目录

[测试截图 2](#_Toc55330358)

[登录页面 2](#_Toc55330359)

[超级管理员 3](#_Toc55330360)

[系统管理员 3](#_Toc55330361)

[用户 4](#_Toc55330362)

[上传简历 5](#_Toc55330363)

[上传照片 5](#_Toc55330364)

[日志 5](#_Toc55330365)

[添加事物层 6](#_Toc55330366)

[新增教师 7](#_Toc55330367)

[搜索教师 7](#_Toc55330368)

[删除教师 8](#_Toc55330369)

[查看教师 9](#_Toc55330370)

[分页 9](#_Toc55330371)

[代码 10](#_Toc55330372)

[TeacherMapper.xml 10](#_Toc55330373)

[Teacher.vue 18](#_Toc55330374)

[UploadMapper.xml 35](#_Toc55330375)

[FileServiceBean 36](#_Toc55330376)

[UploadDao 39](#_Toc55330377)

[FileController 39](#_Toc55330378)

# 测试截图

## 登录页面



## 超级管理员

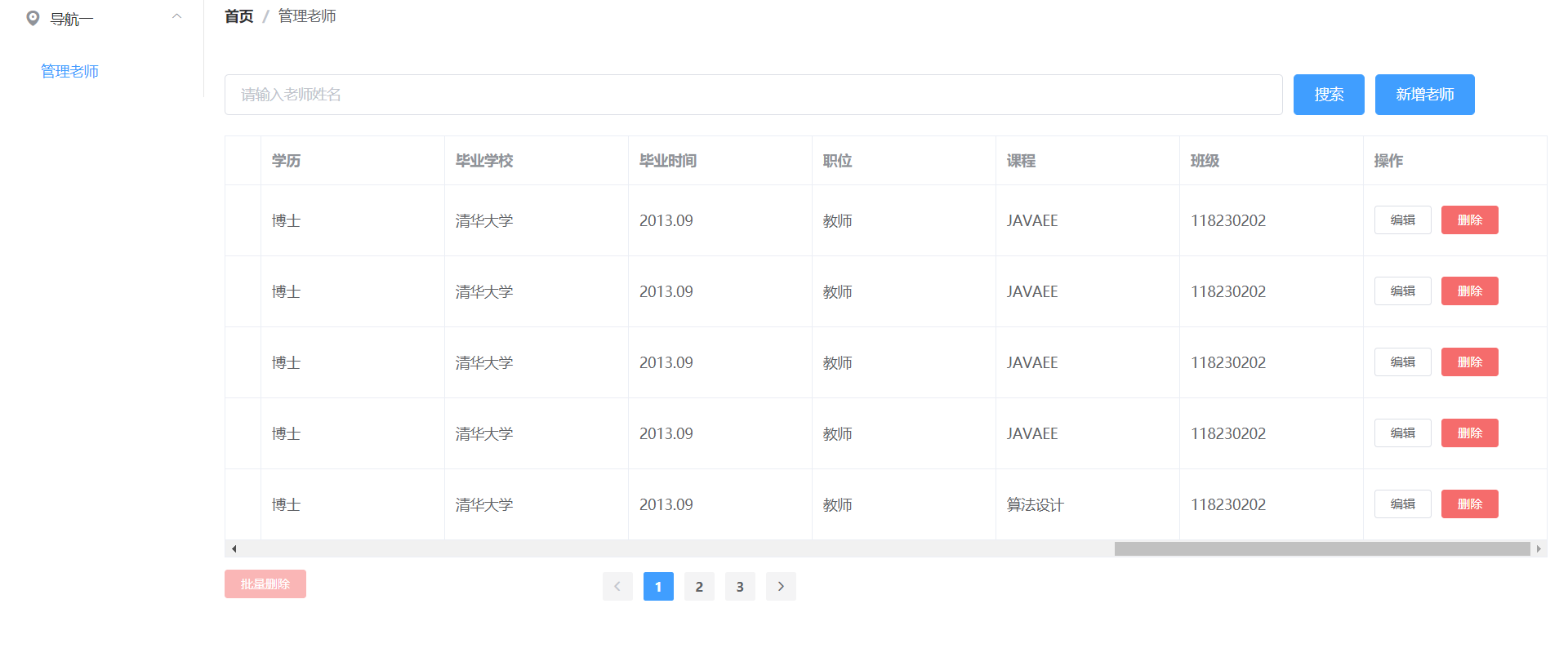
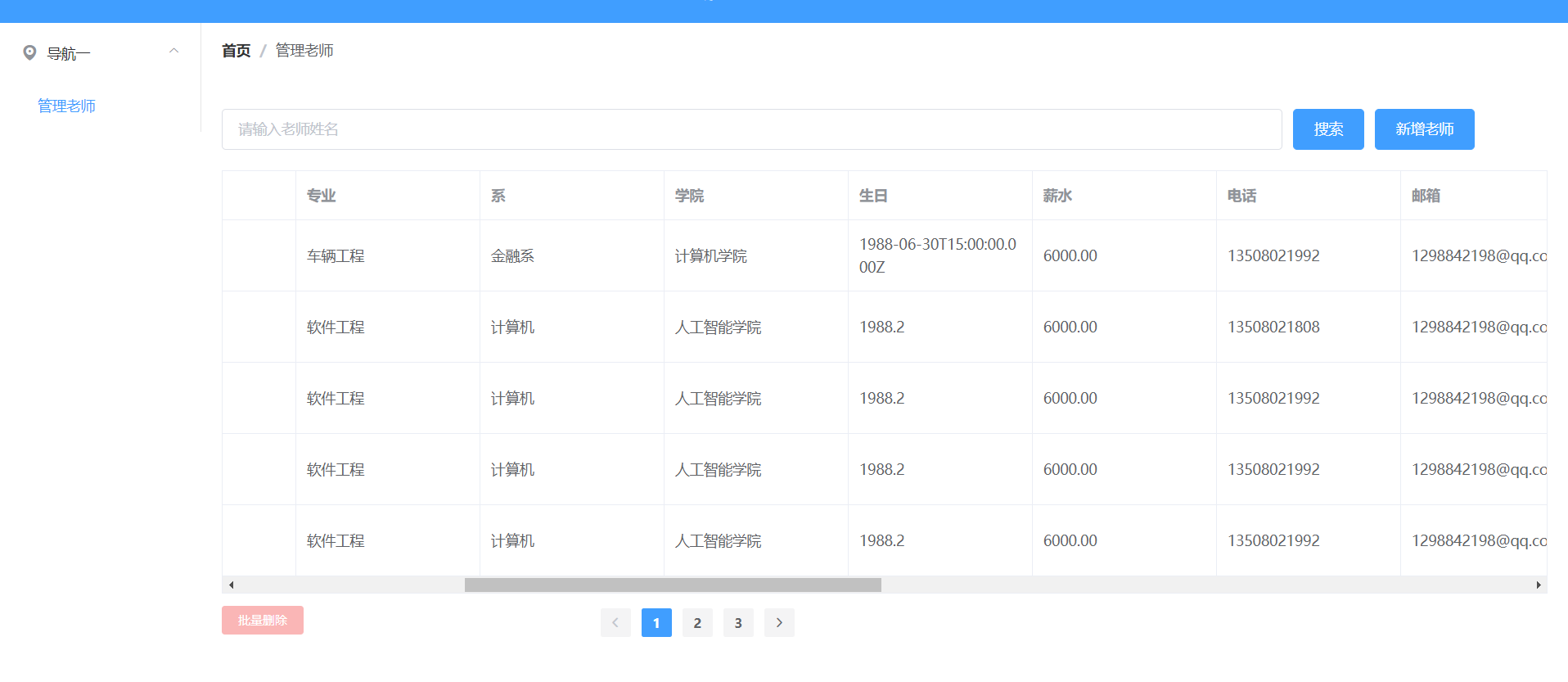


## 系统管理员

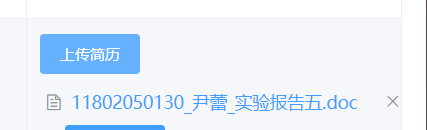


## 用户

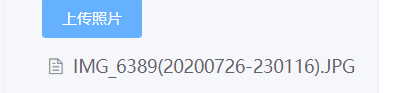




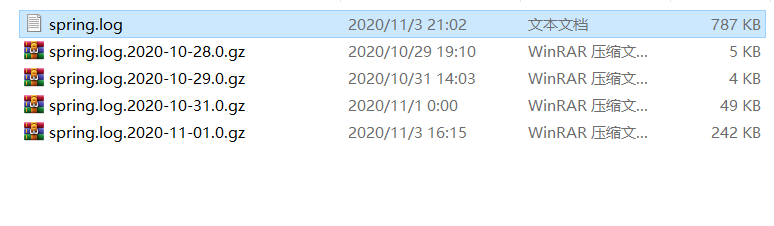
## 上传简历

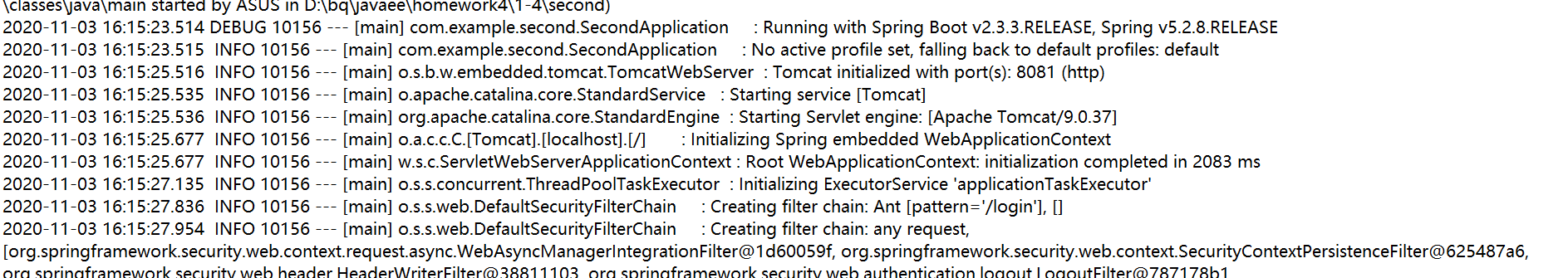


## 上传照片

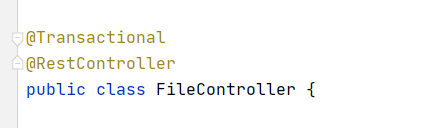


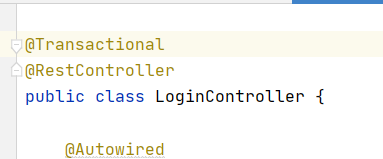
## 日志

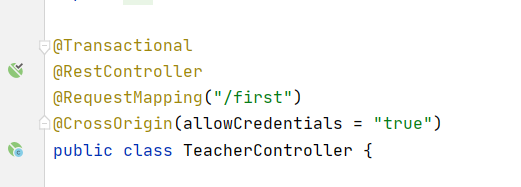


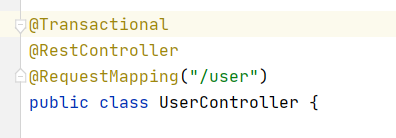


## 添加事物层

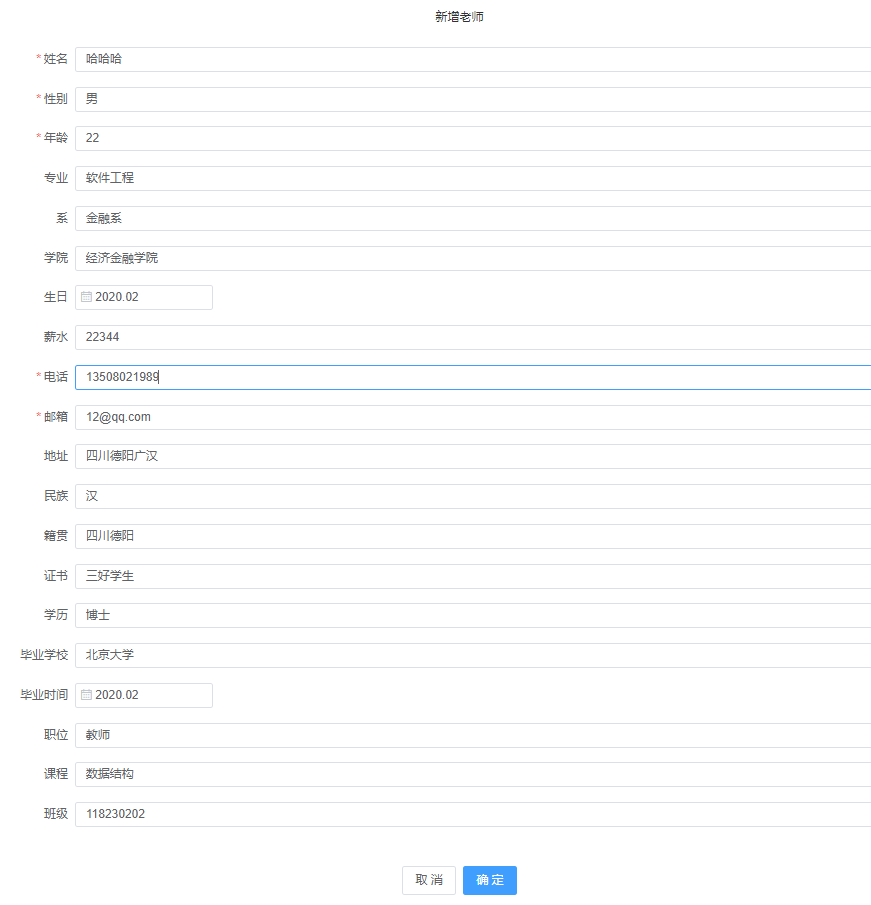




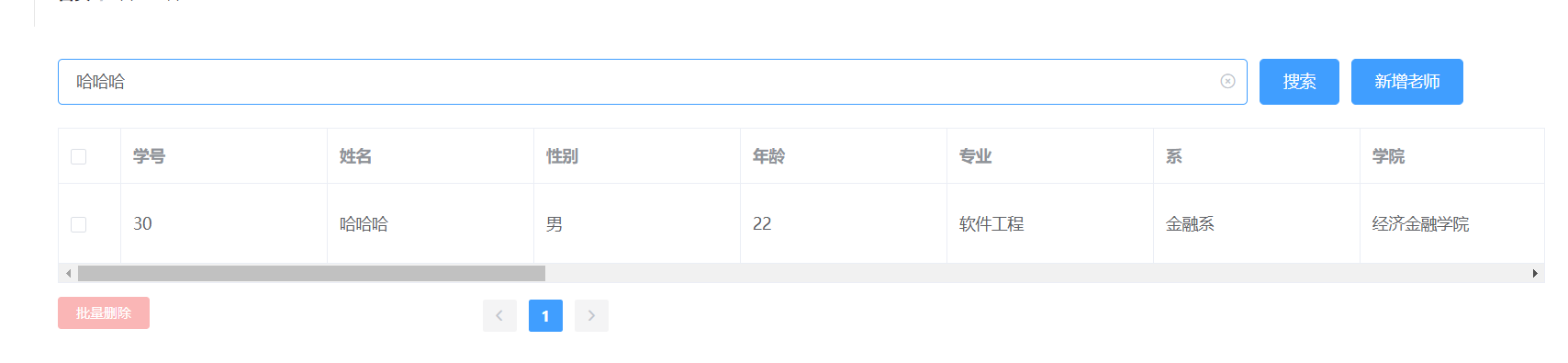




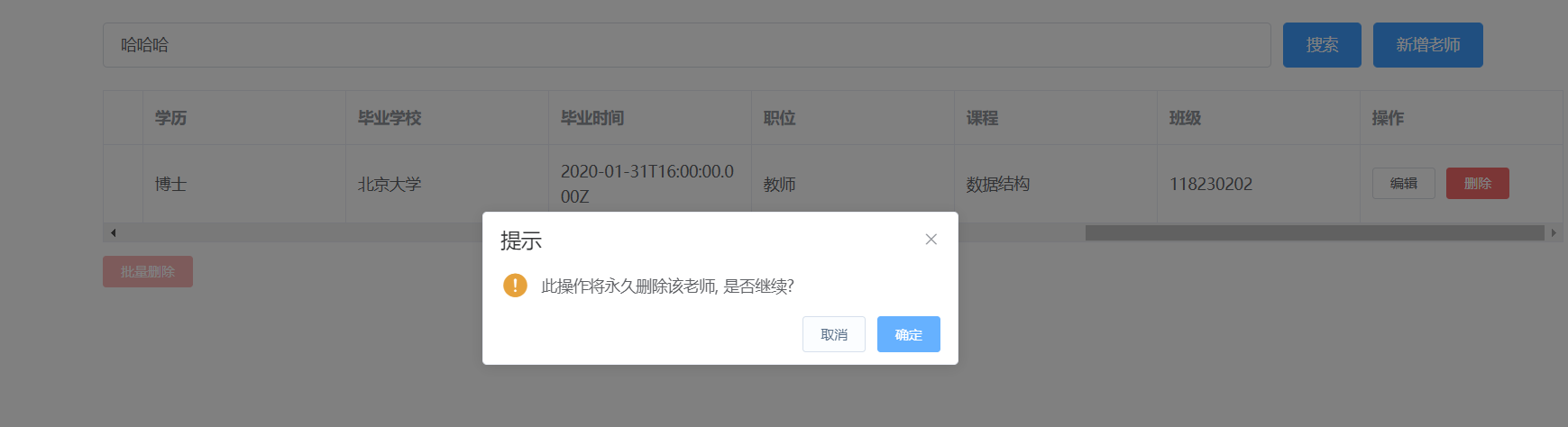
## 新增教师

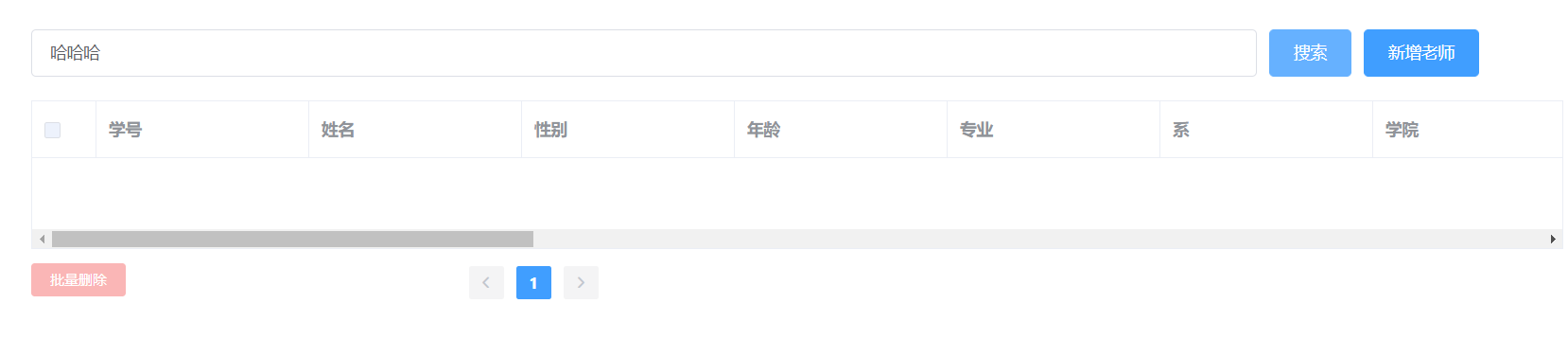


## 搜索教师

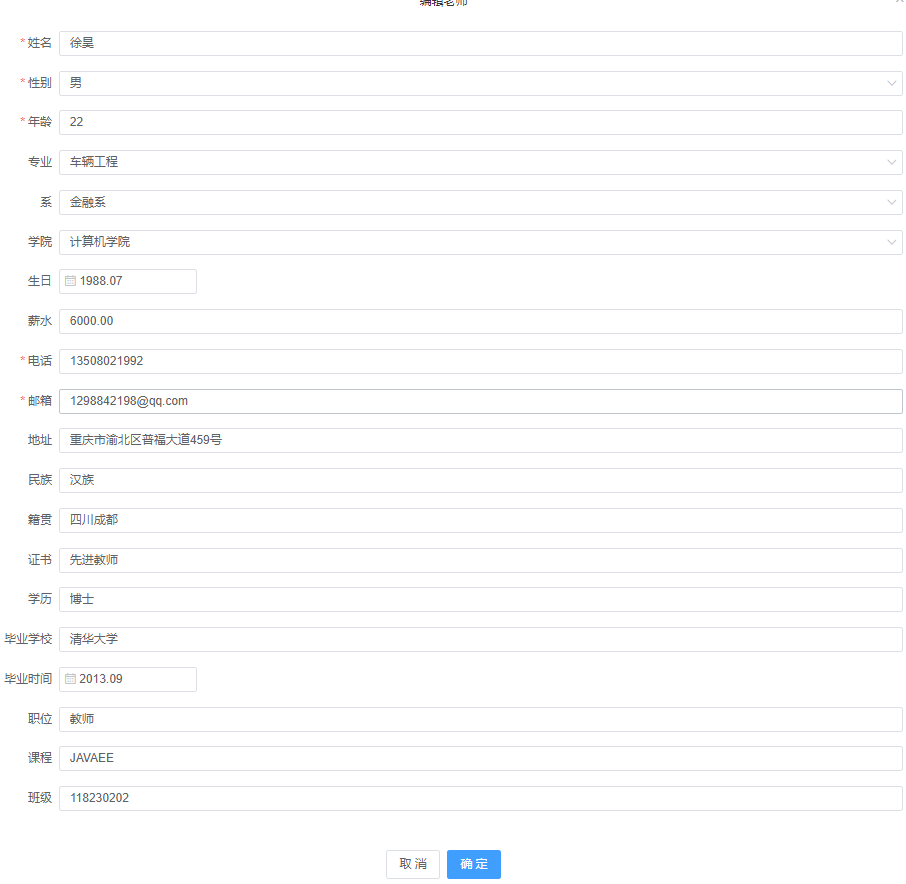


## 删除教师

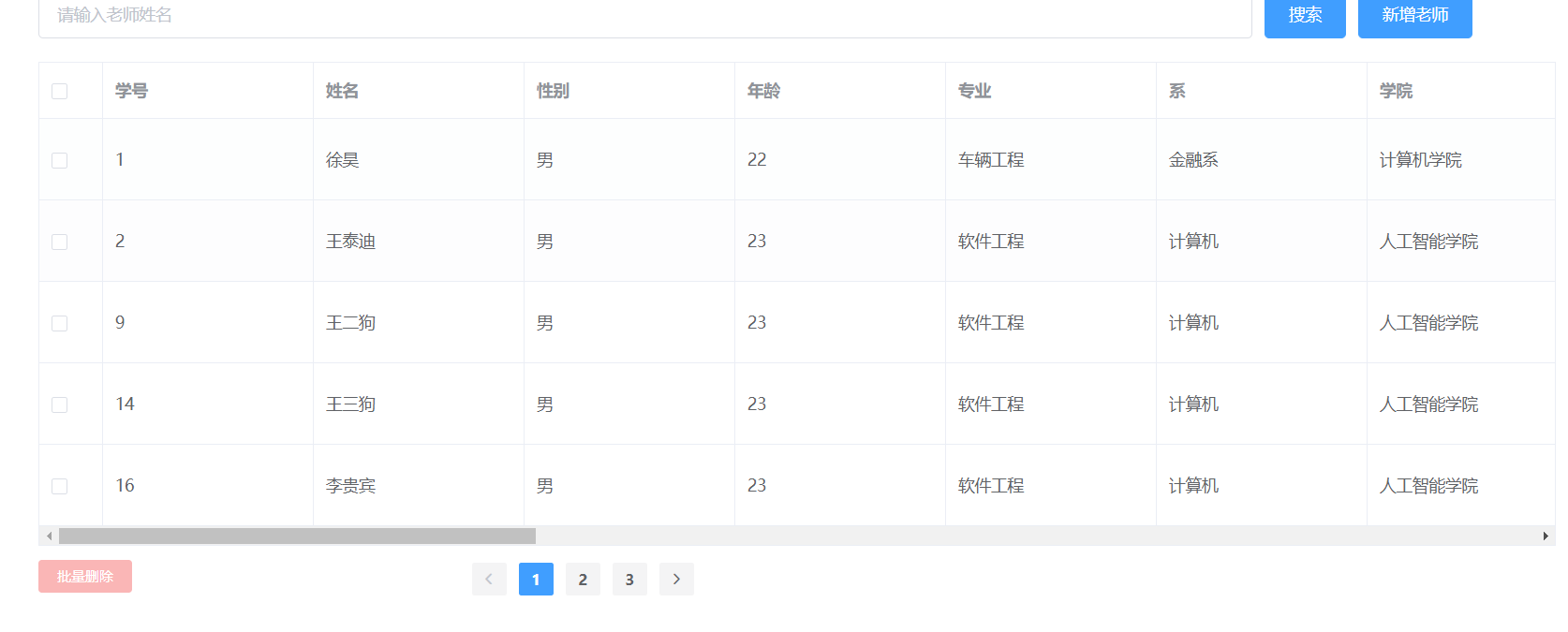


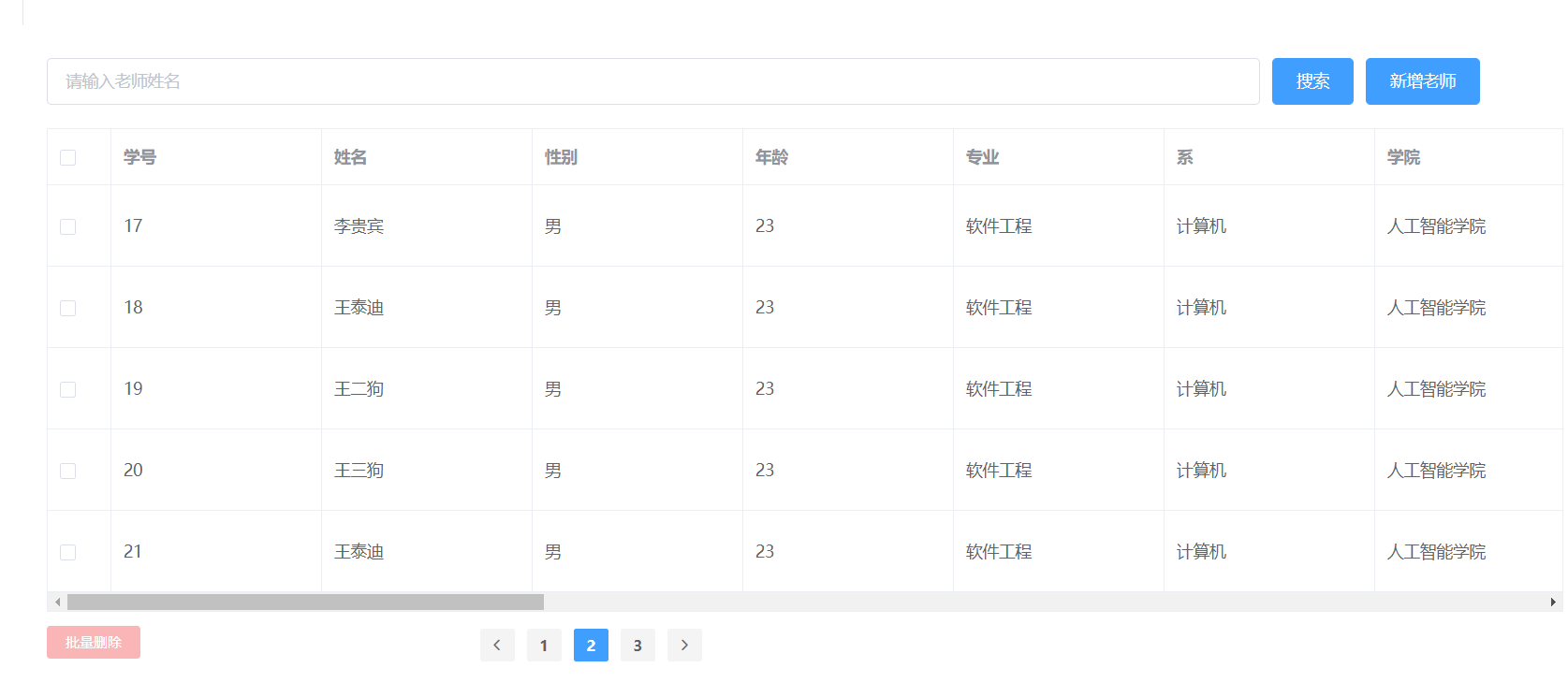


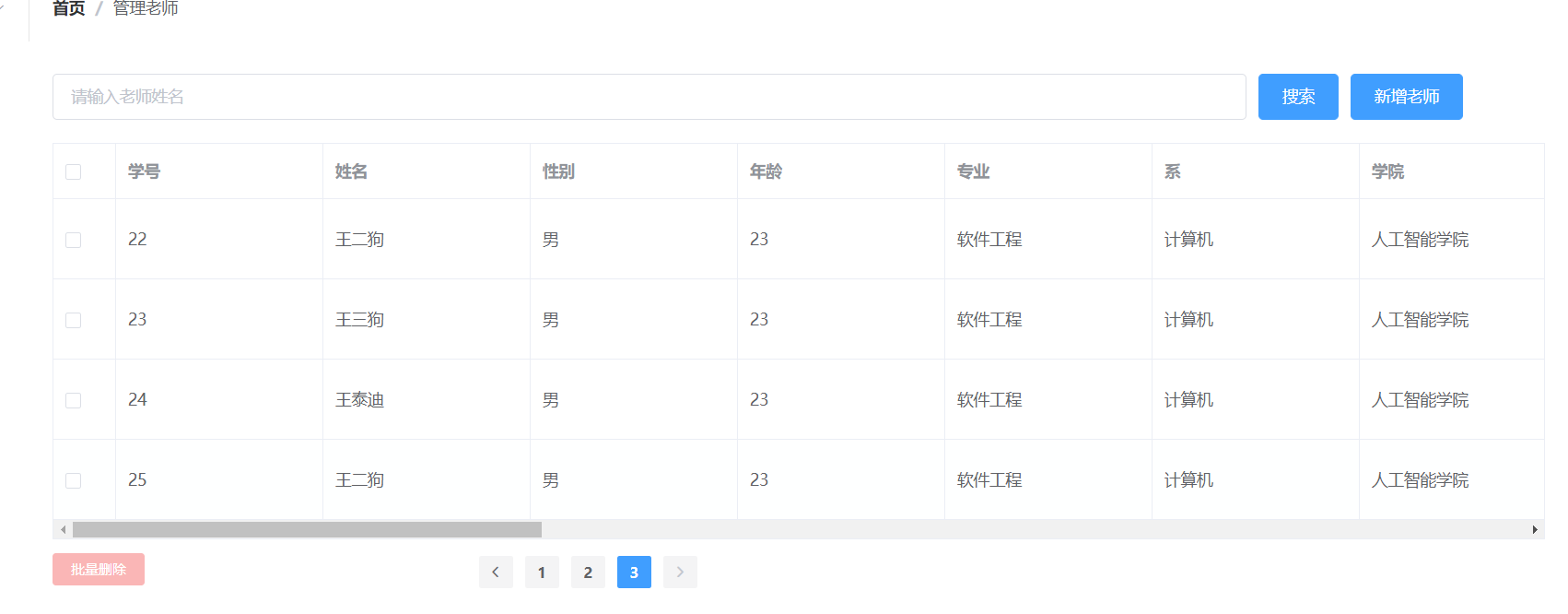
## 查看教师



## 分页







# 代码

## TeacherMapper.xml

*<?*xml version="1.0" encoding="UTF-8" *?>*<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd" *>*<mapper namespace="com.example.second.dao.TeacherDao">  
 <resultMap id="BaseResultMap" type="com.example.second.entity.Teacher">  
 <id column="uid" property="uid" jdbcType="BIGINT"/>  
 <result column="name" property="name" jdbcType="VARCHAR"/>  
 <result column="age" property="age" jdbcType="INTEGER"/>  
 <result column="sex" property="sex" jdbcType="VARCHAR"/>  
 <result column="major" property="major" jdbcType="VARCHAR"/>  
 <result column="department" property="department" jdbcType="VARCHAR"/>  
 <result column="college" property="college" jdbcType="VARCHAR"/>  
 <result column="birth" property="birth" jdbcType="VARCHAR"/>  
 <result column="salary" property="salary" jdbcType="VARCHAR"/>  
 <result column="phone" property="phone" jdbcType="VARCHAR"/>  
 <result column="email" property="email" jdbcType="VARCHAR"/>  
 <result column="address" property="address" jdbcType="VARCHAR"/>  
 <result column="ethnic" property="ethnic" jdbcType="VARCHAR"/>  
 <result column="nativePlace" property="nativePlace" jdbcType="VARCHAR"/>  
 <result column="certificate" property="certificate" jdbcType="VARCHAR"/>  
 <result column="diploma" property="diploma" jdbcType="VARCHAR"/>  
 <result column="schoolOfGraduation" property="schoolOfGraduation" jdbcType="VARCHAR"/>  
 <result column="timeOfGraduation" property="timeOfGraduation" jdbcType="VARCHAR"/>  
 <result column="positions" property="positions" jdbcType="VARCHAR"/>  
 <result column="course" property="course" jdbcType="VARCHAR"/>  
 <result column="clazz" property="clazz" jdbcType="VARCHAR"/>  
 </resultMap>  
 <sql id="Base\_Column\_List">  
 uid, name, age, sex, major, department, college, birth, salary,phone, email, address, ethnic, nativePlace, certificate, diploma, schoolOfGraduation, timeOfGraduation, positions, course, clazz  
 </sql>  
  
 <select id="findAllTeachers" resultType="com.example.second.entity.Teacher">  
 select *\** from teacher  
 </select>  
  
 <select id="findTeacherByName" parameterType="java.lang.String" resultMap="BaseResultMap">  
 select *\** from Teacher where name = #{name,jdbcType=VARCHAR}  
 </select>  
  
 <delete id="deleteTeachersByIds">  
 delete from Teacher where uid in  
 <foreach collection="ids" item="id" separator="," open="(" close=")">  
 #{id}  
 </foreach>  
 </delete>  
  
 <select id="selectByPrimaryKey" resultMap="BaseResultMap" parameterType="java.lang.Long">  
 select  
 <include refid="Base\_Column\_List"/>  
 from Teacher  
 where uid = #{uid,jdbcType=BIGINT}  
 </select>  
 <delete id="deleteTeacherById" parameterType="java.lang.Long">  
 delete from Teacher  
 where uid = #{uid,jdbcType=BIGINT}  
 </delete>  
  
 <insert id="insert" parameterType="com.example.second.entity.Teacher">  
 insert into Teacher (uid, name, age,  
 sex,major, department, college, birth, salary, phone, email, address, ethnic, nativePlace, certificate, diploma, schoolOfGraduation, timeOfGraduation, positions, course, clazz)  
 values (#{uid,jdbcType=BIGINT}, #{name,jdbcType=VARCHAR}, #{age,jdbcType=INTEGER},  
 #{sex,jdbcType=VARCHAR}, #{majorjdbcType=VARCHAR}, #{department,jdbcType=VARCHAR}, #{college,jdbcType=VARCHAR}, #{birth,jdbcType=VARCHAR}, #{salary,jdbcType=VARCHAR}, #{phone,jdbcType=VARCHAR}, #{email,jdbcType=VARCHAR}, #{address,jdbcType=VARCHAR}, #{ethnic,jdbcType=VARCHAR}, #{nativePlace,jdbcType=VARCHAR}, #{certificate,jdbcType=VARCHAR}, #{diploma,jdbcType=VARCHAR}, #{schoolOfGraduation,jdbcType=VARCHAR}, #{timeOfGraduation,jdbcType=VARCHAR}, #{positions,jdbcType=VARCHAR}, #{course,jdbcType=VARCHAR}, #{clazz,jdbcType=VARCHAR},  
 </insert>  
 <insert id="addTeacher" parameterType="com.example.second.entity.Teacher">  
 insert into Teacher  
 <trim prefix="(" suffix=")" suffixOverrides=",">  
 <if test="uid != null">  
 uid,  
 </if>  
 <if test="name != null">  
 name,  
 </if>  
 <if test="age != null">  
 age,  
 </if>  
 <if test="sex != null">  
 sex,  
 </if>  
 <if test="major != null">  
 major,  
 </if>  
 <if test="department != null">  
 department,  
 </if>  
 <if test="college != null">  
 college,  
 </if>  
 <if test="birth != null">  
 birth,  
 </if>  
 <if test="salary != null">  
 salary,  
 </if>  
 <if test="phone != null">  
 phone,  
 </if>  
 <if test="email != null">  
 email,  
 </if>  
 <if test="address != null">  
 address,  
 </if>  
 <if test="ethnic != null">  
 ethnic,  
 </if>  
 <if test="nativePlace != null">  
 nativePlace,  
 </if>  
 <if test="certificate != null">  
 certificate,  
 </if>  
 <if test="diploma != null">  
 diploma,  
 </if>  
 <if test="schoolOfGraduation != null">  
 schoolOfGraduation,  
 </if>  
 <if test="timeOfGraduation != null">  
 timeOfGraduation,  
 </if>  
 <if test="positions != null">  
 positions,  
 </if>  
 <if test="course != null">  
 course,  
 </if>  
 <if test="clazz != null">  
 clazz,  
 </if>  
 </trim>  
 <trim prefix="values (" suffix=")" suffixOverrides=",">  
 <if test="uid != null">  
 #{uid,jdbcType=BIGINT},  
 </if>  
 <if test="name != null">  
 #{name,jdbcType=VARCHAR},  
 </if>  
 <if test="age != null">  
 #{age,jdbcType=INTEGER},  
 </if>  
 <if test="sex != null">  
 #{sex,jdbcType=VARCHAR},  
 </if>  
 <if test="major != null">  
 #{major,jdbcType=VARCHAR},  
 </if>  
 <if test="department != null">  
 #{department,jdbcType=VARCHAR},  
 </if>  
 <if test="college != null">  
 #{college,jdbcType=VARCHAR},  
 </if>  
 <if test="birth != null">  
 #{birth,jdbcType=VARCHAR},  
 </if>  
 <if test="salary != null">  
 #{salary,jdbcType=VARCHAR},  
 </if>  
 <if test="phone != null">  
 #{phone,jdbcType=VARCHAR},  
 </if>  
 <if test="email != null">  
 #{email,jdbcType=VARCHAR},  
 </if>  
 <if test="address != null">  
 #{address,jdbcType=VARCHAR},  
 </if>  
 <if test="ethnic != null">  
 #{ethnic,jdbcType=VARCHAR},  
 </if>  
 <if test="nativePlace != null">  
 #{nativePlace,jdbcType=VARCHAR},  
 </if>  
 <if test="certificate != null">  
 #{certificate,jdbcType=VARCHAR},  
 </if>  
 <if test="diploma != null">  
 #{diploma,jdbcType=VARCHAR},  
 </if>  
 <if test="schoolOfGraduation != null">  
 #{schoolOfGraduation,jdbcType=VARCHAR},  
 </if>  
 <if test="timeOfGraduation != null">  
 #{timeOfGraduation,jdbcType=VARCHAR},  
 </if>  
 <if test="positions != null">  
 #{positions,jdbcType=VARCHAR},  
 </if>  
 <if test="course != null">  
 #{course,jdbcType=VARCHAR},  
 </if>  
 <if test="clazz != null">  
 #{clazz ,jdbcType=VARCHAR},  
 </if>  
 </trim>  
 </insert>  
 <update id="updateTeacher" parameterType="com.example.second.entity.Teacher">  
 update Teacher  
 <set>  
 <if test="name != null">  
 name = #{name,jdbcType=VARCHAR},  
 </if>  
 <if test="age != null">  
 age = #{age,jdbcType=INTEGER},  
 </if>  
 <if test="sex != null">  
 sex = #{sex,jdbcType=VARCHAR},  
 </if>  
 <if test="major != null">  
 major = #{major,jdbcType=VARCHAR},  
 </if>  
 <if test="department != null">  
 department = #{department,jdbcType=VARCHAR},  
 </if>  
 <if test="college != null">  
 college = #{college,jdbcType=VARCHAR},  
 </if>  
 <if test="birth != null">  
 birth = #{birth,jdbcType=VARCHAR},  
 </if>  
 <if test="salary != null">  
 salary = #{salary,jdbcType=VARCHAR},  
 </if>  
 <if test="phone != null">  
 phone = #{phone,jdbcType=VARCHAR},  
 </if>  
 <if test="email != null">  
 email = #{email,jdbcType=VARCHAR},  
 </if>  
 <if test="address != null">  
 address = #{address,jdbcType=VARCHAR},  
 </if>  
 <if test="ethnic != null">  
 ethnic = #{ethnic,jdbcType=VARCHAR},  
 </if>  
 <if test="nativePlace != null">  
 nativePlace = #{nativePlace,jdbcType=VARCHAR},  
 </if>  
 <if test="certificate != null">  
 certificate = #{certificate,jdbcType=VARCHAR},  
 </if>  
 <if test="diploma != null">  
 diploma = #{diploma,jdbcType=VARCHAR},  
 </if>  
 <if test="schoolOfGraduation != null">  
 schoolOfGraduation = #{schoolOfGraduation,jdbcType=VARCHAR},  
 </if>  
 <if test="timeOfGraduation != null">  
 timeOfGraduation = #{timeOfGraduation,jdbcType=VARCHAR},  
 </if>  
 <if test="positions != null">  
 positions = #{positions,jdbcType=VARCHAR},  
 </if>  
 <if test="course != null">  
 course = #{course,jdbcType=VARCHAR},  
 </if>  
 <if test="clazz != null">  
 clazz = #{clazz ,jdbcType=VARCHAR},  
 </if>  
 </set>  
 where uid = #{uid,jdbcType=BIGINT}  
 </update>  
 <update id="updateByPrimaryKey" parameterType="com.example.second.entity.Teacher">  
 update Teacher  
 set name = #{name,jdbcType=VARCHAR},  
 age = #{age,jdbcType=INTEGER},  
 sex = #{sex,jdbcType=VARCHAR}  
 major= #{major,jdbcType=VARCHAR}  
 department = #{department,jdbcType=VARCHAR}  
 college= #{college,jdbcType=VARCHAR}  
 birth = #{birth,jdbcType=VARCHAR}  
 salary = #{salary,jdbcType=VARCHAR}  
 phone = #{phone,jdbcType=VARCHAR}  
 email = #{email,jdbcType=VARCHAR}  
 address = #{address,jdbcType=VARCHAR}  
 ethnic= #{ethnic,jdbcType=VARCHAR}  
 nativePlace = #{nativePlace,jdbcType=VARCHAR}  
 certificate = #{certificate,jdbcType=VARCHAR}  
 diploma = #{diploma,jdbcType=VARCHAR}  
 schoolOfGraduation = #{schoolOfGraduation,jdbcType=VARCHAR}  
 timeOfGraduation = #{timeOfGraduation,jdbcType=VARCHAR}  
 positions = #{positions,jdbcType=VARCHAR}  
 course = #{course,jdbcType=VARCHAR}  
 clazz = #{clazz,jdbcType=VARCHAR}  
 where uid = #{uid,jdbcType=BIGINT}  
 </update>  
 <update id="updateFileByPrimaryKey" parameterType="com.example.second.entity.Teacher">  
 update Teacher  
 set file = #{file,jdbcType=VARCHAR}  
 where uid = #{uid,jdbcType=BIGINT}  
 </update>  
</mapper>

## Teacher.vue

<template>  
 <div>  
 <div style="margin-top:50px;">  
 <el-input v-model="teacherName" placeholder="请输入老师姓名" style="width:80%" clearable></el-input>  
 <el-button type="primary" style="margin-left: 10px;" @click="searchTeacherByName">搜索</el-button>  
 <el-button v-if="this.user.name==='超级管理员'" type="primary" style="margin-left:10px;margin-bottom: 20px;"  
 @click="showDialog">新增老师  
 </el-button>  
 </div>  
 <div>  
 <el-table  
 @selection-change="handleSelectionChange"  
 :data="temp"  
 style="width: 100%"  
 border>  
 <el-table-column  
 type="selection"  
 width="55">  
 </el-table-column>  
 <el-table-column  
 prop="uid"  
 label="学号"  
 width="180">  
 </el-table-column>  
 <el-table-column  
 prop="name"  
 label="姓名"  
 width="180">  
 </el-table-column>  
 <el-table-column  
 prop="sex"  
 label="性别"  
 width="180">  
 </el-table-column>  
 <el-table-column  
 prop="age"  
 label="年龄"  
 width="180">  
 </el-table-column>  
 <el-table-column  
 prop="major"  
 label="专业"  
 width="180">  
 </el-table-column>  
 <el-table-column  
 prop="department"  
 label="系"  
 width="180">  
 </el-table-column>  
 <el-table-column  
 prop="college"  
 label="学院"  
 width="180">  
 </el-table-column>  
 <el-table-column  
 prop="birth"  
 label="生日"  
 width="180">  
 </el-table-column>  
 <el-table-column  
 prop="salary"  
 label="薪水"  
 width="180">  
 </el-table-column>  
 <el-table-column  
 prop="phone"  
 label="电话"  
 width="180">  
 </el-table-column>  
 <el-table-column  
 prop="email"  
 label="邮箱"  
 width="180">  
 </el-table-column>  
 <el-table-column  
 prop="address"  
 label="地址"  
 width="180">  
 </el-table-column>  
 <el-table-column  
 prop="ethnic"  
 label="民族"  
 width="180">  
 </el-table-column>  
 <el-table-column  
 prop="nativePlace"  
 label="籍贯"  
 width="180">  
 </el-table-column>  
 <el-table-column  
 prop="certificate"  
 label="证书"  
 width="180">  
 </el-table-column>  
 <el-table-column  
 prop="diploma"  
 label="学历"  
 width="180">  
 </el-table-column>  
 <el-table-column  
 prop="schoolOfGraduation"  
 label="毕业学校"  
 width="180">  
 </el-table-column>  
 <el-table-column  
 prop="timeOfGraduation"  
 label="毕业时间"  
 width="180">  
 </el-table-column>  
 <el-table-column  
 prop="positions"  
 label="职位"  
 width="180">  
 </el-table-column>  
 <el-table-column  
 prop="course"  
 label="课程"  
 width="180">  
 </el-table-column>  
 <el-table-column  
 prop="clazz"  
 label="班级"  
 width="180">  
 </el-table-column>  
 <el-table-column  
 label="操作"  
 width="180"  
 align="left">  
 <template slot-scope="scope">  
 <el-button  
 size="mini"  
 @click="handleEdit(scope.row)">编辑  
 </el-button>  
 <el-button  
 size="mini"  
 type="danger"  
 v-if="user.name==='超级管理员'"  
 @click="handleDelete(scope.row)">删除  
 </el-button>  
 </template>  
 </el-table-column>  
 <el-table-column  
 label="简历"  
 width="300"  
 align="left"  
 >  
 <template slot-scope="scope">  
 <el-upload  
 style=" display: inline-block "  
 action="/upload"  
 multiple  
 :data="uploadDate"  
 :before-upload="handleBefore"  
 :limit="3"  
 :on-success="filesuccess"  
 :file-list="fileList">  
 <el-button size="small" type="primary">上传简历</el-button>  
 </el-upload>  
 <el-upload  
 style="display: inline-block;margin-left: 20px"  
 action="/uploadActionUrl"  
 accept="image/jpeg,image/gif,image/png"  
 multiple  
 :limit="3"  
 :on-preview="handlePreview"  
 :on-exceed="handleExceed">  
 <el-button size="small" type="primary">上传照片</el-button>  
 </el-upload>  
  
 <a style="display: inline-block;margin-left: 20px">下载简历</a>  
  
 </template>  
 </el-table-column>  
 </el-table>  
 <div v-if="this.user.name==='超级管理员'" style="position: absolute">  
 <el-button type="danger" size="mini" style="position:relative; float: left; margin-top: 12px"  
 :disabled="multipleSelection.length === 0"  
 @click="deleteTeachers">批量删除  
 </el-button>  
 <el-pagination  
 background  
 layout="prev, pager, next"  
 page-size="5"  
 :current-page="page"  
 :total="this.teachers.length"  
 style="position: relative;margin-top: 12px;margin-left: 280px; float: right"  
 @prev-click="pre"  
 @next-click="next"  
 @current-change="currentChange"  
 >  
 </el-pagination>  
 </div>  
 </div>  
 <el-dialog :title="dialogTitle[dialogStatus]" :visible.sync="dialogFormVisible" center>  
 <el-form :model="form" :rules="rules" ref="teacher">  
 <el-form-item label="姓名" :label-width="formLabelWidth" prop="name">  
 <el-input v-model="form.name" autocomplete="off"></el-input>  
 </el-form-item>  
 <el-form-item label="性别" :label-width="formLabelWidth" prop="sex">  
 <el-select v-model="form.sex" placeholder="请选择" style="width: 100%">  
 <el-option  
 v-for="item in options"  
 :key="item.value"  
 :value="item.label">  
 </el-option>  
 </el-select>  
 </el-form-item>  
 <el-form-item label="年龄" :label-width="formLabelWidth" prop="age">  
 <el-input v-model.number="form.age" autocomplete="off"></el-input>  
 </el-form-item>  
 <el-form-item label="专业" :label-width="formLabelWidth" prop="major">  
 <el-select v-model="form.major" placeholder="请选择" style="width: 100%" filterable allow-create>  
 <el-option  
 v-for="item in majors"  
 :key="item.value"  
 :value="item.label">  
 </el-option>  
 </el-select>  
 </el-form-item>  
 <el-form-item label="系" :label-width="formLabelWidth" prop="department">  
 <el-select v-model="form.department" placeholder="请选择" style="width: 100%" filterable allow-create>  
 <el-option  
 v-for="item in departments"  
 :key="item.value"  
 :value="item.label">  
 </el-option>  
 </el-select>  
 </el-form-item>  
 <el-form-item label="学院" :label-width="formLabelWidth" prop="college">  
 <el-select v-model="form.college" placeholder="请选择" style="width: 100%" filterable allow-create>  
 <el-option  
 v-for="item in colleges"  
 :key="item.value"  
 :value="item.label">  
 </el-option>  
 </el-select>  
 </el-form-item>  
 <el-form-item label="生日" :label-width="formLabelWidth" prop="birth">  
 <el-date-picker  
 v-model="form.birth"  
 type="month"  
 format="yyyy.MM"  
 placeholder="选择">  
 </el-date-picker>  
 </el-form-item>  
 <el-form-item label="薪水" :label-width="formLabelWidth" prop="salary">  
 <el-input v-model.number="form.salary" autocomplete="off"></el-input>  
 </el-form-item>  
 <el-form-item label="电话" :label-width="formLabelWidth" prop="phone">  
 <el-input v-model="form.phone" autocomplete="off"></el-input>  
 </el-form-item>  
 <el-form-item label="邮箱" :label-width="formLabelWidth" prop="email">  
 <el-input v-model="form.email" autocomplete="off"></el-input>  
 </el-form-item>  
 <el-form-item label="地址" :label-width="formLabelWidth" prop="address">  
 <el-input v-model="form.address" autocomplete="off"></el-input>  
 </el-form-item>  
 <el-form-item label="民族" :label-width="formLabelWidth" prop="ethnic">  
 <el-input v-model="form.ethnic" autocomplete="off"></el-input>  
 </el-form-item>  
 <el-form-item label="籍贯" :label-width="formLabelWidth" prop="nativePlace">  
 <el-input v-model="form.nativePlace" autocomplete="off"></el-input>  
 </el-form-item>  
 <el-form-item label="证书" :label-width="formLabelWidth" prop="certificate">  
 <el-input v-model="form.certificate" autocomplete="off"></el-input>  
 </el-form-item>  
 <el-form-item label="学历" :label-width="formLabelWidth" prop="diploma">  
 <el-input v-model="form.diploma" autocomplete="off"></el-input>  
 </el-form-item>  
 <el-form-item label="毕业学校" :label-width="formLabelWidth" prop="schoolOfGraduation">  
 <el-input v-model="form.schoolOfGraduation" autocomplete="off"></el-input>  
 </el-form-item>  
 <el-form-item label="毕业时间" :label-width="formLabelWidth" prop="timeOfGraduation">  
 <el-date-picker  
 v-model="form.timeOfGraduation"  
 type="month"  
 format="yyyy.MM"  
 placeholder="选择">  
 </el-date-picker>  
 </el-form-item>  
 <el-form-item label="职位" :label-width="formLabelWidth" prop="positions">  
 <el-input v-model="form.positions" autocomplete="off"></el-input>  
 </el-form-item>  
 <el-form-item label="课程" :label-width="formLabelWidth" prop="course">  
 <el-input v-model="form.course" autocomplete="off"></el-input>  
 </el-form-item>  
 <el-form-item label="班级" :label-width="formLabelWidth" prop="clazz">  
 <el-input v-model="form.clazz" autocomplete="off"></el-input>  
 </el-form-item>  
 </el-form>  
 <div slot="footer" class="dialog-footer">  
 <el-button @click="dialogFormVisible = false">取 消</el-button>  
 <el-button type="primary" @click="submitTeacher">确 定</el-button>  
 </div>  
 </el-dialog>  
 </div>  
</template>  
  
<script>  
 *// @ is an alias to /src* export default {  
 name: 'Home',  
 components: {},  
 data() {  
 return {  
 user: JSON.parse(window.sessionStorage.getItem("user")),  
 url:'',  
 uploadDate: {  
 uid: '',  
 },  
 fileList: [],  
 teacherName: '',  
 multipleSelection: [],  
 dialogTitle: {  
 addTeacher: "新增老师",  
 editTeacher: "编辑老师"  
 },  
 dialogStatus: "",  
 rules: {  
 name: [  
 {required: true, message: '请输入姓名', trigger: 'change'}  
 ],  
 sex: [  
 {required: true, message: '请选择性别', trigger: 'change'}  
 ],  
 age: [  
 {required: true, message: '年龄不能为空'},  
 {type: 'number', message: '年龄必须为数字值'}  
 ],  
 email: [{required: true, message: '请输入邮箱地址', trigger: 'blur'},  
 {type: 'email', message: '请输入正确的邮箱地址', trigger: ['blur', 'change']}  
 ],  
 phone:  
 [{required: true, message: '请输入手机号码', trigger: 'blur'},  
 {  
 validator: function (rule, value, callback) {  
 if (/^1[34578]\d{9}$/.test(value) == false) {  
 callback(new Error("请输入正确的手机号"));  
 } else {  
 callback();  
 }  
 }, trigger: 'change'  
 }],  
 },  
 options: [{  
 value: '1',  
 label: '男'  
 }, {  
 value: '2',  
 label: '女'  
 }],  
 majors: [{  
 value: '软件工程',  
 label: '软件工程'  
 }, {  
 value: '会计学',  
 label: '会计学'  
 }, {  
 value: '车辆工程',  
 label: '车辆工程'  
 }, {  
 value: '国际经济与贸易',  
 label: '国际经济与贸易'  
 }, {  
 value: '电子信息科学与技术',  
 label: '电子信息科学与技术'  
 }, {  
 value: '经济学',  
 label: '经济学'  
 }],  
 departments: [  
 {  
 value: '金融系',  
 label: '金融系'  
 }, {  
 value: '会计系',  
 label: '会计系'  
 }, {  
 value: '计算机系',  
 label: '计算机系'  
 }, {  
 value: '法律系',  
 label: '法律系'  
 }  
 ],  
 colleges: [  
 {  
 value: '经济金融学院',  
 label: '经济金融学院'  
 }, {  
 value: '会计学院',  
 label: '会计学院'  
 }, {  
 value: '计算机学院',  
 label: '计算机学院'  
 }, {  
 value: '人工智能学院',  
 label: '人工智能学院'  
 }, {  
 value: '车辆学院',  
 label: '车辆学院'  
 }  
 ],  
  
 msg: '',  
 teachers: [],  
 temp: [],  
 page: 1,  
 max: '',  
 search: '',  
 dialogFormVisible: false,  
 form: {  
 name: '',  
 age: '',  
 sex: '',  
 major: '',  
 department: '',  
 college: '',  
 birth: '',  
 salary: '',  
 phone: '',  
 email: '',  
 address: '',  
 ethnic: '',  
 nativePlace: '',  
 certificate: '',  
 diploma: '',  
 schoolOfGraduation: '',  
 timeOfGraduation: '',  
 positions: '',  
 course: '',  
 clazz: '',  
 },  
 formLabelWidth: '120px',  
 }  
 },  
 created() {  
 let user = JSON.parse(window.sessionStorage.getItem("user"))  
 console.log(user)  
 if (user.name === "超级管理员")  
 this.findAllTeachers();  
 else  
 this.searchTeacherByName(user.name)  
  
 },  
 watch: {  
 teacherName() {  
 this.findAllTeachers();  
 }  
 },  
 methods: {  
 searchTeacherByName(name) {  
 if (this.teacherName !== '') {  
 this.$axios.get("/first/findTeacherByName?name=" + this.teacherName).then((resp) => {  
 this.teachers = resp.data;  
 this.temp = this.teachers  
  
 }).catch((error) => {  
 this.$message({  
 type: 'error',  
 message: "查询失败，原因是" + error.data.message  
 });  
 })  
 }else{  
 this.$axios.get("/first/findTeacherByName?name=" + name).then((resp) => {  
 this.teachers = resp.data;  
 console.log(this.teachers)  
 this.temp = this.teachers  
  
 }).catch((error) => {  
 this.$message({  
 type: 'error',  
 message: "查询失败，原因是" + error.data.message  
 });  
 })  
 }  
 },  
 deleteTeachers() {  
 this.$confirm('此操作将永久删除老师, 是否继续?', '提示', {  
 confirmButtonText: '确定',  
 cancelButtonText: '取消',  
 type: 'warning'  
 }).then(() => {  
 let ids = '?';  
 this.multipleSelection.forEach((item) => {  
 ids += 'ids=' + item.uid + '&'  
 });  
 this.$axios.post("/first/deleteTeachersByIds" + ids).then((resp) => {  
 if (resp) {  
 this.findAllTeachers();  
 this.$message.success("删除成功！");  
 }  
 })  
 }).catch(() => {  
 this.$message({  
 type: 'info',  
 message: '已取消删除'  
 });  
 });  
 },  
 handleSelectionChange(val) {  
 this.multipleSelection = val;  
 console.log(val)  
 },  
 submitTeacher() {  
 if (this.dialogStatus === 'addTeacher') {  
 this.addTeacher();  
 } else if (this.dialogStatus === "editTeacher") {  
 this.editTeacher();  
 }  
 },  
 findAllTeachers() {  
 this.$axios.get("/first/findAllTeachers")  
 .then((res) => {  
 console.log(res)  
 this.teachers = res.data  
 console.log(this.teachers.length)  
 if (this.teachers.length % 5 === 0)  
 this.max = *parseInt*(this.teachers.length / 5)  
 else  
 this.max = *parseInt*(this.teachers.length / 5) + 1  
 console.log(this.max)  
 this.temp = this.teachers.slice((this.page - 1) \* 5, this.page \* 5)  
  
 })  
 .catch(err => {  
 *alert*("查询失败");  
 console.log(err);  
 });  
 },  
 handleEdit(row) {  
 console.log(row)  
 this.dialogFormVisible = true;  
 this.dialogStatus = "editTeacher";  
 this.form.uid = row.uid;  
 this.form.name = row.name;  
 this.form.age = row.age;  
 this.form.sex = row.sex;  
 this.form.major = row.major;  
 this.form.department = row.department;  
 this.form.college = row.college;  
 this.form.birth = row.birth;  
 this.form.salary = row.salary;  
 this.form.phone = row.phone;  
 this.form.email = row.email;  
 this.form.address = row.address;  
 this.form.ethnic = row.ethnic;  
 this.form.nativePlace = row.nativePlace;  
 this.form.certificate = row.certificate;  
 this.form.diploma = row.diploma;  
 this.form.schoolOfGraduation = row.schoolOfGraduation;  
 this.form.timeOfGraduation = row.timeOfGraduation;  
 this.form.positions = row.positions;  
 this.form.course = row.course;  
 this.form.clazz = row.clazz;  
 console.log("form:" + this.form.diploma)  
  
 },  
 handleDelete(row) {  
 this.$confirm('此操作将永久删除该老师, 是否继续?', '提示', {  
 confirmButtonText: '确定',  
 cancelButtonText: '取消',  
 type: 'warning'  
 }).then(() => {  
 this.$axios.post("/first/deleteTeacher?uid=" + row.uid).then((resp) => {  
 this.$message.success("删除成功!")  
 this.findAllTeachers();  
 })  
 }).catch(() => {  
 this.$message({  
 type: 'info',  
 message: '已取消删除'  
 });  
 });  
 },  
 showDialog() {  
 this.form.uid = '';  
 this.form.name = '';  
 this.form.age = '';  
 this.form.sex = '';  
 this.form.major = '';  
 this.form.department = '';  
 this.form.college = '';  
 this.form.birth = '';  
 this.form.salary = '';  
 this.form.phone = '';  
 this.form.email = '';  
 this.form.address = '';  
 this.form.ethnic = '';  
 this.form.nativePlace = '';  
 this.form.certificate = '';  
 this.form.diploma = '';  
 this.form.schoolOfGraduation = '';  
 this.form.timeOfGraduation = '';  
 this.form.positions = '';  
 this.form.course = '';  
 this.form.clazz = '';  
 this.dialogFormVisible = true;  
 this.dialogStatus = "addTeacher";  
 },  
 addTeacher() {  
 this.$refs.teacher.validate((valid) => {  
 if (valid) {  
 this.$axios.post("/first/addTeacher", this.form).then((resp) => {  
 if (resp) {  
 this.dialogFormVisible = false;  
 this.$message.success("添加成功！");  
 this.findAllTeachers();  
 }  
 }).catch((error) => {  
 this.$message({  
 message: '添加数据失败，原因是' + error.data.message,  
 type: 'error'  
 })  
 })  
 } else {  
 this.$message({  
 message: '请输入所有字段',  
 type: 'error'  
 })  
 return false;  
 }  
 });  
 },  
 editTeacher() {  
 this.$axios.post("/first/updateTeacher", this.form).then((resp) => {  
 this.dialogFormVisible = false;  
 this.$message.success("修改成功!");  
 this.findAllTeachers();  
 }).catch((error) => {  
 this.$message({  
 message: '数据更新失败，原因是' + error.data.message,  
 type: 'error'  
 })  
 })  
 },  
 pre() {  
 if (this.page != 1) {  
 this.page--  
 this.temp = this.teachers.slice((this.page - 1) \* 5, this.page \* 5)  
 console.log("pre:")  
 console.log(+this.temp)  
 }  
 },  
 next() {  
 if (this.page != this.max) {  
 this.page++  
 this.temp = this.teachers.slice((this.page - 1) \* 5, this.page \* 5)  
 *// for(var i = 0;i < this.temp.length;i++){  
 // console.log(this.temp[i]);  
 // if(this.temp[i].icon == 'el-icon-check'){  
 // console.log("mmm")  
 // }  
 // }* this.$forceUpdate()  
 }  
 },  
 currentChange(val) {  
 this.page = val  
 this.temp = this.teachers.slice((this.page - 1) \* 5, this.page \* 5)  
 *// for(var i = 0;i < this.temp.length;i++){  
 // console.log(this.temp[i]);  
 // if(this.temp[i].icon == 'el-icon-check'){  
 // console.log("mmm")  
 // }  
 // }* console.log("page:" + this.page)  
 this.$forceUpdate()  
 },  
 handleBefore() {  
 this.uploadDate.uid = this.user.id  
 },  
  
 handlePreview(file){  
 console.log("当前文件的id"+file.id);  
 var a = document.createElement('a')  
 var event = new MouseEvent('click')  
 console.log("当前教师信息文件id：" + file.id)  
 a.href = '/download?id=' + file.id  
 a.dispatchEvent(event)  
 },  
  
 }  
 }  
</script>  
<style scoped>  
 .table {  
 margin: 0 auto;  
 clear: both;  
 }  
</style>

## UploadMapper.xml

<!DOCTYPE mapper  
 PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  
 "http://mybatis.org/dtd/mybatis-3-mapper.dtd"*>*<mapper namespace="com.example.second.dao.UploadDao">  
  
 <insert id="insertFile" parameterType="com.example.second.entity.TeacherFile">  
 insert into file (name,newname,uid,url) values (#{name},#{newname},#{uid},#{url})  
 </insert>  
  
 <select id="findAllFiles" parameterType="com.example.second.entity.User" resultType="com.example.second.entity.TeacherFile">  
 select *\** from file  
 </select>  
  
 <select id="findById" parameterType="java.lang.Integer" resultType="com.example.second.entity.TeacherFile">  
 select *\** from file where id=#{id}  
 </select>  
  
</mapper>

FileService

package com.example.second.service;  
  
  
import org.springframework.web.multipart.MultipartFile;  
  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
  
public interface FileService {  
  
 public String uploadFile(MultipartFile file, HttpServletRequest request,int uid);  
  
 public void uploadActionUrl(MultipartFile file, HttpServletRequest request);  
  
 public void download(HttpServletResponse response, HttpServletRequest request);  
  
}

## FileServiceBean

package com.example.second.service.impl;  
  
  
import com.example.second.dao.UploadDao;  
import com.example.second.entity.TeacherFile;  
import com.example.second.service.FileService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import org.springframework.web.multipart.MultipartFile;  
  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.\*;  
import java.net.URLEncoder;  
import java.util.UUID;  
  
@Service  
public class FileServiceBean implements FileService {  
  
 @Autowired  
 UploadDao uploadDao;  
  
 public String uploadFile(MultipartFile file, HttpServletRequest request, int uid) {  
 try {  
 String name = file.getOriginalFilename();  
 String realPath = "E:\\file\_upload\\";  
 String newname = UUID.*randomUUID*().toString() + name;  
 File folder = new File(realPath, newname);  
  
 if (!folder.exists()) {  
 folder.mkdirs();  
 }  
 TeacherFile teacherFile = new TeacherFile();  
 teacherFile.setName(name);  
 teacherFile.setNewname(newname);  
 teacherFile.setUid(uid);  
 teacherFile.setUrl(realPath + newname);  
 uploadDao.insertFile(teacherFile);  
 file.transferTo(folder);  
 return name;  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return null;  
 }  
  
 public void uploadActionUrl(MultipartFile file, HttpServletRequest request) {  
 try {  
 String name = file.getOriginalFilename();  
 String realPath = "E:\\image\_upload\\";  
 File folder = new File(realPath, name);  
 if (!folder.exists()) {  
 folder.mkdirs();  
 }  
 file.transferTo(folder);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 public void download(HttpServletResponse response, HttpServletRequest request) {  
  
 Integer id = Integer.*parseInt*(request.getParameter("id"));  
 TeacherFile teacherFile = uploadDao.findById(id);  
  
 try {  
 *// path是指想要下载的文件的路径* File file = new File(teacherFile.getUrl());  
 FileInputStream fileInputStream = new FileInputStream(file);  
 InputStream fis = new BufferedInputStream(fileInputStream);  
 byte[] buffer = new byte[fis.available()];  
 fis.read(buffer);  
 fis.close();  
  
 *// 清空response* response.reset();  
 *// 设置response的Header* response.setCharacterEncoding("UTF-8");  
 *//Content-Disposition的作用：告知浏览器以何种方式显示响应返回的文件，用浏览器打开还是以附件的形式下载到本地保存  
 //attachment表示以附件方式下载 inline表示在线打开 "Content-Disposition: inline; filename=文件名.mp3"  
 // filename表示文件的默认名称，因为网络传输只支持URL编码的相关支付，因此需要将文件名URL编码后进行传输,前端收到后需要反编码才能获取到真正的名称* response.addHeader("Content-Disposition", "attachment;filename=" + URLEncoder.*encode*(teacherFile.getName(), "UTF-8"));  
 *// 告知浏览器文件的大小* response.addHeader("Content-Length", "" + file.length());  
 OutputStream outputStream = new BufferedOutputStream(response.getOutputStream());  
 response.setContentType("application/octet-stream");  
 outputStream.write(buffer);  
 outputStream.flush();  
 System.*out*.println("下载完成");  
 } catch (IOException ex) {  
 ex.printStackTrace();  
 }  
 }  
}

## UploadDao

package com.example.second.dao;  
  
  
import com.example.second.entity.TeacherFile;  
import com.example.second.entity.User;  
import org.apache.ibatis.annotations.Mapper;  
import org.springframework.stereotype.Component;  
  
import java.util.List;  
@Mapper  
@Component(value = "UploadDao")  
public interface UploadDao {  
 void insertFile(TeacherFile teacherFile);  
 List<TeacherFile> findAllFiles(User user);  
 TeacherFile findById(Integer id);  
  
}

## FileController

package com.example.second.controller;  
  
  
import com.example.second.service.FileService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.transaction.annotation.Transactional;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestParam;  
import org.springframework.web.bind.annotation.RestController;  
import org.springframework.web.multipart.MultipartFile;  
  
  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.IOException;  
  
@Transactional  
@RestController  
public class FileController {  
  
 @Autowired  
 private FileService fileService;  
  
  
 @PostMapping("/upload")  
 public String uploadFile(MultipartFile file, HttpServletRequest request,int uid) throws IOException {  
 return this.fileService.uploadFile(file, request,uid);  
 }  
  
 @PostMapping("/uploadActionUrl")  
 public void uploadActionUrl(MultipartFile file, HttpServletRequest request) throws IOException {  
 this.fileService.uploadActionUrl(file, request);  
 }  
  
 @RequestMapping("/download")  
 public void download(HttpServletResponse response, HttpServletRequest request) {  
  
 this.fileService.download(response, request);  
 }  
  
  
}