

《专业技能训练—数据库应用系统开发》 报告

课程设计题目：学生选课管理信息系统

班 级： 18 科技 2 班

姓 名： 刘文鑫

学 号： 201824111260

任课教师： 李泓波

开始时间： 2020 年 9 月 15 日

结束时间： 2020 年 12 月 8 日

成 绩： _____

目录

1、 项目前言.....	3
1.1 项目背景.....	3
1.2 开发环境.....	3
2、 需求分析.....	5
2.1 功能要求.....	5
2.2 处理要求.....	7
2.3 安全性与完整性要求.....	8
3、 模型设计.....	9
3.1 E-R 图.....	9
3.2 数据库表.....	9
3.3 UI 设计.....	11
4、 主要模块的设计以及代码.....	18
4.1 项目模块.....	18
4.2 前端项目架构.....	19
4.3 后端项目架构.....	21
4.4 代码思想.....	22
4.5 代码实现.....	22
5、 总结.....	30
6、 参考文献.....	34

1、项目前言

1.1 项目背景

时代在进步，科技也不断进步，时代对于人才的需求也在不断的发生改变，教育制度的改革也在进行着，而大学的教育制度尤其重要。现在越来越多的高校采用学分制，高校大学生可根据自身兴趣爱好选择自己喜爱的课程来学习。基于高校校园网的学生选课系统不仅克服了以往手工报送选课方式所暴露出的缺点与不足，同时也极大提高了教务工作人员的效率，极大地方便了师生。学生选课系统采用 B/S 架构进行开发，用户只需输入相应的网址即可登录学生选课系统，并在系统进行相应的操作。

1.2 开发环境

后端:

jdk: jdk1.8

语言: java+springboot+mybatis 框架

开发工具: intellij idea

数据库:mysql5.5+navicat

数据交互格式:json

前后端通信协议:http 协议

服务器:tomcat 9.0

版本控制器:maven

缓存:redis

部署环境:

前端:阿里云 centos7 下的 nginx 服务器

后端:阿里云 centos7 下的 springboot 内置 tomcat 服务器

项目接口:http://47.112.145.214

项目管理:git

远程仓库:gitee

测试工具:postman

前端:

语言:vue(前端页面数据渲染)+element-ui(页面组件和样式显示)+iview(页面组件和
样式显示)+echart(数据可视化)

开发工具:webstorm

打包:webpack

架构:cli3

浏览器:火狐和 chrome

模块化:es6

网络: axios 框架(对 ajax 的进一步封装)

服务器: nginx

2、需求分析

2.1 功能要求

1. “信息查询”功能模块

- 1) 查询学生信息
- 2) 查询教师信息
- 3) 查询课程信息
- 4) 查询院系信息
- 5) 查询选课信息

注：信息查询可基于单条件查询也可基于多条件复合查询。

2. “信息录入”功能模块

- 1) 录入学生信息
- 2) 录入教师信息
- 3) 录入课程信息
- 4) 录入院系信息
- 5) 录入选课信息

注：信息录入时，需要考虑数据信息的完整性、有效性等。

3. “信息删除”功能模块

- 1) 删除学生信息
- 2) 删除教师信息
- 3) 删除课程信息
- 4) 删除院系信息
- 5) 删除选课信息

注：可单条记录删除，也可批量删除，执行该项操作时需要考虑删除所需的约束条件

4. “信息修改”功能模块

- 1) 修改学生信息
- 2) 修改教师信息
- 3) 修改课程信息
- 4) 修改院系信息
- 5) 修改选课信息

注：进行修改操作时，需要保证数据的一致性

5. “信息浏览”功能模块

- 1) 浏览学生信息
- 2) 浏览教师信息
- 3) 浏览课程信息
- 4) 浏览院系信息
- 5) 浏览选课信息

注：通过系统提供的相关界面对 student、teacher、course、department、sct 等数据表中的内容进行浏览。

6. “数据报表”功能模块

- 1) 学生信息报表 //注册总人数 男女比例
- 2) 教师信息报表 //教师总人数 各学历人数 各职称人数
- 3) 课程信息报表 //课程总数 根据学分统计课程
- 4) 院系信息报表 //学院总数 各学院总人数 男女比例
- 5) 选课信息报表 //每个学生修了多少门课 每门课程多少学生修

注：按照一定的格式在相应的窗口界面上显示学生、教师、课程、院系、选课等信息，能按一定条件进行统计信息显示，并提供通过**打印机打印输出的功能**。

7. 用户管理与用户登陆功能模块

系统可根据需要添加、删除用户，并可对已有的用户信息进行修改操作；在添加新用户时，可定义其操作权限（查询、更新、浏览、报表等权限）；用户密码可进行随时修改；各种不同身份的用户登陆系统后，可享有不同的系统操作权限。

8. 系统帮助及使用说明功能模块

为用户提供必要的在线帮助功能和简要的操作使用说明。

2.2 处理要求

学生选课系统查询操作相应时间不超过 10s，允许每月系统停机一小时，进行系统服务。

系统服务的时间要安排在学习非工作时间进行。

2.3 安全性与完整性要求

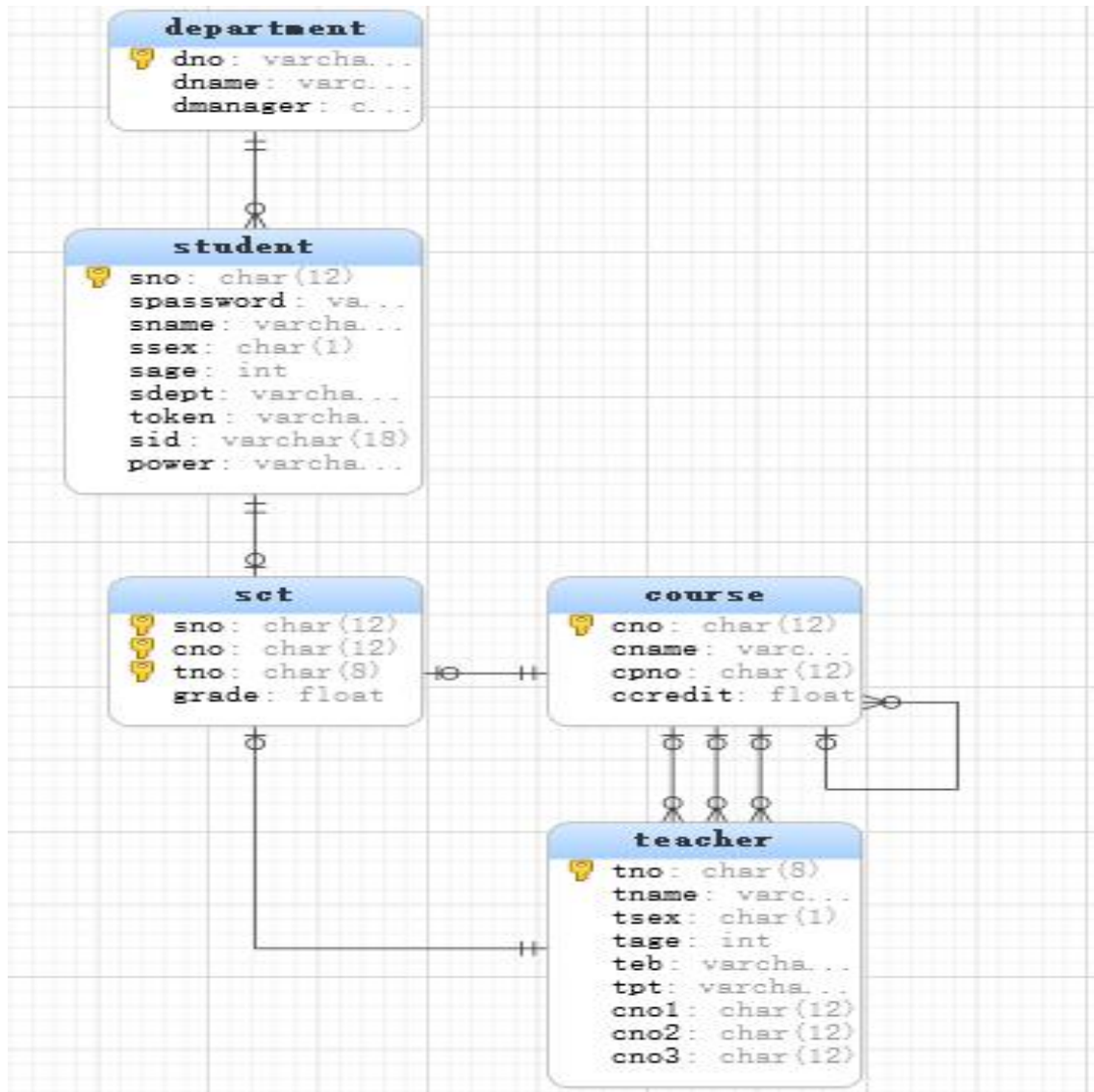
(1) 针对本系统我们对用户密码进行加密，以保证各级用户对系统访问的安全性。生成的口令不可逆转（用 MD5 加密是一种 32 位字符的加密方法）。输入的口令不应显示在显示终端上。

(2) 针对系统运行出现的异常，跟踪调查出现异常的情况，了解操作意图，有针对性的解决问题。系统日志:便于查看系统的运行情况。操作日志:提供用户在系统中增加、修改系统数据信息时记录日志。用于跟踪用户的操作，了解信息的变更，在需要时对事情进行调查。

(3) 页面不可以直接访问，需要携带 token 令牌进行验证，验证通过方可访问页面，否则跳回登录页面重新登录获取 token。

3、模型设计

3.1 E-R 图



3.2 数据库表

student 表

用于记录学生信息

字段名	字段数据类型	字段属性	备注
sno	char(12)	not null	学号 主键自增
spassword	varchar(25)	not null	密码

sname	varchar(8)	not null	学生姓名
ssex	char(1)	not null	性别 男或女
sage	int(4)	not null	年龄
sid	varchar(18)	not null	身份证
sdept	varchar(30)	not null	部门 与 department 表中的 dname 形成外键关系
power	varchar(5)	not null	普通学生、学生管理员

course 表

用于记录课程信息

字段名	字段数据类型	字段属性	备注
cno	char(12)	not null	课程号 主键自增
cname	varchar(50)	not null	课程名
cpno	char(12)	null	先行课编号 与 cno 形成外键关系
ccredit	float(2,1)	not null	课程学分

teacher 表

用于记录教师信息

字段名	字段数据类型	字段属性	备注
tno	char(8)	not null	教工号 主键自增
tname	varchar(8)	not null	教师名
tsex	char(1)	not null	教师性别
tage	int(4)	not null	教师年龄
teb	varchar(10)	not null	学历 (学士、硕士、博士)
tpt	varchar(10)	not null	职称(助教、讲师、副教授、教授)
cno1	char(12)	null	主讲课程一 与 course 表的 cno 形成外键关系
cno2	char(12)	null	主讲课程二 与 course 表的 cno 形成外键关系
cno3	char(12)	null	主讲课程三 与 course 表的 cno 形成外键关系

department 表
用于记录院系信息

字段名	字段数据类型	字段属性	备注
dno	varchar(30)	not null	系编号 主键自增
dname	varchar(30)	not null	系名
dmanager	char(8)	not null	系主任

sct 表
选课信息表

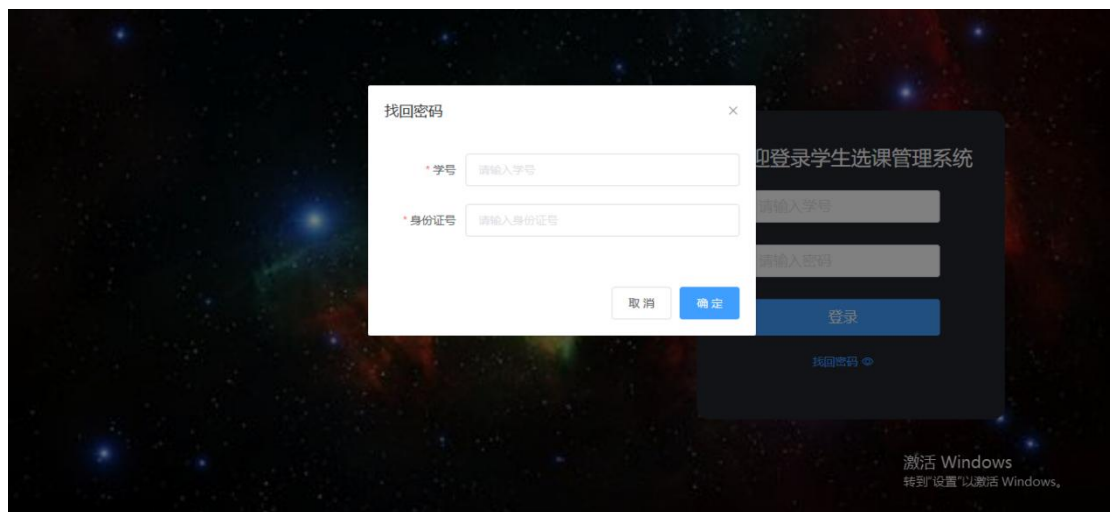
字段名	字段数据类型	字段属性	备注
sno	char(12)	not null	学号 主键 与 student 表的 sno 形成外键关系
cno	char(12)	not null	课程号 主键 与 course 表的 cno 形成外键关系
tno	char(8)	not null	教工号 主键 与 teacher 表的 tno 形成外键关系
grade	float(4,1)	默认值 0	成绩

3.3 UI 设计

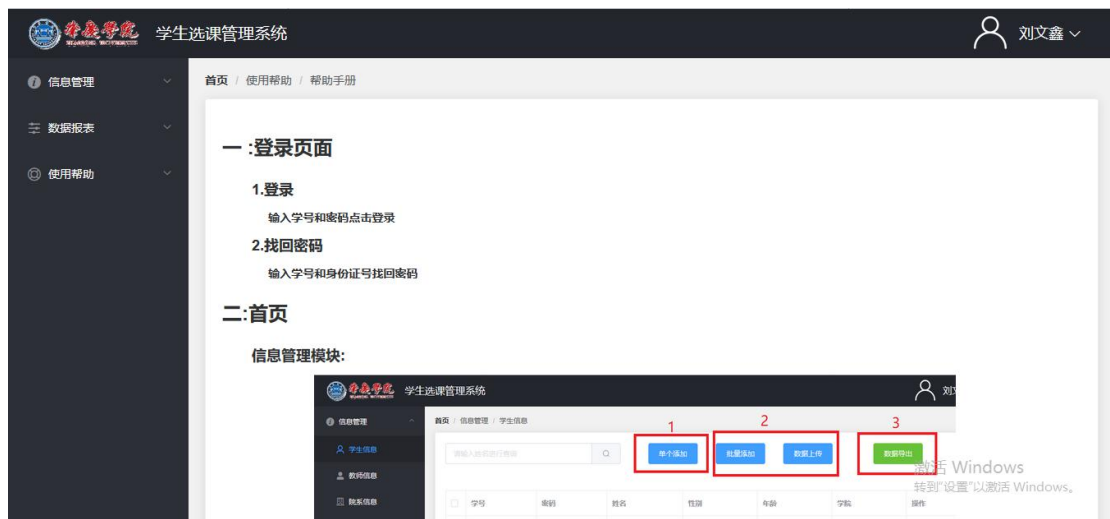
(1)登录页面



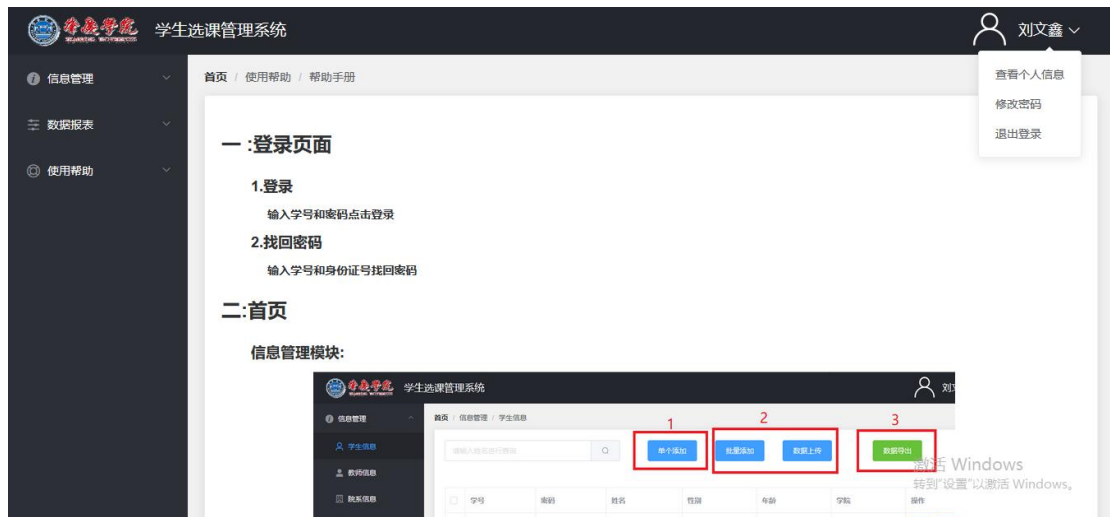
(2) 找回密码



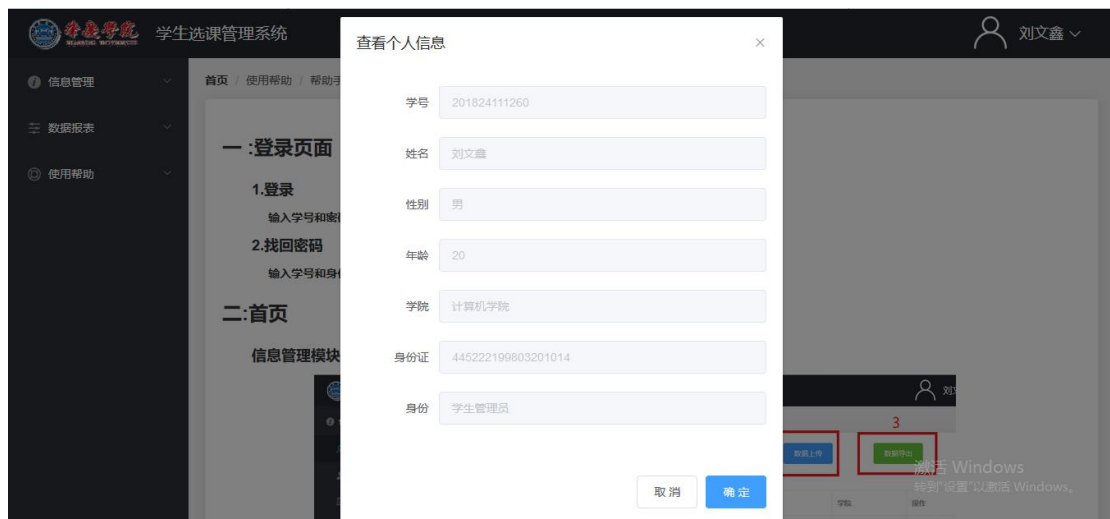
(3) 首页(帮助页)



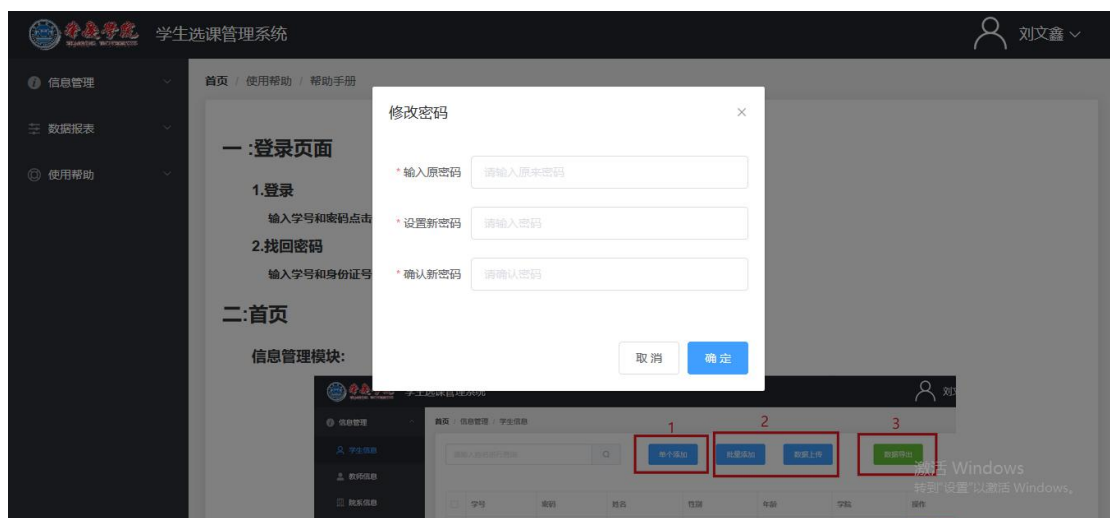
(4) 个人模块(注意右上角)



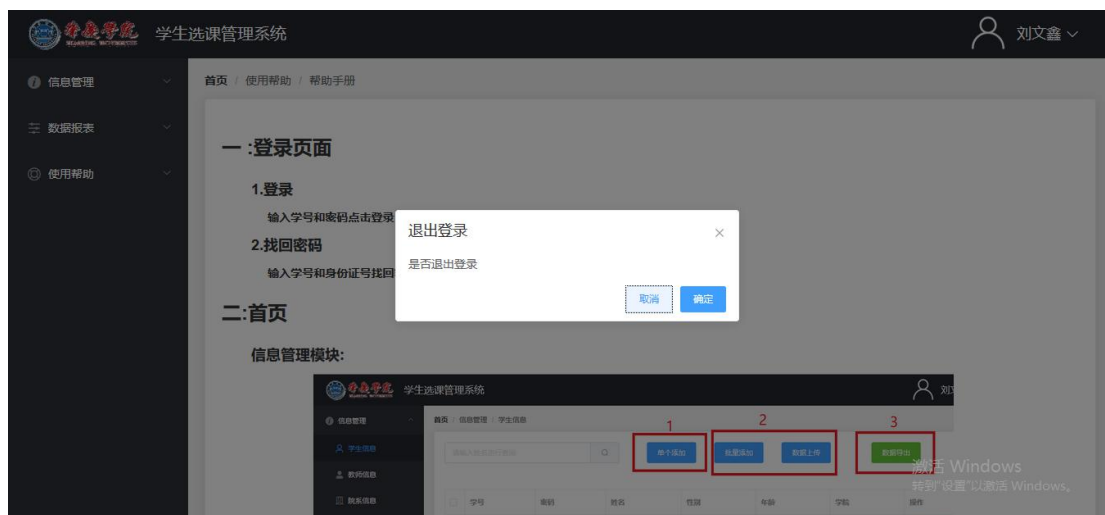
(5) 查看个人信息



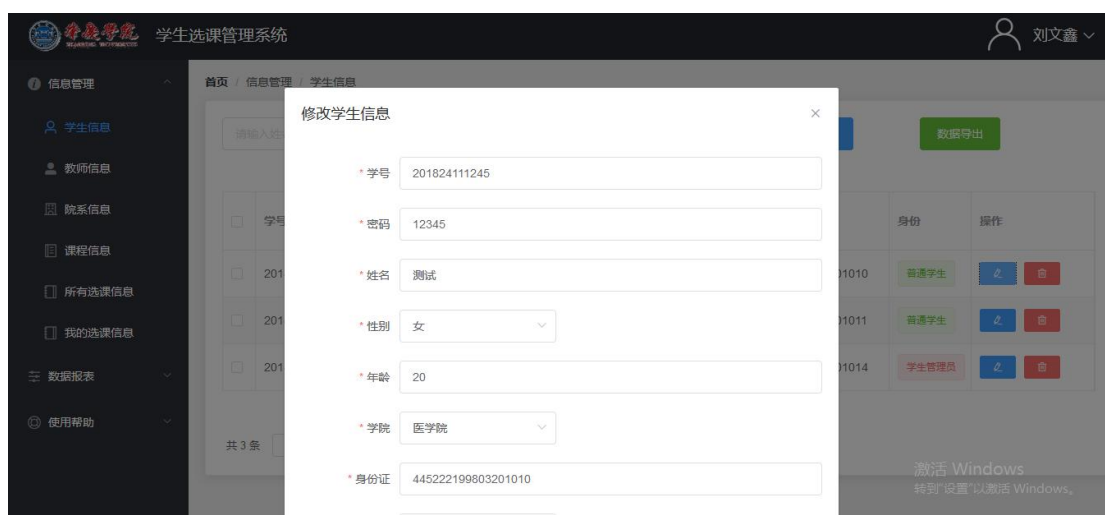
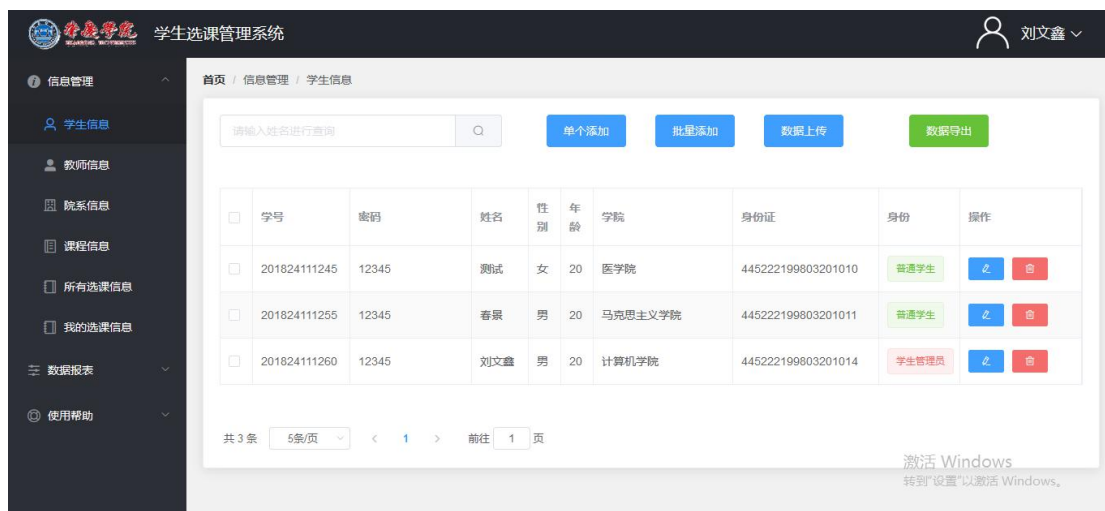
(6) 修改密码




(7) 退出登录



(8) 信息管理





学生选课管理系统

刘文章

信息管理

学生信息

教师信息

院系信息

课程信息

所有选课信息

我的选课信息

数据报表


使用帮助

首页 / 信息管理 / 选课信息

<input type="checkbox"/>	学生	教师	课程编号	课程名	学分	成绩	操作
<input type="checkbox"/>	春景	吴招根	11	android	1	0	<input type="button" value="修改"/> <input type="button" value="删除"/>
<input type="checkbox"/>	刘文章	吴招根	11	android	1	90	<input type="button" value="修改"/> <input type="button" value="删除"/>
<input type="checkbox"/>	刘文章	张弹	22	数据结构	1	0	<input type="button" value="修改"/> <input type="button" value="删除"/>

共 3 条 前往 页

激活 Windows
转到“设置”以激活 Windows。



学生选课管理系统

刘文章

信息管理

学生信息

教师信息

院系信息

课程信息

所有选课信息

我的选课信息

数据报表

使用帮助


首页 / 信息管理 / 我的选课信息

<input type="checkbox"/>	学生	教师	课程编号	课程名	学分	成绩	操作
<input type="checkbox"/>	刘文章	吴招根	11	android	1	90	<input type="button" value="修改"/> <input type="button" value="删除"/>
<input type="checkbox"/>	刘文章	张弹	22	数据结构	1	0	<input type="button" value="修改"/> <input type="button" value="删除"/>

共 2 条 前往 页

激活 Windows
转到“设置”以激活 Windows。

(9) 信息报表



学生选课管理系统

刘文章

信息管理

数据报表

学生报表

教师报表

院系报表


课程报表

选课报表

使用帮助


首页 / 数据报表 / 学生报表

学生总人数




3

男生总人数



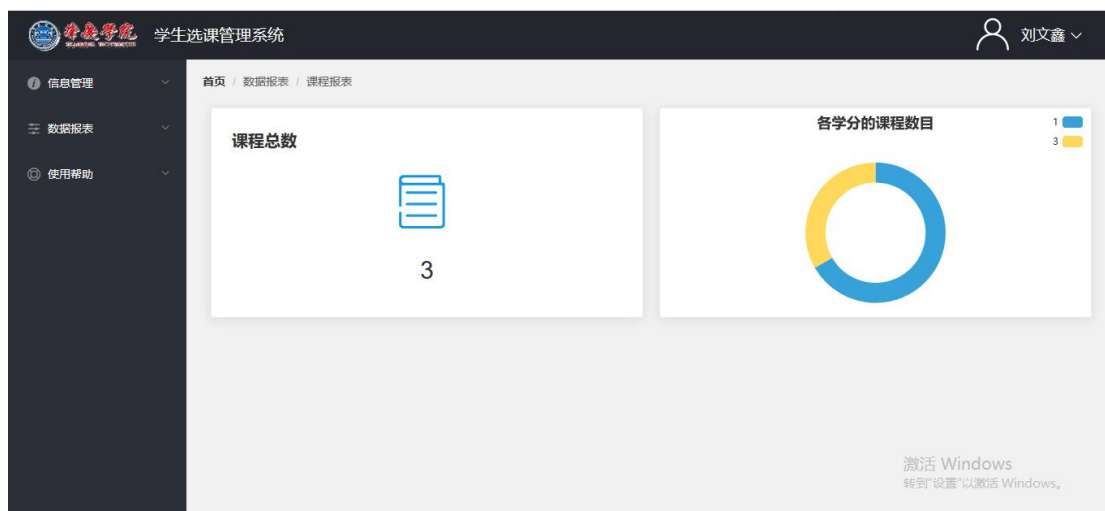
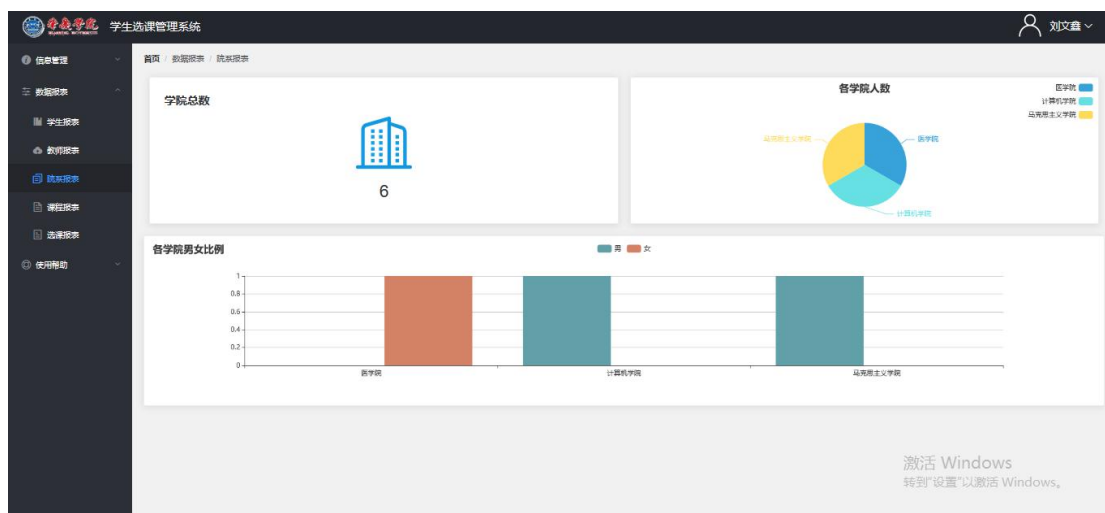
2

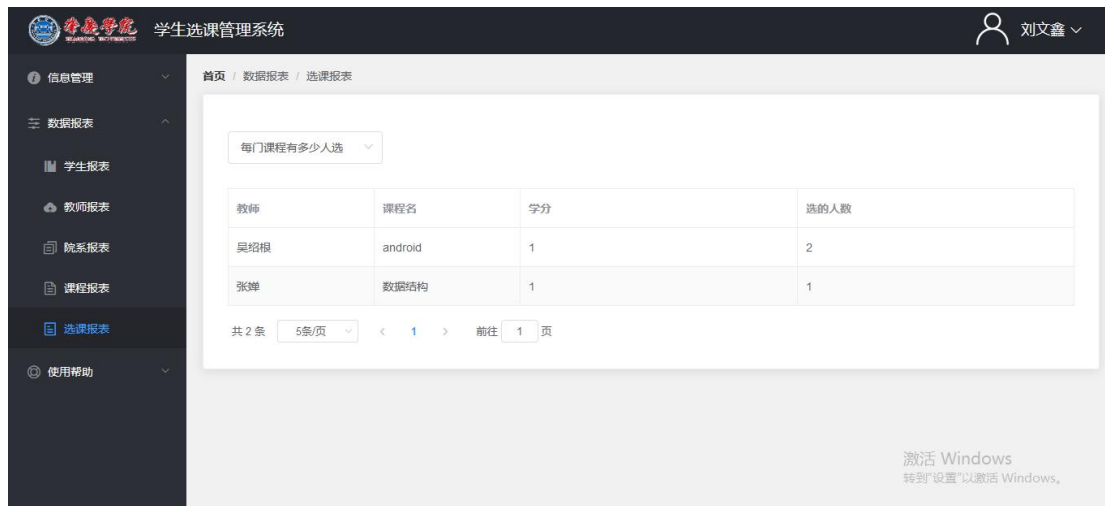
女生总人数



1

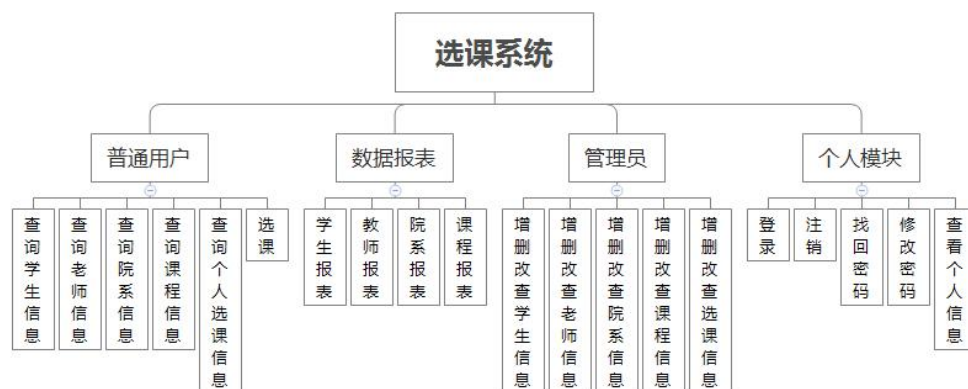
激活 Windows
转到“设置”以激活 Windows。



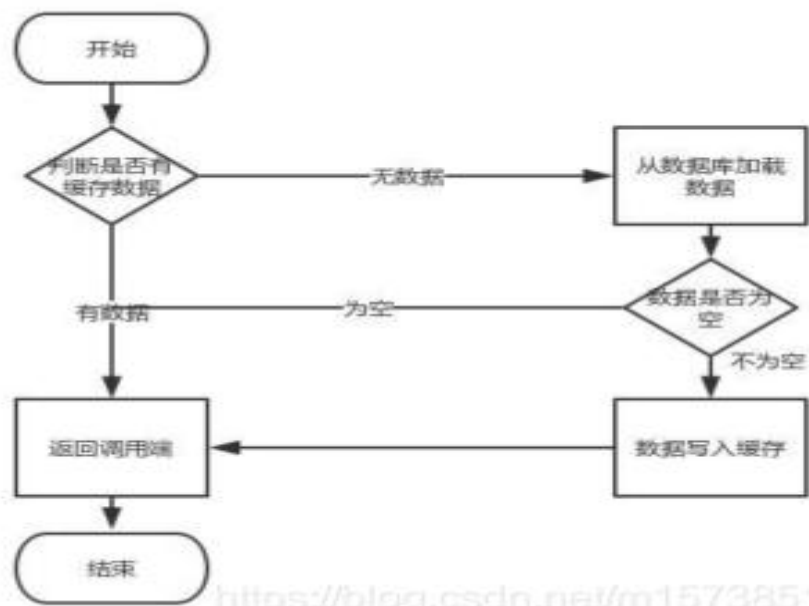


4、主要模块的设计以及代码

4.1 项目模块

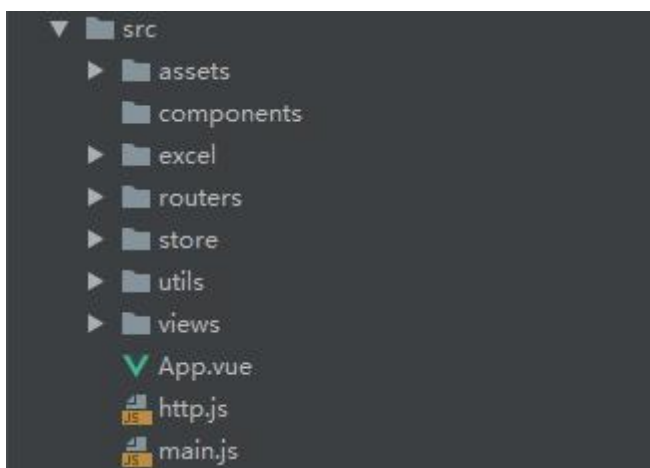


功能模块



缓存原理

4.2 前端项目架构



assets 下放静态资源如:image、js、css

components 下放组件

excel 下是 excel 工具类，将表格数据导出到 excel 表中

router 下是路由，记录前端页面跳转的路径

store 下是状态管理管家，用于管理全局变量

utils 下是 url 接口

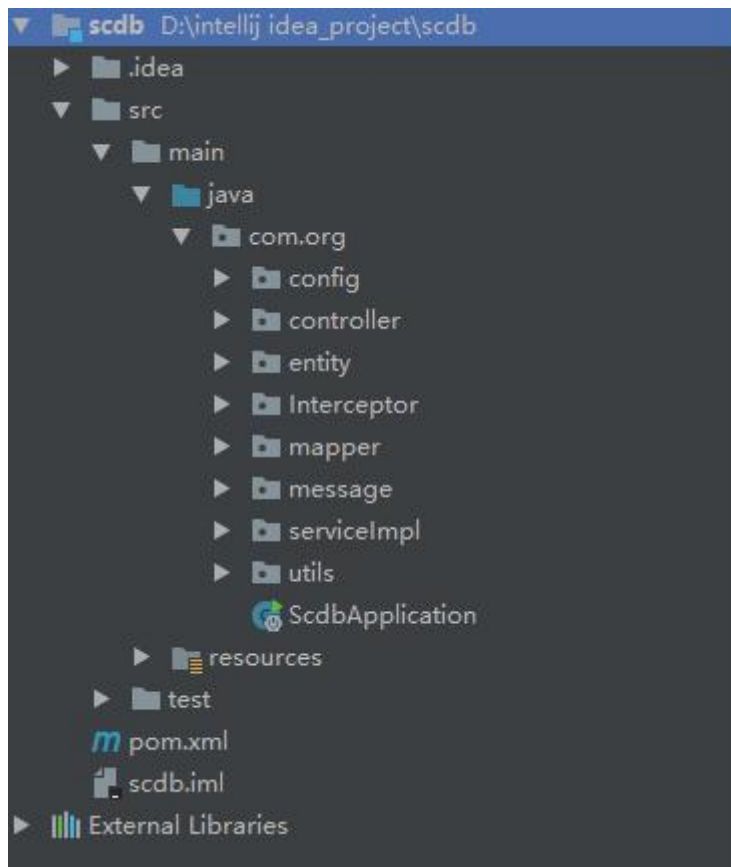
views 下是页面视图以及视图逻辑部分以及渲染部分

App.vue 是全局视图，是所有视图的父视图

http.js 是 http 访问的拦截请求工具，用于判断请求是否携带 token，以及服务器端相应是否携带 token 数据

main.js 是定义全局变量、创建 vue 实例以及实现模块通信的重要文件

4.3 后端项目架构



config 下是配置类

controller 下是将 serviceImpl 文件下的业务逻辑接口实现类的处理结果以 json 的数据格式请求转发到前端

entity 下是实体类的定义，是对象与表一对一的关系映射

Interceptor 下是拦截器类

mapper 下是数据库相关操作

service 下是定义业务逻辑的接口

serviceImpl 下是业务逻辑接口的实现类

message 下是后端返回给前端数据的一个封装类，用于规范后端返回给前端的数据类型

utils 是工具类，对 excel 文件的读取、token 令牌的判断

最后一个后端程序的启动类

4.4 代码思想

用户登录之后会在后台通过 UUID 随机生成 token 字符串，并把它保存在 redis 缓存里，之后返回给用户浏览器，用户浏览器会将 token 保存到本地，以后每次访问都会自动携带 token，用于唯一标识用户。此时后端添加拦截器，用于判断用户是否携带 token 令牌，若无携带，则提示用户重新登陆，重新获取 token 令牌。若有携带令牌，则完成相对应的业务操作。此项目用到了 redis 缓存，redis 缓存的目的是为了减轻 mysql 数据库的压力，特别是当数据量大、并发量大的时候。redis 是为了提高查询速度和查询效率，redis 适用于经常查询但不经常修改的数据。当前端需要完成查询操作，拦截器放行前端的请求之后，会先去 redis 缓存里查询数据，若缓存中存在数据，则将查询到的结果返回给前端。若缓存中不存在数据，则去 mysql 中查询相应数据，并把读取到的结果写入缓存和返回给前端。缓存更新的时间是 1 分钟。

4.5 代码实现

前端:

```
//get 的请求方式
this.$axios({
  method:'get',
  url:url    //后端接口
},{
  params:params    //参数列表
}).then(response=>{

  //response 为请求成功时，后端返回的结果，并将结果赋值给 data()，然后再由
  data 渲染到<template>中
```

```
}).catch(error=>{  
    error 为请求失败时的结果  
})
```

//post 的请求方式

```
this.$axios({  
    method:'post',  
    data: data,    //访问后端携带的数据  
    url:url    //后端接口  
}).then(response=>{  
    //response 为请求成功时，后端返回的结果，并将结果赋值给 data()，然后再由  
    data 渲染到<template>中
```

```
}).catch(error=>{  
    error 为请求失败时的结果  
})
```

后端:

config:

```
Configuration  
public class InterceptorConfig extends WebMvcConfigurationSupport{  
  
    @Autowired  
    Interceptor interceptor;  
  
    @Override  
    protected void addResourceHandlers(ResourceHandlerRegistry  
registry) {  
        //放行的静态资源  
        registry.addResourceHandler("/**").addResourceLocations(  
            "classpath:/static/");  
  
        registry.addResourceHandler("swagger-ui.html").addResourceLocations(  
            "classpath:/META-INF/resources/");  
  
        registry.addResourceHandler("/webjars/**").addResourceLocations(  
            "classpath:/META-INF/resources/webjars/");  
        super.addResourceHandlers(registry);  
    }  
  
    @Override
```

```

protected void addInterceptors(InterceptorRegistry registry) {
    //设置拦截器拦截请求，以及拦截器放行的请求
    registry.addInterceptor(interceptor).addPathPatterns("/**")
        .excludePathPatterns("/login")
        .excludePathPatterns("/findPassword")
        .excludePathPatterns("/swagger-resources/**",
"/webjars/**", "/v2/**", "/swagger-ui.html/**");
}
}

```

```

@Configuration
public class RedisConfig {
    //设置数据存入 redis 的序列化规则，不设置的话数据存入 redis 会出现乱码
    @Bean
    public RedisTemplate<String, Object>
redisTemplate(RedisConnectionFactory redisConnectionFactory) throws
UnknownHostException {
    RedisTemplate<String, Object> template = new RedisTemplate<>();
    //修改默认的序列化规则
    //1.创建序列化规则对象
    Jackson2JsonRedisSerializer jackson2JsonRedisSerializer=new
Jackson2JsonRedisSerializer(Object.class);

    //2.更改默认的序列化规则
    template.setKeySerializer(new StringRedisSerializer());
    template.setValueSerializer(jackson2JsonRedisSerializer);

    template.setConnectionFactory(redisConnectionFactory);
    return template;
}
}

```

mapper:

//操作数据库语句，并将查询到的结果映射成对象

@Select("select * from student") Student selectStudent();

@service //表面此类是业务逻辑类

serviceImpl:

```
@resource
StudentMapper studentMapper;

Gson gson=new Gson();

Msg selectStudent (HttpServletRequest request,HttpServletResponse response){
    TokenUtil.addToken(request,response);

    //Msg 里面封装了请求的状态码、数据结果、以及错误原因
    return Msg.success().add("data",studentMapper.selectStudentByNo(sno));
}
```

Interceptor: //拦截器对有无 token 令牌的拦截具体实现

```
@Component
public class Interceptor implements HandlerInterceptor{

    @Resource
    StringRedisTemplate stringRedisTemplate;
    //拦截器类
    public boolean preHandle(HttpServletRequest request,
    HttpServletResponse response, Object handler) throws Exception {
        if(request.getMethod().equals("OPTIONS")){
            //浏览器在发送请求时会默认先发送一次类型为'option'且不帶任何参
            //数的请求，请求成功后才会发送真正的 Post 或者 get 请求
            // //设置 option 请求通过
            return true;
        }

        String token=request.getHeader("token");
        if(token!=null){
            //有 token
            String sno=stringRedisTemplate.opsForValue().get(token);
            if (sno!=null){
                //token 值存在
                return true;
            }else {
                //token 值不存在
                returnJson(response);
                return false;
            }
        }
        }else {
```

```

        //无 token
        returnJson(response);
        return false;
    }
}

public void returnJson(HttpServletResponse response){
    PrintWriter writer=null;
    response.setCharacterEncoding("utf-8");
    response.setContentType("application/json; charset=utf-8");
    try {
        writer=response.getWriter();
        Msg msg=Msg.fail().add("detailMessage","没有 token 或 token
失效");
        Gson gson=new Gson();
        writer.print(gson.toJson(msg));
    }catch (Exception e){
        e.printStackTrace();
    }finally {
        if(writer!=null){
            writer.close();
        }
    }
}
}
}

```

@RestController //表面此类是控制器类

@CrossOrigin(allowCredentials = "true", allowedHeaders = "*") //解决跨域问题
controller

@resource

ServiceImpl serviceImpl;

```

Message selectStudent (HttpServletRequest req,HttpServletResponse res){
    return serviceImpl.selectStudent (req,res);
}

```

utils: //里面放的都是工具类

RedisUtil: //封装了对 redis 操作的方法

//封装了对 redis 操作的方法

@Component

public class RedisUtil {

@Resource

RedisTemplate redisTemplate;

@Resource

StringRedisTemplate stringRedisTemplate;

//redis 中操作 string 类型数据

public Object getStringValue(String key, Object Object, int time){

//获取 string 的 value 值

ValueOperations<String, Object>

operations=redisTemplate.opsForValue();

if(!redisTemplate.hasKey(key)){

operations.set(key, Object, time, TimeUnit.SECONDS);

}

return operations.get(key);

}

public String getStringValue(String key, String data){

ValueOperations<String, String>

operations=stringRedisTemplate.opsForValue();

if(!stringRedisTemplate.hasKey(key)){

operations.set(key, data, 60, TimeUnit.SECONDS);

}

return operations.get(key);

}

//redis 中操作 list 类型的数据

public List<Object> getListValue(String key, Object object){

ListOperations<String, Object>

operations=redisTemplate.opsForList();

if(!redisTemplate.hasKey(key)){

operations.leftPushAll(key, object);

redisTemplate.expire(key, 60, TimeUnit.SECONDS);

}

System.out.println(operations.range(key, 0, -1));

return operations.range(key, 0, -1);

}

}

TokenUtil: //从请求头中读取 token, 以及将 token 添加到响应头中

```
public class TokenUtil {
    //验证 token 是否失效
    /* public static Student testToken(HttpServletRequest request,
    StudentMapper studentMapper) {
        String token = request.getHeader("token");
        Student
selectStudentByToken=studentMapper.selectStudentByToken(token);
        if(token!=null&& !token.equals("")&& !token.equals("0") &&
selectStudentByToken!=null ){
            return selectStudentByToken;
        }else{
            return null;
        }
    }*/
    //在响应头添加 token
    public static void addToken(HttpServletRequest
request,HttpServletResponse response) {

        response.setHeader("Access-Control-Expose-Headers",
"Cache-Control,Content-Type,Expires,Pragma,Content-Language,Last-Mod
ified,token");
        response.addHeader("token", request.getHeader("token"));
    }
}
```

ExcelUtil: //读取 excel 文件的工具类

```
//excel 的读取工具类
public class ExcelUtil {

    public static Workbook getWorkbook(InputStream in, String fileName)
throws Exception {
        Workbook workbook = null;
        String fileType =
fileName.substring(fileName.lastIndexOf(".")); //获得后缀
        if (".xls".equals(fileType)) {
            workbook = new HSSFWorkbook(in);
        } else if (".xlsx".equals(fileType)) {
            workbook = new XSSFWorkbook(in);
        } else {
            throw new Exception("请上传 excel 文件!");
        }
    }
}
```

```

    }
    return workbook;
}

public static List<Student> getMultipleStudent(InputStream in,
String fileName){
    List<Student> students=new ArrayList<>();
    try{
        Workbook wb = getWorkbook(in, fileName);//获得正确的流
        Sheet sheet=wb.getSheetAt(0);
        for(int r = 1; r <= sheet.getLastRowNum(); r++) {    //遍历
行
            Row row = sheet.getRow(r);
            if(row ==null){
                continue;
            }
            row.getCell(0).setCellType(CellType.STRING);
            row.getCell(1).setCellType(CellType.STRING);
            row.getCell(2).setCellType(CellType.STRING);
            row.getCell(3).setCellType(CellType.STRING);
            row.getCell(5).setCellType(CellType.STRING);
            row.getCell(6).setCellType(CellType.STRING);
            row.getCell(7).setCellType(CellType.STRING);
            Student student=new Student();
            student.setSno(row.getCell(0).getStringCellValue());

student.setSpassword(row.getCell(1).getStringCellValue());
            student.setSname(row.getCell(2).getStringCellValue());
            student.setSsex(row.getCell(3).getStringCellValue());

student.setSage((int)row.getCell(4).getNumericCellValue());
            student.setSdept(row.getCell(5).getStringCellValue());
            student.setSid(row.getCell(6).getStringCellValue());
            student.setPower(row.getCell(7).getStringCellValue());
            students.add(student);
        }
    }catch (Exception e){
        e.printStackTrace();
    }
    return students;
}
}

```

5、总结

开发中遇到的问题以及解决方法

(1)问题:前端页面出现不适配的现象,在不同的浏览器下不兼容,页面上的组件随着浏览器窗口的改变而改变。

解决:页面提前布局好,将页面分成四个模块:顶部、侧边栏、内容区、底部,宽度用百分比

(2)问题:前后端通讯时出现跨域问题

解决:在后端 `controller` 加上一行注解

`@CrossOrigin(allowCredentials = "true", allowedHeaders = "*")`,解决跨域问题

(3)问题:后端如何识别任意用户

解决:在学生表中增添一个 `token` 字段,用户初次登录便使用 `UUID` 随机生成一个字符串赋值给 `token`,并保存到数据库当中,每次访问后端前,前端的请求头都要携带 `token`,只有 `token` 存在才能实现访问后端操作。处理完后端操作,在响应请求头中将 `token` 带回。

(4)问题:前端页面组件中如何通信

解决:`vue` 的思想是模块化思想,各个模块之间即相互独立又可以相互协同,每次需要不同模块的数据,需要在

```
<script>
```

```
import xxx from xxx
```

```
</script>
```

，引入别的模块

如果是全局都会使用到，则在 `main.js` 中引入，

通过 `Vue.prototype.$axios = axios` 将要使用到的对象传递给全局

(5)问题:前端项目如何打包

解决: `npm run build`

(6)问题:后端项目如何打包

解决:通过 `maven` 中的 `package`

(7)问题:项目需求到底是什么

解决:项目需求就是项目想要实现什么样的功能、实现什么样的效果。要根据不同情况进行分析。

(8) 问题:将本地项目添加到远程仓库 `gitee` 怎么样都不成功

解决: 在 `git push origin master` 之前需要 `git pull --rebase origin master` 将远程仓库与本地仓库进行同步，如果远程仓库为空，记得将项目保存一份，要不然可能发现代码丢失的风险。

(9) 问题: 拦截器拦截不成功

解决: 浏览器在发送请求时会默认先发送一次类型为 `option` 且不携带任何参数的请求，请求成功后才会发送真正的 `post` 或者 `get` 请求，所以拦截器要设置 `option` 类型的请求通过

```
if(request.getMethod().equals("OPTIONS")){  
    //浏览器在发送请求时会默认先发送一次类型为'option'且不帶任何参数的请求，请求成功后才会发送真正的 Post 或者 get 请求  
    // /设置 option 请求通过  
    return true;  
}
```

(10) 问题: 前端项目加载缓慢、项目加载出现空白

解决: npm run build 将所有 js 都打包在一个 js 中, 包括所有项目引用到的第三方工具, 而这些第三方工具是非常大的, 所以要将这些第三方工具进行压缩。通过 gzip 减小文件体积, 提高文件首页加载速度。通过 externals 加载外部 cdn 资源, 凡是声明在 externals 中的第三方依赖包, 都不会打包到 dist 文件夹。

(11) 问题: 通过 externals 加载外部 cdn 资源失败

解决:

在 vue.config.js 中添加如下代码

key 为要引入的包名,value 为 cdn 中的全局对象, 也是你项目中要引入的对象

```
config.set('externals',{  
  
    vue:'Vue',  
  
    'vue-router':'router',  
  
    axios:'axios',  
  
    echarts: 'echarts',  
  
    nprogress:'nprogress'  
  
}))
```


在 public 中的 index.html 的头部加上这个:

```
<!--nprogress 样式-->
<link rel="stylesheet"
href="https://cdn.staticfile.org/nprogress/0.2.0/nprogress.min.css">

<script src="https://cdn.staticfile.org/vue/2.6.11/vue.min.js"></script>
<script
src="https://cdn.staticfile.org/axios/0.18.0/axios.min.js"></script>
<script
src="https://cdn.staticfile.org/vue-router/3.1.1/vue-router.min.js"></script>

<script
src="https://cdn.staticfile.org/element-ui/2.13.2/index.js"></script>
<script
src="https://cdn.staticfile.org/echarts/4.7.0/echarts.min.js"></script>
<script
src="https://cdn.staticfile.org/nprogress/0.2.0/nprogress.min.js"></script>
<script src="https://cdn.bootcss.com/vuex/3.5.1/vuex.min.js"></script>
<!-- <script src="https://cdn.bootcss.com/qs/6.5.2/qs.min.js"></script>-->
<!-- element-ui 的 css 文件 -->
<link rel="stylesheet"
href="https://cdn.staticfile.org/element-ui/2.13.2/theme-chalk/index.css">
<script
src="https://cdn.staticfile.org/xlsx/0.16.6/xlsx.core.min.js"></script>
```

去掉原先的 import xx from xx 避免资源冲突

开发体会:

一个人独自完成前后端项目的设计、代码编写、维护、升级,感觉成就感巨大,我不仅仅是一个只会前端或者只会后端的程序员,而是全栈的程序员,极大增强了我对编程的信息与兴趣,让我对软件开发流程又有了一个全新、全面的了解和认识。但同时也暴露了一个问题---就是消耗时间比较多,虽然一个人能做出所有东西,但效率和速度始

终比不上团队协作，如果团队合作分工，能极大减轻工作量，最重要的是能提高开发速度，软件行业中提高开发速度是非常重要的，代表着你的软件产品提前一天上市，就多一点自主权，多一些用户青睐。

个人感觉编码其实不是特别难，最难的是确立需求，怎么将用户需求转化成代码实现，并且达到用户想要的效果。所以通常做项目的时候，确立需求是花相对比较长的时间，而编码的时间相对来说比较短。编码完成之后不代表着项目结束，它还涉及到项目维护以及功能升级，此过程持续时间需要很长。

本次开发使用很多框架，我使用框架的目的就是为了提高开发速度，并无任何要求，我也可以用原生，但是原生的开发速度非常慢，关键是很多代码冗余，造成空间资源浪费。程序员千万不能只会框架而脱离原生基础，因为任何的框架都是通过原生封装而成，如果不了解原生，当框架出现问题时，程序员就会无从下手，无所适从，感觉到非常迷惑和迷惘。所以有空的时候多去阅读阅读框架背后的源代码，找到框架与框架之间的联系。

个人觉得学校的技术太落后了，起码落后外面两年，如果刚毕业想要找到好的工作，并且适应外面的工作压力，大学期间就要好好努力，打扎实基础，增加学习时长和项目经验，这样方可提高就业竞争力，拿到一个可观的 offer 和薪资，利于实现自己的目标与理想。

6、参考文献

jdk1.8----->jdk1.8 帮助文档

mysql-----><https://www.runoob.com/mysql/mysql-tutorial.html>

springboot-----><https://spring.io/projects/spring-boot/>

mybatis-----><https://mybatis.org/mybatis-3/zh/index.html>

vue-----><https://cn.vuejs.org/>

cli3-----><https://cli.vuejs.org/>

element-ui-----><https://element.eleme.cn/#/zh-CN/component/installation>

iview-----><http://v1.iviewui.com/components/layout>

echart-----><https://echarts.apache.org/zh/tutorial.html>

maven-----><https://mvnrepository.com/tags/maven>