沧州交通学院

《Web 程序设计》

智慧团建管理系统
设计说明书

课程名称	Web 程序设计
学　　院	计算机与信息技术学院
专业班级	计科 2109
学　　号	21851195
姓　　名	单佳龙
日　　期	2023-07-07

《Web 程序设计》课程考核说明

一、考试对象

计科 2021 级 2104——2019 共 6 个班级。

二、考核方式

采用大作业及答辩方式。

三、考试内容

使用 Spring、Spring MVC、MyBatis 框架，完成某个管理系统的增、删、改、查功能，题目自拟：

如：客户管理、供应商管理、课程管理、合同管理、……：

四、技术要求

1．要求使用 Spring、Spring MVC、Mybatis 技术实现。

2．包路径命名：com.学生姓名的拼音首字母.ssm.xxxx

3．包、类、变量命名符合通用规范（包名必须小写、类名首字母必须大写、变量名必须小写，jsp/html 页面文件名小写，类名、变量名符合变量命名规范等）

五、提交物

平时练习、期末大作业、设计说明书电子版、设计说明书纸质打印版

六、期末成绩构成

由项目成绩和答辩成绩组成，其中项目成绩 60 分，答辩成绩 40 分

项目成绩评分标准如下：

综合/单项类评分	分值
1 提交物	**10**
1.1 内容完整性	10
1.2 目录结构	10
2 代码	**20**
2.1 代码包结构	5
2.2 项目、代码命名规范	5
2.3 代码编写规范性	5
2.4 代码可读性	5
3 程序演示	**30**
3.1 结果正确，功能完整	15
3.2 页面美观	15

《Web 程序设计》课程考核成绩

项目成绩（60 分）	
答辩成绩（40 分）	
期末总成绩	

答辩评语：

提交物内容（完整、较完整、基本完整、不完整），目录结构（正确、较正确、基本正确、不正确）；项目、代码命名（符合规范、较符合规范、基本符合规范、不规范），代码注释（完整、较完整、基本完整、不完整），可读性（好、较好、尚可、差）；程序功能（完整、较完整、基本完整、不完整），页面（美观、较美观、设计尚可、不美观），演示过程（顺畅、较顺畅、基本顺畅、不顺畅），问题回答（正确、较正确、基本正确、错误较多），思路（清晰、较清晰、基本清晰、不清晰），概念（清楚、较清楚、基本清楚、不清楚）。

智慧团建管理系统设计说明书

从智慧团建管理的功能设计、智慧团建管理的流程设计、智慧团建管理的类图设计、智慧团建管理的流程图设计、智慧团建管理的数据库表设计这五个方面进行详细的设计。

1．系统设计

智慧团建管理模块是实现用户操作团组织与团日活动的功能，可以对用户信息进行新增、修改、删除、查询等操作，对团组织信息进行修改，查询等操作，对团日活动进行新增、修改、删除、查询等操作。查询组织信息通过组织名称条件进行查询，查询团日活动信息通过活动时间条件进行查询。

2．流程设计

智慧团建管理模块是由用户登录系统，根据提供的信息进行查询组织与当前组织下的团日活动，登录后可对组织与团日活动进行修改、删除、查询等。所有操作的数据均保存在数据库中。智慧团建管理流程图如上图 1 所示。

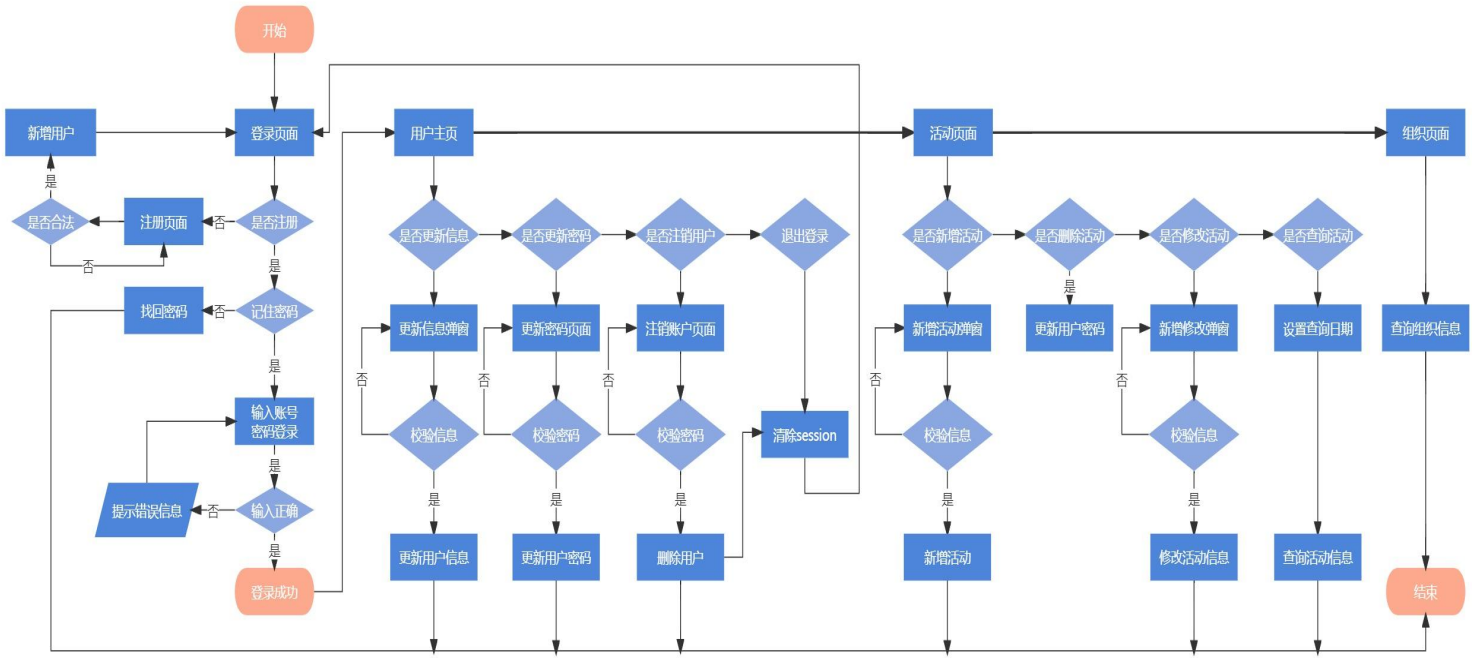

图 1　智慧团建管理流程图

3．功能设计

（1）用户注册

在用户注册功能中，访问 localhost:8080/pages/register.html，用户按要求提交表单后，向控制层 UserController 发送异步请求，控制层 UserController 的 save 方法，首先调用业务层 OrganizationService 接口实现类 OrganizationServiceImpl 里面的 getByName 方法通过组织 name 获取组织 id 并赋值给该用户的 organization，然后调用业务层 UserService 接口实现类 UserServiceImpl 里面的 Save 方法，UserServiceImpl 的 Save 方法调用 UserDao 的 getByCard 方法对数据库中的 zhtj 库下的 user 表中的用户信息进行查询操作，将相关数据返回进行用户是否已经注册的判断，如果已经注册，那么将相关数据返回到前端页面显示。如果未注册，UserServiceImpl 的 save 方法调用 UserDao 的 save 方法，对数据库中的 zhtj 库下的 user 表中的用户信息进行新增插入操作，然后将相关数据返回到前端页面显示。该功能模块的设计如图 2 所示。

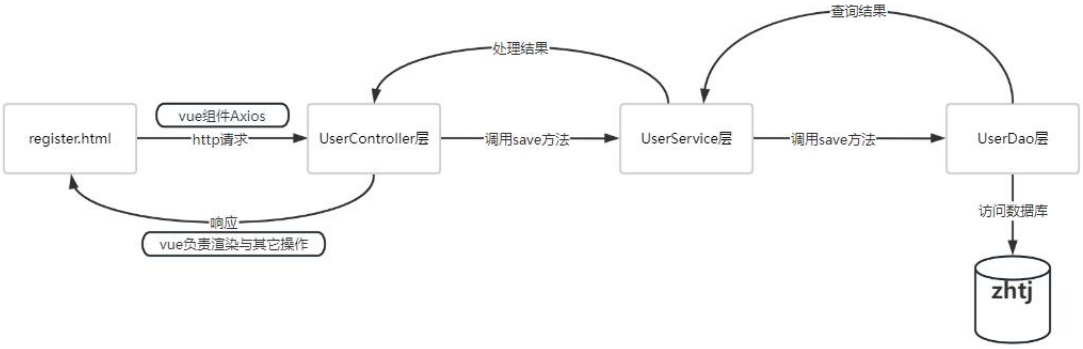

图 2　用户注册

（2）用户登陆

在用户登陆功能中，访问 localhost:8080/pages/index.html，向控制层 LoginController 发送异步请求，控制层 LoginController 的 getCode 方法使用 HuTool Captcha 工具生成验证码，用二进制流输出到前端页面显示，用户按要求提交表单后，向控制层 LoginController 发送异步请求，控制层 LoginController 的 getId 方法，首先判断验证码，如果错误，将相关数据返回到前端页面显示。如果正确，调用业务层 UserService 接口实现类 UserServiceImpl 里面的 getId 方法，UserServiceImpl 的 getId 方法调用 UserDao 的 getId 方法对数据库中的 zhtj 库下的 user 表中的用户信息进行身份证与密码匹配的查询操作，将相关数据返回进行用户身份证与密码是否匹配的判断，如果不匹配，将相关数据返回到前端页面显示。如果匹配，存下用户的 session 并将相关数据返回到前端页面显示，之后前端跳转到用户主页。该功能模块的设计如图 3 所示。

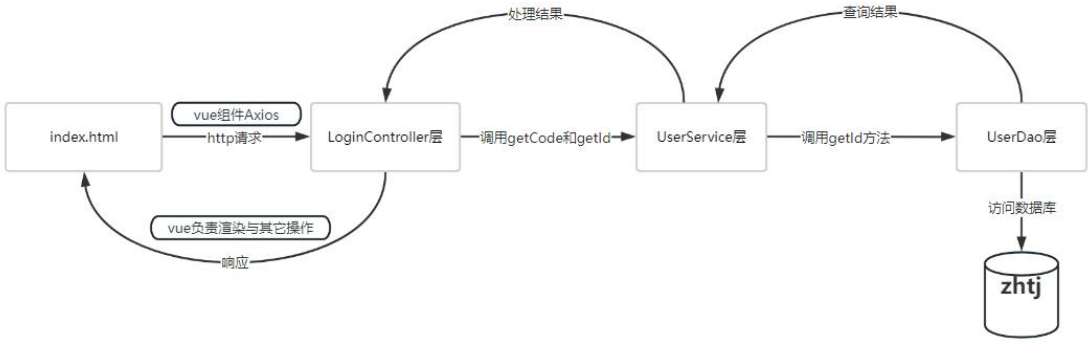

图 3　用户登陆

（3）登陆后显示用户个人信息（用户主页）

在显示用户个人信息功能中，访问 localhost:8080/pages/mine/home.html，向控制层 UserController 发送异步请求，控制层 UserController 的 getByCard 方法，首先从 session 读取用户属性中的身份证信息，然后调用业务层 UserService 接口实现类 UserServiceImpl 里面的 getByCard 方法，UserServiceImpl 的 getByCard 方法调用 UserDao 的 getByCard 方法对数据库中的 zhtj 库下的 user 表中的用户信息进行查询操作，然后将相关数据返回到前端页面显示。该功能模块的设计如图 4 所示。

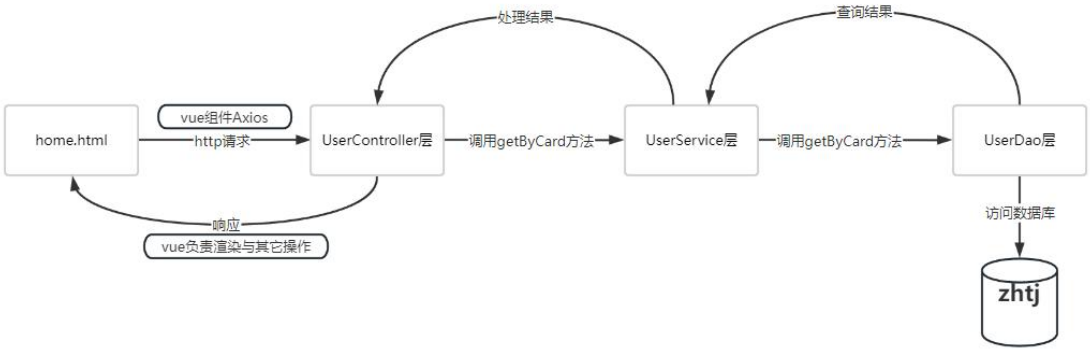

图 4　显示用户个人信息

（4）登陆后用户更新个人信息

在用户更新个人信息功能中，访问 localhost:8080/pages/mine/home.html，用户点击更新信息按钮后，弹出更新表单的弹窗，用户按要求提交表单后，向控制层 UserController 发送异步请求，控制层 UserController 的 update 方法调用业务层 UserService 接口实现类 UserServiceImpl 里面的 update 方法，UserServiceImpl 的 update 方法调用数据层 UserDao 的 update 方法对数据库中的 zhtj 库下的 user 表中的用户信息进行更新操作，然后将相关数据返回到前端页面显示。该功能模块的设计如图 5 所示。

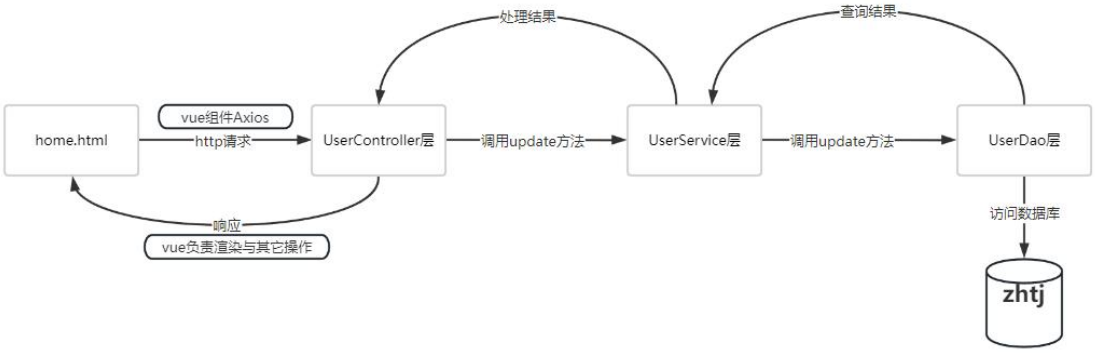

图 5　用户更新个人信息

（5）登陆后用户修改个人密码

在用户修改个人密码功能中，成功登录后，用户点击在系统设置中点击修改密码后，跳转页面到 localhost:8080/pages/mine/changepwd.html，用户按要求提交表单后，向控制层 UserController 发送异步请求，控制层 UserController 的 updatePwd 方法调用业务层 UserService 接口实现类 UserServiceImpl 里面的 updatePwd 方法，UserServiceImpl 的 updatePwd 方法调用数据层 UserDao 的 updatePwd 方法对数据库中的 zhtj 库下的 user 表中的用户密码进行更新操作，然后清除 session 中的用户信息并将相关数据返回到前端页面显示，之后前端跳转到登陆页面。该功能模块的设计如图 6 所示。

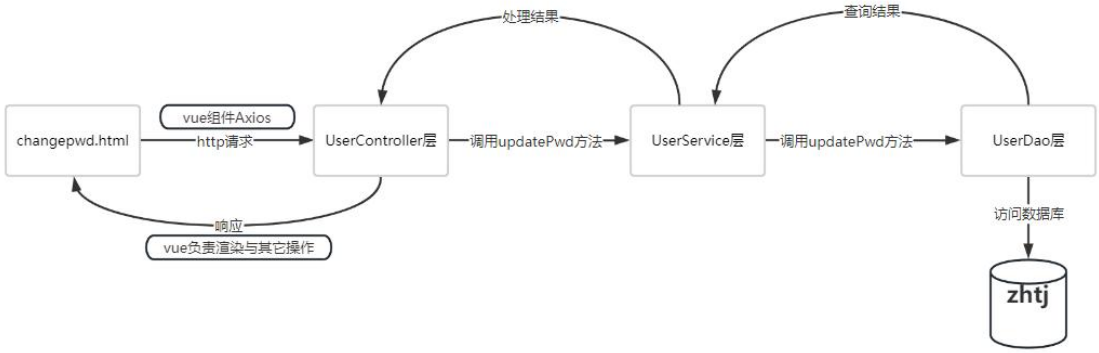

图 6　用户修改个人密码

（6）登陆后用户注销个人账户

在用户注销个人账户功能中，成功登录后，用户点击在系统设置中点击注销账户后，跳转页面到 localhost:8080/pages/mine/destroy.html，用户按要求成功提交表单后，向控制层 UserController 发送异步请求，控制层 UserController 的 delete 方法，首先从 session 读取用户属性中的身份证信息，然后调用业务层 UserService 接口实现类 UserServiceImpl 里面的 delete 方法，UserServiceImpl 的 delete 方法调用数据层 UserDao 的 delete 方法对数据库中的 zhtj 库下的 user 表中的用户以及信息进行删除操作，然后清除 session 中的用户信息并将相关数据返回到前端页面显示，之后前端跳转到登陆页面。该功能模块的设计如图 7 所示。

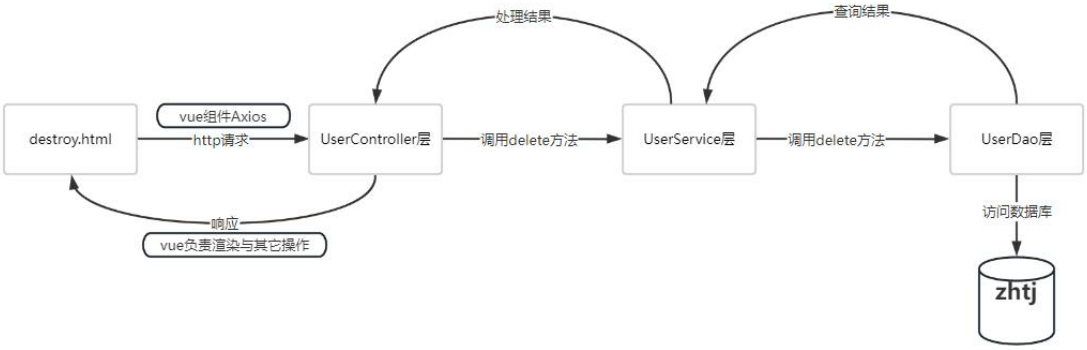

图 7　用户注销个人账户

（7）登陆后用户退出登录

在用户注销个人账户功能中，成功登录后，用户点击在系统设置中点击退出后，向控制层 UserController 发送异步请求，控制层 UserController 的 logout 方法，然后清除 session 中的用户信息并将相关数据返回到前端页面显示，之后前端跳转到登陆页面。该功能模块的设计如图 8 所示。

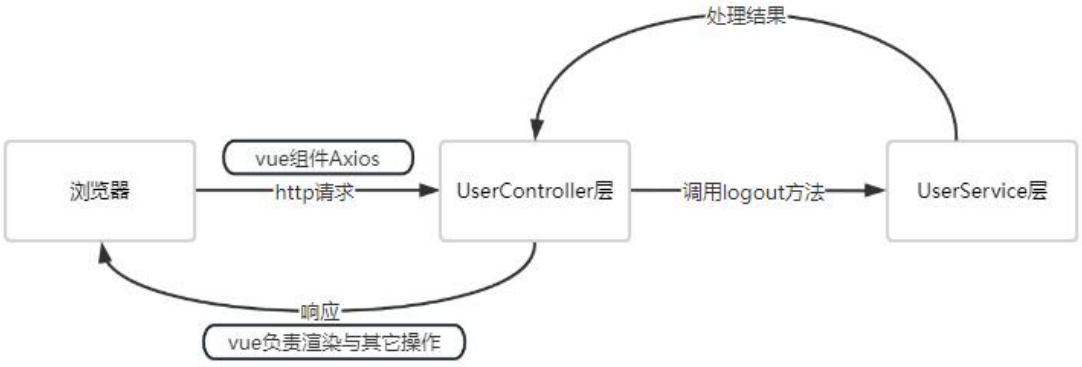

图 8　用户退出登录

（8）登陆后用户查看组织信息与组织成员

在用户查看组织信息与组织成员功能中，成功登录后，用户点击在我的组织后，跳转页面到 localhost:8080/pages/mine/myorganization.html，向控制层 UserController 发送异步请求，控制层 UserController 的 getOrg 方法，首先从 session 读取用户属性中的组织 id，然后调用业务层 UserService 接口实现类 UserServiceImpl 里面的 getOrg 方法，UserServiceImpl 的 getOrg 方法调用数据层 UserDao 的 getOrg 方法对数据库中的 zhtj 库下的 user 表中的相同组织的用户信息进行查询操作，然后将相关数据返回到前端页面显示。该功能模块的设计如图 9 所示。

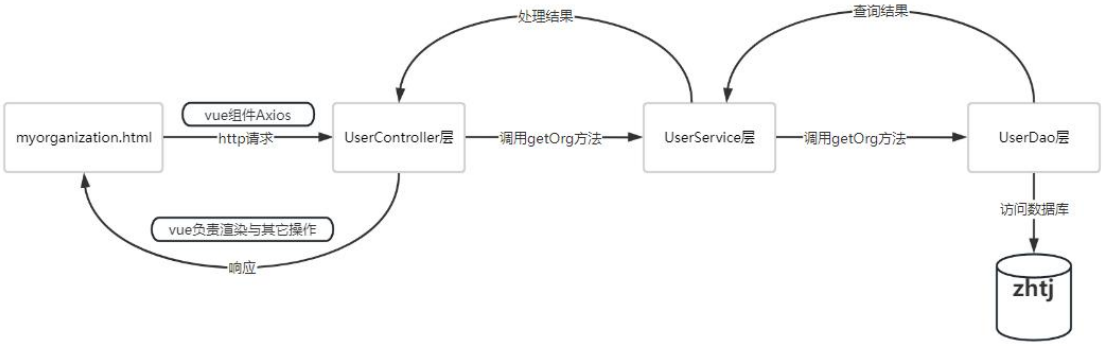

图 9　用户查看组织信息与组织成员

（9）登陆后用户修改同组织成员信息

在用户修改同组织成员信息功能中，成功登录后，在用户查看组织信息与组织成员功能页面 localhost:8080/pages/mine/myorganization.html 下，用户点击查看的眼睛图标后，跳转页面到对应组织成员页面 localhost:8080/pages/mine/myorganizationX.html,用户按要求提交表单后，向控制层 UserController 发送异步请求，控制层 UserController 的 updateOther 方法调用业务层 UserService 接口实现类 UserServiceImpl 里面的 updateOther 方法，UserServiceImpl 的 updateOther 方法调用数据层 UserDao 的 updateOther 方法对数据库中的 zhtj 库下的 user 表中的相对应的用户信息进行更新操作，然后将相关数据返回到前端页面显示。因为该功能模块涉及到用户权限模块的设计，所以该功能尚未完善。该功能模块的设计如图 10 所示。

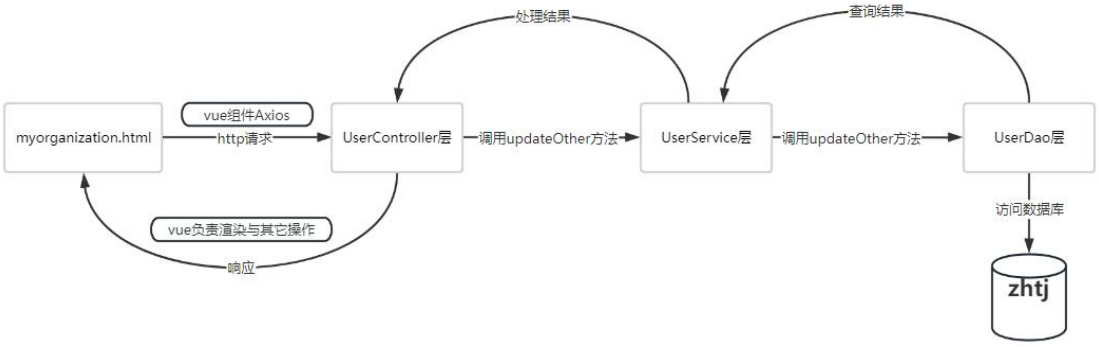

图 10　用户修改同组织成员信息

（10）登陆后用户查看同组织下全部团日活动

在用户查看同组织下全部团日活动功能中，成功登录后，用户点击在团日活动后，跳转页面到 localhost:8080/pages/mine/activity.html，向控制层 ActivityController 发送异步请求，控制层 ActivityController 的 getAllByOrg 方法，首先从 session 读取用户属性中的身份证信息，其次调用业务层 userService 的 getByCard 方法，调用数据层 userDao 的 getByCard 方法对数据库中的 zhtj 库下的 activity 表中的用户信息进行查询操作，以便获取组织 id，然后调用业务层 ActivityService 接口实现类 ActivityServiceImpl 里面的 getOrg 方法，ActivityServiceImpl 的 getOrg 方法调用数据层 ActivityDao 的 getOrg 方法对数据库中的 zhtj 库下的 activity 表中的相同组织的团日活动信息进行查询操作，然后

将相关数据返回到前端页面显示。该功能模块的设计如图 11 所示。

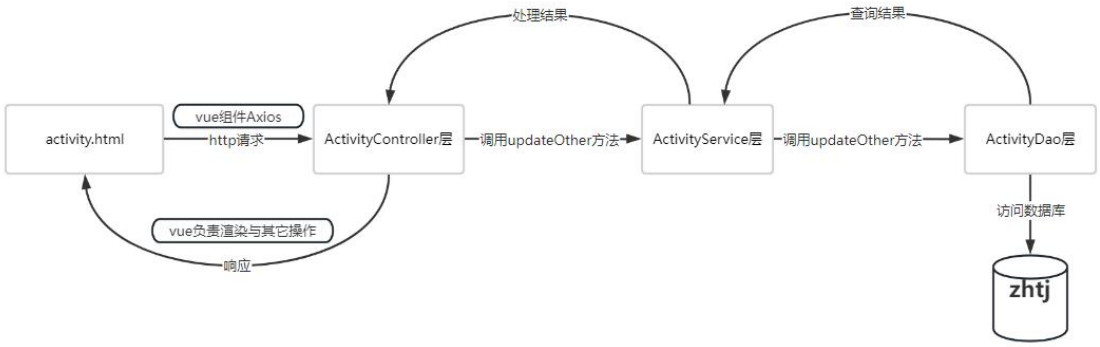

图 11　用户查看同组织下全部团日活动

（11）登陆后用户为组织增加团日活动

在用户为组织增加团日活动功能中，成功登录后，用户在团日活动页面点击添加活动按钮，向控制层 ActivityController 发送异步请求，控制层 ActivityController 的 save 方法，调用业务层 ActivityService 接口实现类 ActivityServiceImpl 里面的 save 方法，首先，ActivityServiceImpl 的 save 方法调用数据层 ActivityDao 的 getId 方法对数据库中的 zhtj 库下的 activity 表中的团日活动信息进行查询操作，将相关数据返回业务层进行判断。如果有相同的团日活动将相关数据返回到前端页面显示，如果没有相同的团日活动信息，继续调用数据层 ActivityDao 的 save 方法对数据库中的 zhtj 库下的 activity 表中的团日活动信息进行新增插入操作，然后将相关数据返回到前端页面显示。该功能模块的设计如图 12 所示。

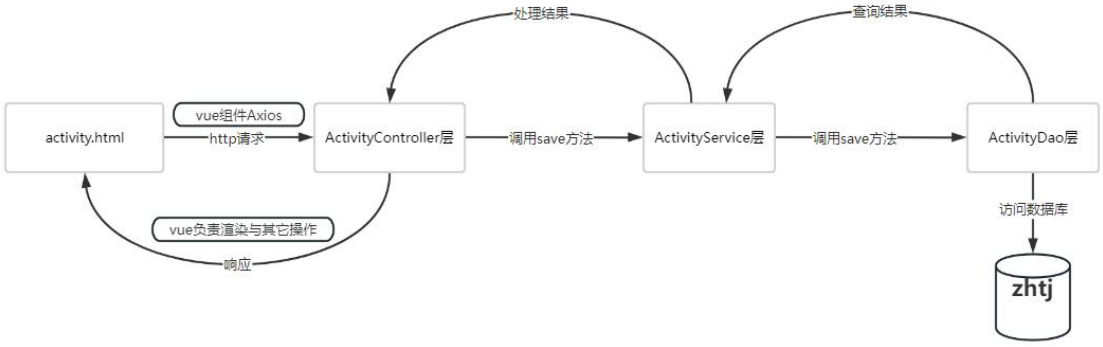

图 12　用户为组织增加团日活动

（12）登陆后用户为组织删除团日活动

在用户为组织删除团日活动功能中，成功登录后，用户在团日活动页面点击删除活动按钮，向控制层 ActivityController 发送异步请求，控制层 ActivityController 的 delete 方法，通过路径变量活动 id，调用业务层 ActivityService 接口实现类 ActivityServiceImpl 里面的 delete 方法，ActivityServiceImpl 的 delete 方法调用数据层 ActivityDao 的 delete 方法对数据库中的 zhtj 库下的 activity 表中的团日活动信息进行删除操作，然后将相关数据返回到前端页面显示。该功能模块的设计如图 13 所示。

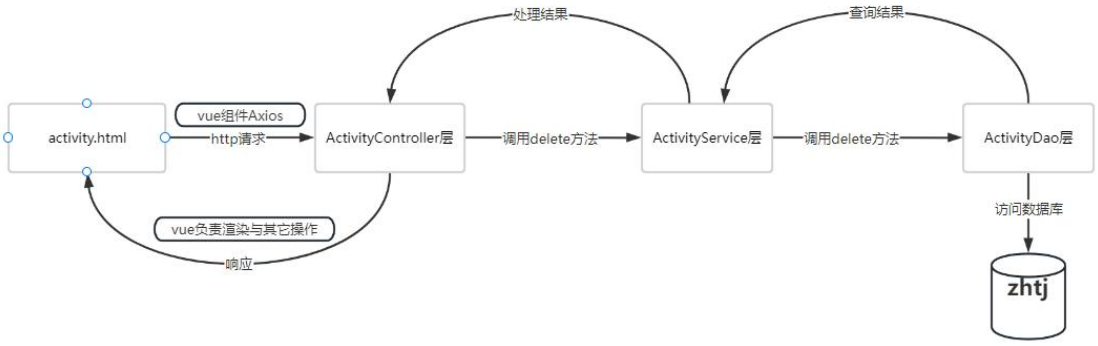

图 13　用户为组织删除团日活动

（13）登陆后用户为组织更新修改已有的团日活动

在用户为组织更新修改已有的团日活动功能中，成功登录后，用户在团日活动页面点击删除活动按钮，向控制层 ActivityController 发送异步请求，控制层 ActivityController 的 update 方法，首先从 session 读取用户属性中的身份证信息，其次调用业务层 userService 的 getByCard 方法，调用数据层 userDao 的 getByCard 方法对数据库中的 zhtj 库下的 activity 表中的用户信息进行查询操作，以便获取组织 id，然后调用业务层 ActivityService 接口实现类 ActivityServiceImpl 里面的 update 方法，ActivityServiceImpl 的 update 方法调用数据层 ActivityDao 的 update 方法对数据库中的 zhtj 库下的 activity 表中的团日活动信息进行更新操作，然后将相关数据返回到前端页面显示。该功能模块的设计如图 14 所示。

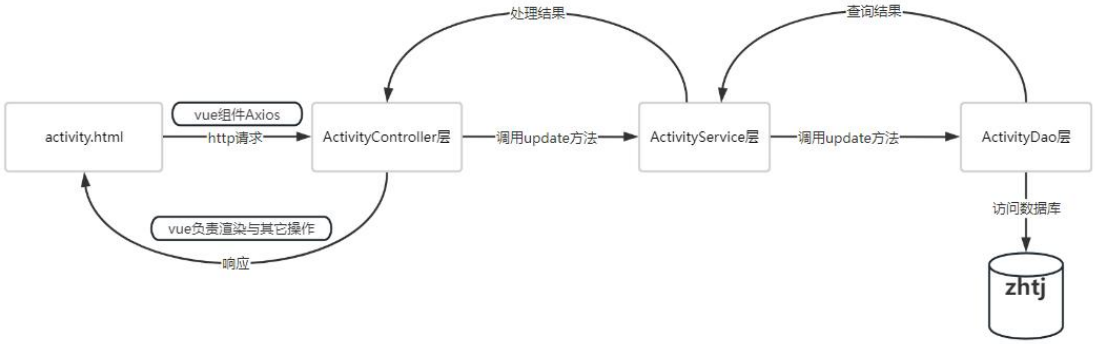

图 14　用户为组织更新修改已有的团日活动

（14）登陆后用户根据时间查询团日活动

在用户查看同组织下全部团日活动功能中，成功登录后，用户按要求设置时间表单后点击查询按钮，向控制层 ActivityController 发送异步请求，控制层 ActivityController 的 getAllByOrg 方法，首先从 session 读取用户属性中的身份证信息，其次调用业务层 userService 的 getByCard 方法，调用数据层 userDao 的 getByCard 方法对数据库中的 zhtj 库下的 activity 表中的用户信息进行查询操作，以便获取组织 id，然后调用业务层 ActivityService 接口实现类 ActivityServiceImpl 里面的 getAllByDate 方法，ActivityServiceImpl 的 getAllByDate 方法调用数据层 ActivityDao 的 getAllByDate 方法对数据库中的 zhtj 库下的 activity 表中的相同组织的团日活动信息进行查询操作，然后将相关数据返回到前端页面显示。该功能模块的设计如图 15 所示。

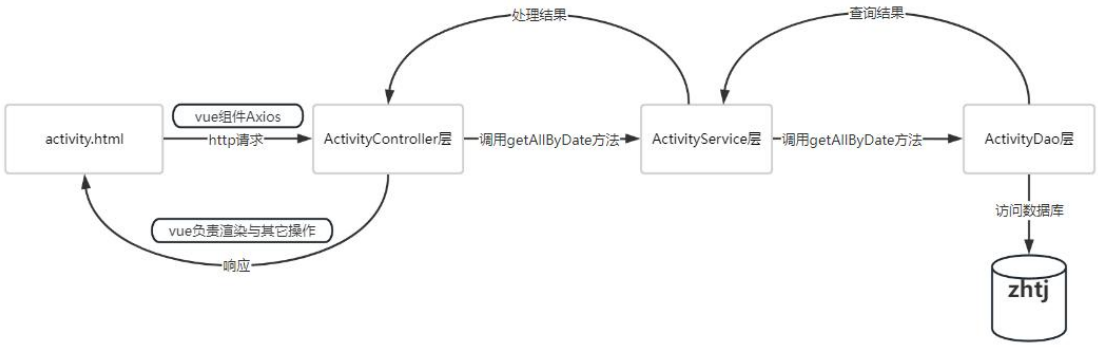

图 15　用户根据时间查询团日活动

4．数据库表结构设计

本模块使用的数据库表主要使用用户信息表，活动信息表，组织信息表。

用户信息表包括序号、团组织序号、姓名、身份证、民族、手机、密码，其中序号为表主键，表结构设计如表 1 所示。

活动信息表包括序号、团组织序号、活动类型、活动日期、活动主持人、活动地点、活动内容，其中序号为表主键，表结构设计如表 2 所示。

组织信息表包括序号、组织名称，其中序号为表主键，表结构设计如表 3 所示。

表 1 用户信息表

名	类型	长度	小数点	不是 null	虚拟	键	注释
id	int			☑	☐	1	序号
organization	int			☐	☐		团组织序号
name	varchar	20		☐	☐		姓名
card	varchar	18		☐	☐		身份证
ethnic	varchar	10		☐	☐		民族
phone	varchar	11		☐	☐		手机
pwd	varchar	20		☐	☐		密码

表 2 活动信息表

名	类型	长度	小数点	不是 null	虚拟	键	注释
id	int			☑	☐	1	序号
organization	int			☐	☐		团组织序号
type	varchar	20		☐	☐		活动类型
date	varchar	10		☐	☐		活动日期
host	varchar	20		☐	☐		活动主持人
place	varchar	20		☐	☐		活动地点
content	varchar	600		☐	☐		活动内容

表 3 组织信息表

名	类型	长度	小数点	不是 null	虚拟	键	注释
id	int			☑	☐	1	序号
name	varchar	50		☑	☐		组织名称

5．系统实现

（1）登录界面

在用户登陆功能中，访问 localhost:8080/pages/index.html，向控制层 LoginController 发送异步请求，控制层 LoginController 的 getCode 方法使用 HuTool Captcha 工具生成验证码，用二进制流输出到前端页面显示，用户按要求提交表单后，向控制层 LoginController 发送异步请求，控制层 LoginController 的 getId 方法，首先判断验证码，如果错误，将相关数据返回到前端页面显示。如果正确，调用业务层 UserService 接口实现类 UserServiceImpl 里面的 getId 方法，UserServiceImpl 的 getId 方法调用 UserDao 的 getId 方法对数据库中的 zhtj 库下的 user 表中的用户信息进行身份证与密码匹配的查询操作，将相关数据返回进行用户身份证与密码是否匹配的判断，如果不匹配，将相关数据返回到前端页面显示。如果匹配，存下用户的 session 并将相关数据返回到前端页面显示，之后前端跳转到用户主页。

① 原型设计

登录页面如图 17 所示。

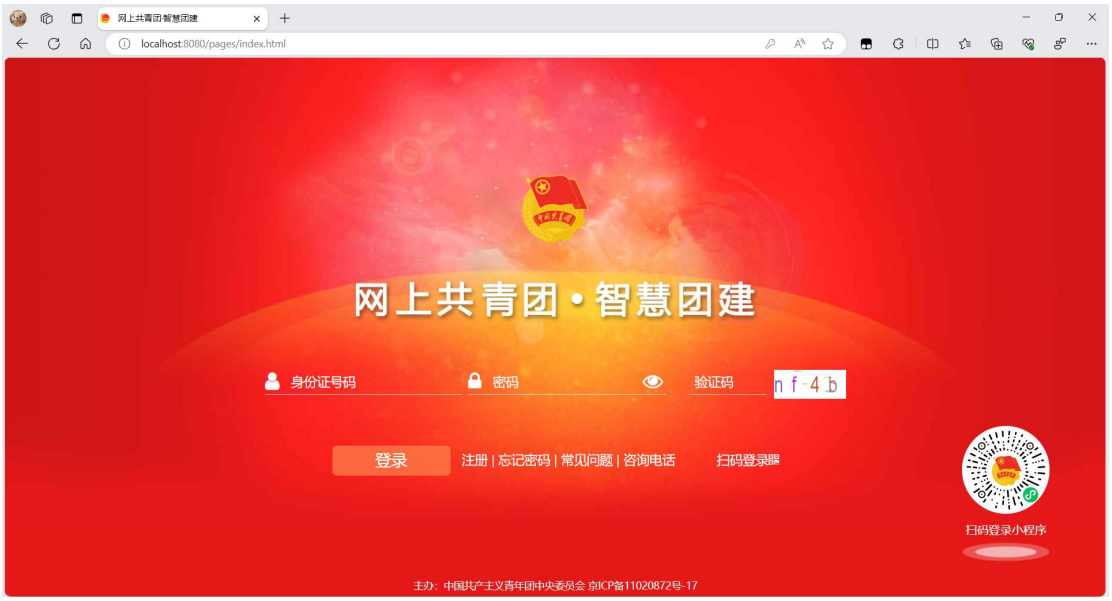

图 17 登录界面

② 核心代码

```
@RestController
@RequestMapping("login")
public class LoginController {
    @Autowired
    private UserService userService;
    private LineCaptcha captcha;

    //登录
    @PostMapping
    public boolean getId(HttpSession session, @RequestBody List<Object> list) {
        //应该写个包装类好了,或者只传身份证，密码，验证码
/*      public class UserYzm {
        private User user;
        private String yzm;
    }
*/          //json 里的 Object 转 String,Object 转 User
        //这里写的不好
        String yzm = String.valueOf(list.get(1)).substring(5, 9);
        User user = new ObjectMapper().convertValue(list.get(0), User.class);
        //先判断验证码是否匹配
        System.out.println(yzm);
        if (!yzm.equals(captcha.getCode())) return false;
        //存下 session
        session.setAttribute("user", user);
        Integer userId = userService.getId(user);
        //判断身份证与密码是否匹配，匹配返回 true，不匹配返回 false
        boolean flag;
        flag = userId != null;
        return flag;
    }
    //先获取验证码
    @GetMapping
    public void getCode(HttpServletResponse response) {
```

```
        //定义图形验证码的长和宽
        captcha = CaptchaUtil.createLineCaptcha(116, 36, 4, 5);
        //保存到本地
        //captcha.write("/img/captcha.png");
        response.setContentType("image/jpeg");
        response.setHeader("Pragma", "No-cache");
        try {
            // 输出到页面
            captcha.write(response.getOutputStream());
            // 关闭流
            response.getOutputStream().close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

（2）注册界面

在用户注册功能中，访问 localhost:8080/pages/register.html，用户按要求提交表单后，向控制层 UserController 发送异步请求，控制层 UserController 的 Save 方法，首先调用业务层 OrganizationService 接口实现类 OrganizationServiceImpl 里面的 getByName 方法通过组织 name 获取组织 id 并赋值给该用户的 organization，然后调用业务层 UserService 接口实现类 UserServiceImpl 里面的 Save 方法，UserServiceImpl 的 Save 方法调用 UserDao 的 getByCard 方法对数据库中的 zhtj 库下的 user 表中的用户信息进行查询操作，将相关数据返回进行用户是否已经注册的判断，如果已经注册，那么将相关数据返回到前端页面显示。如果未注册，UserServiceImpl 的 Save 方法调用 UserDao 的 Save 方法，对数据库中的 zhtj 库下的 user 表中的用户信息进行新增插入操作，然后将相关数据返回到前端页面显示。

① 原型设计

注册界面如图 18 所示。

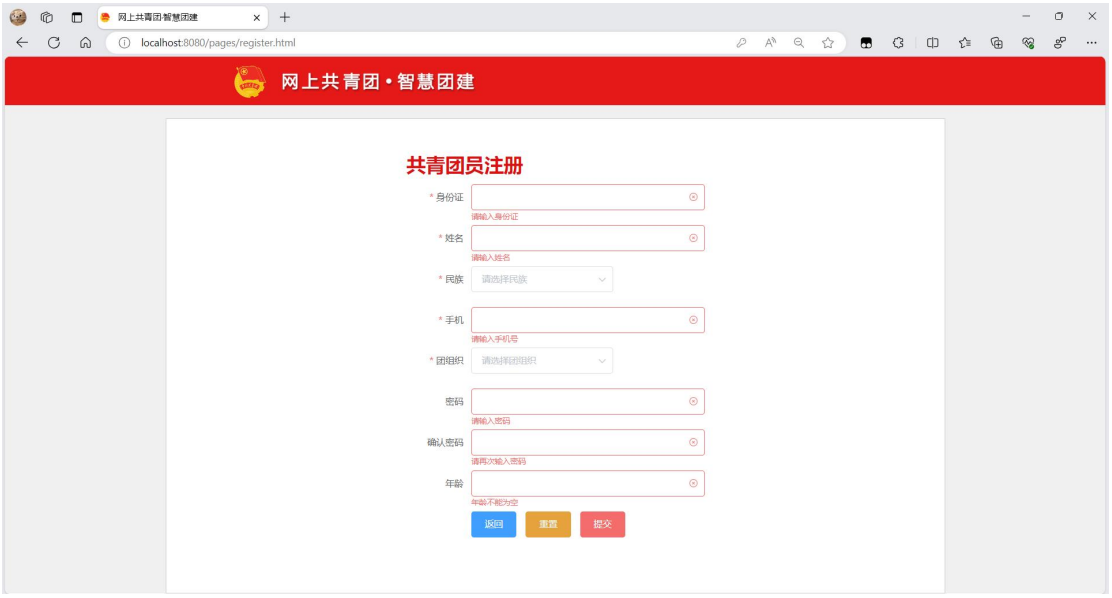

图 18　注册界面

② 核心代码

用户注册的核心代码如下。

```
//注册用户
@PostMapping
public boolean Save(@RequestBody Organization organization) {
    User user = organization.getUser();

user.setOrganization(organizationService.getByName(organization.getName()));
    return userService.save(user);
}
```

（3）我的首页

在显示用户个人信息功能中，访问 localhost:8080/pages/mine/home.html，向控制层 UserController 发送异步请求，控制层 UserController 的 getByCard 方法，首先从 session 读取用户属性中的身份证信息，然后调用业务层 UserService 接口实现类 UserServiceImpl 里面的 getByCard 方法，UserServiceImpl 的 getByCard 方法调用 UserDao 的 getByCard 方法对数据库中的 zhtj 库下的 user 表中的用户信息进行查询操作，然后将相关数据返回到前端页面显示。

在用户更新个人信息功能中，访问 localhost:8080/pages/mine/home.html，用户点击更新信息按钮后，弹出更新表单的弹窗，用户按要求提交表单后，向控制层 UserController 发送异步请求，控制层 UserController 的 update 方法调用业务

层 UserService 接口实现类 UserServiceImpl 里面的 update 方法，UserServiceImpl 的 update 方法调用数据层 UserDao 的 update 方法对数据库中的 zhtj 库下的 user 表中的用户信息进行更新操作，然后将相关数据返回到前端页面显示。

在用户注销个人账户功能中，成功登录后，用户点击在系统设置中点击退出后，向控制层 UserController 发送异步请求，控制层 UserController 的 logout 方法，然后清除 session 中的用户信息并将相关数据返回到前端页面显示，之后前端跳转到登陆页面。

① 原型设计

我的首页如图 19 所示。

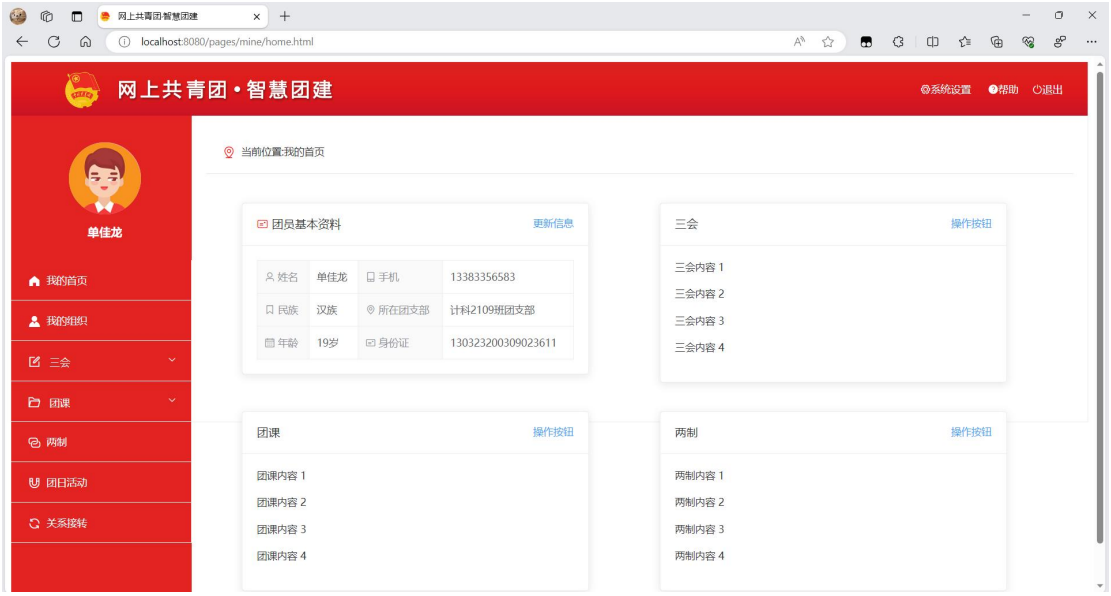

图 19　我的首页

用户更新个人信息界面如图 20 所示。

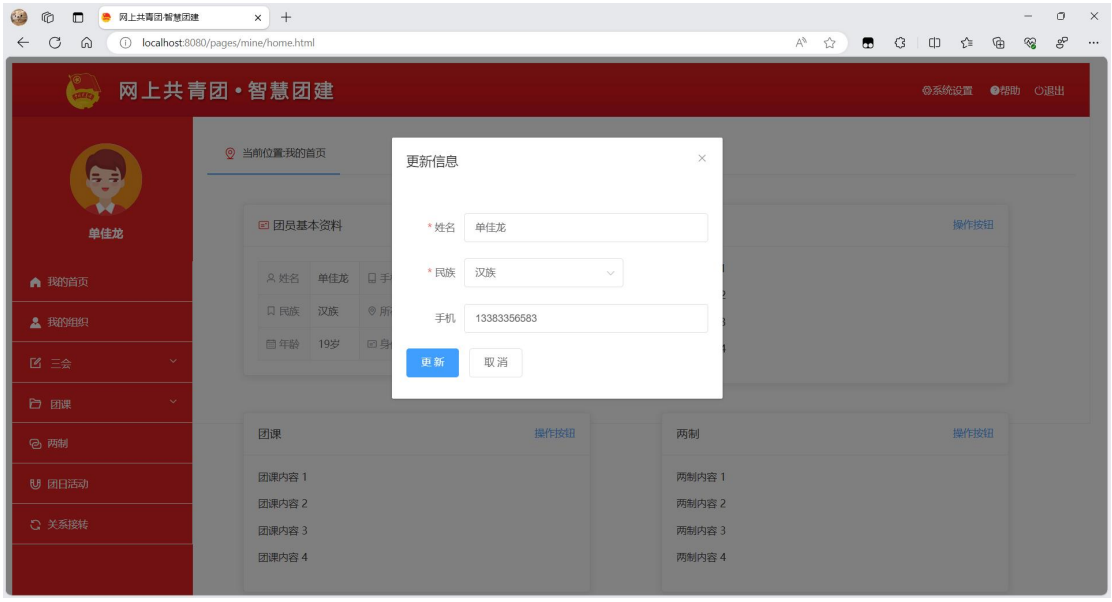

图 20　用户更新个人信息界面

② 核心代码

显示用户个人信息的核心代码如下。

```
@GetMapping("/1")
public Organization getByCard(HttpSession session) {
    //从 session 读属性 user
    User user = (User) session.getAttribute("user");
    //用 user 的身份证读其他信息
    user = userService.getByCard(user.getCard());
    //用组织 id 读组织名
    String name=organizationService.getById(user.getOrganization());
    return new Organization(null,name,user);
}
```

用户更新个人信息的核心代码如下。

```
//更新个人信息
@PutMapping
public boolean update(@RequestBody Organization organization){
    userService.update(organization.getUser());
    return true;
}
```

用户退出登录的核心代码如下。

```
//退出登录，清除一下 session 就 ok，让前端跳到登陆页面
@GetMapping("/logout")
public boolean logout(HttpSession session){
    session.removeAttribute("user");
    return true;
}
```

（4）修改密码界面

在用户修改个人密码功能中，成功登录后，用户点击在系统设置中点击修改密码后，跳转页面到 localhost:8080/pages/mine/changepwd.html，用户按要求提交表单后，向控制层 UserController 发送异步请求，控制层 UserController 的 updatePwd 方法调用业务层 UserService 接口实现类 UserServiceImpl 里面的 updatePwd 方法，UserServiceImpl 的 updatePwd 方法调用数据层 UserDao 的

updatePwd 方法对数据库中的 zhtj 库下的 user 表中的用户密码进行更新操作，然后清除 session 中的用户信息并将相关数据返回到前端页面显示，之后前端跳转到登陆页面。

① 原型设计

修改密码界面如图 21 所示。

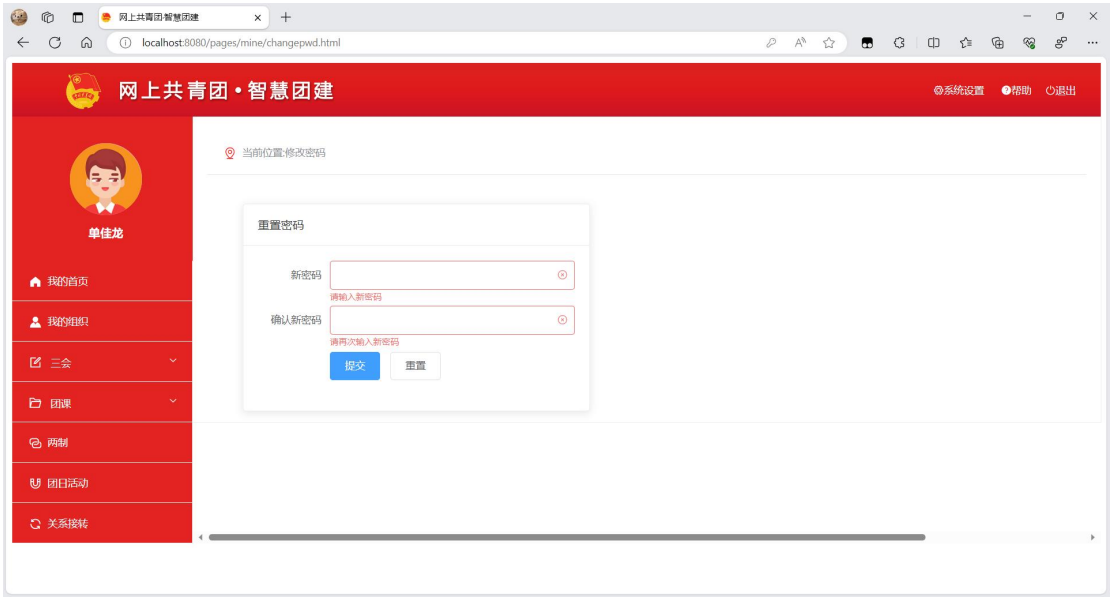

图 21　修改密码界面

② 核心代码

用户修改个人密码的核心代码如下。

```
//更新用户密码并且清除 session
@PutMapping("/pwd")
public boolean updatePwd(HttpSession session,@RequestBody Organization organization){
    userService.updatePwd(organization.getUser());
    session.removeAttribute("user");
    return true;
}
```

（5）注销账户界面

① 原型设计

注销账户界面如图 22 所示。

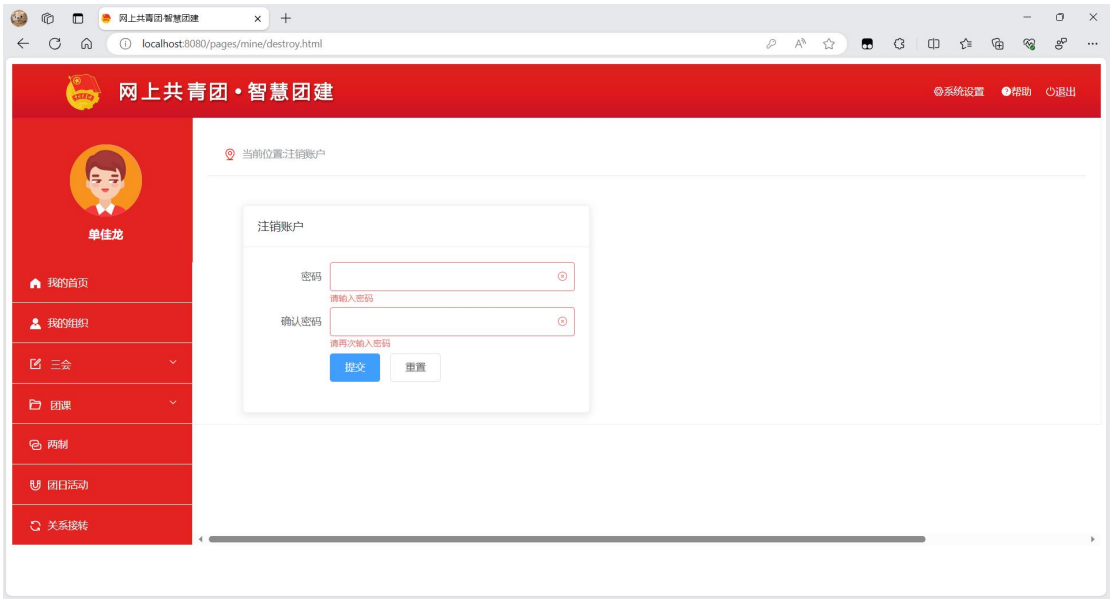

图 22　注销账户界面

② 核心代码

用户注销个人账户的核心代码如下。

```
//现在是前端根据登录的 session 判定密码，删除用户并且清除 session，让前端跳到登陆页面
//其实可以在访问一次数据库看看密码对不对，下次再改！！！
@DeleteMapping
public boolean delete(HttpSession session){
    String card=((User) session.getAttribute("user")).getCard();
    System.out.println(card);
    userService.delete(card);
    session.removeAttribute("user");
    return true;
}
```

（6）我的组织界面

在用户查看组织信息与组织成员功能中，成功登录后，用户点击在我的组织后，跳转页面到 localhost:8080/pages/mine/myorganization.html，向控制层 UserController 发送异步请求，控制层 UserController 的 getOrg 方法，首先从 session 读取用户属性中的组织 id，然后调用业务层 UserService 接口实现类 UserServiceImpl 里面的 getOrg 方法，UserServiceImpl 的 getOrg 方法调用数据层

UserDao 的 getOrg 方法对数据库中的 zhtj 库下的 user 表中的相同组织的用户信息进行查询操作，然后将相关数据返回到前端页面显示。

在用户修改同组织成员信息功能中，成功登录后，在用户查看组织信息与组织成员功能页面 localhost:8080/pages/mine/myorganization.html 下，用户点击查看的眼睛图标后，跳转页面到对应组织成员页面 localhost:8080/pages/mine/myorganizationX.html,用户按要求提交表单后，向控制层 UserController 发送异步请求，控制层 UserController 的 updateOther 方法调用业务层 UserService 接口实现类 UserServiceImpl 里面的 updateOther 方法，UserServiceImpl 的 updateOther 方法调用数据层 UserDao 的 updateOther 方法对数据库中的 zhtj 库下的 user 表中的相对应的用户信息进行更新操作，然后将相关数据返回到前端页面显示。因为该功能模块涉及到用户权限模块的设计，所以该功能尚未完善。

① 原型设计

我的组织界面如图 23 所示。

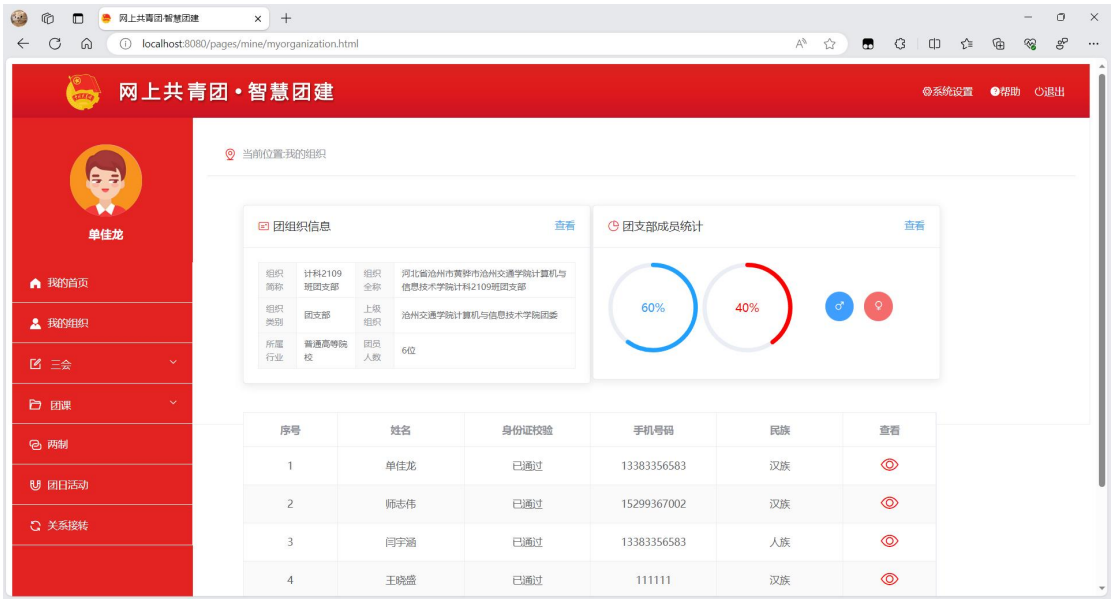

图 23　我的组织界面

② 核心代码

用户查看组织信息与组织成员的核心代码如下。

```
//查组织下全部成员
@GetMapping
public List<User> getOrg(HttpSession session) {
    //从 session 读属性 user
    User user = (User) session.getAttribute("user");
```

```
        //用 user 的身份证读其他信息
        user = userService.getByCard(user.getCard());
        //用组织 id 读组织名
        List<User> userList = userService.getOrg(user.getOrganization());
        return userList;
    }
```

（7）团日活动界面

在用户查看同组织下全部团日活动功能中，成功登录后，用户点击在团日活动后，跳转页面到 localhost:8080/pages/mine/activity.html，向控制层 ActivityController 发送异步请求，控制层 ActivityController 的 getAllByOrg 方法，首先从 session 读取用户属性中的身份证信息，其次调用业务层 userService 的 getByCard 方法，调用数据层 userDao 的 getByCard 方法对数据库中的 zhtj 库下的 activity 表中的用户信息进行查询操作，以便获取组织 id，然后调用业务层 ActivityService 接口实现类 ActivityServiceImpl 里面的 getOrg 方法，ActivityServiceImpl 的 getOrg 方法调用数据层 ActivityDao 的 getOrg 方法对数据库中的 zhtj 库下的 activity 表中的相同组织的团日活动信息进行查询操作，然后将相关数据返回到前端页面显示。

在用户为组织增加团日活动功能中，成功登录后，用户在团日活动页面点击添加活动按钮，向控制层 ActivityController 发送异步请求，控制层 ActivityController 的 save 方法，调用业务层 ActivityService 接口实现类 ActivityServiceImpl 里面的 save 方法，首先，ActivityServiceImpl 的 save 方法调用数据层 ActivityDao 的 getId 方法对数据库中的 zhtj 库下的 activity 表中的团日活动信息进行查询操作，将相关数据返回业务层进行判断。如果有相同的团日活动将相关数据返回到前端页面显示，如果没有相同的团日活动信息，继续调用数据层 ActivityDao 的 save 方法对数据库中的 zhtj 库下的 activity 表中的团日活动信息进行新增插入操作，然后将相关数据返回到前端页面显示。

在用户为组织删除团日活动功能中，成功登录后，用户在团日活动页面点击删除活动按钮，向控制层 ActivityController 发送异步请求，控制层 ActivityController 的 delete 方法，通过路径变量活动 id，调用业务层 ActivityService 接口实现类 ActivityServiceImpl 里面的 delete 方法，ActivityServiceImpl 的 delete 方法调用数据层 ActivityDao 的 delete 方法对数据库中的 zhtj 库下的 activity 表中的团日活动信息进行删除操作，然后将相关数据返回到前端页面显示。

在用户为组织更新修改已有的团日活动功能中，成功登录后，用户在团日活动页面点击删除活动按钮，向控制层 ActivityController 发送异步请求，控制层 ActivityController 的 update 方法，首先从 session 读取用户属性中的身份证信息，其次调用业务层 userService 的 getByCard 方法，调用数据层 userDao 的 getByCard 方法对数据库中的 zhtj 库下的 activity 表中的用户信息进行查询操作，以便获取组织 id，然后调用业务层 ActivityService 接口实现类 ActivityServiceImpl 里面的 update 方法，ActivityServiceImpl 的 update 方法调用数据层 ActivityDao 的 update 方法对数据库中的 zhtj 库下的 activity 表中的团日活动信息进行更新操作，然后将相关数据返回到前端页面显示。

在用户查看同组织下全部团日活动功能中，成功登录后，用户按要求设置时间表单后点击查询按钮，向控制层 ActivityController 发送异步请求，控制层 ActivityController 的 getAllByOrg 方法，首先从 session 读取用户属性中的身份证信息，其次调用业务层 userService 的 getByCard 方法，调用数据层 userDao 的 getByCard 方法对数据库中的 zhtj 库下的 activity 表中的用户信息进行查询操作，以便获取组织 id，然后调用业务层 ActivityService 接口实现类 ActivityServiceImpl 里面的 getAllByDate 方法，ActivityServiceImpl 的 getAllByDate 方法调用数据层 ActivityDao 的 getAllByDate 方法对数据库中的 zhtj 库下的 activity 表中的相同组织的团日活动信息进行查询操作，然后将相关数据返回到前端页面显示。

① 原型设计

团日活动界面如图 24 所示。

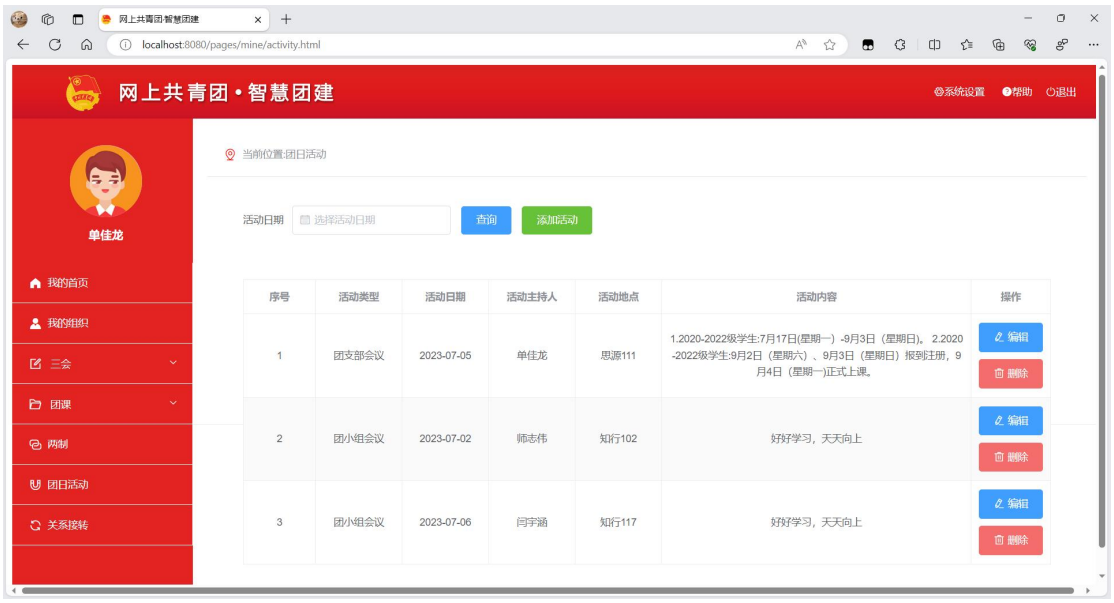

图 24 团日活动界面

添加团日活动界面如图 25 所示。

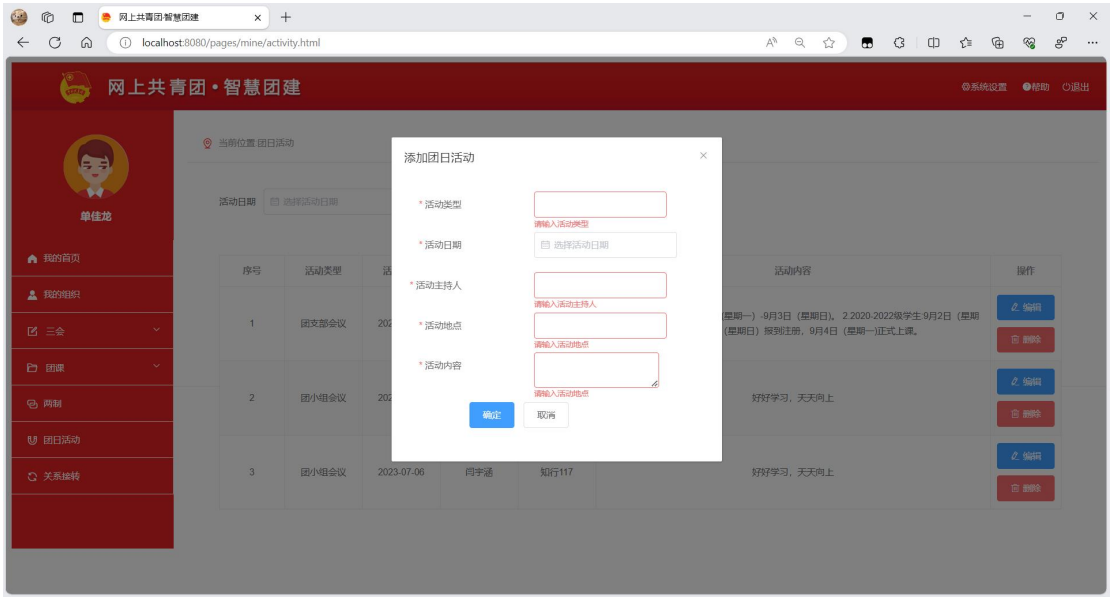

图 25 添加团日活动界面

删除团日活动界面如图 26 所示。

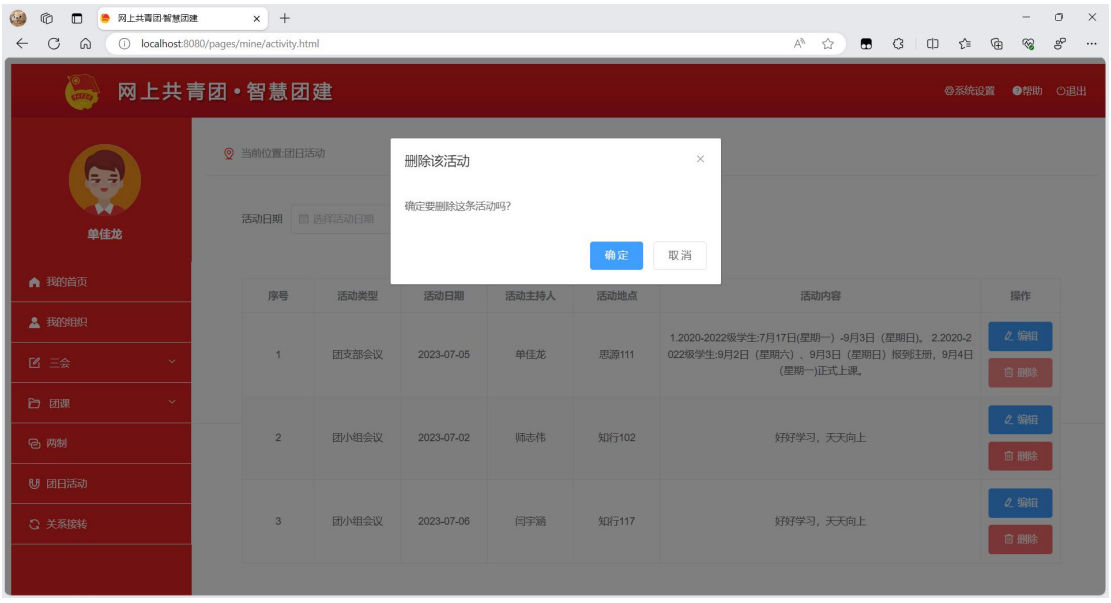

图 26 删除团日活动界面

编辑团日活动界面如图 27 所示。

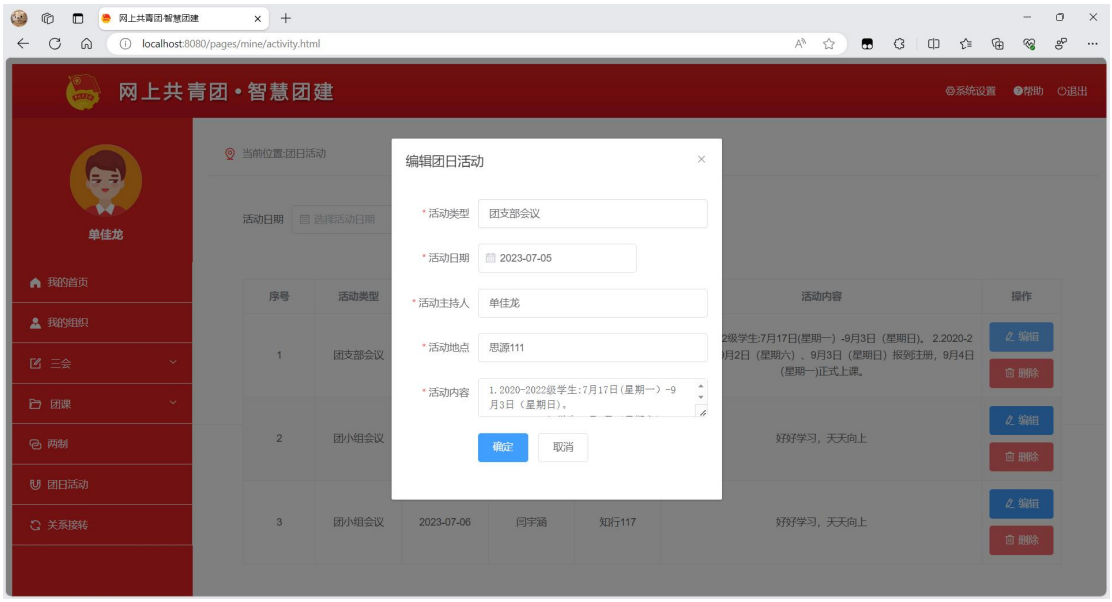

图 27 编辑团日活动界面

查询团日活动界面如图 28 所示。

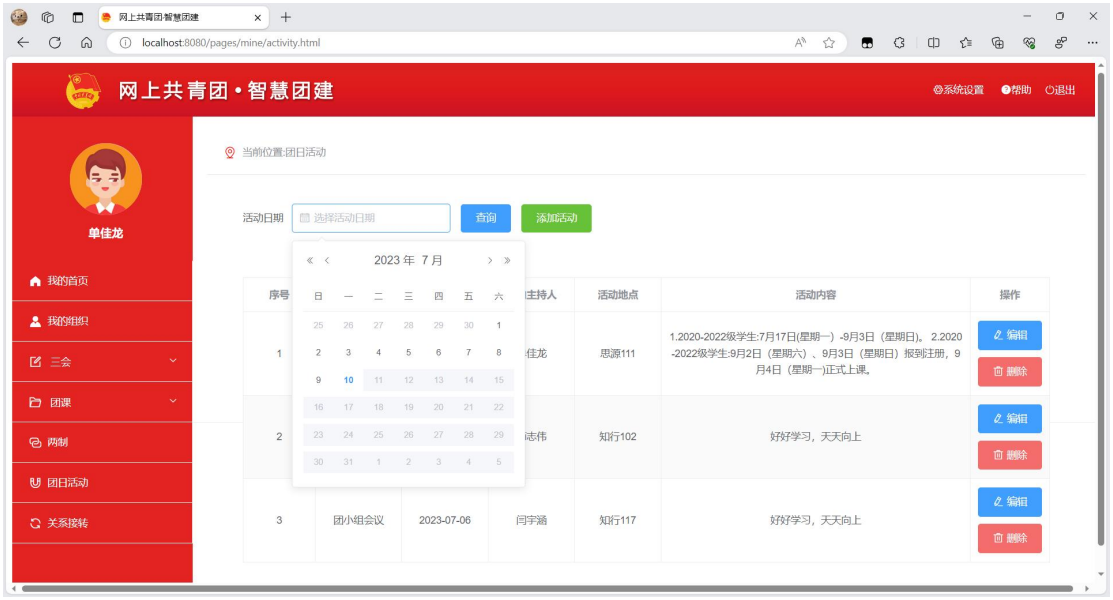

图 28 查询团日活动界面

② 核心代码

用户查看同组织下全部团日活动的核心代码如下。

```
//根据团组织获取活动列表
@GetMapping
public List<Activity> getAllByOrg(HttpSession session) {
    //从 session 读属性 user
    User user = (User) session.getAttribute("user");
    //用 user 的身份证读其他信息
    //每次都读两次，并且还得调用用户业务层，下次把组织 id 存在 session 里！！！！！
    user = userService.getByCard(user.getCard());
    return activityService.getAllByOrg(user.getOrganization());
}
```

用户为组织增加团日活动的核心代码如下。

```
//增加活动
@PostMapping
public Boolean save(@RequestBody Activity activity) {
    return activityService.save(activity);
}
```

用户为组织删除团日活动的核心代码如下。

```
//删除活动
@DeleteMapping("/{id}")
public Boolean delete(@PathVariable Integer id) {
    return activityService.delete(id);
}
```

用户为组织更新修改已有的团日活动的核心代码如下。

```
//修改活动
@PutMapping
public Boolean update(HttpSession session, @RequestBody Activity activity) {
    //从 session 读属性 user
```

```
        User user = (User) session.getAttribute("user");
        //用 user 的身份证读其他信息
        //每次都读两次，并且还得调用用户业务层，下次把组织 id 存在 session 里！！！！！
        user = userService.getByCard(user.getCard());
        activity.setOrganization(user.getOrganization());
        return activityService.update(activity);
    }
```

用户根据时间查询团日活动核心代码如下。

```
    //根据时间查询活动
    @GetMapping("/{date}")
    public List<Activity> getAllByDate(HttpSession session, @PathVariable String date) {
        //从 session 读属性 user
        User user = (User) session.getAttribute("user");
        //用 user 的身份证读其他信息
        //每次都读两次，并且还得调用用户业务层，下次把组织 id 存在 session 里！！！！！
        user = userService.getByCard(user.getCard());
        System.out.println("getAllByDate"+date);
        return activityService.getAllByDate(user.getOrganization(), date);
    }
```

6．总结

经过本次《Web 程序设计》课程的学习，我对 ssm 框架有了初步的认识。所谓 ssm 就是指 spring+springMVC+mybatis 。spring 用于实现业务的对象，springMVC 负责转发请求和视图，而 mybatis 是对 jdbc 的封装，负责数据库的操作。

Spring 是一个开源框架，Spring 是于 2003 年兴起的一个轻量级的 Java 开发框架。它是为了解决企业应用开发的复杂性而创建的。Spring 的用途不仅限于服务器端的开发。从简单性、可测试性和松耦合的角度而言，任何 Java 应用都可以从 Spring 中受益。简单来说，Spring 是一个轻量级的控制反转（IoC）和面向切面（AOP）的容器框架。

Spring MVC 属于 SpringFrameWork 的后续产品，已经融合在 Spring Web Flow 里面。Spring MVC 分离了控制器、模型对象、分派器以及处理程序对象的角色，这种分离让它们更容易进行定制。

MyBatis 本是 apache 的一个开源项目 iBatis, 2010 年这个项目改名为 MyBatis 。MyBatis 是一个基于 Java 的持久层框架。iBATIS 提供的持久层框架包括 SQL Maps 和 Data Access Objects（DAO）MyBatis 消除了几乎所有的 JDBC 代码和参数的手工设置以及结果集的检索。MyBatis 使用简单的 XML 或注解用于配置和原始映射，将接口和 Java 的 POJO 映射成数据库中的记录。我更喜欢使用 MyBatis 注解配置的方式。

SSM 框架的理解：

学习了框架之后编写代码的效率提高了，框架封装了普通项目中程序员需要重复书写的代码和简化了调用过程，就比如说在传统的 jsp 项目中，我们的 Controller 接收到前端的请求然后程序员就需要去开发 Dao 层，里面还涉及数据库的连接和存储过程的代码，大部分都是冗余的代码，而有了 SSM 框架后就极大的简化了 controller 以下层的开发，只需要一个 Service 层和 Dao 层就行了，直接用在 Dao 中使用 MyBatis 注解做 SQL 语句的开发就行了，而什么驱动程序、数据库连接的、存储的过程和结果集都直接由 Mybatis 负责了，我们只需要负责传递形参和接收返回数据就行了，这样就完成了一次完整的数据库交互！

Vue 和 Element UI 的理解：

本来是想前端随便写写的，但还是希望前端页面好看美观，所以浅浅的学习了一下前端的知识，之前已经学过了简单的 js、jQuery、ajax 和 axios。

Vue 是一套用于构建用户界面的渐进式 JavaScript 框架，开发者只需要关注视图层，它不仅易于上手，还便于与第三方库或既有项目的整合。是基于 MVVM（Model-View-ViewModel 即：视图层-视图模型层-模型层）设计思想。提供 MVVM 数据双向绑定的库，专注于 UI 层面。我感觉非常好用，比原生 js 和 jQ 方便且好用。

渐进式框架：就是一开始不需要你完全掌握它的全部功能特性，可以后续逐步增加功能。没有多做职责之外的事情

Element 是基于 Vue 实现的一套不依赖业务的 UI 组件库，提供了丰富的 PC 端组件，减少用户对常用组件的封装，降低了开发的难易程度。我感觉非常棒，在官网挑选自己喜欢的样式，粘粘改改一个还算说得过去的前端页面就好了。

总而言之，我认为学习是一辈子的事情，学计算机更是这样，科学技术永远在向前发展，我所需要学习的知识还有很多很多，前路漫漫还需加倍努力！