网站端功能测试用例

网站登录功能测试如表：

表6.1 登录功能测试表

前提条件	输入动作	期望输出	实际显示	结果
输入正确的用户名、密码、验证码	输入“admin”，密码“123456”	跳转网站主界面	跳转成功	成功

输入错误的密码	输入“admin”，密码“123”	提示用户名或者密码输入错误	提示用户名或密码错误	成功
输入未赋予权限的教师用户	输入“lmm”，密码“123456”	提示用户登录失败	提示用户登录失败	成功

网站权限功能测试如表：

表6.2 权限功能测试表

前提条件	输入动作	期望输出	实际显示	结果
输入管理员用户	输入“admin”，密码“123456”	跳转网站主界面，所有权限	跳转网站主界面，所有权限	成功
输入教师用户	输入“lmm”，密码“123456”	跳转网站主界面，教师权限	跳转网站主界面，教师权限	成功
输入学生用户	输入“xjj”，密码“123456”	跳转网站主界面，学生权限	跳转网站主界面，学生权限	成功

任课功能测试如表：

表6.3任课功能测试表

前提条件	输入动作	期望输出	实际显示	结果
输入正确的班级，教师，课程	班级“16级汉语言一班”，教师“lmm”，课程“数学”	添加成功	任课添加成功	成功
输入重复的班级，教师，课程	班级“16级汉语言一班”，教师“lmm”，课程“数学”	教师记录已存在	教师记录已存在	成功

修改教师状态功能测试如表：

表6.4 教师状态功能测试表

前提条件	输入动作	期望输出	实际显示	结果
点击教师启用	点击教师启用	教师启用成功	教师启用成功	成功

学生成绩评定功能测试如表：

表6.5 成绩评定功能测试表

前提条件	输入动作	期望输出	实际显示	结果
教师用户登录	点击成绩表格输入“80”，评定“优秀”	数据保存数据库，无输出	数据保存数据库，无输出	成功

角色功能测试如表：

表6.6角色功能测试表

前提条件	输入动作	期望输出	实际显示	结果
管理员登录进行添加	角色名称“测试”，描述“测试”	角色添加成功	角色添加成功	成功

角色功能测试如表：

表6.7角色功能测试表

前提条件	输入动作	期望输出	实际显示	结果
管理员登录进行修改	查询院长角色，授予“系统管理权限”	授权成功，请刷新页面	授权成功，请刷新页面	成功
管理员登录进行修改	查询院长角色，取消授予“系统管理权限”	取消授权成功，请刷新页面	取消授权成功，请刷新页面	成功

6.13.测试报告

经过多次各种条件的测试，证实本系统已达到预期目标，基本的功能模块与工作流程都可正常实现，经过各种试错，测试，调整最终是本系统更适合高校管理者的需求，但还是存在着一些不足与缺陷，后期的开发会进一步完善。

结 论

经过长时间的代码编写与测试，终于完成了本次的毕业设计，从选题、开题，开始设计系统、对系统进行需求分析、概要设计，经历了无数次的报错，修改代码，再试错的过程。直到系统的测试成功，期间做了许多工作，对程序开发积累了很多经验。

在设计过程中我深刻的体验与学习了系统开发的整个过程。前期的需求分析、功能设计、数据库设计、各种角色与角色之间的逻辑关系、办公流程的逻辑非常重要，不要急于编码，前期的设计不完善、不清晰，可能使后期工作变得更复杂。本次毕业设计是对大学四年学习的一次综合能力的锻炼和实践，而且培养了坚强的毅力、耐力，更需要持之以恒，要远超在学生时所需要的。

由于时间和个人能力有限，各个功能模块的实现与逻辑并不完美。往后的开发应该以功能逻辑的完善与各个功能的扩展为主要目标。在整个系统的开发过程中使我意识到必须掌握足够的知识才能在工作中得心应手，适应社会的需要。

致 谢

本文是在向金明指导老师的热情关心和指导下完成的，他渊博的知识和耐心的讲解使我受益匪浅，对顺利完成本课题起到了极大的作用。在此向他表示我最衷心的感谢！

在系统的测试完成过程中，本人在学校与家里得到学校和父母的支持与帮助，让我能够全身心的参与到系统的开发与论文的编写当中，在这里对学校，对父母表示崇高的谢意。

在此，我还要感谢帮助过我写完论文的所有同学，正是由于你们的帮助和支持，我才能克服一个一个的困难和疑感，我的论文才能顺利完成。在这里请大家接受我真诚的谢意！

参考文献

一、中文部分

[1] 秦朝明.学生工作管理信息系统在高校学生管理中的应用[J].学理论,2019(08):136-137.

[2]潘蕊.SSH框架的Web网站设计与实现研究[J].成才之路,2019(36):58-59.

[3] 胡强.MySQL数据库常见问题分析与研究[J].电脑编程技巧与维护,2019(12):91-92.

[4]余炀,曲毅,孙亦乐.基于Apache Tomcat的一站式Java应用服务器解决方案[J].中国金融电脑,2018(01):59-63.

[5]陈华平.民办高职院校学生诚信档案管理信息系统的研建[J].电子世界,2018(24):16-17.

[6]刘巍峰. 基于框架模式的工作流程网站设计与实现[D].吉林大学,2015.

[7]何晶.以SSH框架与jQuery技术为基础的Java-Web开发应用探讨[J].计算机产品与流通,2019(11):104-105.

[8]曹灿,刘志刚.基于SSH和Layui的工程科学前沿与实践系统[J].工业控制计算机,2019,32(02):91-92+96.

二、英文部分

[1] Privacy Act of 1974; System of Records--Migrant Student Information Exchange[J]. The Federal Register / FIND, 2019,84(132).

[2] Hong ZHANG. Research on the Information Management System of University Student Status Archives[P]. Proceedings of the 2019 3rd International Conference on Education, Management Science and Economics (ICEMSE 2019),2019.

- **说明：**

相似片段中"综合"包括：

《中文主要报纸全文数据库》 《中国专利特色数据库》 《中国主要会议论文特色数据库》 《港澳台文献资源》 《图书资源》 《维普优先出版论文全文数据库》 《年鉴资源》 《古籍文献资源》 《IPUB原创作品》

- **声明：**

报告编号系送检论文检测报告在本系统中的唯一编号。

本报告为维普论文检测系统算法自动生成，仅对您所选择比对资源范围内检验结果负责，仅供参考。
