# 截图



图1 学生管理系统页面



图2 教师管理系统页面(仅管理员)



图3 查看教师信息



图4 新增教师

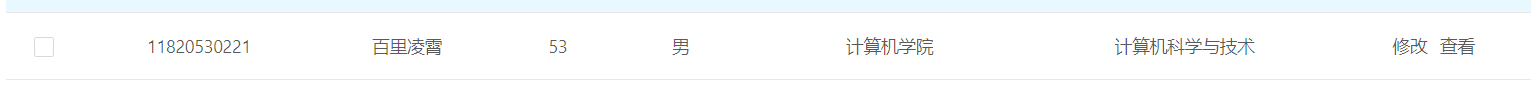


图5 添加成功



图6 修改教师信息

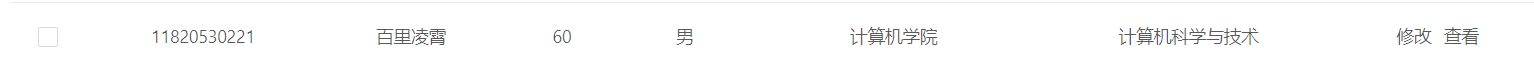


图7 修改成功



图8 删除教师信息



图9 删除成功



图10 登录



图11 个人中心(仅教师)



图12 更改头像



图13 修改个人信息



图14 修改成功



图15 下载简历

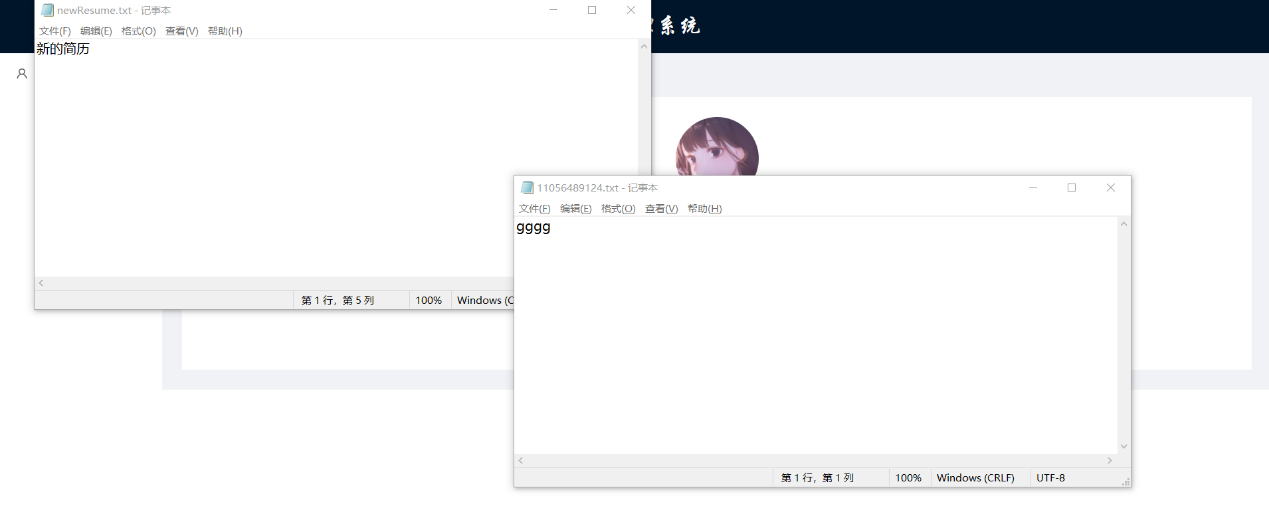


图16 准备上传的新简历

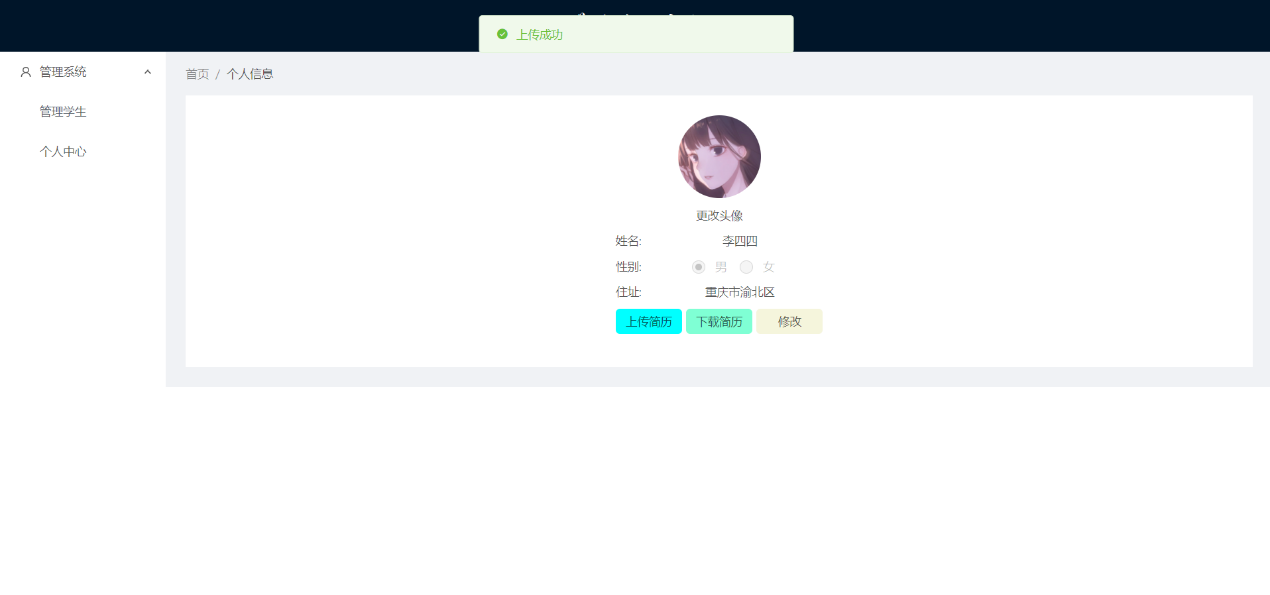


图17 上传成功

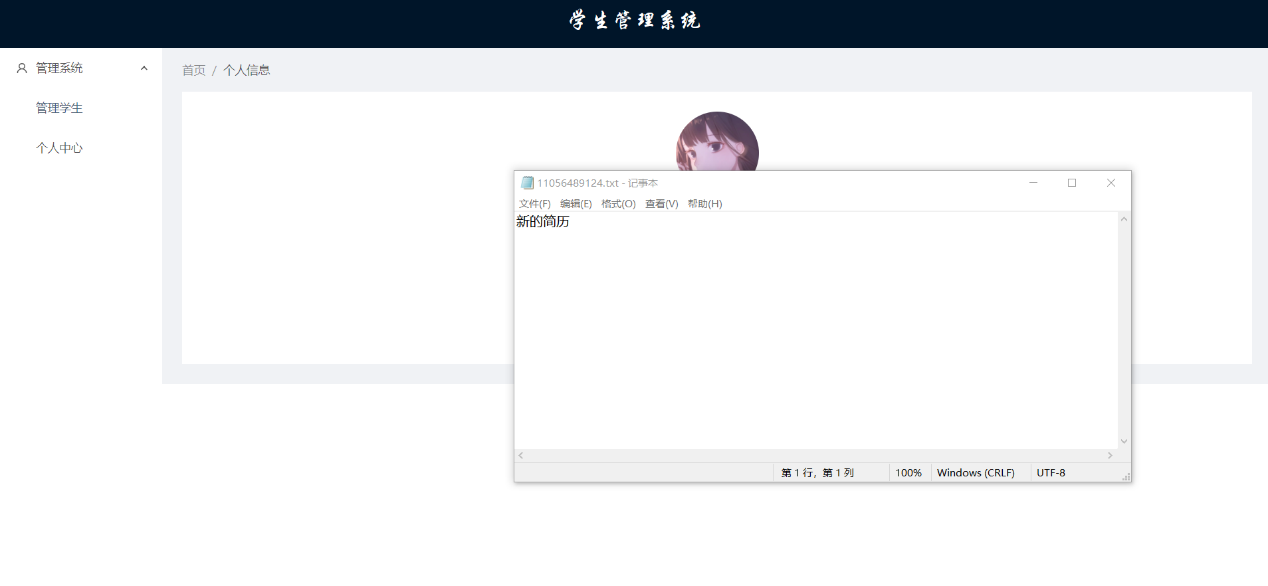


图18 重新上传后的简历

# 代码

JWT代码：

package edu.cqut.ouyangjianrong.studentmanagemybatis.utils;  
  
import com.auth0.jwt.JWT;  
import com.auth0.jwt.JWTCreator;  
import com.auth0.jwt.algorithms.Algorithm;  
import com.auth0.jwt.interfaces.DecodedJWT;  
  
import java.util.Calendar;  
import java.util.Map;  
  
public class JWTUtils {  
 private static final String *SIGNATURE* = "!jfioJF@%J@L^&jpojf/f.>!%sdfGJ";  
  
 */\*\*  
 \* 生成token  
 \*/* public static String getToken(Map<String, String> map) {  
 Calendar instance = Calendar.*getInstance*();  
 *// 过期时间* instance.add(Calendar.*DATE*, 1);  
  
 *// 创建JWT builder* JWTCreator.Builder builder = JWT.*create*();  
  
 map.forEach(builder::withClaim);  
  
 return builder.withExpiresAt(instance.getTime())  
 .sign(Algorithm.*HMAC256*(*SIGNATURE*));  
 }  
  
 */\*\*  
 \* 验证token合法性，并返回token信息  
 \* @param token  
 \*/* public static DecodedJWT verify(String token) {  
 return JWT.*require*(Algorithm.*HMAC256*(*SIGNATURE*)).build().verify(token);  
 }  
}

JWT的拦截器：

package edu.cqut.ouyangjianrong.studentmanagemybatis.config;  
  
import com.auth0.jwt.exceptions.AlgorithmMismatchException;  
import com.auth0.jwt.exceptions.SignatureVerificationException;  
import com.auth0.jwt.exceptions.TokenExpiredException;  
import com.fasterxml.jackson.databind.ObjectMapper;  
import edu.cqut.ouyangjianrong.studentmanagemybatis.utils.JWTUtils;  
import org.springframework.web.servlet.HandlerInterceptor;  
  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.util.HashMap;  
import java.util.Map;  
  
public class JWTInterceptor implements HandlerInterceptor {  
 @Override  
 public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {  
 Map<String, Object> map = new HashMap<>();  
 *// 获取请求中令牌* String token = request.getHeader("token");  
  
 try {  
 *// 验证令牌* JWTUtils.*verify*(token);  
 return true;  
 } catch (SignatureVerificationException e) {  
 e.printStackTrace();  
 map.put("msg", "无效签名");  
 } catch (TokenExpiredException e) {  
 e.printStackTrace();  
 map.put("msg", "token过期");  
 } catch (AlgorithmMismatchException e) {  
 e.printStackTrace();  
 map.put("msg", "token算法不一致！");  
 } catch (Exception e) {  
 e.printStackTrace();  
 map.put("msg", "token无效!!");  
 }  
 map.put("state", false);  
 *// 将map转为json* String json = new ObjectMapper().writeValueAsString(map);  
 response.setContentType("application/json;charset=UTF-8");  
 response.getWriter().println(json);  
  
 return false;  
 }  
}

注册拦截器：

package edu.cqut.ouyangjianrong.studentmanagemybatis.config;  
  
import org.springframework.context.annotation.Configuration;  
import org.springframework.web.servlet.config.annotation.InterceptorRegistry;  
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;  
  
@Configuration  
public class InterceptorConfig implements WebMvcConfigurer {  
 @Override  
 public void addInterceptors(InterceptorRegistry registry) {  
 registry.addInterceptor(new JWTInterceptor())  
 .addPathPatterns("/\*\*")  
 .excludePathPatterns("/user/login");  
 }  
}

前端路由拦截：

router.beforeEach((to, from, next) => {  
 if (to.meta.requireAuth) {  
 if (window.localStorage.return\_state !== undefined && window.localStorage.return\_state == "true") {  
 *// token有效* const role = to.meta.role.filter(el => el == window.localStorage.role)  
 if (role.length !== 0) {  
 next()  
 } else {  
 router.back()  
 }  
 } else {  
 *// token无效或者过期* next({  
 path: '/'  
 })  
 }  
 } else {  
 next()  
 }  
})

前端网络响应拦截：

request.interceptors.response.use(  
 response => {  
 if (response.data.state !== undefined) {  
 store.commit(STATE, response.data.state)  
 }  
 return response  
 },  
 err => {  
 if (err.response) {  
 switch (err.response.status) {  
 case 404:  
 console.log("404 code")  
 break  
 case 500:  
 console.log("500 code")  
 }  
 }  
 return Promise.reject(err)  
 }  
)