项目二：学生成绩管理系统的设计

1、目的和要求：

通过设计一个实用的数据库应用系统，使学生掌握数据库应用系统的设计方法，并加深对于数据库系统相关概念、原理的理解和掌握。

2、主要内容：

通过对学校日常教学管理中的课程、选课、学生、教师、成绩等相关内容进行分析，完成具有学生管理、成绩管理、课程管理等相关功能的小型数据库管理应用系统。

（1） 根据要求完成有关表的设计（表的设计需要遵循3FN）。

（2） 完成有关表的维护（如学生、班级、课程及成绩表等）。

（3） 完成成绩的浏览查找 。

（4） 按班级、按课程计算平均成绩、最高成绩、最低成绩、及格率。

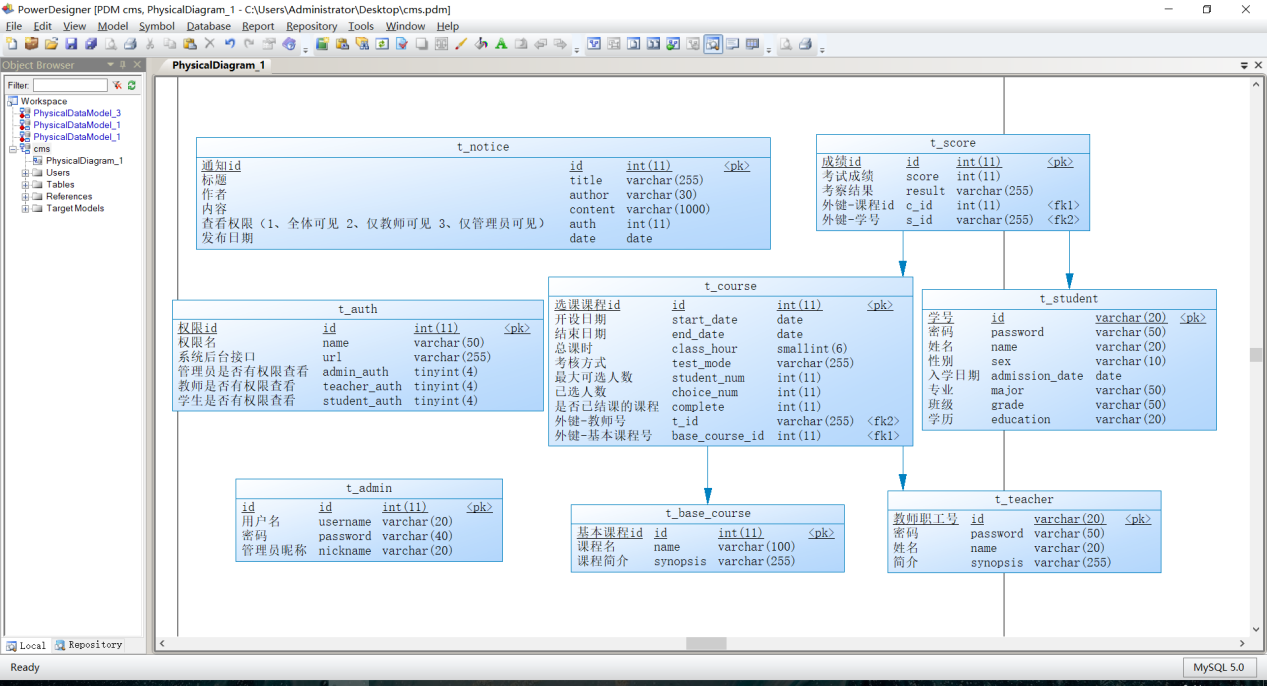
（5） 统计某学生、某学期的所有课程的平均成绩。

（6） 实现简单的用户管理功能。

（7） 完成数据备份与恢复功能。

1. 系统设计与实现

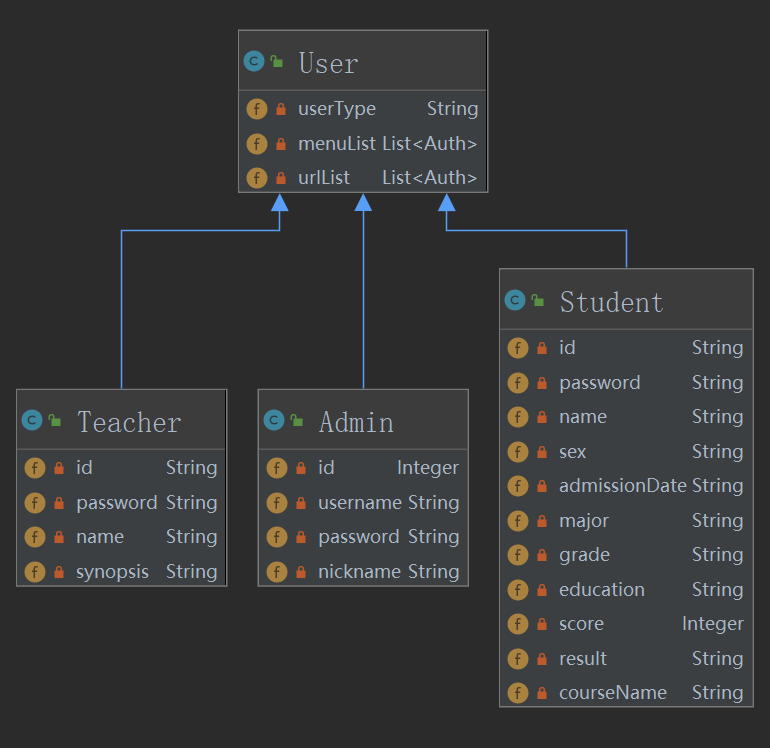
一、ER图



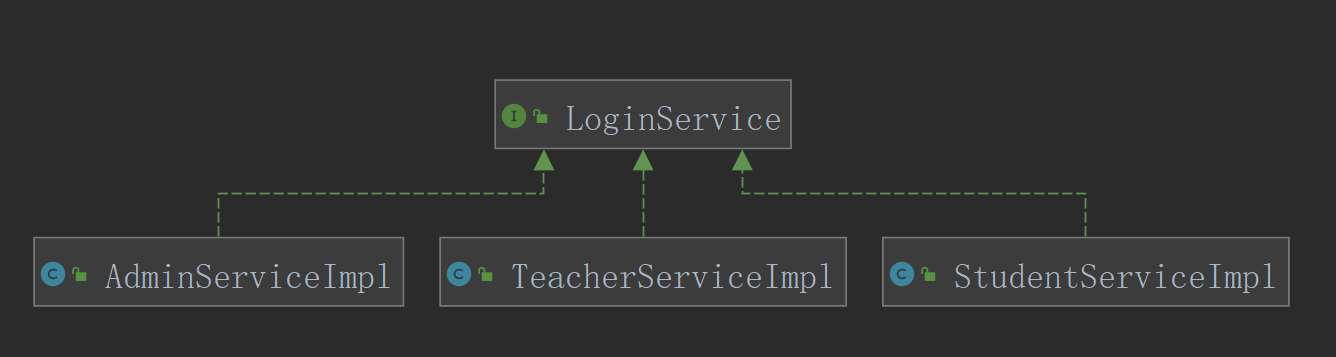
二、接口SQL分析

1. 登录用户权限控制的设计与实现：
2. Auth实体类属性：权限id、名称、需要控制的接口url、3个标志位（管理员、教师、学生是否可访问该页面，0不可访问，1可访问）

（2）实体类的继承关系：



（3）登录逻辑使用**策略模式**：



public interface LoginService {  
 */\*\*  
 \* 登录验证方法  
 \** ***@return*** *\*/* public User loginValidate(String username, String password);  
}

AdminServiceImpl：

@Override  
public User loginValidate(String username, String password) {  
 Admin admin = new Admin();  
 admin.setUsername(username);  
 admin.setPassword(password);  
 admin = selectAdmin(admin);  
 if(admin != null) admin.setUserType("admin");  
 return admin;  
}

StudentServiceImpl

@Override  
public User loginValidate(String username, String password) {  
 Student student = new Student();  
 student.setId(username);  
 student.setPassword(password);  
 student = selectStudent(student);  
 if(student != null) student.setUserType("student");  
 return student;  
}

TeacherServiceImpl：

@Override  
public User loginValidate(String username, String password) {  
 Teacher teacher = new Teacher();  
 teacher.setId(username);  
 teacher.setPassword(password);  
 teacher = selectTeacher(teacher);  
 if(teacher != null) teacher.setUserType("teacher");  
 return teacher;  
}

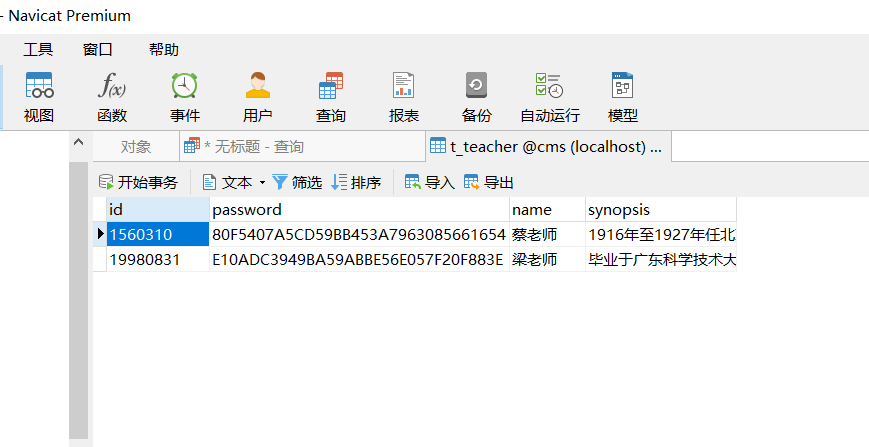
1. 登录逻辑：

//1、比较验证码  
String sessionVerifyCode = (String) session.getAttribute(StrUtil.*VERIFY\_CODE*);  
if (sessionVerifyCode == null || !sessionVerifyCode.equals(verifyCode.toUpperCase()) ) {  
 return StrUtil.*CODE\_ERROR*;  
}

// 策略模式实现  
LoginService loginServeice = null;  
if (userType == 1) {  
 loginServeice = (LoginService) adminServiceImpl;  
} else if(userType == 2) {  
 loginServeice = (LoginService) teacherServiceImpl;  
} else if(userType == 3) {  
 loginServeice = (LoginService) studentServiceImpl;  
}  
//2、获得验证后user对象  
User user = loginServeice.loginValidate(username, password.toUpperCase());  
//若登录成功  
if (user != null) {  
 //3、获取角色对应的可访问页面  
 List<Auth> urlList = authService.getUrlList(user.getUserType());  
 user.setUrlList(urlList);  
 session.setAttribute(StrUtil.*USER*, user);  
 return JSON.*toJSONString*(user);  
}  
return StrUtil.*RESULT\_FALSE*;

2、用户密码md5加密通过前端js实现：





3、用户权限拦截，若未登录重定向登录页，若没有权限则重定向404

public class SecurityInterceptor implements HandlerInterceptor {  
 @Override  
 public boolean preHandle(HttpServletRequest request,  
 HttpServletResponse response, Object handler) throws Exception   
 HttpSession session = request.getSession();  
 User user = (User) session.getAttribute(StrUtil.*USER*);  
 if (user == null) {  
 response.sendRedirect(request.getContextPath()+"/");  
 return false;  
 } else {  
 String url = request.getRequestURI();  
 List<Auth> list = user.getUrlList();  
 for (Auth auth: list) {  
 if (url.indexOf(auth.getUrl()) >= 0) return true;  
 }  
 response.sendRedirect(request.getContextPath()+"/404");  
 }  
 return false;  
 }  
 @Override  
 public void postHandle(HttpServletRequest request,  
 HttpServletResponse response, Object handler,  
 ModelAndView modelAndView) throws Exception {  
 }  
 @Override  
 public void afterCompletion(HttpServletRequest request,  
 HttpServletResponse response, Object handler, Exception ex)  
 throws Exception {  
 }  
}

4、查询该学生可选的课程

（1）Dao接口定义：

List<Course> getCourseListBySid(Pagination<Course> page, int isAll, String searchKey,  
 String id);

1. 思路：

查询结果映射为pojo对象Course；

首先需要课程表 t\_course、教师表t\_teacher、基本课程表t\_base\_course进行等值连接；

开课日期需要必须比当前时间晚

若有传入关键字，则模糊匹配课程名称

必须是未结课的，且是该学生未选修的课程

（3）动态SQL语句：

<select id="getCourseListBySid" resultMap="BaseResultMap">

select

C.id, start\_date, end\_date, class\_hour, test\_mode, student\_num,

choice\_num, B.name as courseName, T.name as teacherName

from t\_course as C, t\_teacher as T, t\_base\_course as B

<where>

C.base\_course\_id = B.id and C.t\_id = T.id and start\_date > CURRENT\_DATE

<if test="param3 != null and param3 != '' ">

and(

(B.name like CONCAT ('%',#{param3},'%') )

or (T.name like CONCAT ('%',#{param3},'%') )

)

</if>

and C.complete = 0

and C.id

<if test="param2 == 1 ">

not

</if>

in ( select c\_id

from t\_score, t\_student as S

where S.id= #{param4} and S.id = t\_score.s\_id )

</where>

<if test="param1 != null">

limit #{param1.start}, #{param1.pageSize}

</if>

</select>

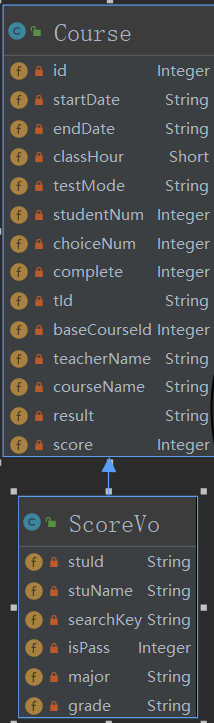
5、用于打印的成绩单查询

（1）Dao接口定义：

List<ScoreVo> getScoreListForExport(Pagination<ScoreVo> page,  
 ScoreVo scoreVo);

（2）思路：

查询结果映射为pojo对象ScoreVo；



成绩表、课程表、学生表、基本课程表、教师表等值连接；

必须是已结课的课程；

若查询参数不为空，根据传入的日期段、课程是否通过、or连接的条件子句（课程名称、专业、分数）进行模糊匹配；

使用limit关键字分页；

1. SQL语句：

<select id="getScoreListForExport" resultMap="ResultMap">

select start\_date, Stu.id as stuId, Stu.name as stuName, major, grade,

Bc.name as courseName, T.name as teacherName, test\_mode, score, result

from t\_score as Sc, t\_course as C, t\_student as Stu, t\_base\_course as Bc, t\_teacher as T

where Sc.c\_id = C.id and Sc.s\_id = Stu.id and C.base\_course\_id = Bc.id

and C.t\_id = T.id and C.complete = 1

<if test="param2 != null">

<if test="param2.startDate != null and param2.startDate != '' ">

and C.start\_date >= #{param2.startDate}

</if>

<if test="param2.endDate != null and param2.endDate != '' ">

and C.start\_date &lt;= #{param2.endDate}

</if>

<if test="param2.isPass == 1">

and (score >= 60 or result = '通过')

</if>

<if test="param2.isPass == 2">

and (60 > score or result = '未通过')

</if>

<if test="param2.searchKey != null and param2.searchKey != '' ">

and (

Stu.major like CONCAT('%', #{param2.searchKey}, '%')

or Stu.grade like CONCAT('%', #{param2.searchKey}, '%')

or Bc.name like CONCAT('%', #{param2.searchKey}, '%')

)

</if>

</if>

<if test="param1 != null">

limit #{param1.start}, #{param1.pageSize}

</if>

</select>

6、选修了该教师该课程的学生列表

（1）Dao接口定义：

List<Student> getStudentListByTid(Pagination<Student> page, String tid,  
 Integer cId);

1. 思路：

查询结果映射为pojo对象Student；

学生表、成绩表、课程表、基本课程表等值连接

若已选定课程id则按照课程id匹配

若传入分页参数，则使用limit分页

（3）SQL语句：

<select id="getStudentListByTid" resultMap="BaseResultMap">

select B.id as id, A.id as sId, A.name as sName, sex, grade, C.complete as complete

score, result, C.id as cId, BC.name as cName

from t\_student as A , t\_score as B , t\_course as C , t\_base\_course as BC

<where>

A.id = B.s\_id and C.id = B.c\_id and C.base\_course\_id = BC.id

and C.complete = 0

and C.t\_id = #{param2}

<if test="param3 != null and param3 != '' ">

and C.id = #{param3}

</if>

</where>

<if test="param1 != null">

limit #{param1.start}, #{param1.pageSize}

</if>

</select>

1. 计算该课程每个班级的平均、最高、最低成绩、及格率

（1）Dao接口定义：

List<AverageScoreVo> getAverageByCid(Integer cid);

（2）思路：

1）连接成绩表、学生表、课程表、基本课程表

2）选了此课程的学生

3）分组统计他们平均成绩、及格率

（3）SQL语句：

SELECT

bc.id AS cid,

bc.NAME AS cname,

major,

grade,

COUNT( 1 ) AS stu\_number,

AVG( result ) AS average,

MAX( result ) AS max,

MIN( result ) AS min,

CONCAT(ROUND( ( sum( CASE WHEN result >= 60 THEN 1 ELSE 0 END ) / COUNT( 1 ) ) \* 100, 2 ),"%" ) pass\_rate

FROM

t\_score score

LEFT JOIN t\_course course ON course.id = score.c\_id

LEFT JOIN t\_base\_course bc ON bc.id = course.base\_course\_id

LEFT JOIN t\_student s ON s.id = score.s\_id

WHERE

complete = 1

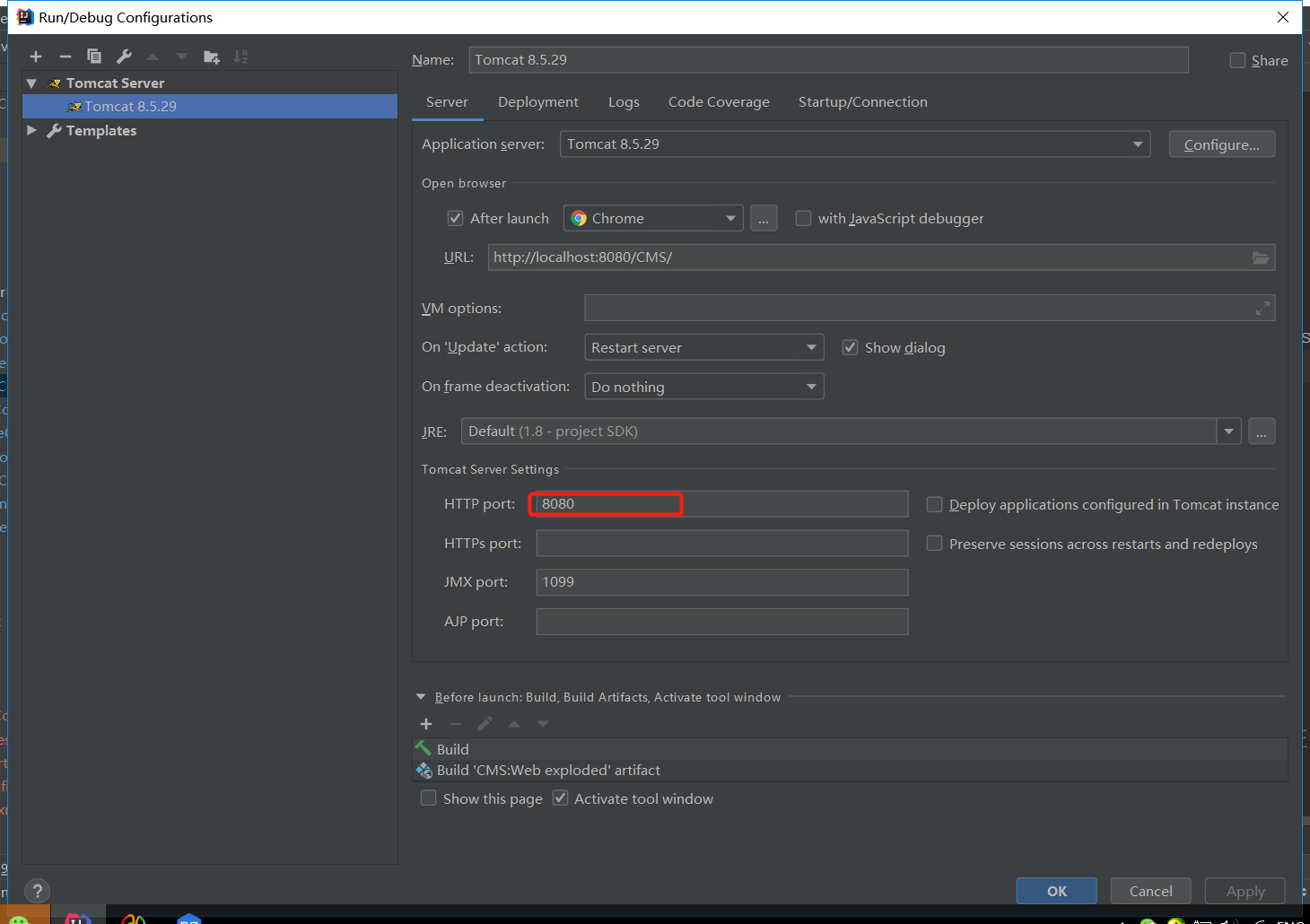
AND bc.id = 2

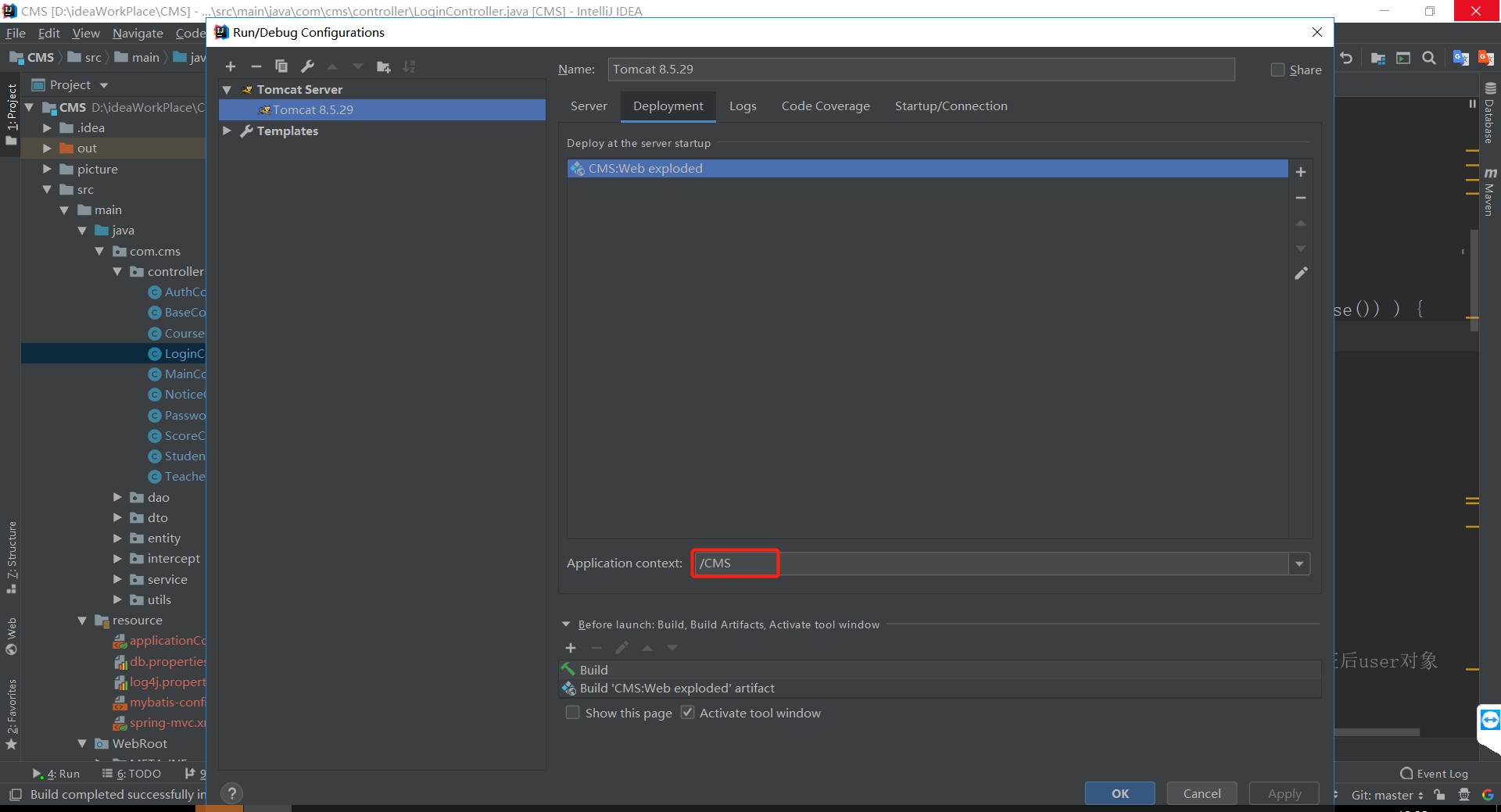
GROUP BY

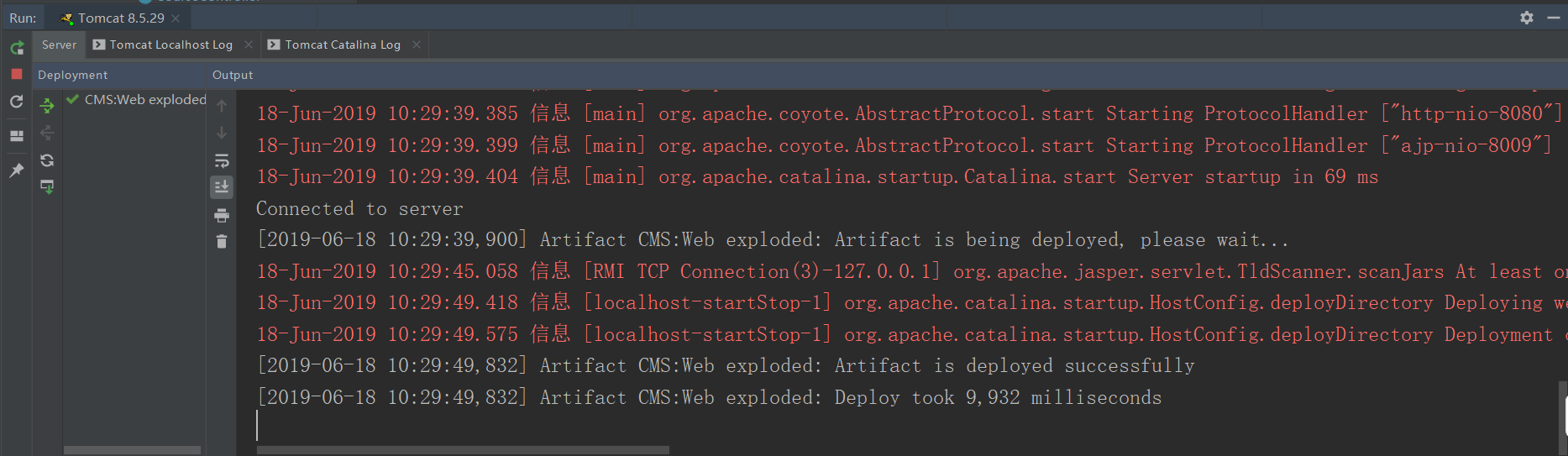
CONCAT( major, grade )

三、效果截图

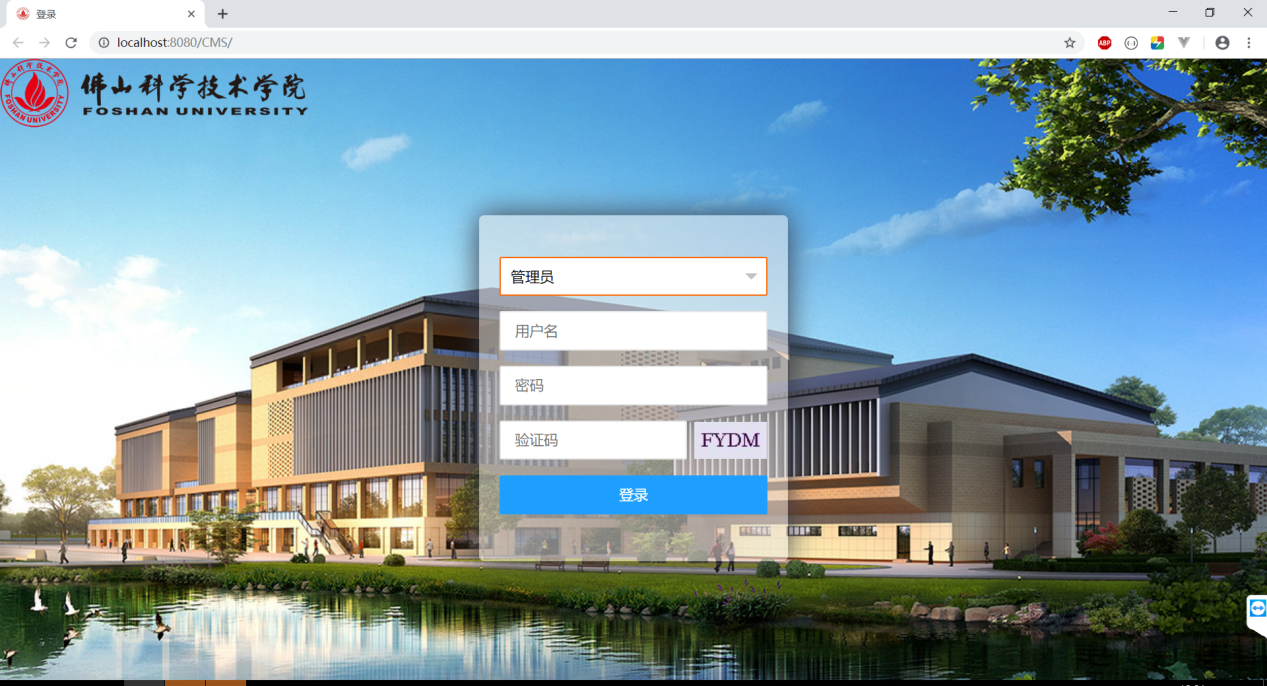
1. 部署运行在8080端口上



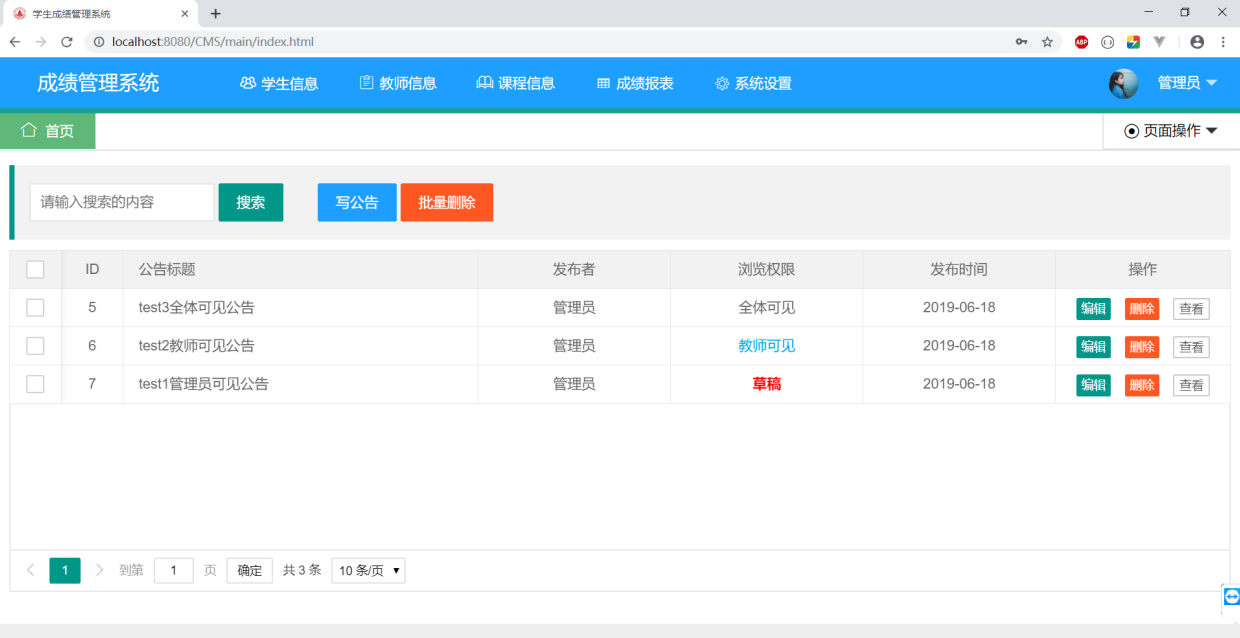




2、首页登录界面

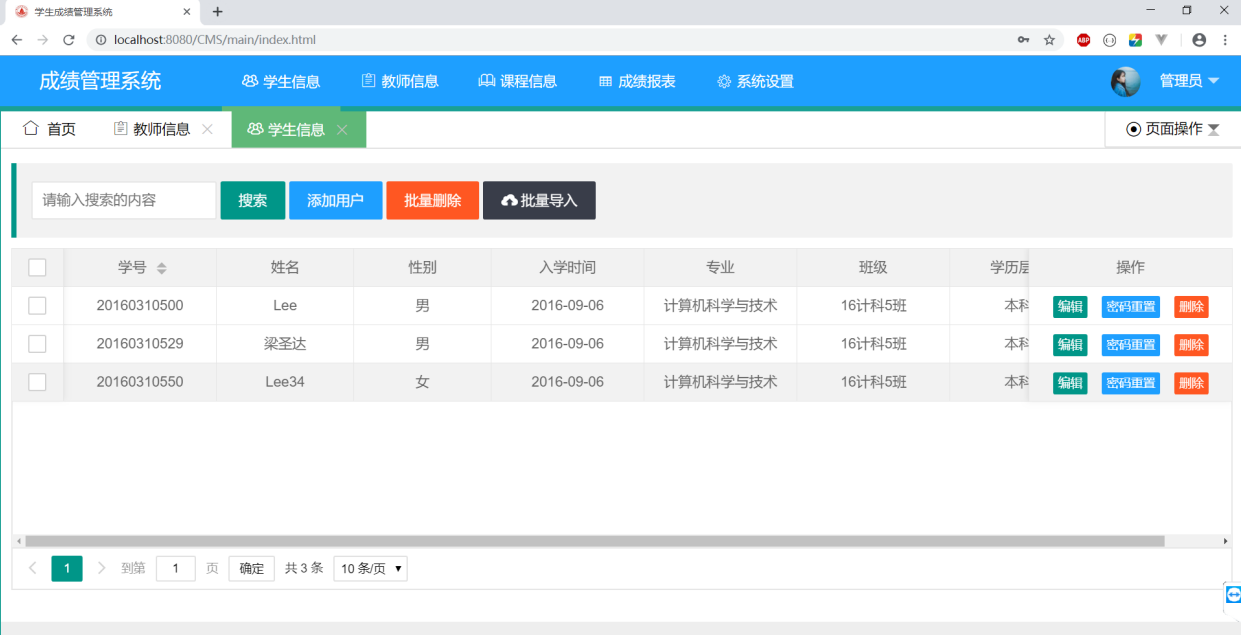


3、登录管理员角色



1. 学生管理



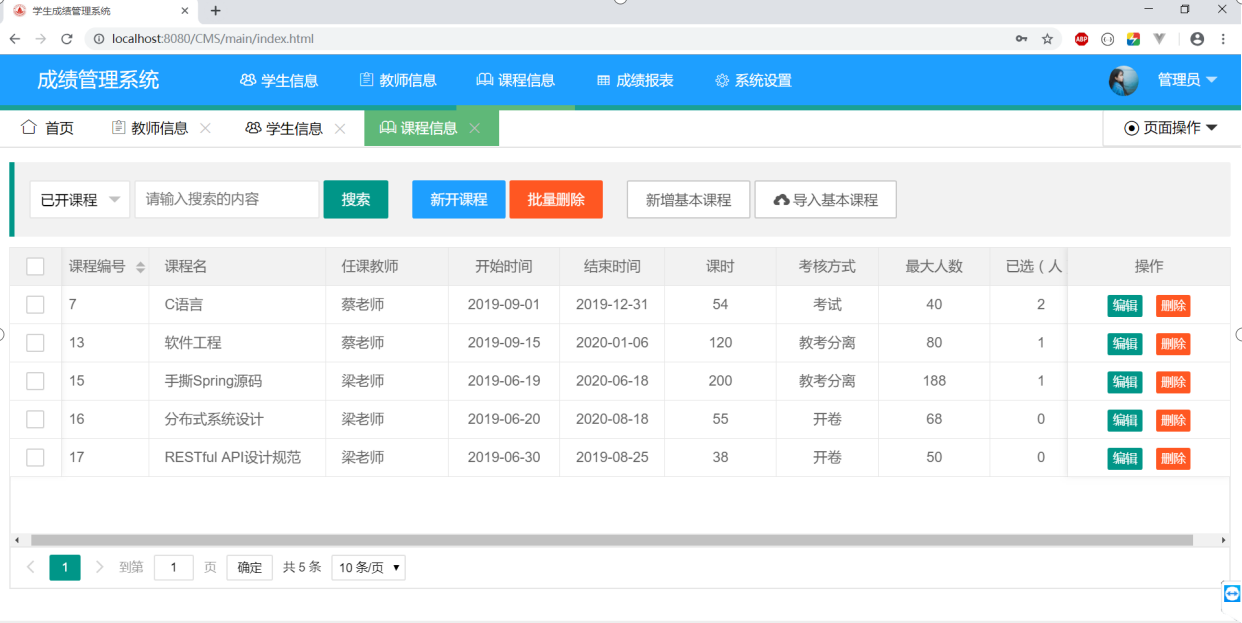


1. 教师管理

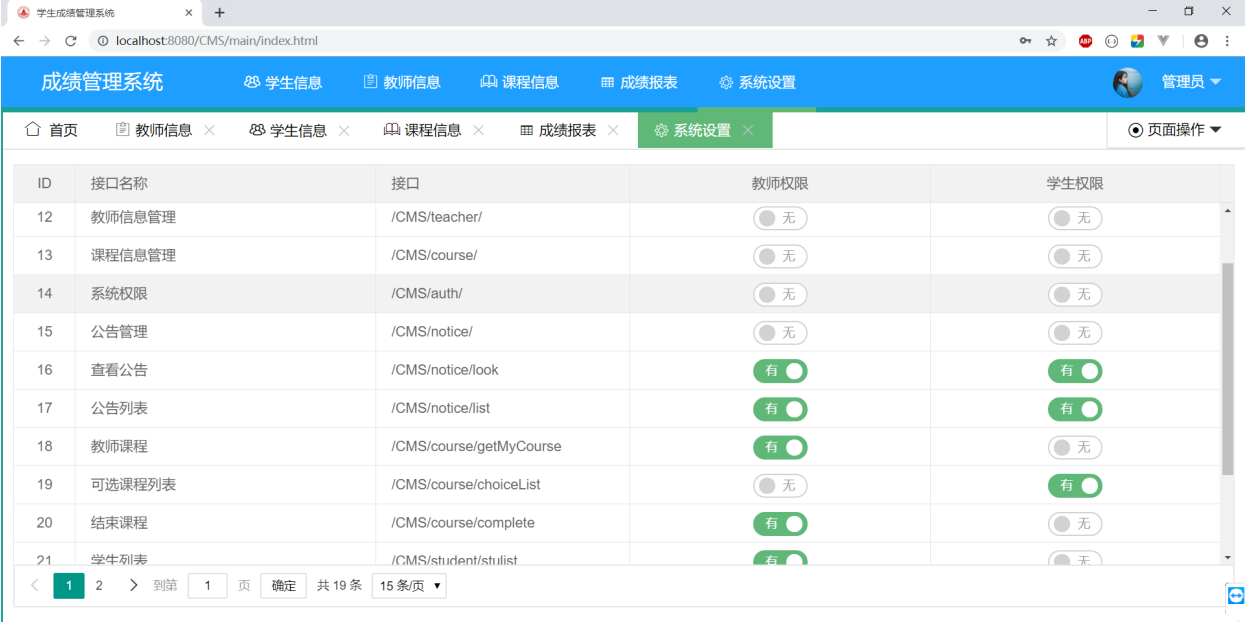


1. 课程管理

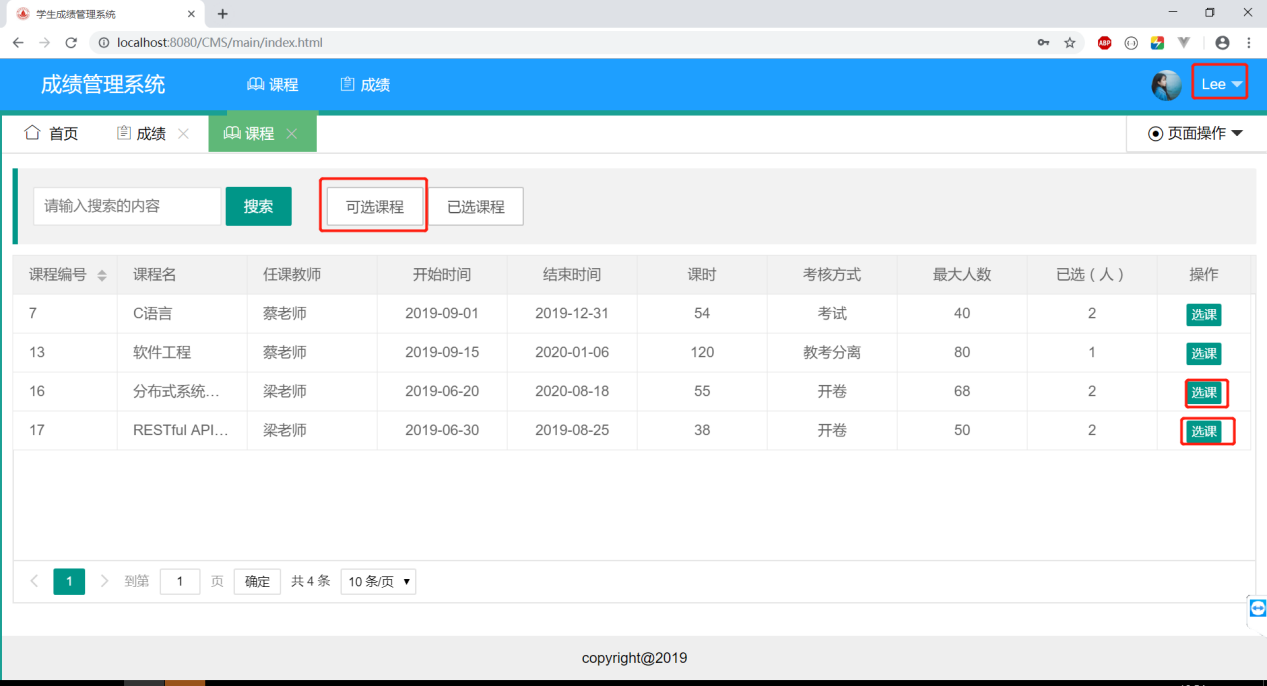


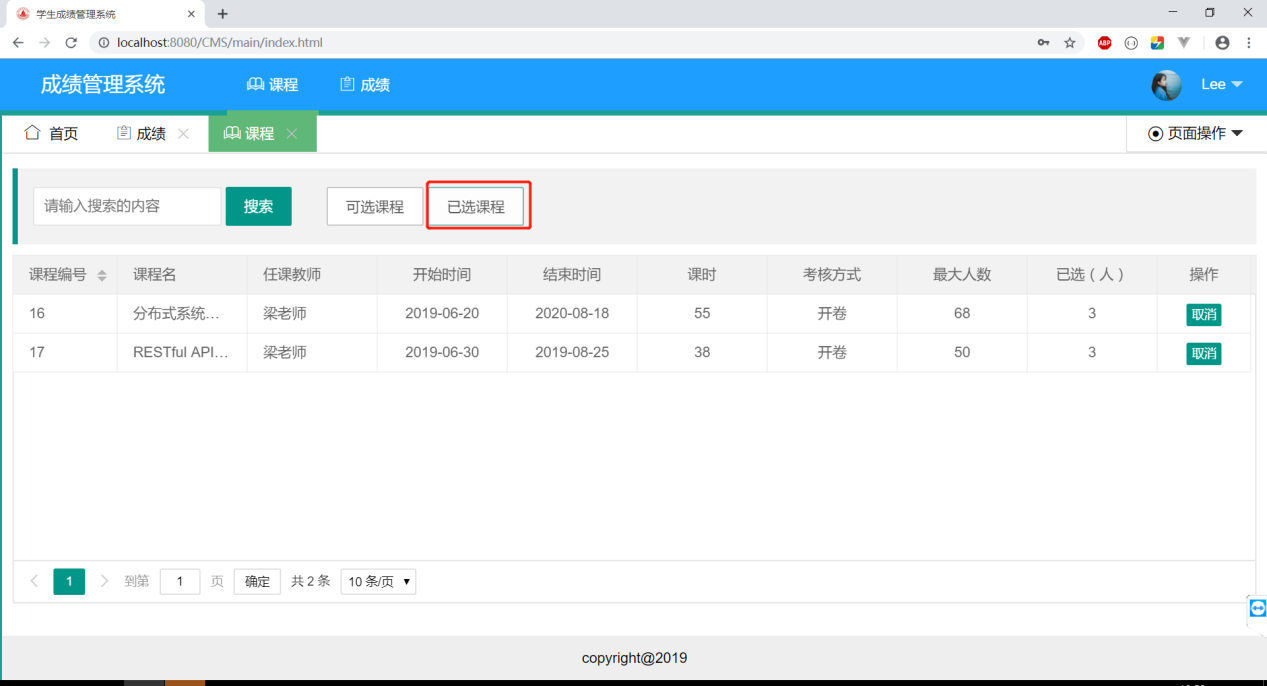


7、系统权限管理

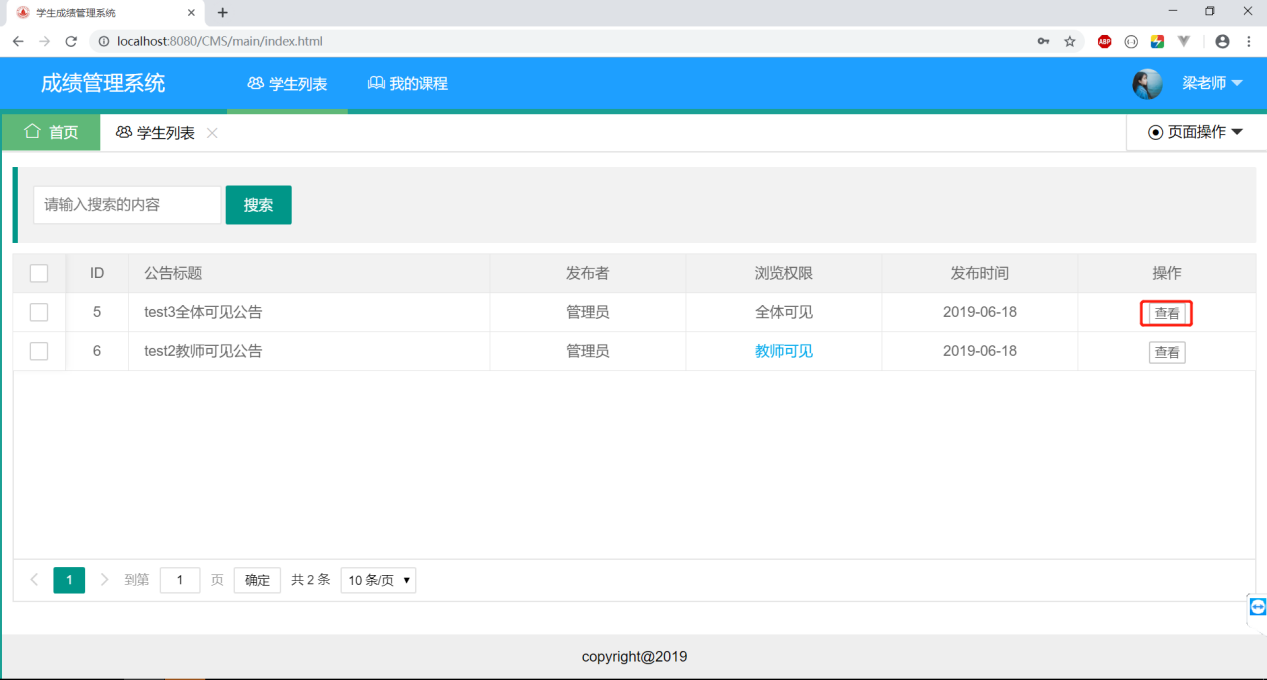


8、学生选课

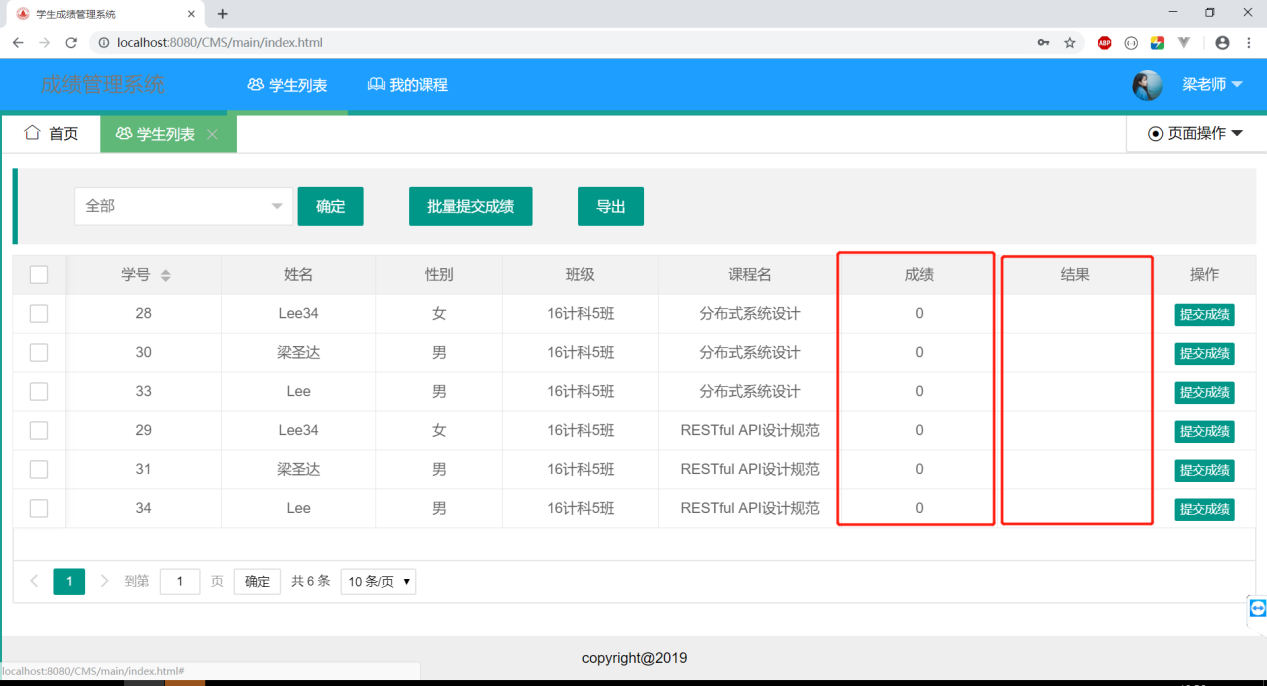


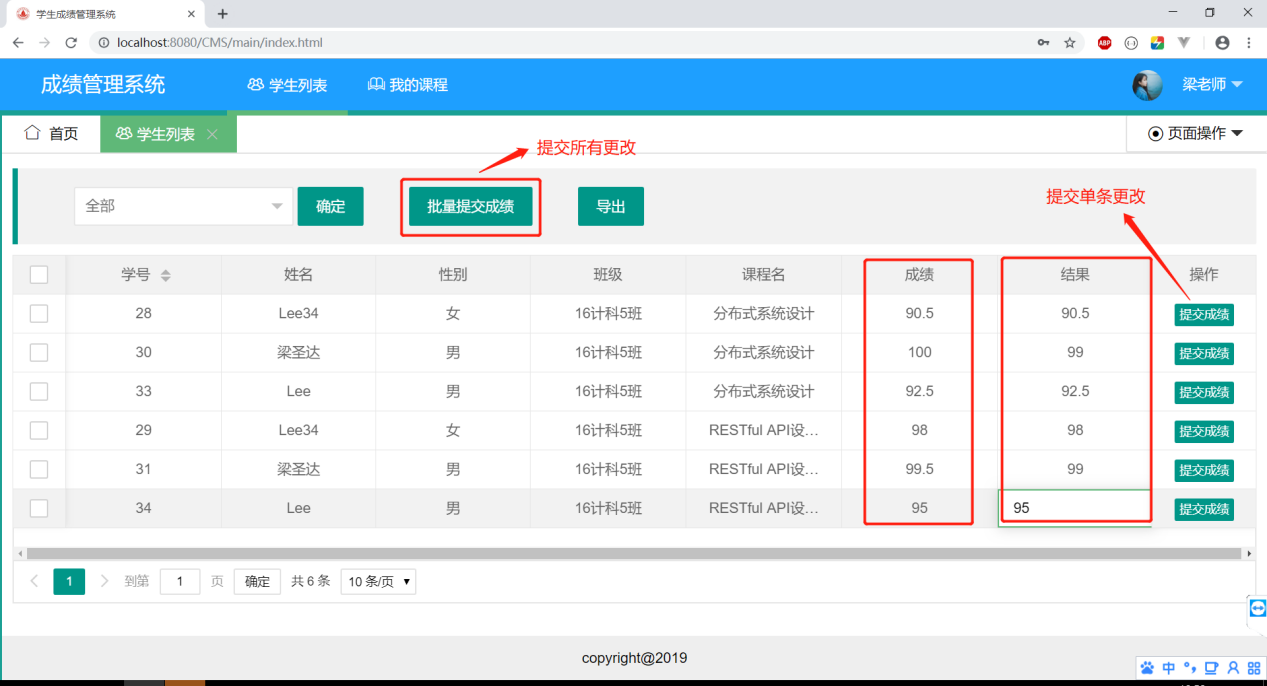


9、查看公告

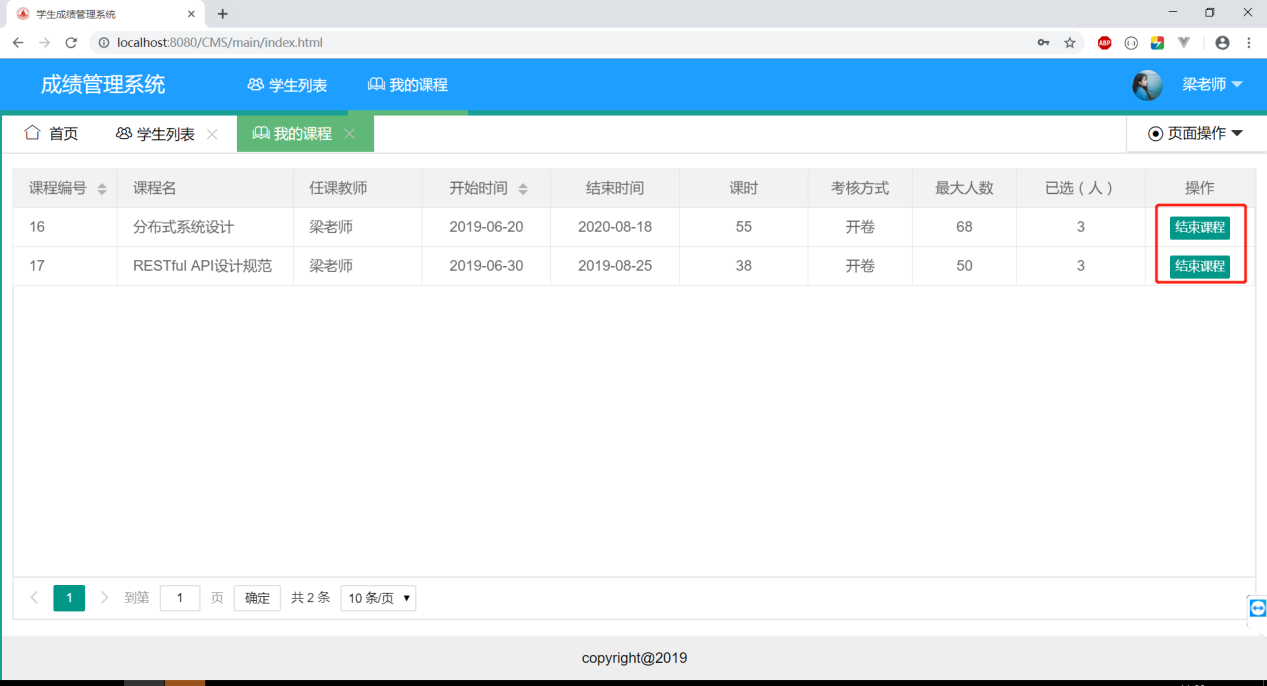


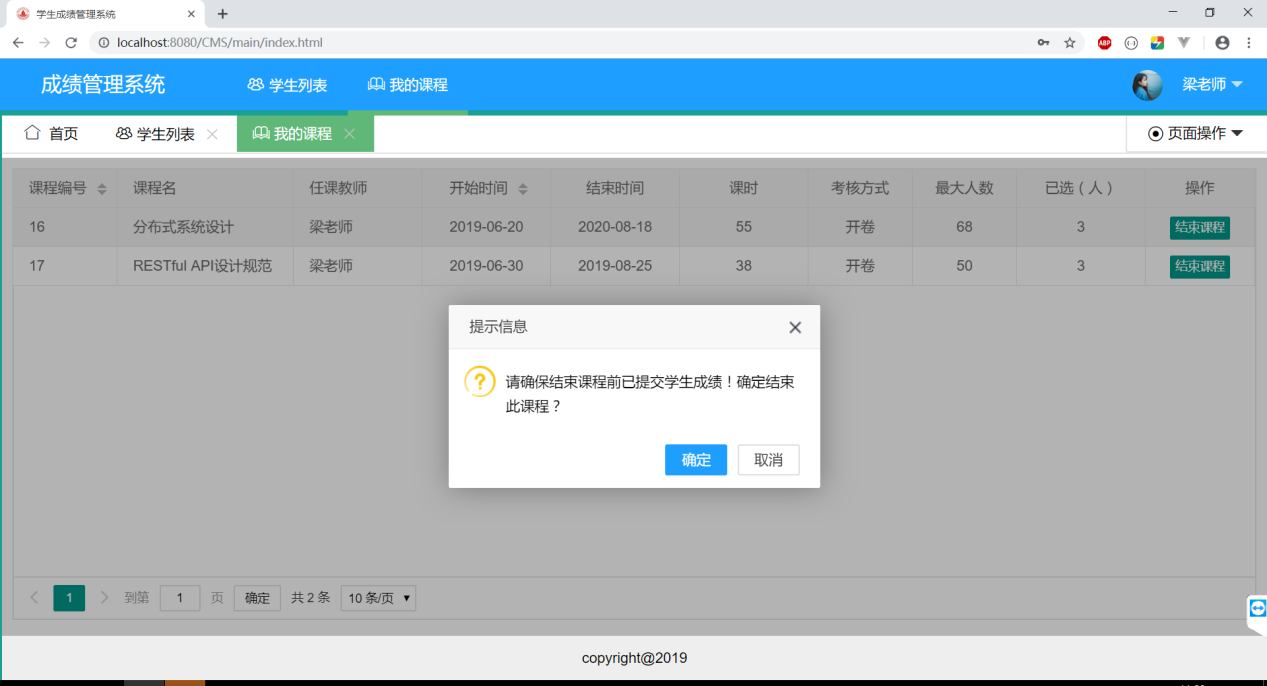
10、给学生录入成绩

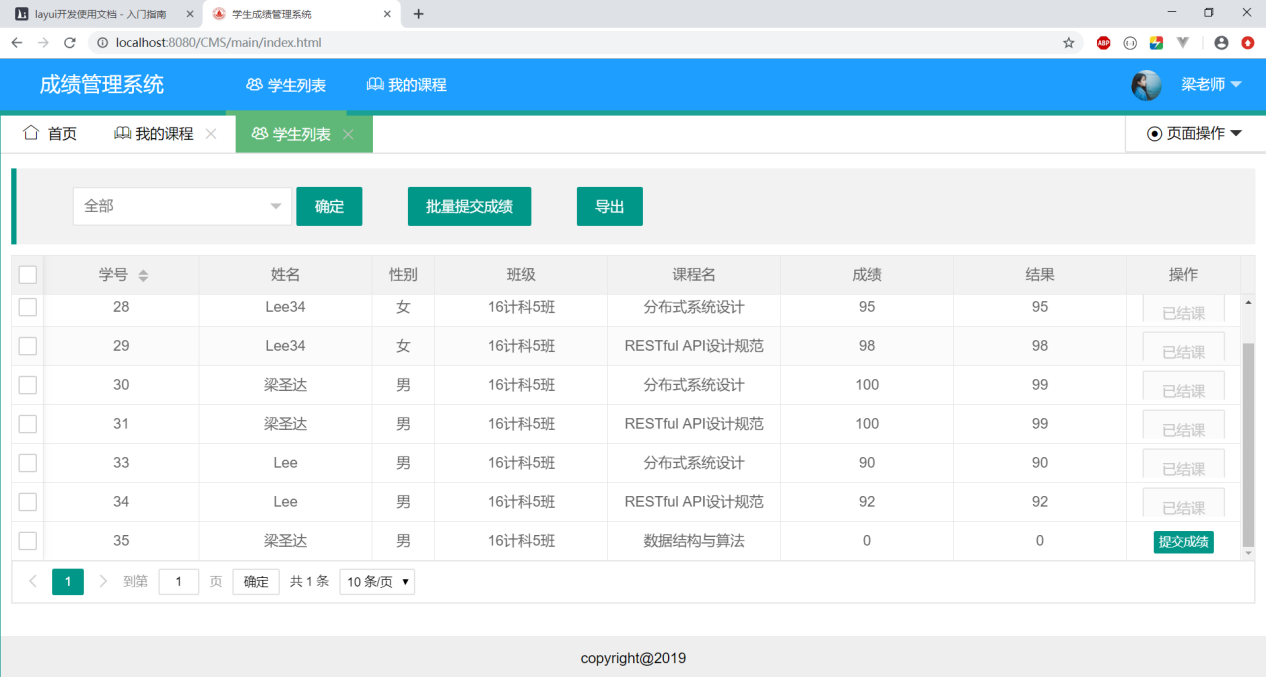




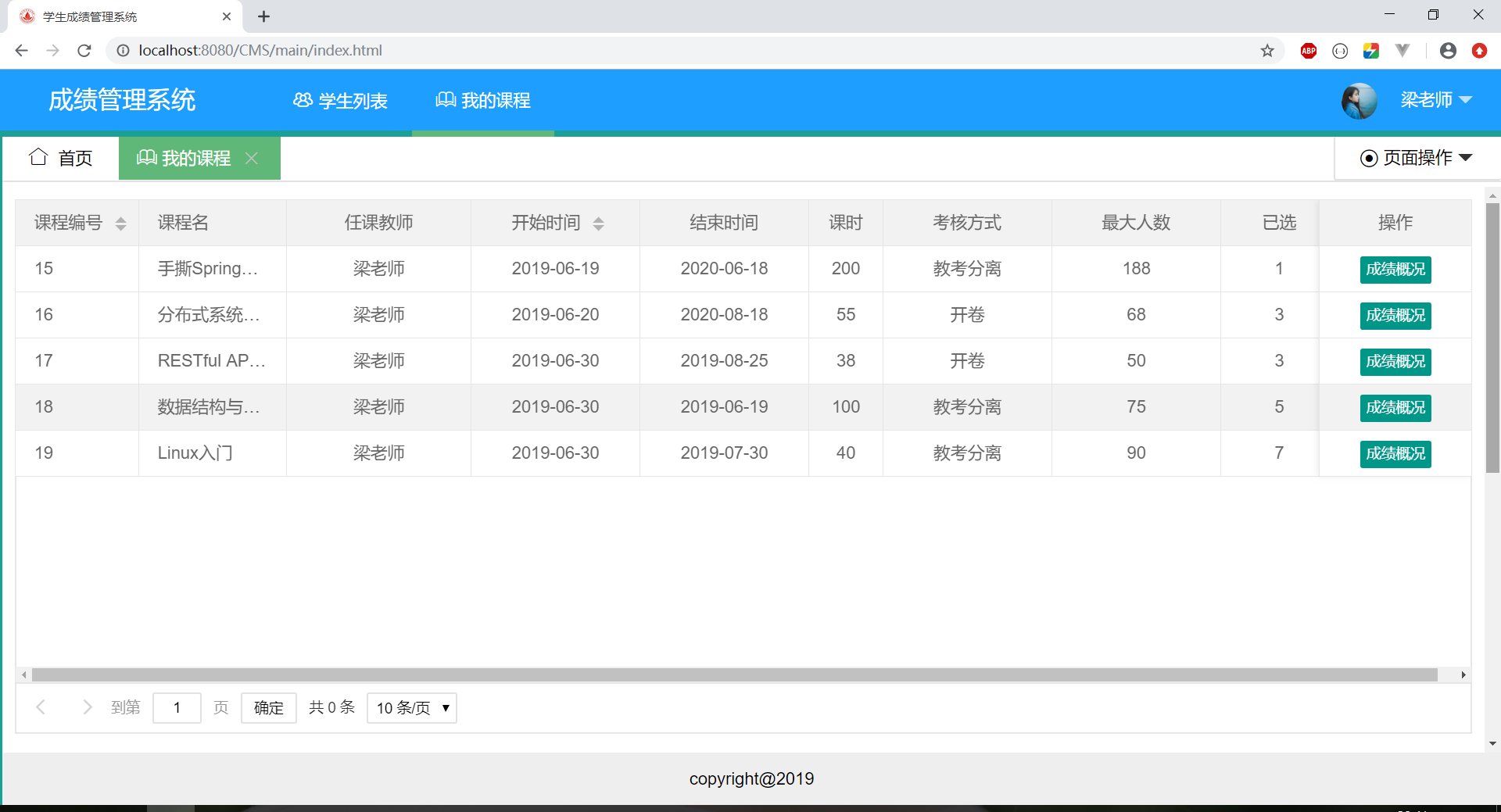
11、老师录入完所有学生成绩后结课

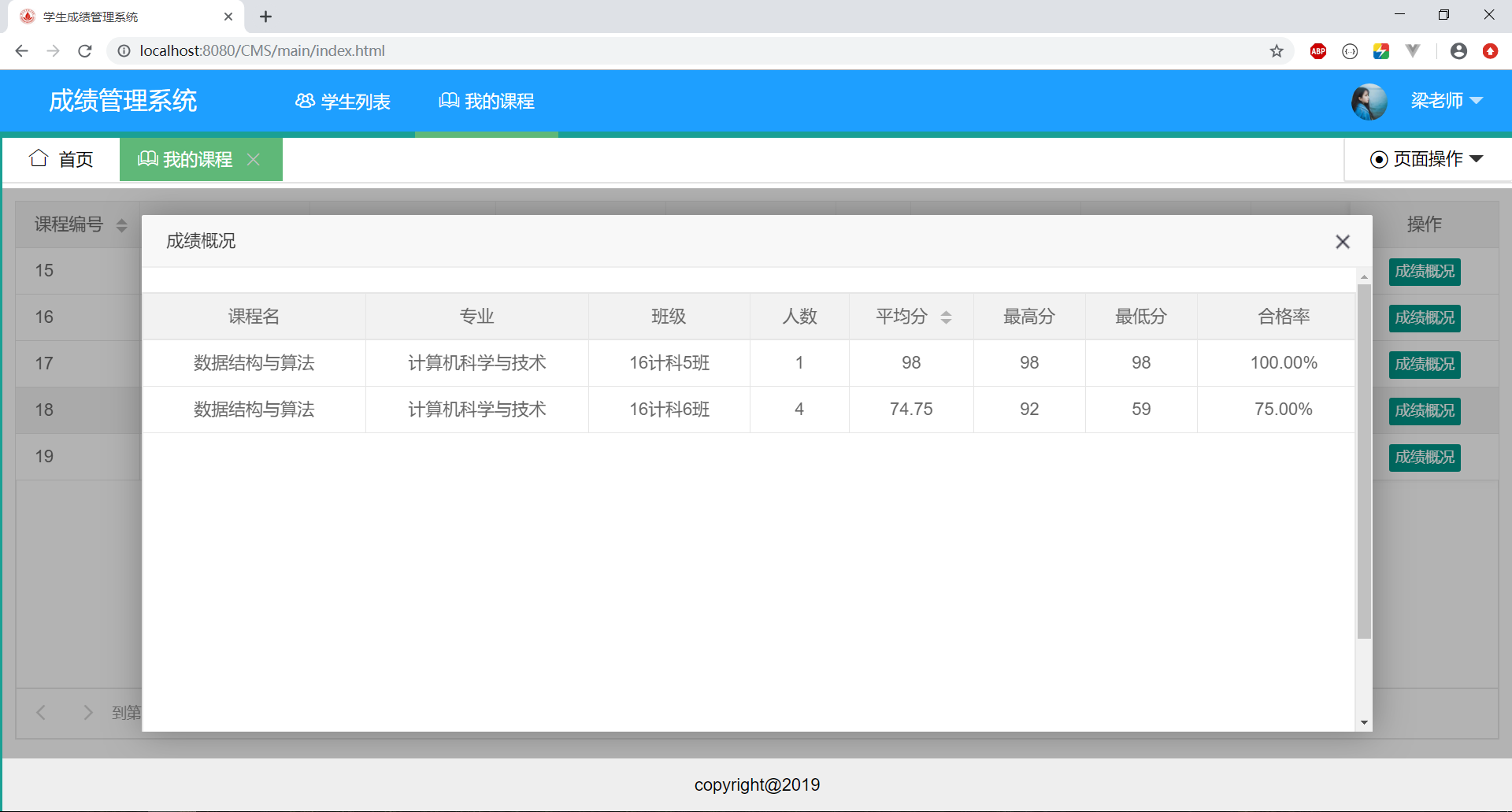




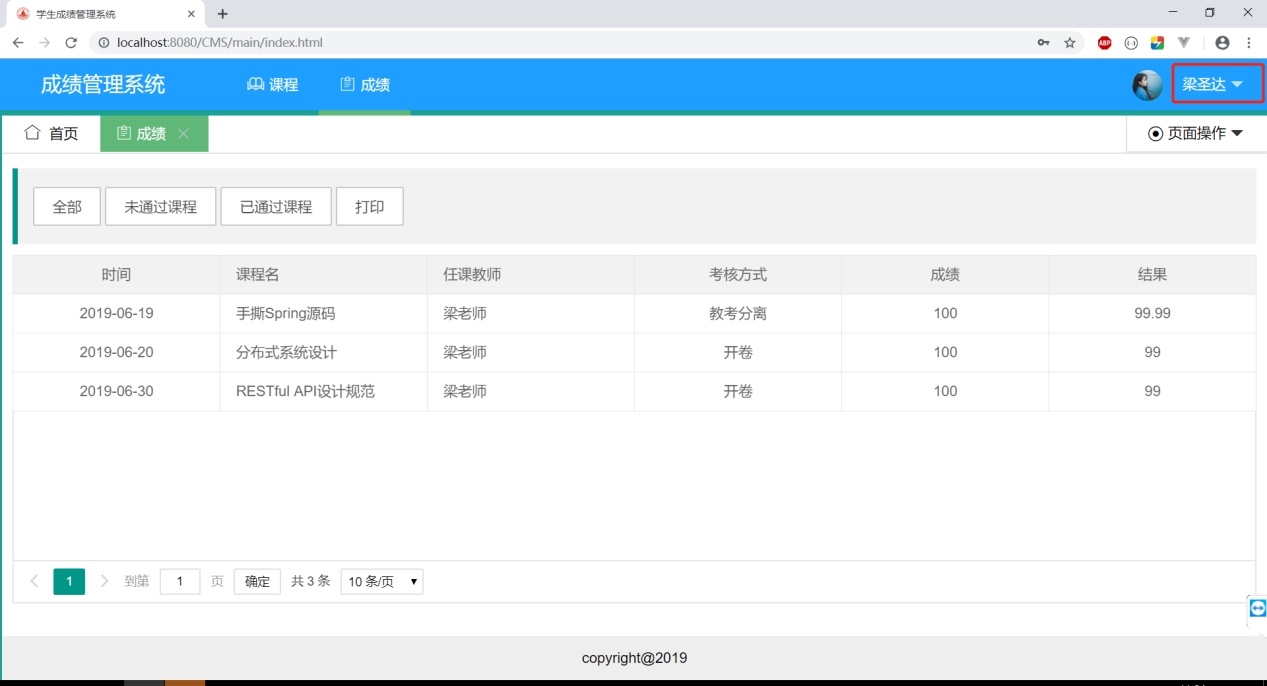


1. 若课程已经结束，则显示课程概况（查看课程的班级平均分）按钮

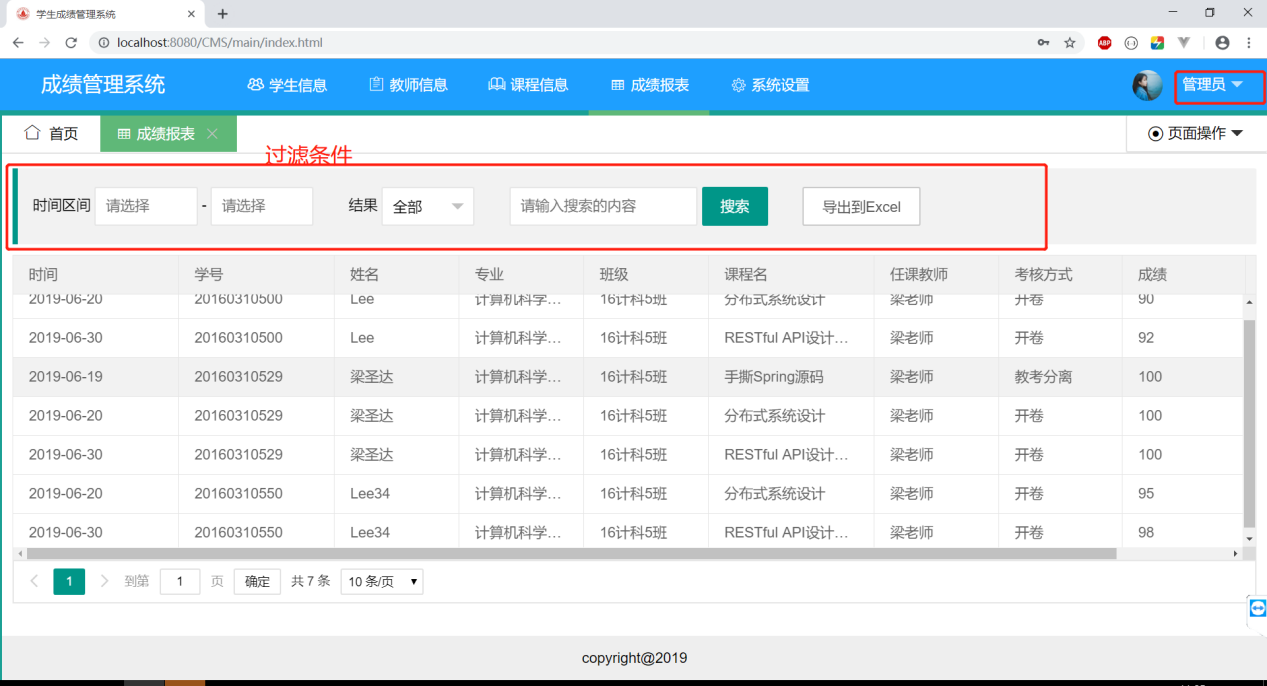


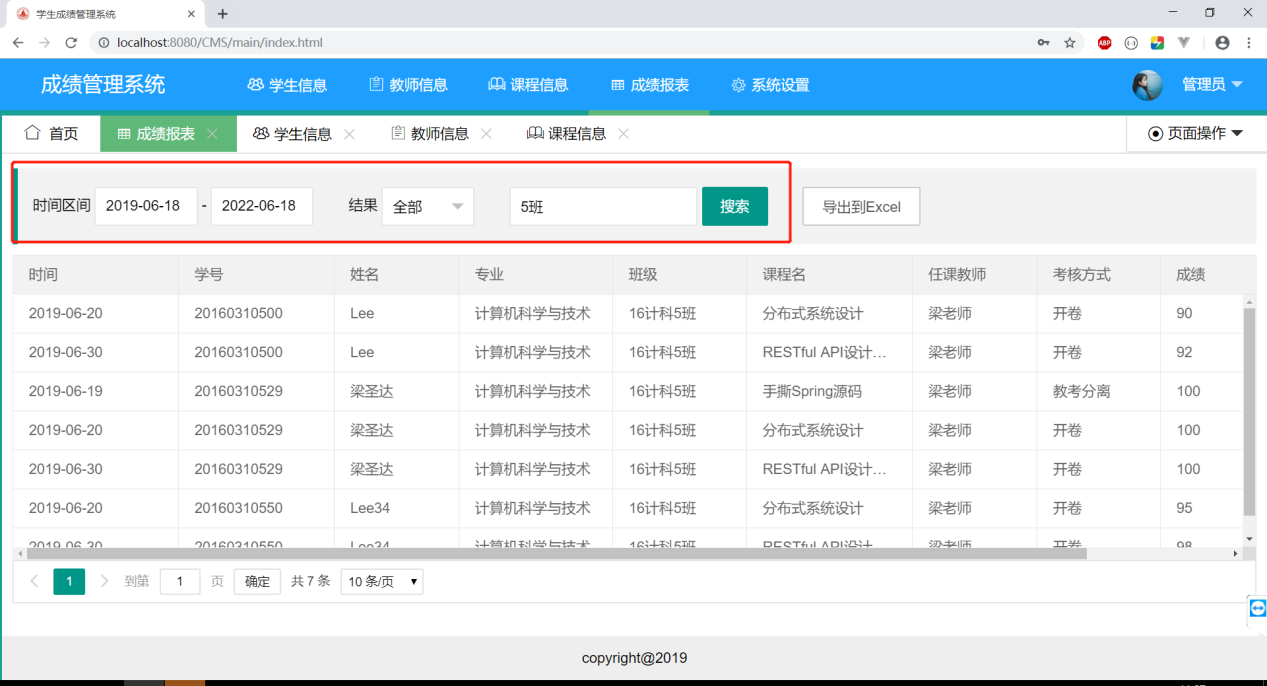


12、学生查看课程成绩



13、管理员查看成绩汇总概况





14、导出为Excel报表

