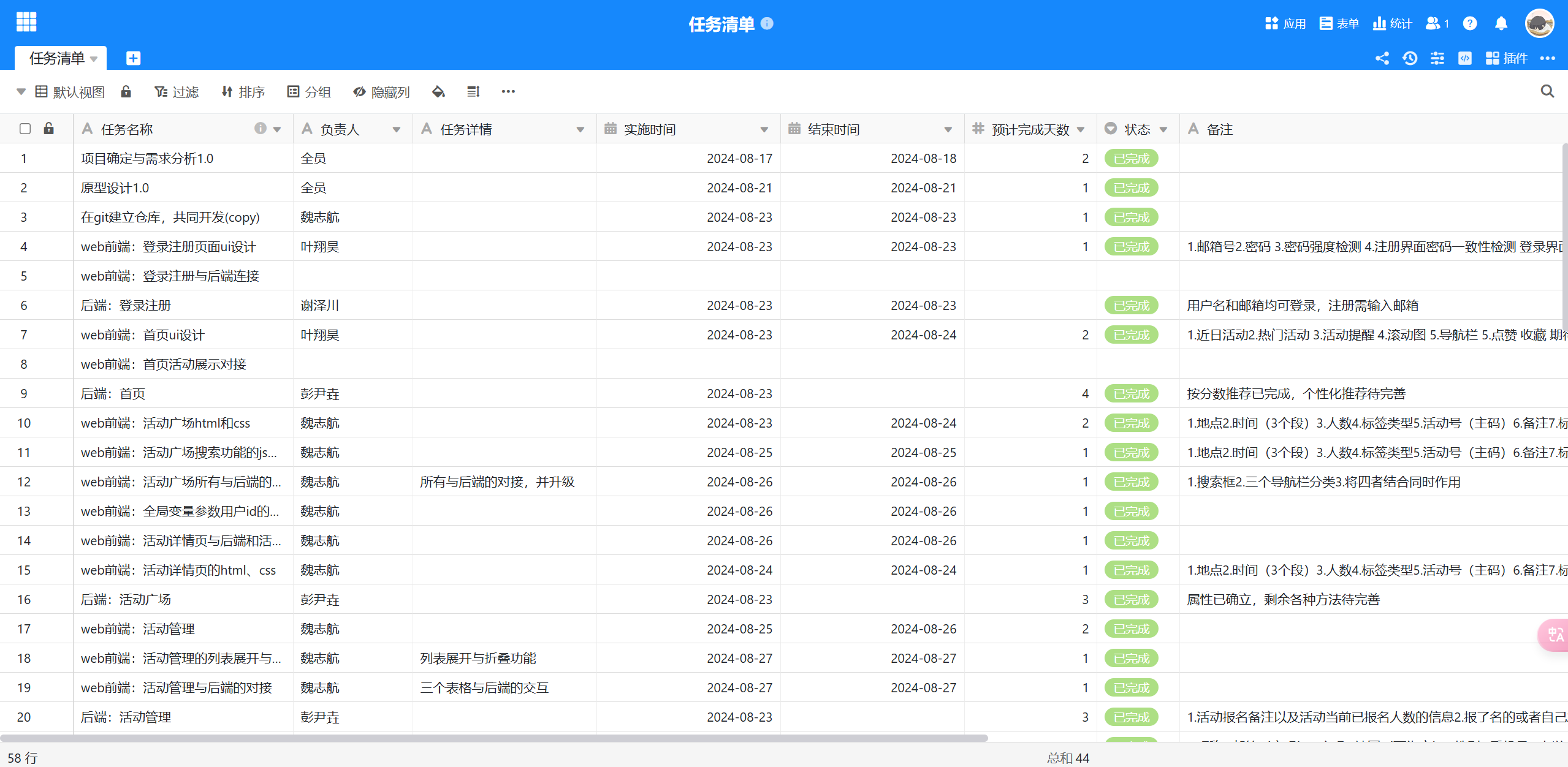
组长:魏志航

组员:兰天阳，彭尹垚，张世纪，叶翔昊，谢泽川

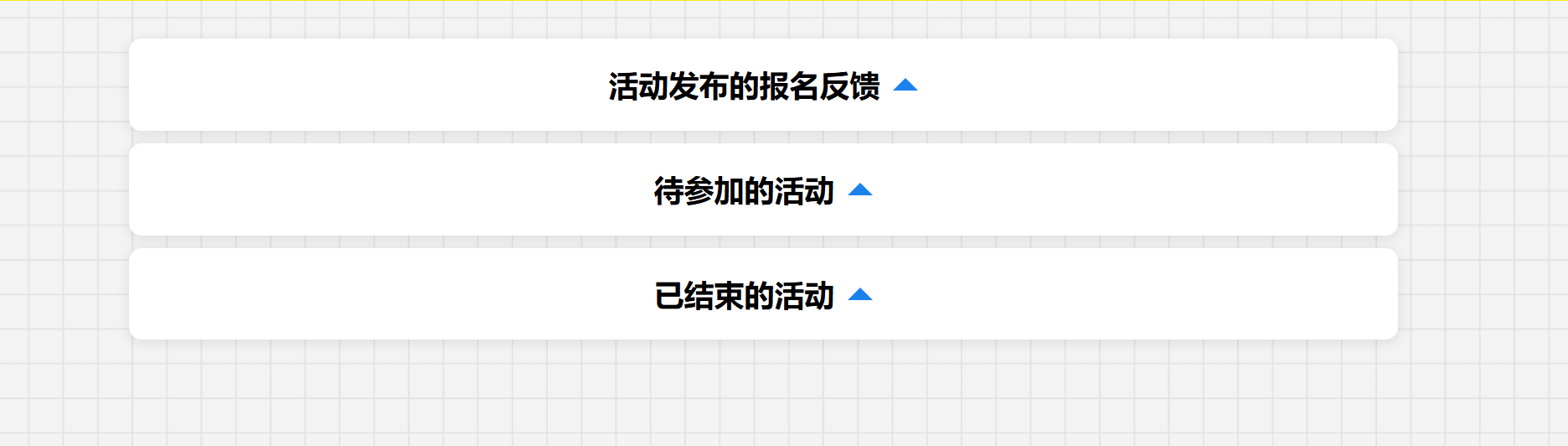
总体情况：

进度在计划中，网页端与后端的对接基本完成，手机端还剩下少许。明天做大屏和手机端剩下的部分，和项目整体的优化。

**魏志航**



**1.给活动管理页面升级添加了表格展开和折叠功能**





**<div class="mainbody-area">**

**<!-- 管理申请 -->**

**<div class="activityManagement">**

**<div>**

**<h2 class="table-title"><span class="title-text">活动发布的报名反馈</span>**

**<div class="collapse-button" onclick="toggleTable('feedbackTable')"></div>**

**</h2>**

**<table id="feedbackTable">**

**<thead>**

**<tr>**

**<th>活动名称</th>**

**<th>活动ID</th>**

**<th>报名人</th>**

**<th>备注</th>**

**<th>人数进展</th>**

**</tr>**

**</thead>**

**<tbody>**

**<tr><td>无</td><td>无</td><td>无</td><td>无</td><td>无</td></tr>**

**<!-- 其他申请可继续添加 -->**

**</tbody>**

**</table>**

**</div>**

**</div>**

**<!-- 我的活动列表模块 -->**

**<div class="MyActivity">**

**<h2 class="table-title"><span class="title-text">待参加的活动</span>**

**<div class="collapse-button" onclick="toggleTable('myActivityTable')"></div>**

**</h2>**

**<table id="myActivityTable">**

**<thead>**

**<tr>**

**<th>活动名称</th>**

**<th>活动ID</th>**

**<th>发布人名称</th>**

**<th>起始时间</th>**

**<th>结束时间</th>**

**<th>参与人数</th>**

**<th>活动地点</th>**

**<th>活动类型</th>**

**<th>发布人电话</th>**

**</tr>**

**</thead>**

**<tbody>**

**<tr><td>无</td><td>无</td><td>无</td><td>无</td><td>无</td><td>无</td><td>无</td><td>无</td><td>无</td></tr>**

**<!-- 其他活动也可以继续添加 -->**

**</tbody>**

**</table>**

**</div>**

**<!-- 结束活动列表模块 -->**

**<div class="EndActivity">**

**<h2 class="table-title"><span class="title-text">已结束的活动</span>**

**<div class="collapse-button" onclick="toggleTable('endActivityTable')"></div>**

**</h2>**

**<table id="endActivityTable">**

**<thead>**

**<tr>**

**<th>活动名称</th>**

**<th>活动ID</th>**

**<th>发布人名称</th>**

**<th>起始时间</th>**

**<th>结束时间</th>**

**<th>活动地点</th>**

**<th>联系方式</th>**

**<th>分数(默认80)</th>**

**<th>评分</th>**

**</tr>**

**</thead>**

**<tbody>**

**<tr><td>无</td><td>无</td><td>无</td><td>无</td><td>无</td><td>无</td><td>无</td><td>无</td><td>无</td></tr>**

**</tbody>**

**</table>**

**</div>**

**// 折叠/展开表格功能**

**function toggleTable(tableId) {**

**var table = document.getElementById(tableId);**

**var button = document.querySelector(`.collapse-button[onclick="toggleTable('${tableId}')"]`);**

**var currentDisplay = window.getComputedStyle(table).display;**

**if (currentDisplay === "none") {**

**table.style.display = "table"; // 显示表格**

**button.style.transform = "rotate(180deg)"; // 指示展开状态**

**} else {**

**table.style.display = "none"; // 隐藏表格**

**button.style.transform = "rotate(0deg)"; // 指示折叠状态**

**}**

**}**

**// 默认表格显示在页面加载时**

**document.addEventListener("DOMContentLoaded", function() {**

**document.querySelectorAll(".feedback-table").forEach(table => {**

**table.style.display = "table"; // 初始化时显示所有表格**

**});**

**var buttons = document.querySelectorAll('.collapse-button');**

**buttons.forEach(button => {**

**button.style.transform = "rotate(180deg)"; // 初始状态指示表格已展开**

**});**

**});**

**/\* 将折叠按钮放到标题的右侧 \*/**

**.collapse-button {**

**width: 0; /\* 初始宽度为0 \*/**

**height: 0; /\* 初始高度为0 \*/**

**border-left: 10px solid transparent; /\* 形成三角形的左边 \*/**

**border-right: 10px solid transparent; /\* 形成三角形的右边 \*/**

**border-bottom: 10px solid hsl(211, 85%, 52%); /\* 形成三角形的底边 \*/**

**cursor: pointer; /\* 鼠标指针效果 \*/**

**margin-left: 10px; /\* 给按钮一点左边距 \*/**

**transition: transform 0.3s; /\* 添加平滑过渡效果 \*/**

**}**

**/\* 初始化时隐藏表格（可选） \*/**

**#feedbackTable {**

**display: table; /\* 默认显示 \*/**

**}**

**2.完成活动管理页面与后端的对接**

// 加载活动反馈的函数

                async function loadActivityFeedback(userId) {

                    try {

                        // 第一步：请求用户活动列表

                        const activitiesResponse = await fetch(`http://192.168.1.43:8080/activities/${userId}/activities`);

                        if (!activitiesResponse.ok) {

                            throw new Error(`HTTP error! status: ${activitiesResponse.status}`);

                        }

                        const activitiesData = await activitiesResponse.json();

                        // 从活动数据中提取 activityid 和 maxceiling

                        const activities = activitiesData.data.map(activity => ({

                            activityid: activity.activityid,

                            maxceiling: activity.maxceiling

                        }));

                        // 第二步：请求活动的报名反馈和人数

                        const feedbackRows = []; // 用于存储行信息

                        for (let { activityid, maxceiling } of activities) {

                            // 请求活动的报名反馈

                            const feedbackResponse = await fetch(`http://192.168.1.43:8080/event/users?activityId=${activityid}`);

                            if (!feedbackResponse.ok) {

                                throw new Error(`HTTP error! status: ${feedbackResponse.status}`);

                            }

                            const feedbackData = await feedbackResponse.json();

                            // 获取报名人数

                            const countResponse = await fetch(`http://192.168.1.43:8080/event/${activityid}/count`);

                            let participantCount = 0; // 默认人数为0

                            if (countResponse.ok) {

                                const countData = await countResponse.json();

                                participantCount = countData.data; // 获取人数

                            }

                            // 处理反馈数据，并生成表格行

                            feedbackData.data.forEach(item => {

                                // 格式化为 当前人数/总人数

                                const participantFormat = `${participantCount}/${maxceiling || '无'}`; // maxceiling 可能为 null 或 undefined

                                feedbackRows.push(`

                                    <tr>

                                        <td>${item.activityname || '无'}</td>

                                        <td>${item.activityid || '无'}</td>

                                        <td>${item.username || '无'}</td>

                                        <td>${item.reason || '无'}</td>

                                        <td>${participantFormat || '无'}</td> <!-- 使用当前人数/总人数 -->

                                    </tr>

                                `);

                            });

                        }

                        // 更新报名反馈表格

                        const feedbackTableBody = document.getElementById("feedbackTable").getElementsByTagName("tbody")[0];

                        feedbackTableBody.innerHTML = feedbackRows.join(''); // 连接并更新表格内容

                    } catch (error) {

                        console.error('Fetching error: ', error);

                    }

                }

                // 加载用户活动的函数

                async function loadMyActivities(userId) {

                    try {

                        const response = await fetch(`http://192.168.1.43:8080/activities/user/activities?userId=${userId}`); // 修改为最新用户 ID 进行请求

                        if (!response.ok) {

                            throw new Error(`HTTP error! status: ${response.status}`);

                        }

                        const activities = await response.json();

                        updateMyActivityTable(activities); // 更新活动表格

                    } catch (error) {

                        console.error('Fetching error: ', error);

                    }

                }

                // 更新我的活动列表表格的函数

                async function updateMyActivityTable(activities) {

                    const tbody = document.getElementById("myActivityTable").getElementsByTagName("tbody")[0];

                    tbody.innerHTML = ""; // 清空现有的内容

                     // 逐个遍历活动

                    for (const activity of activities) {

                        const pnumber = await fetchPNumber(activity.headid); // 异步获取联系信息

                        const row = document.createElement("tr");

                        // 创建单元格并添加到行

                        row.innerHTML = `

                            <td>${activity.activityname || '无'}</td>

                            <td>${activity.activityid || '无'}</td>

                            <td>${activity.headusername || '无'}</td>

                            <td>${formatDate(activity.timeproceed1) || '无'}</td>

                            <td>${formatDate(activity.timeproceed2) || '无'}</td>

                            <td>${activity.maxceiling || '无'}</td>

                            <td>${activity.location || '无'}</td>

                            <td>${activity.type || '无'}</td>

                            <td>${pnumber || '无'}</td> <!-- 此处更新为获取的 pnumber -->

                        `;

                        tbody.appendChild(row);

                    }

                }

                // 异步请求 pnumber 的函数

                async function fetchPNumber(activityId) {

                    try {

                        const response = await fetch(`http://192.168.1.43:8080/auth/showuser/${activityId}`); // 请求端口以获取 pnumber

                        if (!response.ok) {

                            throw new Error(`HTTP error! status: ${response.status}`);

                        }

                        const userInfo = await response.json(); // 获取用户信息

                        return userInfo.data.pnumber || '无'; // 提取 pnumber 值

                    } catch (error) {

                        console.error('Fetching pnumber error: ', error);

                        return '未知'; // 返回默认值或者处理错误

                    }

                }

                // 加载已结束活动的函数

                async function loadEndActivities(userId) {

                    try {

                        const response = await fetch(`http://192.168.1.43:8080/activities/user/finished-activities?userId=${userId}`); // 修改为最新用户 ID 进行请求

                        if (!response.ok) {

                            throw new Error(`HTTP error! status: ${response.status}`);

                        }

                        const activities = await response.json();

                        updateEndActivityTable(activities); // 更新活动表格

                    } catch (error) {

                        console.error('Fetching error: ', error);

                    }

                }

                // 更新我的活动列表表格的函数

                async function updateEndActivityTable(activities) {

                    const tbody = document.getElementById("endActivityTable").getElementsByTagName("tbody")[0];

                    tbody.innerHTML = ""; // 清空现有的内容

                    // 逐个遍历活动

                    for (const activity of activities) {

                        const pnumber = await fetchPNumber(activity.headid); // 异步获取联系信息

                        const row = document.createElement("tr");

                        const activityId = activity.activityid; // 提取活动ID

                        // 创建单元格并添加到行

                        row.innerHTML = `

                            <td>${activity.activityname || '无'}</td>

                            <td>${activity.activityid || '无'}</td>

                            <td>${activity.headusername || '无'}</td>

                            <td>${formatDate(activity.timeproceed1) || '无'}</td>

                            <td>${formatDate(activity.timeproceed2) || '无'}</td>

                            <td>${activity.location || '无'}</td>

                            <td>${pnumber || '无'}</td> <!-- 此处更新为获取的 pnumber -->

                            <td><input type="number" min="0" max="100" step="1" placeholder="${activity.score || 80}" class="score-input" data-activity-id="${activityId}"></td>

                            <td><button type="button" class="submit-score-button">提交</button></td>

                        `;

                        // 设置提交按钮的事件处理程序

                        row.querySelector('.submit-score-button').onclick = async function() {

                            // 确保在事件处理程序里获取最新的输入值

                            const scoreInput = row.querySelector('.score-input');

                            const score = parseInt(scoreInput.value, 10); // 将输入框的值转换为整数

                            const userId = parseInt(savedUserId, 10); // 将用户ID转换为整数

                            // 读取活动ID并转换为整数

                            const activityId = parseInt(scoreInput.getAttribute('data-activity-id'), 10);

                            // 日志输出

                            console.log('userId:', userId);

                            console.log('score:', score);

                            console.log('activityId:', activityId);

                            // 提交分数的逻辑

                            try {

                                // 确保将参数转换为整数

                                const userIdInt = parseInt(userId, 10);

                                const activityIdInt = parseInt(activityId, 10);

                                const scoreInt = parseInt(score, 10); // 确保分数也是整数

                                // 创建 URLSearchParams 对象

                                const params = new URLSearchParams();

                                params.append('userid', userIdInt);

                                params.append('activityid', activityIdInt);

                                params.append('score', scoreInt);

                                const response = await fetch(`http://192.168.1.43:8080/event/${userIdInt}/activities/${activityIdInt}/rate`, {

                                    method: 'POST',

                                    headers: {

                                        'Content-Type': 'application/x-www-form-urlencoded', // 设置请求内容类型

                                    },

                                    body: params.toString() // 转换为 x-www-form-urlencoded 格式

                                });

                                if (!response.ok) {

                                    throw new Error(`HTTP error! status: ${response.status}`);

                                }

                                const result = await response.json();

                                console.log('评分提交成功:', result);

                                alert('评分提交成功'); // 可以根据需要将其更改为提醒或其他处理

                            } catch (error) {

                                console.error('提交评分失败', error);

                                alert('提交评分失败，请稍后再试。');

                            }

                        };

                        // 将行添加到表格

                        tbody.appendChild(row);

                    }

                }

                // 时间格式化函数

                function formatDate(dateString) {

                    console.log(dateString);

                    if (!dateString) {

                        return '未设置'; // 如果日期无效，返回“未设置”

                    }

                    const date = new Date(dateString);

                    // 格式化为 YYYY-MM-DD HH:MM

                    const year = date.getFullYear();

                    const month = (date.getMonth() + 1).toString().padStart(2, '0'); // 月份从 0 开始，所以需要加 1

                    const day = date.getDate().toString().padStart(2, '0');

                    const hours = date.getHours().toString().padStart(2, '0');

                    const minutes = date.getMinutes().toString().padStart(2, '0');

                    return `${year}-${month}-${day} ${hours}:${minutes}`; // 返回格式化的字符串

                }

            </script>

**3.报名、评分功能和后端的对接**

// 在点击报名时提交数据

            document.getElementById('registerButton').addEventListener('click', function() {

                const urlParams = new URLSearchParams(window.location.search);

                const activityId = urlParams.get('activityid');

                const remark = document.getElementById('remark').value; // 获取备注内容

                console.log(activityId);

                console.log(savedUserId);

                console.log(remark);

                if (!savedUserId) {

                    alert('您未登录，请先登录！');

                    return;

                }

                if (!activityId) {

                    alert('活动ID未找到。');

                    return;

                }

                // 确保备注字段不是空的

                if (!remark) {

                    alert('请填写备注。');

                    return;

                }

                const registrationData = {

                    userid: parseInt(savedUserId, 10), // 确保是整数

                    activityid: parseInt(activityId, 10), // 确保是整数

                    reason: remark // 备注内容

                };

                fetch('http://192.168.1.43:8080/event/register', {

                    method: 'POST',

                    headers: {

                        'Content-Type': 'application/x-www-form-urlencoded',

                    },

                    body: JSON.stringify(registrationData)

                })

                .then(response => {

                    if (!response.ok) {

                        throw new Error('网络响应错误');

                    }

                    return response.json();

                })

                .then(data => {

                    if (data.code === 200) {

                        alert('报名成功！');

                    } else {

                        alert(`报名失败: ${data.message}`);

                    }

                })

                .catch(error => {

                    console.error('报名时发生错误:', error);

                    alert('网络错误，请稍后再试。');

                });

            });

// 设置提交按钮的事件处理程序

                        row.querySelector('.submit-score-button').onclick = async function() {

                            // 确保在事件处理程序里获取最新的输入值

                            const scoreInput = row.querySelector('.score-input');

                            const score = parseInt(scoreInput.value, 10); // 将输入框的值转换为整数

                            const userId = parseInt(savedUserId, 10); // 将用户ID转换为整数

                            // 读取活动ID并转换为整数

                            const activityId = parseInt(scoreInput.getAttribute('data-activity-id'), 10);

                            // 日志输出

                            console.log('userId:', userId);

                            console.log('score:', score);

                            console.log('activityId:', activityId);

                            // 提交分数的逻辑

                            try {

                                // 确保将参数转换为整数

                                const userIdInt = parseInt(userId, 10);

                                const activityIdInt = parseInt(activityId, 10);

                                const scoreInt = parseInt(score, 10); // 确保分数也是整数

                                // 创建 URLSearchParams 对象

                                const params = new URLSearchParams();

                                params.append('userid', userIdInt);

                                params.append('activityid', activityIdInt);

                                params.append('score', scoreInt);

                                const response = await fetch(`http://192.168.1.43:8080/event/${userIdInt}/activities/${activityIdInt}/rate`, {

                                    method: 'POST',

                                    headers: {

                                        'Content-Type': 'application/x-www-form-urlencoded', // 设置请求内容类型

                                    },

                                    body: params.toString() // 转换为 x-www-form-urlencoded 格式

                                });

                                if (!response.ok) {

                                    throw new Error(`HTTP error! status: ${response.status}`);

                                }

                                const result = await response.json();

                                console.log('评分提交成功:', result);

                                alert('评分提交成功'); // 可以根据需要将其更改为提醒或其他处理

                            } catch (error) {

                                console.error('提交评分失败', error);

                                alert('提交评分失败，请稍后再试。');

                            }

                        };

**4.活动详情页对还在审核或结束的活动隐藏报名键并提醒用户**

                                // 判断活动状态

                                const activityStatus = activity.status; // 获取活动状态

                                const registerSection = document.getElementById('registerSection');

                                const statusMessage = document.getElementById('statusMessage');

                                if (activityStatus === '审核中' || activityStatus === '已结束') {

                                    // 隐藏报名部分，显示状态消息

                                    registerSection.style.display = 'none';

                                    statusMessage.innerText = activityStatus === '审核中'

                                        ? '该活动还在审核中。'

                                        : '该活动已结束。';

                                    statusMessage.style.display = 'block';  // 显示状态消息

                                } else {

                                    // 显示报名部分，隐藏状态消息

                                    registerSection.style.display = 'block';

                                    statusMessage.style.display = 'none';

                                }

**5.拟定大屏的基本框架**

<!DOCTYPE html>

<html lang="zh">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>活动发布平台大屏数据展示</title>

    <link rel="stylesheet" href="BigScreen.css">

</head>

<body>

    <div class="dashboard">

        <div class="panel">

            <h2>人数分类统计</h2>

            <canvas id="rankingChart"></canvas>

        </div>

        <div class="big">

            <h2>当天实时发稿量</h2>

            <div class="total">

                <span class="total-number">3891</span>

                <span>条</span>

            </div>

            <div class="categories">

                <div>宏观经济<span class="category-number">1292条</span></div>

                <div>资本市场<span class="category-number">892条</span></div>

                <div>货币市场<span class="category-number">590条</span></div>

                <div>外汇市场<span class="category-number">970条</span></div>

                <div>债券市场<span class="category-number">1002条</span></div>

                <div>大宗商品<span class="category-number">692条</span></div>

            </div>

        </div>

        <div class="panel">

            <h2>用户评分排行</h2>

            <canvas id="typeStatsChart"></canvas>

        </div>

        <div class="panel">

            <h2>活动评分排行</h2>

            <canvas id="trendChart"></canvas>

        </div>

        <div class="panel">

            <h2>状态类型占比</h2>

            <canvas id="machineVsHumanChart"></canvas>

        </div>

        <div class="panel">

            <h2>时间分布图</h2>

            <canvas id="distributionChart"></canvas>

        </div>

        <div class="panel">

            <h2>分类类型统计图</h2>

            <canvas id="typeStatsChart2"></canvas>

        </div>

    </div>

    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

    <script src="BigScreen.js"></script>

</body>

</html>

body {

    margin: 0;

    font-family: Arial, sans-serif;

    background-color: #0e1623;

    color: #ffffff;

}

.dashboard {

    display: flex; /\* 使用 Grid 布局 \*/

    flex-direction: row;

    flex-wrap: wrap;

    justify-content: center; /\* 水平居中 \*/

    padding: 0; /\* 去掉内边距，以便完全填充 \*/

    height: 100vh; /\* 让 dashboard 填满整个视口高度 \*/

    width: 100vw; /\* 让 dashboard 填满整个视口宽度 \*/

}

.panel {

    background-color: #1e2a38;

    padding: 20px;

    border-radius: 10px;

    box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);

    width: 21vw;

    height: 50vh;

    margin: 0.5%;

}

.big {

    background-color: #1e2a38;

    padding: 20px;

    border-radius: 10px;

    box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);

    width: 45.5vw;

    height: 50vh;

    margin: 0.5% ;

}

.panel h2 {

    margin: 0 0 20px;

    font-size: 18px;

    border-bottom: 2px solid #444;

    padding-bottom: 10px;

}

.total {

    font-size: 48px;

    display: flex;

    align-items: baseline;

}

.total-number {

    color: #00aaff;

    margin-right: 10px;

}

.categories {

    display: flex;

    flex-wrap: wrap;

    gap: 10px;

}

.categories div {

    background-color: #293748;

    padding: 10px;

    border-radius: 5px;

    flex: 1 1 30%;

    display: flex;

    justify-content: space-between;

    align-items: center;

}

.category-number {

    color: #ffaa00;

}

// 示例数据

const rankingData = {

    labels: ["周成武", "陈羽翔", "罗开平", "裴玉娟", "小乔", "张国东", "温汇", "万琳霞", "立诚君", "严平"],

    datasets: [{

        label: '发稿量',

        data: [635, 400, 300, 203, 107, 100, 65, 38, 31, 5],

        backgroundColor: '#00aaff'

    }]

};

const typeStatsData = {

    labels: ["宏观经济", "资本市场", "货币市场", "外汇市场", "债券市场", "大宗商品"],

    datasets: [{

        label: '原创',

        data: [10, 9, 11, 11, 8, 14],

        backgroundColor: '#ffaa00'

    }, {

        label: '编辑',

        data: [20, 15, 21, 18, 13, 17],

        backgroundColor: '#00ff7f'

    }, {

        label: '编译',

        data: [15, 14, 19, 12, 11, 15],

        backgroundColor: '#0033ff'

    }]

};

const trendData = {

    labels: ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"],

    datasets: [{

        label: '宏观经济',

        data: [10, 20, 30, 40, 30, 20, 10, 20, 30, 40, 30, 20],

        borderColor: '#ffaa00',

        fill: false

    }, {

        label: '资本市场',

        data: [15, 25, 35, 45, 35, 25, 15, 25, 35, 45, 35, 25],

        borderColor: '#00ff7f',

        fill: false

    }, {

        label: '货币市场',

        data: [20, 30, 40, 50, 40, 30, 20, 30, 40, 50, 40, 30],

        borderColor: '#0033ff',

        fill: false

    }]

};

const machineVsHumanData = {

    labels: ['机器', '人工'],

    datasets: [{

        data: [22.1, 77.9],

        backgroundColor: ['#ffaa00', '#00aaff']

    }]

};

const distributionData = {

    labels: ['大宗商品', '债券市场', '外汇市场', '货币市场', '资本市场', '宏观经济'],

    datasets: [{

        data: [35, 20, 15, 5, 10, 15],

        backgroundColor: ['#ffaa00', '#00aaff', '#00ff7f', '#0033ff', '#ff00aa', '#ffcc00']

    }]

};

// 渲染图表

function renderChart(id, type, data, options = {}) {

    new Chart(document.getElementById(id), {

        type: type,

        data: data,

        options: options

    });

}

window.onload = function() {

    renderChart('rankingChart', 'bar', rankingData);

    renderChart('typeStatsChart', 'bar', typeStatsData);

    renderChart('trendChart', 'line', trendData);

    renderChart('machineVsHumanChart', 'doughnut', machineVsHumanData);

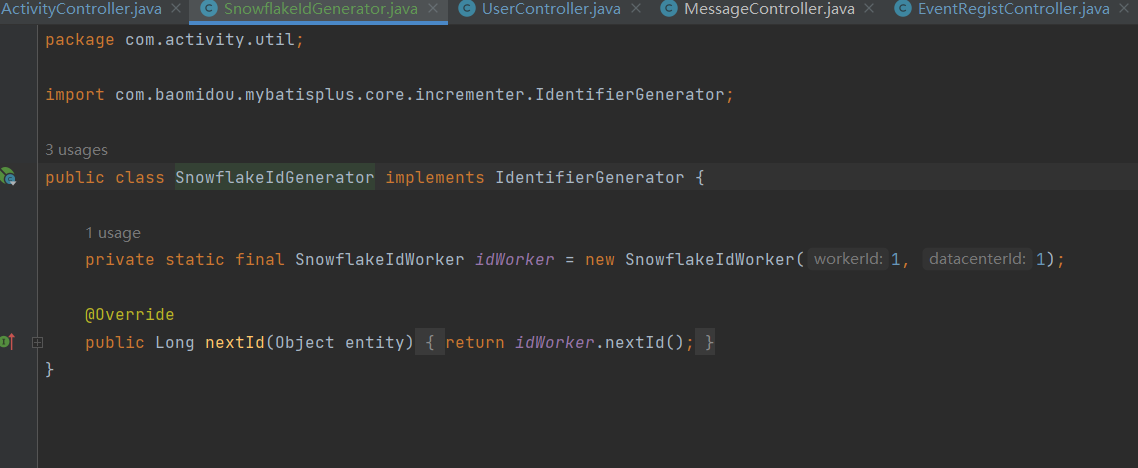
    renderChart('distributionChart', 'pie', distributionData);

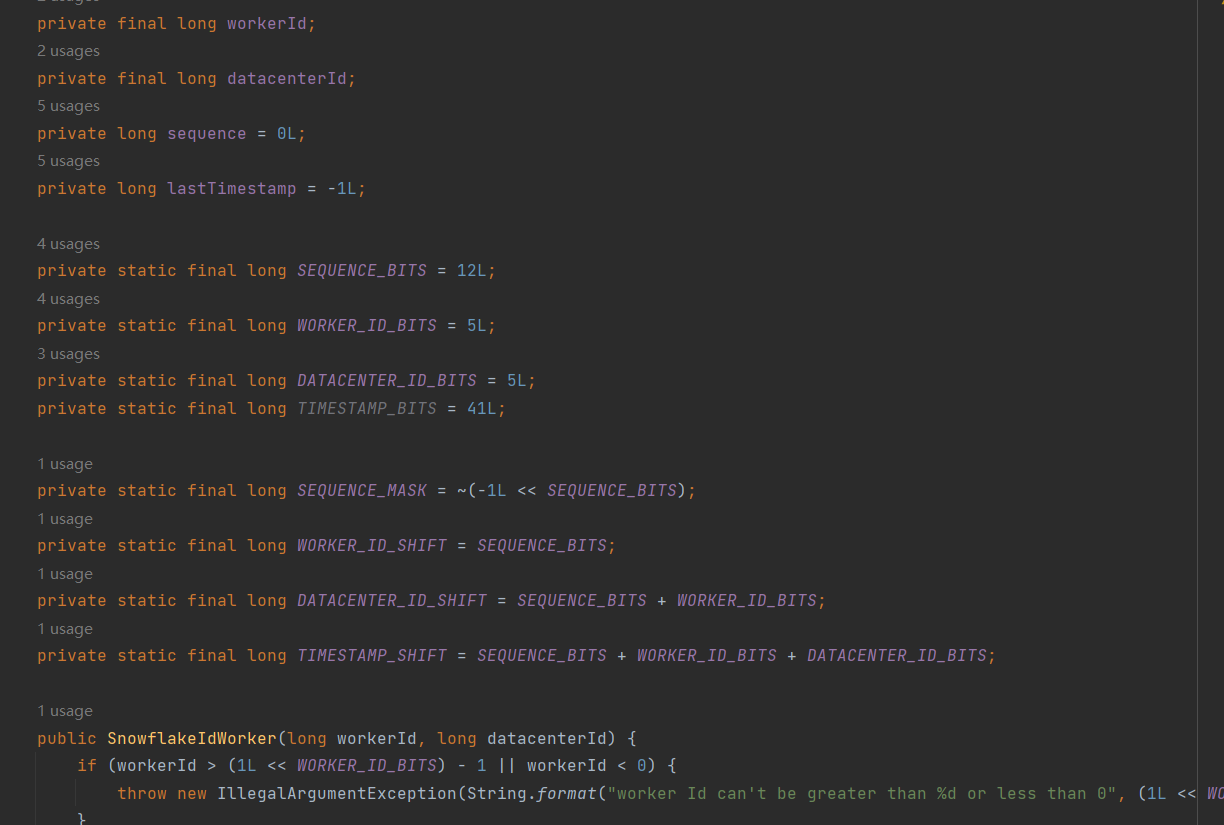
    renderChart('typeStatsChart2', 'bar', typeStatsData);

};

**彭尹垚**

今天主要参与了前端对接，写了id自生成算法，参与了大屏端的建设，登陆检查等







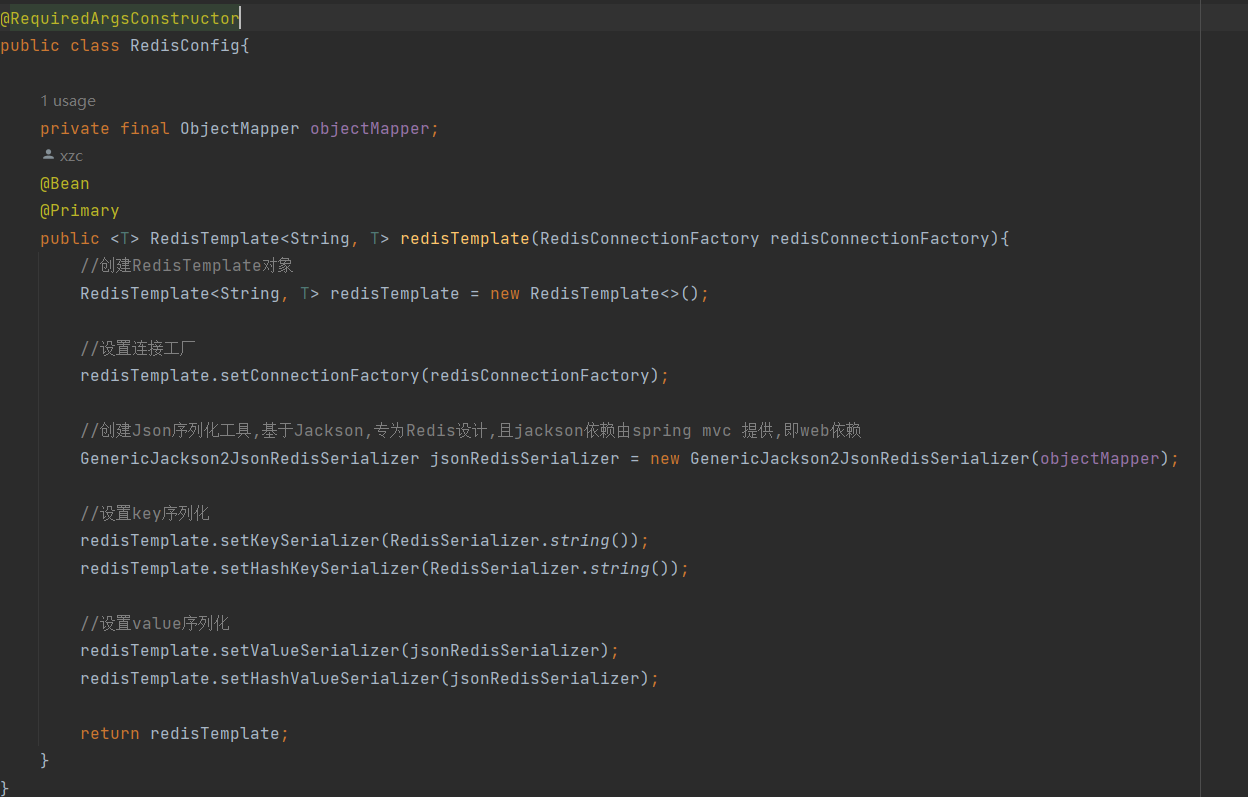
**谢泽川**

##### Java学习日志 - JAVASpringboot对校园活动的编写实现

##### 日期：2024年8月27日

##### 实现内容：

Redis缓存的实现，加快了查询数据的速度：

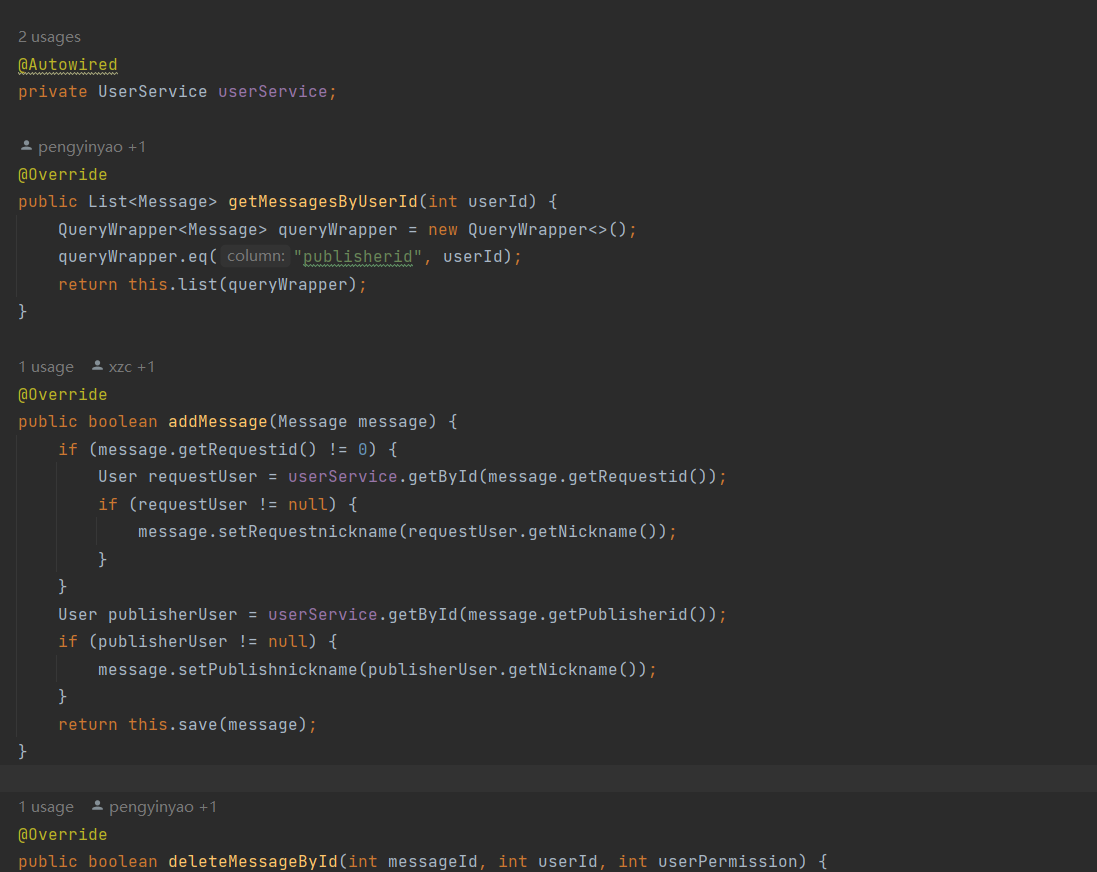


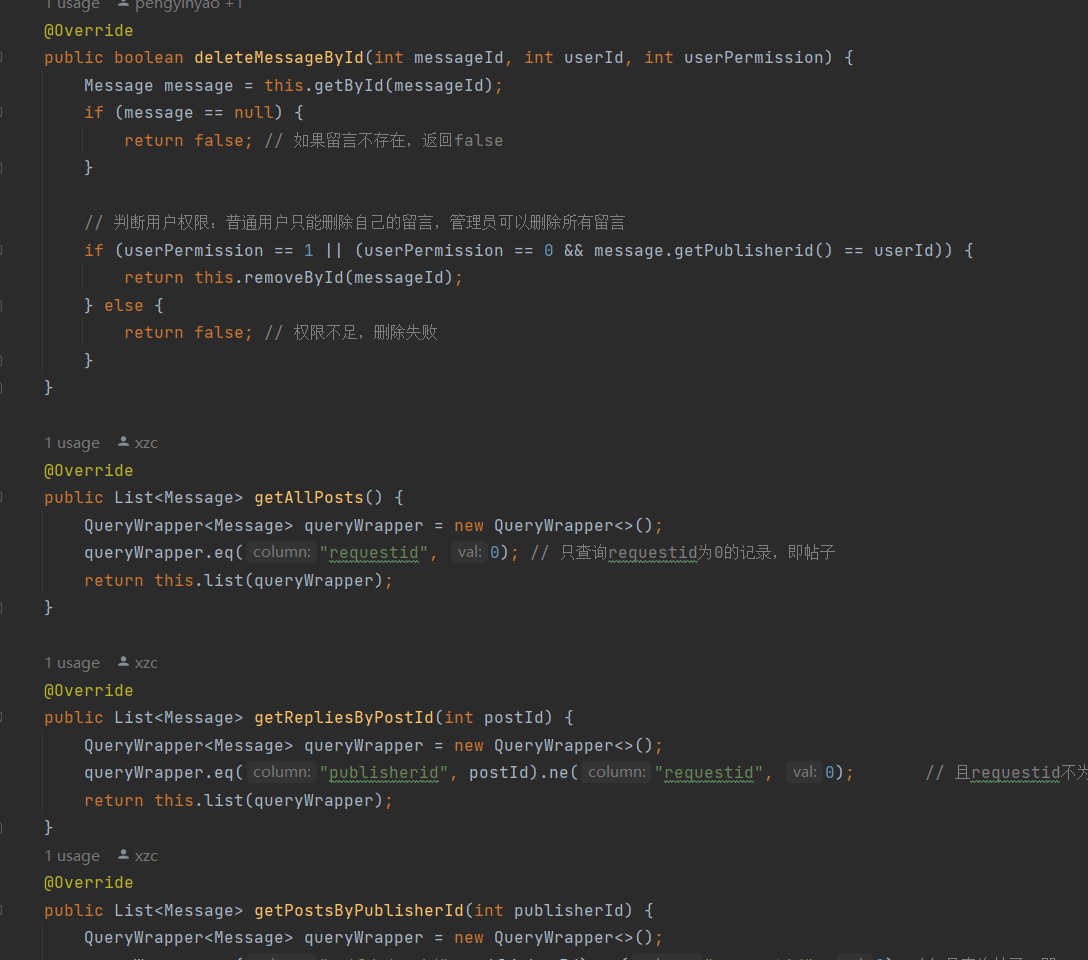


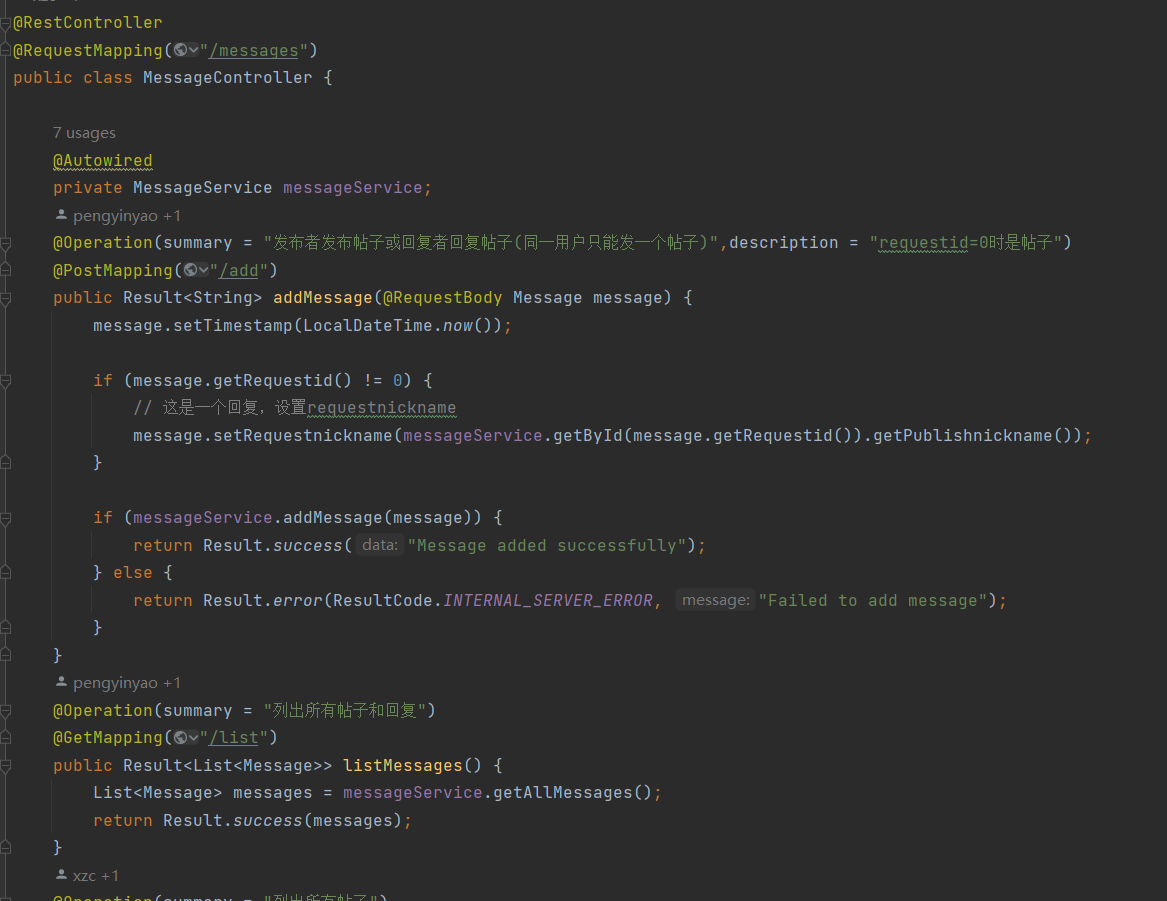


留言板的回复与发布功能：

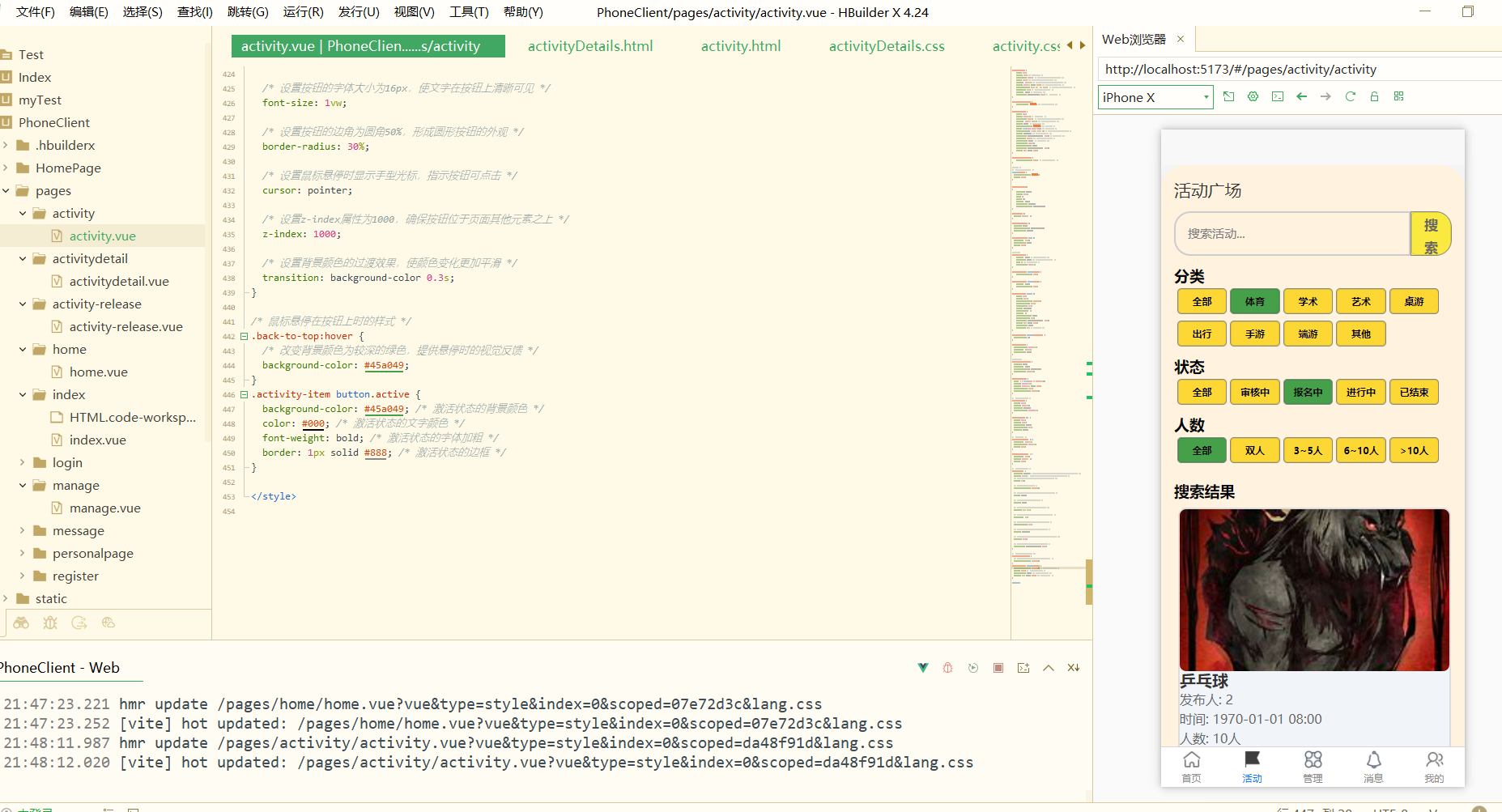
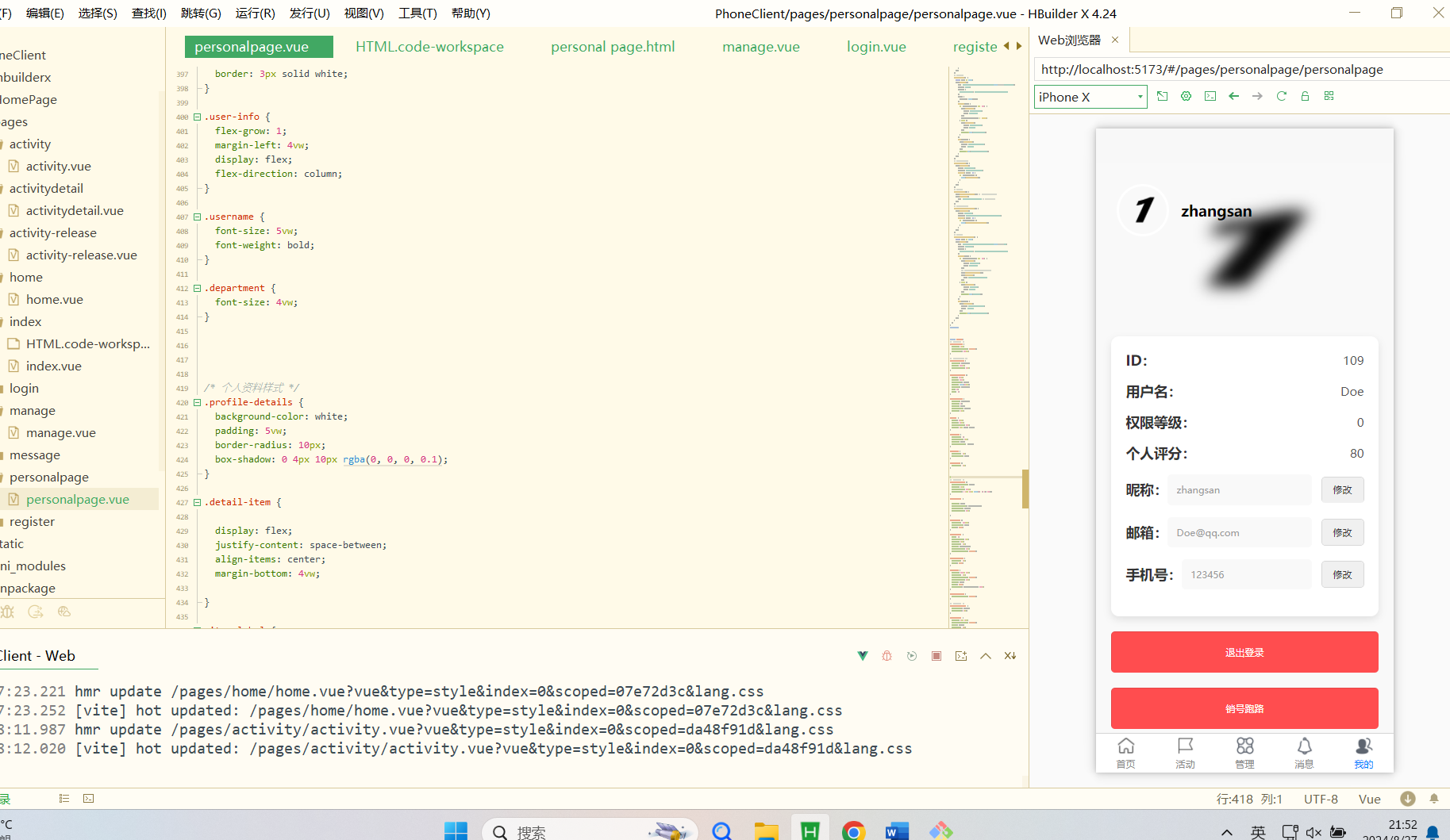








**张世纪**

1. 移动端活动页面口对接完成
2. 移动端活动页面对接完成
3. 优化了登陆界面

**叶翔昊**

8·27实训日志

总体情况：作为前端完成了留言板和登录状态后端的对接。

代码：

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title></title>

<style type="text/css">

\*{

margin: 0;

padding: 0;

}

body{

height: 100vh;

display: flex;

justify-content: center;

align-items: center;

}

.container{

position: absolute;

background-color: #90cbe3;

width:300px;

height: 500px;

border: 1px soid beige;

border-radius: 5%;

box-shadow: 10px 10px 15px 5px rgba(70, 70, 70, 0.6);

}

.content{

display: flex;

flex-direction: column;

padding: 100px 0 0px 0px;

}

.content.input{

margin: 0 auto 50px auto;

width: 300px;

height: 50px;

border: 1 0px solid rgba(187, 110, 225, 5);

}

.content.input:nth-child(2){

margin: 80px auto 50px auto;

width: 300px;

height: 50px;

border: 1px solid rgba(187, 110, 225, 5);

}

.content button{

margin: 40px auto 10px auto;

width: 300px;

height: 35px;

}

.content.span{

margin:40px auto 40px 110px;

width:300px;

height: 35px;

}

</style>

</head>

<body>

<div class="container">

<div class="content">

<input type="text" placeholder="用户名" />

<input type="password" placeholder="密码" />

<button type="button">登录</button>

<span>没有账号？<a href="#">去注册</a></span>

</div>

</div>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>用户登录</title>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css">

<style>

html, body {

height: 100%;

margin: 0;

padding: 0;

font-family: Arial, sans-serif;

overflow: hidden;

}

body {

background: url('123/original.jpg') no-repeat center center fixed;

background-size: cover;

position: relative;

min-height: 100vh;

}

body::before {

content: "";

position: absolute;

top: 0;

left: 0;

width: 100%;

height: 100%;

background-color: rgba(0, 0, 0, 0.5);

z-index: -1;

}

.header {

text-align: center;

color: white;

margin-top: 50px;

font-size: 2em;

text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);

}

.container {

max-width: 400px;

margin: 100px auto;

padding: 20px;

background-color: rgba(255, 255, 255, 0.9);

border-radius: 15px; /\* 圆滑边缘 \*/

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2); /\* 阴影效果 \*/

position: relative;

width: 100%;

height: auto;

}

h2 {

text-align: center;

}

label {

display: block;

margin-top: 10px;

}

.password-container {

position: relative;

}

input[type="password"], input[type="text"] {

width: 100%;

padding: 8px;

margin-top: 5px;

box-sizing: border-box;

border-radius: 5px; /\* 输入框圆滑边缘 \*/

border: 1px solid #ccc; /\* 输入框边框 \*/

}

.toggle-password {

position: absolute;

right: 10px;

top: 50%;

transform: translateY(-50%);

cursor: pointer;

}

button {

width: 100%;

padding: 15px;

background-color: #28a745;

color: white;

border: none;

border-radius: 5px;

margin-top: 20px;

cursor: pointer;

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

font-size: 16px;

font-weight: bold;

text-transform: uppercase;

}

button:hover {

background-color: #218838;

}

.error {

color: red;

margin-top: 10px;

text-align: center;

}

.switch-page {

text-align: center;

margin-top: 5px;

}

.switch-page a {

color: #007bff;

text-decoration: none;

}

.switch-page a:hover {

text-decoration: underline;

}

.quote { /\* 样式定义 \*/

text-align: center;

margin-top: 20px;

font-style: italic;

color: white;

text-shadow: 1px 1px 2px rgba(0, 0, 0, 0.5);

}

</style>

</head>

<body>

<div class="header">学生活动发布系统</div>

<div class="container">

<h2>登录</h2>

<form id="loginForm">

<label for="loginUsername">账号：</label>

<input type="text" id="loginUsername" placeholder="用户名/邮箱" required>

<label for="loginPassword">密码：</label>

<div class="password-container">

<input type="password" id="loginPassword" placeholder="请输入密码" required>

<i class="fas fa-eye-slash toggle-password" onclick="togglePassword('loginPassword')"></i>

</div>

<button type="submit">登录</button>

<p class="error" id="loginError"></p>

</form>

<div class="switch-page">

<p>没有账户？<a href="register.html">点此注册</a></p>

</div>

</div>

<div class="quote" id="quote"></div> <!-- 显示名言的容器 -->

<script>

// 名言数组

const quotes = [

"「生活不只是眼前的苟且，还有诗和远方。」",

"「知之愈明，则行之愈笃;行之愈笃，则知之益明。」",

"「你必须参加生活中的每一场战斗，才能真正享受生活。」",

"「天空没有翅膀的痕迹，但鸟儿已经飞过。」",

"「行动是治愈恐惧的良药，而犹豫、拖延将不断滋养恐惧。」",

"「世界是一本大书，若不到处走走的话，看到的总是同一页的内容。」",

"「如果你不出去走走，你就会以为这就是世界。」",

];

// 随机选择名言并显示

function displayRandomQuote() {

const randomIndex = Math.floor(Math.random() \* quotes.length);

const quoteElement = document.getElementById('quote');

quoteElement.textContent = quotes[randomIndex];

}

// 页面加载时显示随机名言

window.onload = displayRandomQuote;

function togglePassword(inputId) {

const input = document.getElementById(inputId);

const icon = input.nextElementSibling;

if (input.type === "password") {

input.type = "text";

icon.classList.remove("fa-eye-slash");

icon.classList.add("fa-eye");

} else {

input.type = "password";

icon.classList.remove("fa-eye");

icon.classList.add("fa-eye-slash");

}

}

document.getElementById('loginForm').addEventListener('submit', async function (event) {

event.preventDefault();

const username = document.getElementById('loginUsername').value;

const password = document.getElementById('loginPassword').value;

try {

const response = await fetch('http://192.168.1.43:8080/auth/login', {

method: 'POST',

headers: {

'Content-Type': 'application/x-www-form-urlencoded'

},

body: new URLSearchParams({

'usernameoremail': username,

'password': password

})

});

const result = await response.json();

console.log('Response Status:', response.status);

console.log('Response Data:', result);

if (response.ok && result.code === 200) {

// 保存用户信息到 localStorage

localStorage.setItem('user', JSON.stringify(result.data));

////////

const headid= result.data.id; // 这个值应该是从服务器获取的用户 ID

// 保存用户 ID 到 localStorage

localStorage.setItem('headid', headid);

console.log(headid);

///////

// 登录成功后跳转到主页

window.location.href = '../homepage/homepage.html';

} else {

document.getElementById('loginError').textContent = '账号或密码错误';

}

} catch (error) {

console.error('Error:', error); // 在控制台中记录错误

document.getElementById('loginError').textContent = '网络错误，请稍后重试';

}

});

</script>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>用户注册</title>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css">

<style>

html, body {

height: 100%;

margin: 0;

padding: 0;

font-family: Arial, sans-serif;

overflow: hidden;

}

body {

background: url('123/original.jpg') no-repeat center center fixed;

background-size: cover;

position: relative;

min-height: 100vh;

}

body::before {

content: "";

position: absolute;

top: 0;

left: 0;

width: 100%;

height: 100%;

background-color: rgba(0, 0, 0, 0.5);

z-index: -1;

}

.header {

font-family: "微软雅黑", sans-serif;

text-align: center;

color: white;

margin-top: 50px;

font-size: 2em;

text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);

}

.container {

max-width: 400px;

margin: 50px auto;

padding: 20px;

background-color: rgba(255, 255, 255, 0.9);

border-radius: 15px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);

position: relative;

width: 100%;

height: auto;

}

h2 {

text-align: center;

}

label {

display: block;

margin-top: 10px;

}

.password-container {

position: relative;

}

input[type="password"], input[type="text"], input[type="email"] {

width: 100%;

padding: 8px;

margin-top: 5px;

box-sizing: border-box;

border-radius: 5px;

border: 1px solid #ccc;

}

.toggle-password {

position: absolute;

right: 10px;

top: 50%;

transform: translateY(-50%);

cursor: pointer;

}

.check-mark {

position: absolute;

right: 35px;

top: 50%;

transform: translateY(-50%);

color: green;

display: none;

}

.strength-bar-container {

display: none;

position: relative;

top: -20px; /\* 调整这个值来控制与输入框的距离 \*/

margin-bottom: -20px; /\* 确保与下一个元素有合适的间距 \*/

width: 50%;

margin-left: 55px;

}

.strength-bar {

display: flex;

height: 5px; /\* 调整整体高度 \*/

}

.strength-bar div {

flex: 1;

background-color: #ccc;

margin: 0 2px; /\* 为各个进度条添加间距 \*/

}

.strength-text {

font-size: 0.8em;

margin-top: 5px;

}

.error {

color: red;

margin-top: 10px;

text-align: center;

}

.switch-page {

text-align: center;

margin-top: 20px;

}

.switch-page a {

color: #007bff;

text-decoration: none;

}

.switch-page a:hover {

text-decoration: underline;

}

button {

width: 100%;

padding: 15px;

background-color: #28a745;

color: white;

border: none;

border-radius: 5px;

margin-top: 20px;

cursor: pointer;

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

font-size: 16px;

font-weight: bold;

text-transform: uppercase;

}

button:hover {

background-color: #218838;

}

</style>

</head>

<body>

<div class="header">学生活动发布系统</div>

<div class="container">

<h2>注册</h2>

<form id="registerForm">

<label for="registerUsername">用户名：</label>

<input type="text" id="registerUsername" name="username" placeholder="请输入用户名" required>

<label for="registerEmail">邮箱：</label>

<input type="email" id="registerEmail" name="email" placeholder="请输入邮箱" required>

<label for="registerPassword">密码：</label>

<div class="strength-bar-container" id="strengthBarContainer">

<div class="strength-bar">

<div id="bar1"></div>

<div id="bar2"></div>

<div id="bar3"></div>

<div id="bar4"></div>

<div id="bar5"></div>

</div>

<div class="strength-text" id="strengthText">密码强度：非常弱</div>

</div>

<div class="password-container">

<input type="password" id="registerPassword" name="password" placeholder="请输入密码" required>

<i class="fas fa-eye-slash toggle-password" onclick="togglePassword('registerPassword')"></i>

<i class="fas fa-check check-mark" id="passwordMatchCheck"></i>

</div>

<label for="confirmPassword">确认密码：</label>

<div class="password-container">

<input type="password" id="confirmPassword" name="confirmPassword" placeholder="请再次输入密码" required>

<i class="fas fa-eye-slash toggle-password" onclick="togglePassword('confirmPassword')"></i>

<i class="fas fa-check check-mark" id="confirmMatchCheck"></i>

</div>

<button type="submit">注册</button>

<p class="error" id="registerError"></p>

</form>

<div class="switch-page">

<p>已有账户？<a href="login.html">点此登录</a></p>

</div>

</div>

<script>

function togglePassword(inputId) {

const input = document.getElementById(inputId);

const icon = input.nextElementSibling;

if (input.type === "password") {

input.type = "text";

icon.classList.remove("fa-eye-slash");

icon.classList.add("fa-eye");

} else {

input.type = "password";

icon.classList.remove("fa-eye");

icon.classList.add("fa-eye-slash");

}

}

function checkPasswordStrength(password) {

const strengthBarContainer = document.getElementById('strengthBarContainer');

const strengthBar = [

document.getElementById('bar1'),

document.getElementById('bar2'),

document.getElementById('bar3'),

document.getElementById('bar4'),

document.getElementById('bar5')

];

const strengthText = document.getElementById('strengthText');

if (password) {

strengthBarContainer.style.display = 'block';

} else {

strengthBarContainer.style.display = 'none';

return;

}

let score = 0;

// 密码长度

if (password.length <= 4) {

score += 5;

} else if (password.length <= 7) {

score += 10;

} else {

score += 25;

}

// 字母

const hasLower = /[a-z]/.test(password);

const hasUpper = /[A-Z]/.test(password);

if (hasLower || hasUpper) {

score += hasLower && hasUpper ? 20 : 10;

}

// 数字

const numMatches = (password.match(/\d/g) || []).length;

if (numMatches > 0) {

score += numMatches === 1 ? 10 : 20;

}

// 符号

const symbolMatches = (password.match(/[^a-zA-Z0-9]/g) || []).length;

if (symbolMatches > 0) {

score += symbolMatches === 1 ? 10 : 25;

}

// 奖励

if (hasLower || hasUpper) {

if (numMatches > 0) {

if (symbolMatches > 0) {

score += hasLower && hasUpper ? 5 : 3;

} else {

score += 2;

}

}

}

// 更新进度条颜色

strengthBar.forEach(bar => bar.style.backgroundColor = '#ccc');

if (score >= 90) {

strengthText.textContent = '密码强度：非常安全';

strengthBar.forEach(bar => bar.style.backgroundColor = 'purple');

} else if (score >= 80) {

strengthText.textContent = '密码强度：安全';

strengthBar.slice(0, 4).forEach(bar => bar.style.backgroundColor = 'purple');

} else if (score >= 70) {

strengthText.textContent = '密码强度：非常强';

strengthBar.slice(0, 3).forEach(bar => bar.style.backgroundColor = 'red');

} else if (score >= 60) {

strengthText.textContent = '密码强度：强';

strengthBar.slice(0, 3).forEach(bar => bar.style.backgroundColor = 'orange');

} else if (score >= 50) {

strengthText.textContent = '密码强度：一般';

strengthBar.slice(0, 2).forEach(bar => bar.style.backgroundColor = 'green');

} else if (score >= 25) {

strengthText.textContent = '密码强度：弱';

strengthBar.slice(0, 1).forEach(bar => bar.style.backgroundColor = 'green');

} else {

strengthText.textContent = '密码强度：非常弱';

}

}

function checkPasswordsMatch() {

const password = document.getElementById('registerPassword').value;

const confirmPassword = document.getElementById('confirmPassword').value;

const passwordMatchCheck = document.getElementById('passwordMatchCheck');

const confirmMatchCheck = document.getElementById('confirmMatchCheck');

if (password && confirmPassword && password === confirmPassword) {

passwordMatchCheck.style.display = 'inline';

confirmMatchCheck.style.display = 'inline';

} else {

passwordMatchCheck.style.display = 'none';

confirmMatchCheck.style.display = 'none';

}

}

document.getElementById('registerPassword').addEventListener('input', function () {

const password = this.value;

checkPasswordStrength(password);

checkPasswordsMatch();

});

document.getElementById('confirmPassword').addEventListener('input', checkPasswordsMatch);

document.getElementById('registerForm').addEventListener('submit', async function(event) {

event.preventDefault(); // 防止表单默认提交

const username = document.getElementById('registerUsername').value;

const email = document.getElementById('registerEmail').value;

const password = document.getElementById('registerPassword').value;

const confirmPassword = document.getElementById('confirmPassword').value;

if (password !== confirmPassword) {

document.getElementById('registerError').textContent = '密码和确认密码不一致';

return;

}

const userData = {

username: username,

password: password,

email: email

};

try {

const response = await fetch('http://192.168.1.43:8080/auth/register', {

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify({

id: 0,

username: userData.username,

password: userData.password,

email: userData.email,

nickname: "",

community: "",

sex: "",

pnumber: 0,

university: "",

hobby: "",

permission: 0,

reputation: 0,

profileimageurl: ""

})

});

const result = await response.json();

if (response.ok&&result.code === 200) {

alert('注册成功！');

window.location.href = 'login.html'; // 跳转到登录页面

} else if (response.status === 400) {

document.getElementById('registerError').textContent = '用户名已存在';

} else {

document.getElementById('registerError').textContent = '用户名已存在';

}

} catch (error) {

console.error('Error:', error);

document.getElementById('registerError').textContent = '注册失败，请稍后再试。';

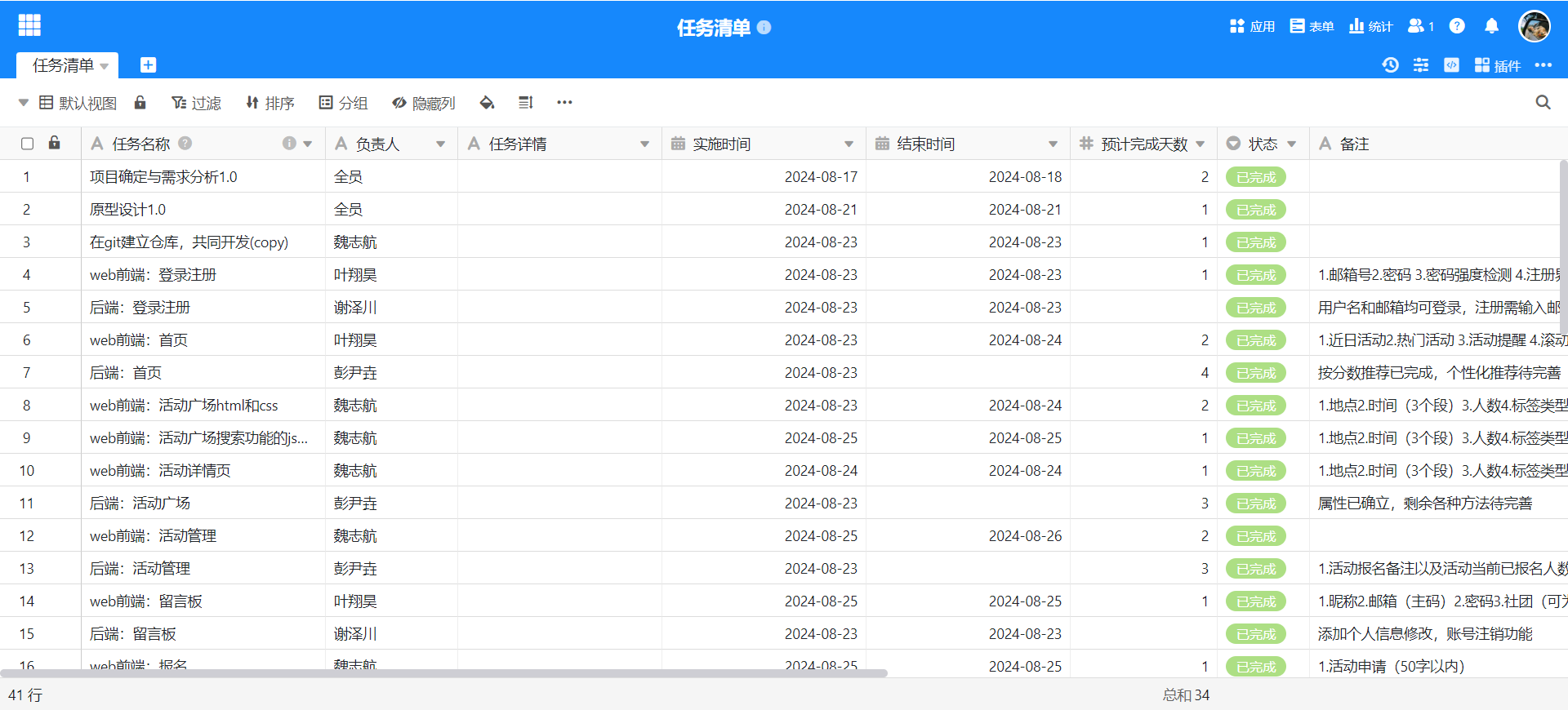
}

});

</script>

</body>

</html>



**兰天阳**

2024.8.27

实现了主页、活动发布、活动详情页、活动跳转的前后端口对接，对接了部分留言板功能，但发帖和回复的问题仍未解决



