系统开发工具及其技术简介

**一、开发环境：**

|  |
| --- |
| 后端：  开发工具：Intellij IDEA 2019年1月  JDK版本： JDK8  项目构建软件：MAVEN 3.3  数据库：MYSQL 5.7  服务器：TOMCAT 7.0  前端：  开发工具： Visual Studio Code  JavaScript框架： Vue v2.4  运行环境 node.js v12.14.1  主要组件：Element-UI  模块加载器： WebPack v2.0 |

**二、技术选型：**

|  |
| --- |
| 开发搭建框架： SpringBoot 2.1.6.RELEASE  Spring 5.1.8.RELEASE  Spring MVC 5.1.8.RELEASE  持久化框架： Mybatis 1.3.2  数据库连接池： com.alibaba.druid 1.0.9  缓存技术： Redis 4.0 |

**三、数据库表设计：**

**User表：**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | KEY | 默认值/描述 |
| unumber | varchar(50) | PRIMARY KEY | 主键/ id |
| upassword | varchar(50) | Not null | 用户密码 |
| uname | varchar(30) | Not null | 用户姓名 |
| uage | int(3) | Not null | 用户年龄 |
| utel | varchar(15) | Not null | 用户电话 |
| uemail | varchar(30) | Not null | 用户电子邮箱 |
| usex | int(2) | Not null | 用户性别 |

**Role表：**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | KEY | 默认值/描述 |
| rid | int(5) | PRIMARY KEY | 主键/ id |
| rname | varchar(20) | Not null | 角色名称 |

**User\_Role表：**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | KEY | 默认值/描述 |
| Urid | int(5) | PRIMARY KEY | 主键/ id |
| useid | varchar(50) | Not null | 外键/用户id |
| Roleid | int(5) | Not null | 外键/角色ID |
| description | varchar(50) | Not null | 角色描述 |

**一个用户对应一个角色**

**Permission表：**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | KEY | 默认值/描述 |
| Pid | int(5) | PRIMARY KEY | 主键/ id |
| Pname | varchar(20) | Not null | 权限名称 |
| purl | varchar(50) | Not null | 接口路径 |

**Role\_permission表：**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | KEY | 默认值/描述 |
| id | int(5) | PRIMARY KEY | 主键/ id |
| Roleid | Int(5) | Not null | 角色id |
| Pid | Int(5) | Not null | 权限id |

**Notice表（公告）：**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | KEY | 默认值/描述 |
| nid | int(10) | PRIMARY KEY | 主键/ id |
| ntitle | varchar(200) | Not null | 公告标题 |
| ncreater | varchar(10) | Not null | 公告发布者 |
| ntime | varchar(40) | Not null | 公告发布时间 |
| nreadcount | int(10) | Not null | 公告阅读量 |
| nroot | int(2) | Not null | 公告阅级别 |
| ncontent | text | Not null | 公告内容 |

**Project表(项目)：**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | KEY | 默认值/描述 |
| pid | int(10) | PRIMARY KEY | 主键/ id |
| ptitle | varchar(100) | Not null | 项目名称 |
| pdes | varchar(100) | Not null | 项目描述 |
| pcreater | varchar(10) | Not null | 发布者 |
| pcharger | varchar(10) | Not null | 负责人 |
| ptime | varchar(50) | Not null | 发布时间 |

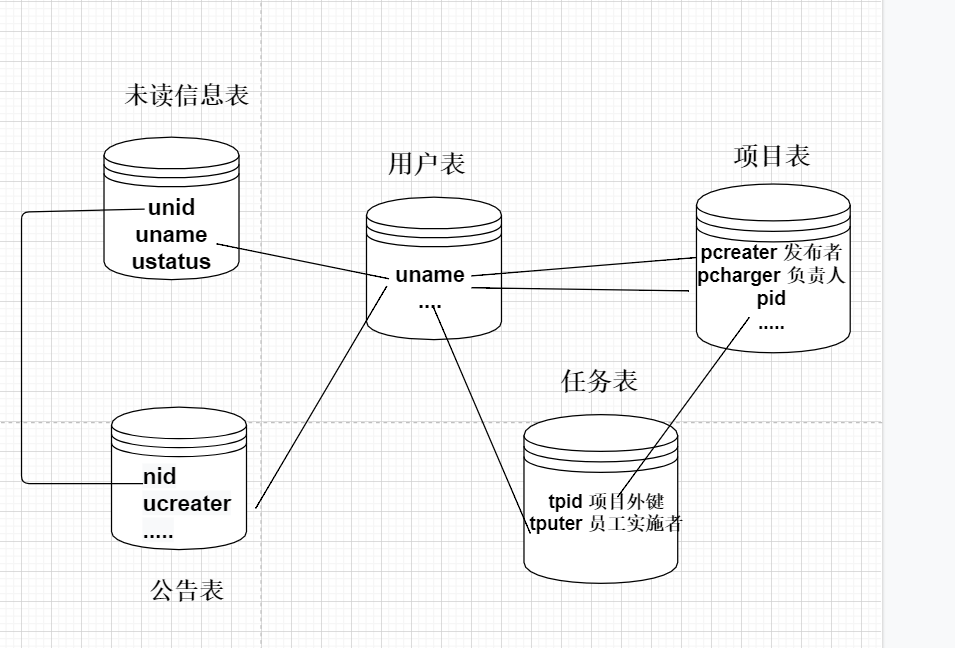
**Task表（任务）：**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | KEY | 默认值/描述 |
| Tid | int(10) | PRIMARY KEY | 主键/ id |
| Tname | varchar(50) | Not null | 任务名称 |
| Tcreater | varchar(50) | Not null | 创建者 |
| Tputer | varchar(20) | Not null | 实施者 |
| Tendtime | varchar(50) | Not null | 截止日期 |
| Tworktime | Int(10) | Not null | 预计工时 |
| Tstatus | int(1) | Not null | 任务状态 |
| Tpid | Int(10) | Not null | 外键/项目外键 |
| Tcontent | text | Not null | 任务内容 |
| Ttype | varchar(10) | Not null | 任务类型 |
| Tpri | varchar(10) | Not null | 任务优先级 |
| Treason | varchar(20) | Not null | 发布任务取消原因 |
| Tquit | Varchar(20) | Not null | 任务取消者 |
| tsum | text | Not null | 任务总结 |

**Unread表（未读消息表）**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | KEY | 默认值/描述 |
| uid | int(10) | PRIMARY KEY | 主键/ id |
| unid | int(10) | Not null | 外键/公告id |
| uname | varchar(50) | Not null | 外键/阅读者姓名 |
| ustatus | int(2) | Not null | 阅读状态 |

**表与表之间关联：**



**未读消息表需要公告的id作为外键（公告索引）**

**需要用户的姓名作为外键（需要通知的人）**

**公告表需要用户的姓名作为外键（发布人）**

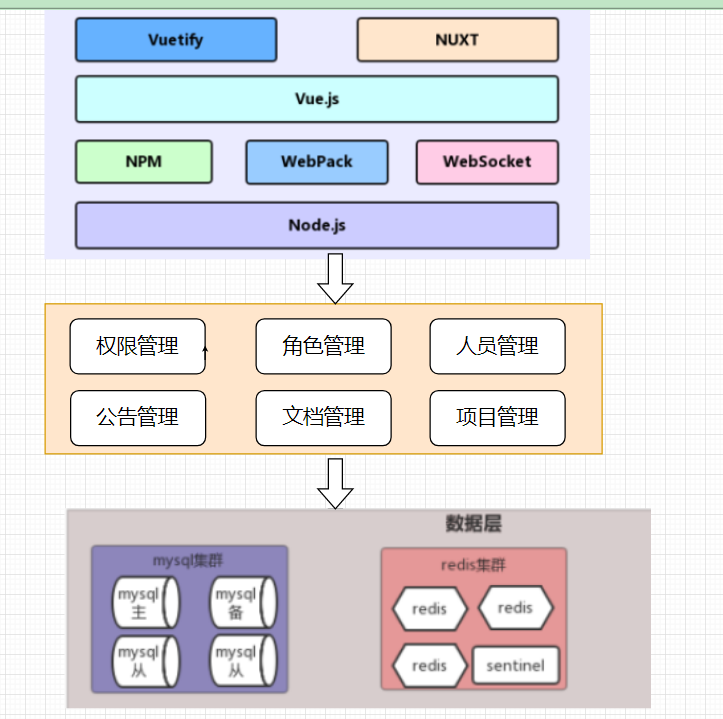
**项目表需要获取可以用户表姓名作为外键（项目负责人、实施者）**

**任务表需要获取用户的姓名作为外键人（执行项目任务的员工）**

**需要项目表的id（归属于哪个项目）**

**第三章 系统分析**

**1、架构图展示：**



**架构图解读：**

**（1）权限管理：**

|  |
| --- |
| 无论是员工经理还是管理员，每个人的极别不同，为了方便开发 我们采用RBAC设计方案对每个用户的权限进行严格划分其在登录时，前端页面接受到后端权限相关信息，利用Vue特性 路由跳转给予用户可点击的页面，在访问与权限不符的页面，会出现空白。  当管理员在权限里关闭某个角色权限，那么相关角色人员在登录后自动关闭该权限相关的内容，进一步完善了公司的保密机制。 |

**（2）角色管理：**

|  |
| --- |
| 系统开发的所有功能将根据角色分配，可设定某个角色具有一些功能，与公司职位相吻合。 |

**（3）人员管理：**

|  |
| --- |
| 录入新员工信息，对员工信息进行修改，删除员工信息，查询某位员工信息。 |

**（4）公告管理：**

|  |
| --- |
| 上级发布公告，员工查看公告，设置公告阅读权限，修改公告内容，删除公告。  普通员工：查看极别相符的公告。  如果出现严重性错误，需要当前负责人员立马解决，可发送紧急消息，我们将以短信的形式发送给用户，必要情况下，我们会实现电话告警。 |

**（5）文档管理：**

|  |
| --- |
| 文档包括：员工任务提交的文档，项目分配人员名单，办公事宜详细内容等等。 |

**（6）项目管理：**

|  |
| --- |
| 管理员查看和发布新项目，交给项目负责人（项目经理）负责。  项目负责人分配项目任务给普通员工，更改项目内容，更改任务内容，取消任务。  普通员工提交项目任务，查看新任务。 |

**第四章 系统实现**

**首先查看后台所有功能实现接口（Controller层）：**

**（1）用户**

|  |
| --- |
| **//查询所有用户**  **@RequestMapping("/queryAlluser")**  **//删除用户**  **@RequestMapping("/deleteUser/{id}")**  **//添加用户**  **@RequestMapping("/insertUser")**  **@ResponseBody**  **//修改用户**  **@RequestMapping("/updateUser")**    **//查询用户**  **@RequestMapping("/selectUser")**  **//查找项目负责人**  **@RequestMapping("/projectUser")**  **//查找普通员工**  **@RequestMapping("/queryStaff")**  **//修改个人信息**  **@RequestMapping("/updateUserOne")**    **//修改密码**  **@RequestMapping("/updatePassword")** |

**（2）公告**

|  |
| --- |
| **//添加公告**  **@RequestMapping("/insertNotice")**    **//删除公告**  **@RequestMapping("/deleteNotice/{id}")**  **//修改公告内容**  **@RequestMapping("/updateNotice")**    **//修改公告权限**  **@RequestMapping("/updateNoticeRoot/{nroot}/{nid}")**  **//查询所有公告**  **@RequestMapping("/queryAllNotice")**  **//增加阅读量**  **@RequestMapping("/insertReadCount/{nid}")**  **//查询公告**  **@RequestMapping("/queryNoticeById/{id}")** |

**（3）用户**

|  |
| --- |
| **//查询所有用户**  **@RequestMapping("/queryAlluser")**    **//删除用户**  **@RequestMapping("/deleteUser/{id}")**  **//添加用户**  **@RequestMapping("/insertUser")**  **@ResponseBody**  **//修改用户**  **@RequestMapping("/updateUser")**  **public int updateUser( @RequestBody user user)**      **//查询用户**  **@RequestMapping("/selectUser")**  **public user selece(@RequestParam("uname") String uname,@RequestParam("upassword") String upassword)**    **//查找项目负责人**  **@RequestMapping("/projectUser")**  **//查找普通员工**  **@RequestMapping("/queryStaff")**  **public List<String> selectStaff()**  **//修改个人信息**  **@RequestMapping("/updateUserOne")**  **public user updateUser1(@RequestBody user user)**  **turn userSerive.updateUserInfo(user);**  **//修改密码**  **@RequestMapping("/updatePassword")**  **public user updatePS(@RequestBody user user)** |

**（4）提示信息**

|  |
| --- |
| **//修改消息状态**  **@RequestMapping("/updateUnreadStatus/{name}/{id}")**  **public void updateStatus(@PathVariable("name") String name,**  **@PathVariable("id") String id)**    **//查看消息**  **@RequestMapping("/selectStatus/{uname}/{unid}")** |

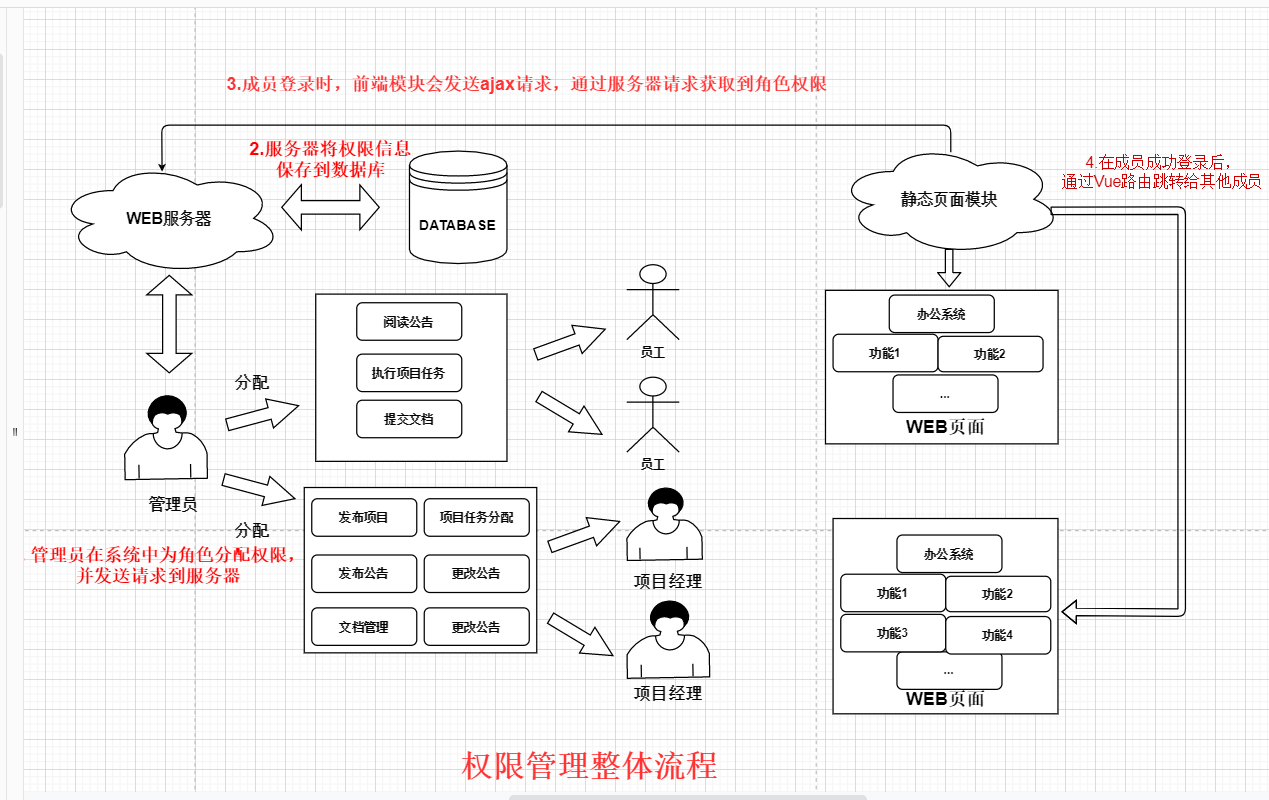
**（5）项目**

|  |
| --- |
| **//查询所有项目**  **@RequestMapping("/queryAllproject")**  **public systemResult queryAll ()**  **//增加项目**  **@RequestMapping("/insertProject")**  **public systemResult insertProject(@RequestBody project project)**    **//修改项目**  **@RequestMapping("/updateProject")**  **public systemResult updateProject(@RequestBody project project)**    **//删除项目**  **@RequestMapping("/deleteProject/{id}")**  **public systemResult deleteProject(@PathVariable("id") String id)**    **//查看项目及任务**  **@RequestMapping("/queryProjectById/{id}")**  **public systemResult queryOne(@PathVariable("id") String id)** |

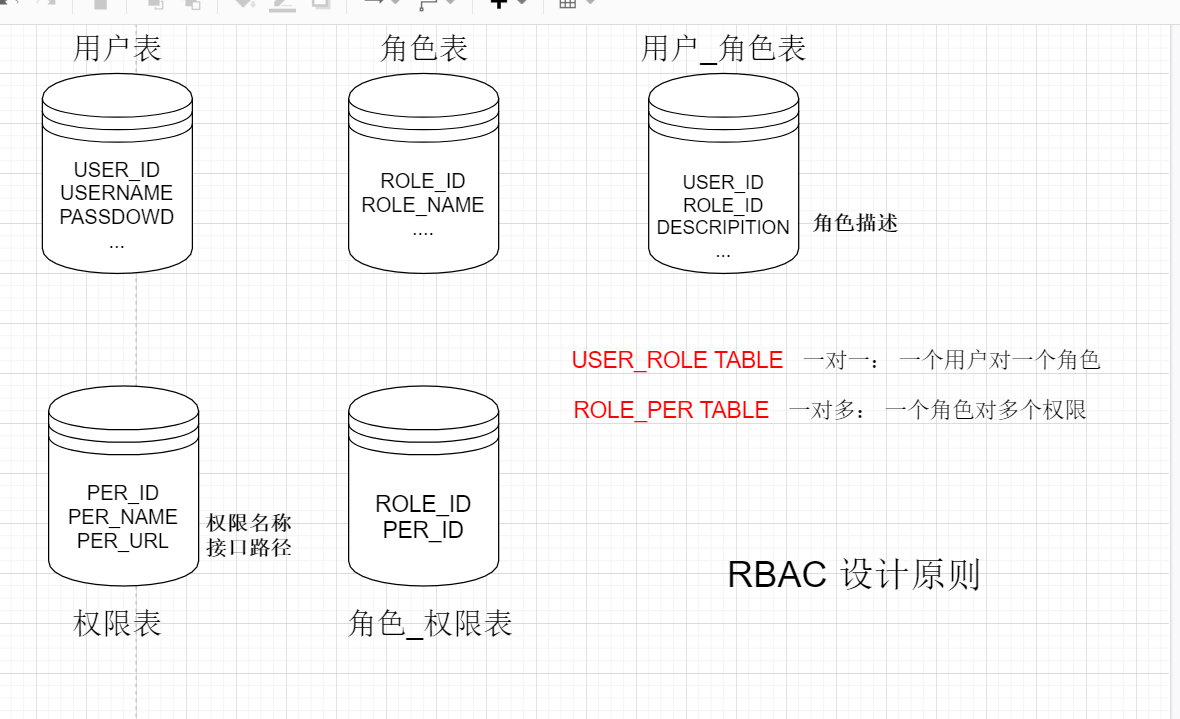
**（6）任务**

|  |
| --- |
| **//新增任务**  **@RequestMapping("/insertTask")**  **public systemResult insertTask(@RequestBody task task)**  **//查询该项目中的所有任务**  **@RequestMapping("selectTaskByPid/{id}")**  **public systemResult selectByID(@PathVariable("id") String id)**    **//修改任务状态**  **@RequestMapping("/UpdateTaskStatus/{status}/{id}")**  **public systemResult updateStatus(@PathVariable("id") String id,**  **@PathVariable("status") String status)**  **//修改任务内容**  **@RequestMapping("/UpdateTask")**  **public systemResult updateTask(@RequestBody task task)**    **//删除任务**  **@RequestMapping("/deleteTask/{id}")**  **public systemResult deleteTask(@PathVariable("id") String id)**    **//项目人取消任务**  **@RequestMapping("/cacelTask")**  **public systemResult cacelTask(@RequestBody task task)**  **//员工提交任务**  **@RequestMapping("/submitTask")**  **public systemResult submitTask(@RequestBody task task)**    **//查找自己负责的项目**  **@RequestMapping("/querymyjob/{uname}")**  **public systemResult querymyjob(@PathVariable("uname") String name)**    **//用户未读消息框**  **//judge为0 只返回未读公告个数 judge为1 +两个任务信息**  **@RequestMapping("/queryReadCount/{name}/{judge}")**  **public systemResult queryUnread(@PathVariable("name") String name,**  **@PathVariable("judge") String judge)** |

**1、权限管理：**



2、设计权限规则：



设计思想：

|  |
| --- |
| （1）俗话说没有规矩不成方圆。在一个办公系统中，公司各个部门，各个人员都有严格的制度，规则，我们必须遵守公司的制度，认真完成自己应该完成的任务，不给其他人添麻烦。  （2）正常公司一般在人员稳定，不浮动的情况下，每个人执行的业务不会发生变动。但是也有可能有特殊情况，如“负责A项目的张三请病假，项目经理也正好出差”，这时我们可以把部分权力划分给其他有能力，能承担该项目的人，从而将公司部分业务损失最小化。  （3）项目归属于安全系统网站，未防止“越级”，“删库跑路”等不可控因素，运用可靠的技术 维护好该网站也是必需的。  （4）初期产品需要良好的口碑才会有良好的后期维护与发展。 |

测试:

普通员工目前**只能查看公告**，我们**设置普通员工可以管理公告**（增加广告，修改公告，删除公告）

（1）先登录普通员工，查看其权限

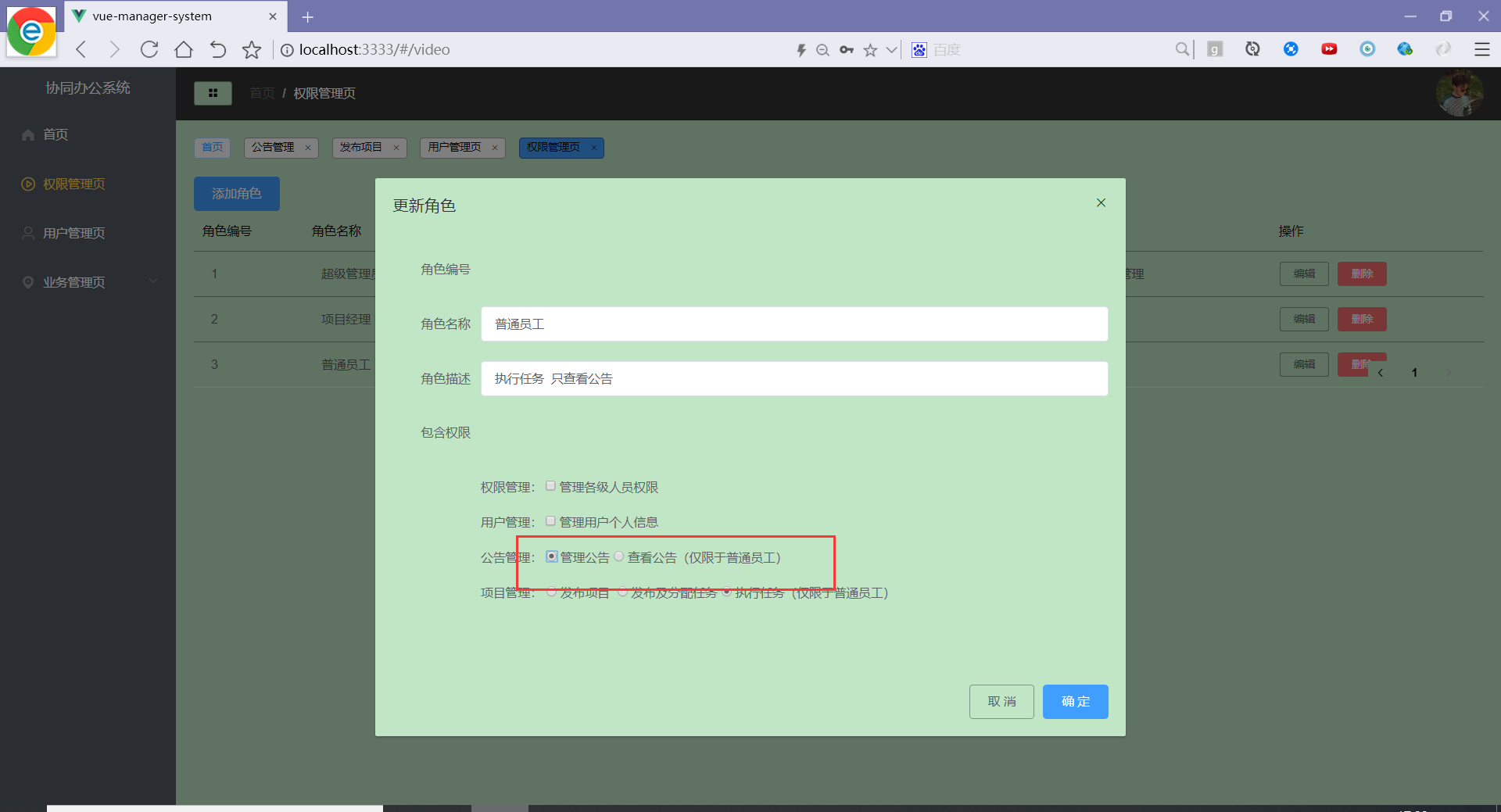




发现当前只能阅读公告

（2）登录超级管理员，修改普通员工权限





（3）再次登录普通员工，查看其权限





关键代码实现：

（1）前台根据所选内容发送ajax请求到后台服务器：

首先，提交表单内容的数组为checknames，利用vue特性**v-model**实时获取用户选定的值（用户勾选，则有值，用户取消，则值消失）

    <el-form-item>

      权限管理：<input type="checkbox" name="crouse" value="a" v-model="checknames.a" /><label>管理各级人员权限</label> <br />用户管理：<input

        type="checkbox"

        name="crouse"

        value="b"

        v-model="checknames.b"

      /><label>管理用户个人信息</label> <br />公告管理：<input type="radio" name="crouse" value="c" v-model="checknames.c" /><label>管理公告</label>

      <input type="radio" name="crouse" value="g" v-model="checknames.g" /><label>查看公告（仅限于普通员工）</label> <br />项目管理：<input

        type="radio"

        name="crouse1"

        value="d"

        v-model="checknames.d"

      /><label>发布项目</label> <input type="radio" name="crouse1" value="e" v-model="checknames.e" /><label>发布及分配任务</label>

      <input type="radio" name="crouse1" value="f" v-model="checknames.f" /><label>执行任务（仅限于普通员工）</label>

    </el-form-item>

将用户选定的值拿过来，发送请求：

//将ajax请求地址写为后台地址

axios1.defaults.baseURL = 'http://localhost:4450/'

//将传递的值类型修改为表单类型

axios1.defaults.headers.post['Content-Type'] = 'application/x-www-form-urlencoded'

//修改角色

async function funB(rid, rname, rdes, rinclude) {

  var id = await axios1

    .post(

      '/updatePermission',

      {

        rid: rid, //传递角色ID

        rname: rname, //角色名称

        rdes: rdes, //角色概述

        rinclude: rinclude //角色包含权限

      },

      { headers: { 'Content-Type': 'application/json' } } //设置头信息为json请求

    )

    .then(res => {

      return res.data

    })

  return id

}

（2）后台接口代码：

@RequestMapping("/updatePermission") //接受get或post请求

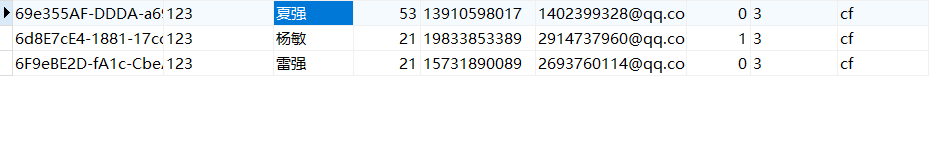
public systemResult updatePermission(@RequestBody permission permission)  
{  
 return permissionService.updatePermission(permission);  
}

@Override

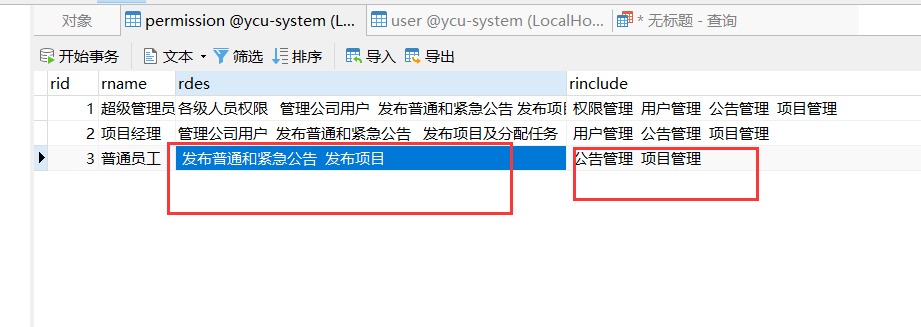
//两个操作 修改用户的权限和修改角色权限 必须一起完成 所以添加事务性操作（要么一起成功，要么一起失败）  
@Transactional  
public systemResult updatePermission(permission permission) {  
 //如果权限的角色id为空（无效请求，直接返回错误）  
 if(StringUtils.*isBlank*((permission.getRid()+"")) || permission.getRid()==0)  
 {  
 return new systemResult(systemStatus.*SELECT\_BY\_ID\_FAIL*,null);  
 }  
 String c=permission.getRinclude();  
 StringBuilder builder=new StringBuilder();  
 StringBuilder builder1=new StringBuilder();  
 builder1.delete(0, builder1.length());  
 //查找该等级下当前用户的人数  
 List<user> users= userService.selectByRoot(permission.getRid()+"");  
 //对该等级的成员进行修改等级  
 for(user user: users)  
 {  
 userService.updateRootByName(c,user.getUname());  
 }  
 if(c.contains("a"))  
 {  
 builder.append(" 权限管理 ");  
 builder1.append(" 各级人员权限 ");  
 }  
 if(c.contains("b"))  
 {  
 builder.append(" 用户管理 ");  
 builder1.append(" 管理公司用户 ");  
 }  
 if(c.contains("c") && (!c.contains("g")))  
 {  
 builder.append(" 公告管理 ");  
 builder1.append(" 发布普通和紧急公告 ");  
 }  
  
 permission.setRdes(builder1.toString());  
 permission.setRinclude(builder.toString());  
 //修改权限，无论成功或失败都返回状态信息  
 *anInt*= permissionMapper.updatePermission(permission);  
 return *anInt*==1?(new systemResult(systemStatus.*UPDATE\_SUCCESS*,null)):( new systemResult(systemStatus.*FAIL*,null));  
  
}

（3）查看数据库：

**用户表：**

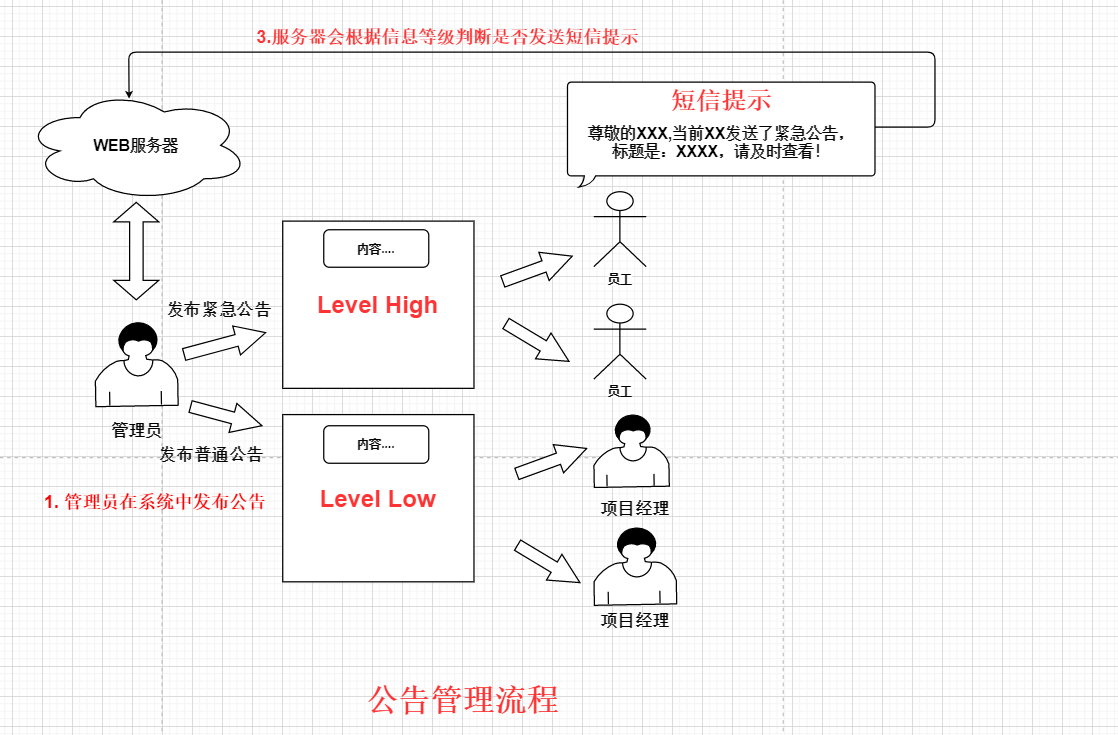


**权限表：**



**2、公司公告管理：**

**流程图展示：**

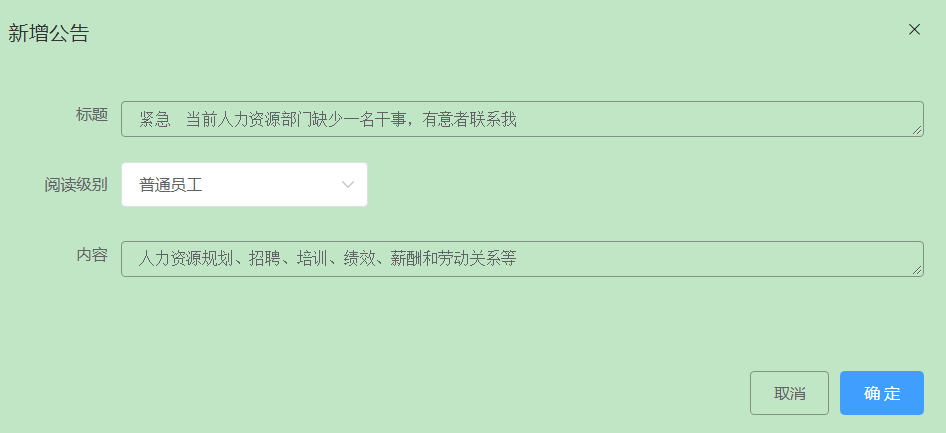


**（1）设计思想：**

|  |
| --- |
| **（1）方便公司通知开会，及其他紧急情况，会及时发送短信，将工作内容发送给该级别成员。**  **（2）正常工作中，会有一些项目或者执行任务的成员出现突发情况，比如 项目出现Bug需要该负责人解决，并不影响整个部门的运转，可以选择正常信息显示。但是，如果出现严重性错误，需要当前负责人员立马解决，可发送紧急消息，我们将以短信的形式发送给用户，必要情况下，我们会实现电话告警。**  **（3）可通过阅读次数来辨别是否应该浏览的全部人员进行查看，系统会根据发布的一定时间内告知通知人员谁存在未阅读问题。** |

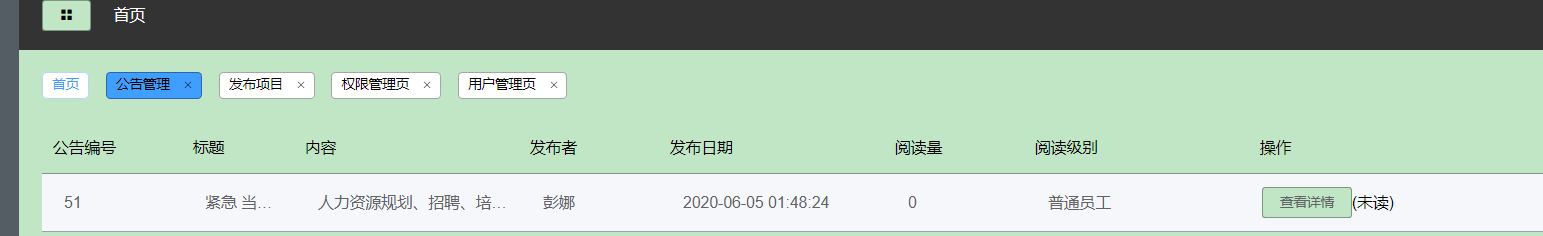
**(2)测试：**

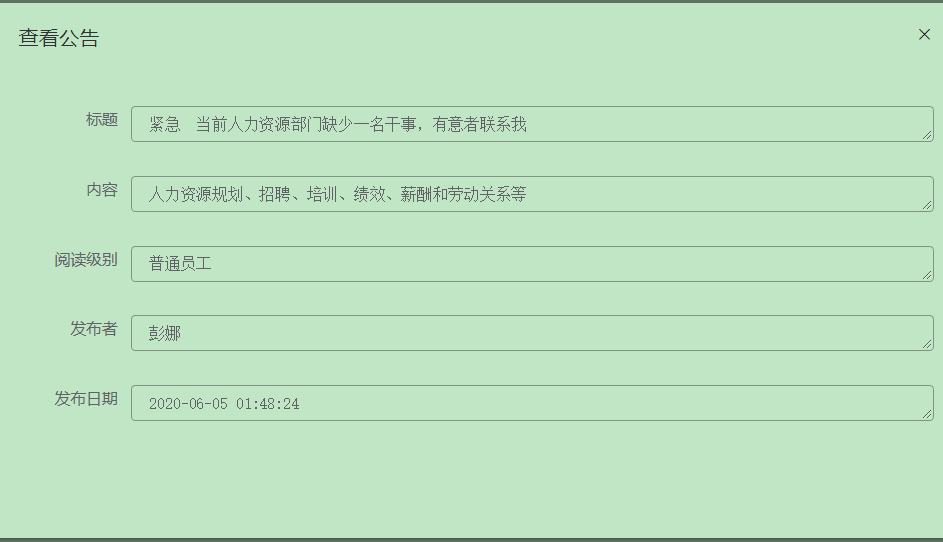
**发送紧急信息公告，会发送短信到用户手机。**



**我们先登录普通员工账号 进入到网站进行查看**



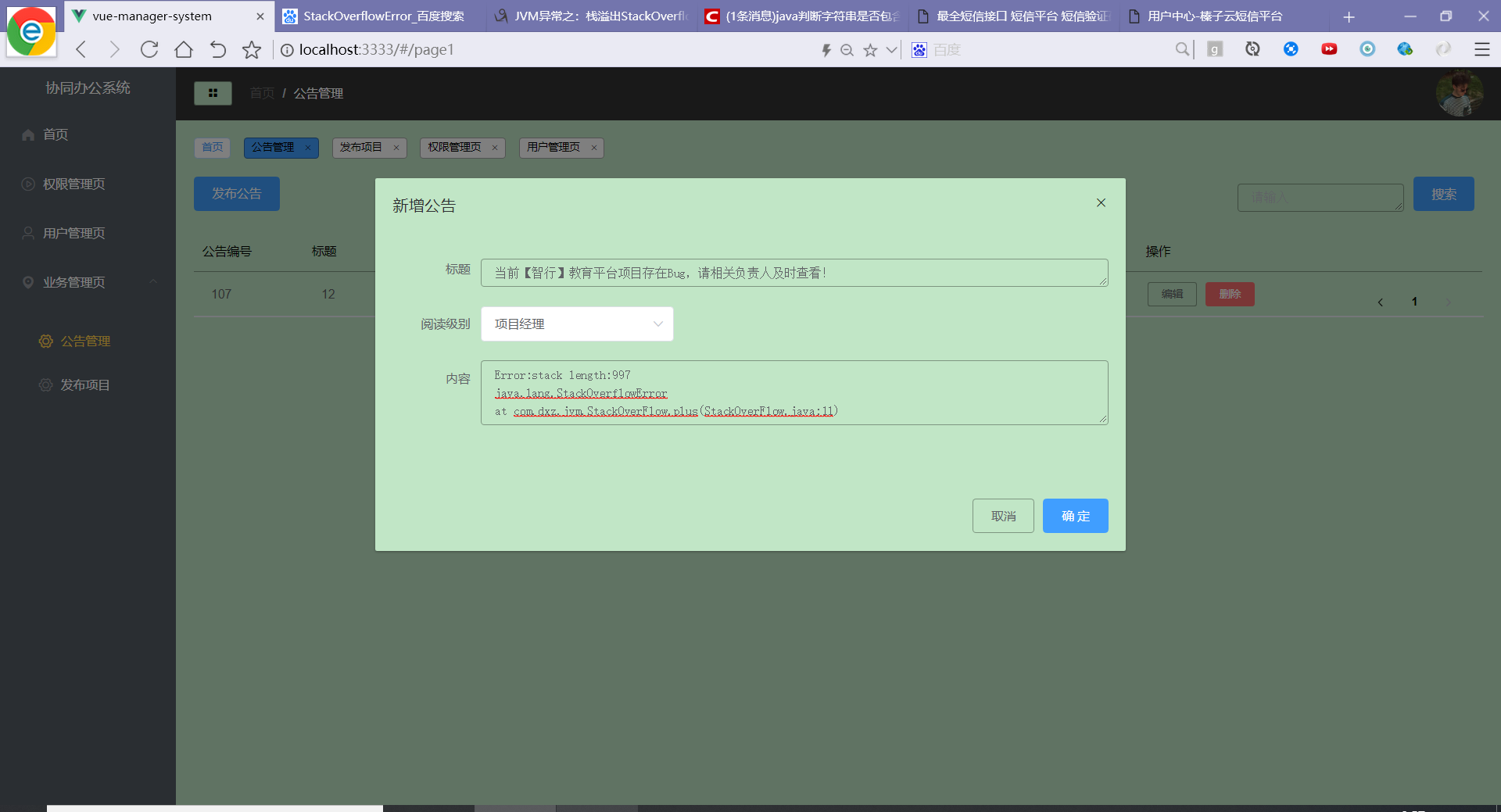


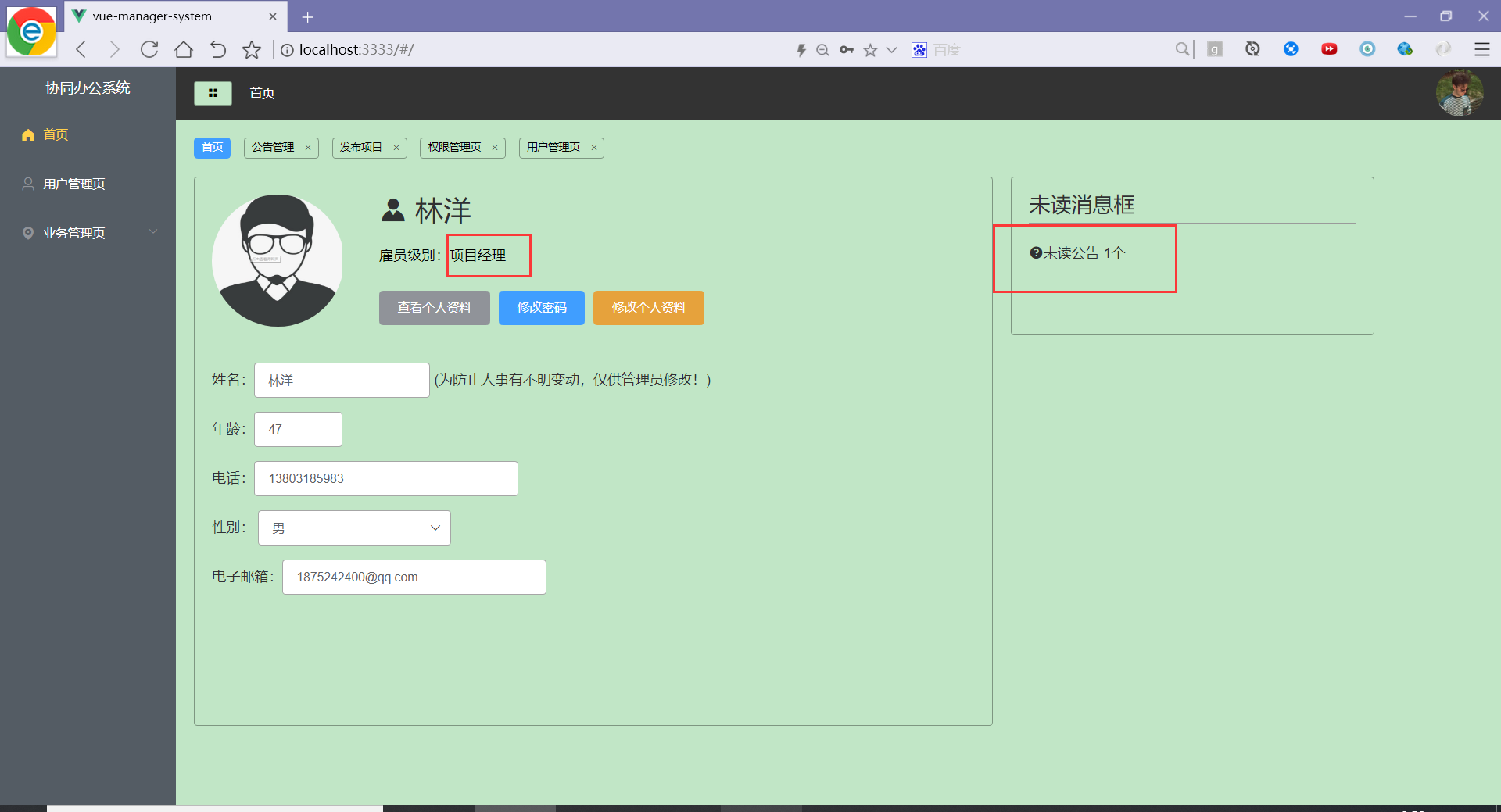


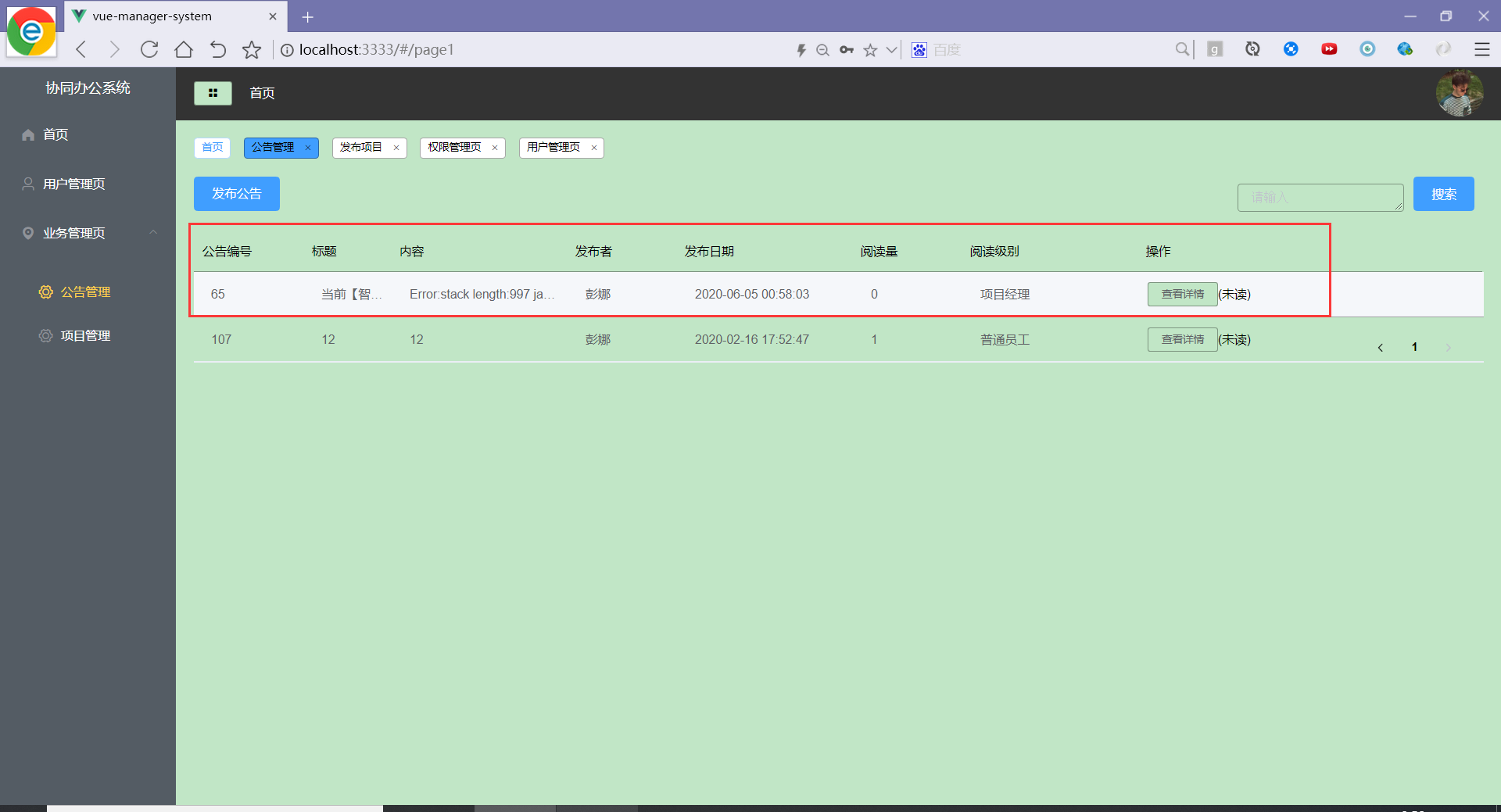
**查看自己是否收到短信：**

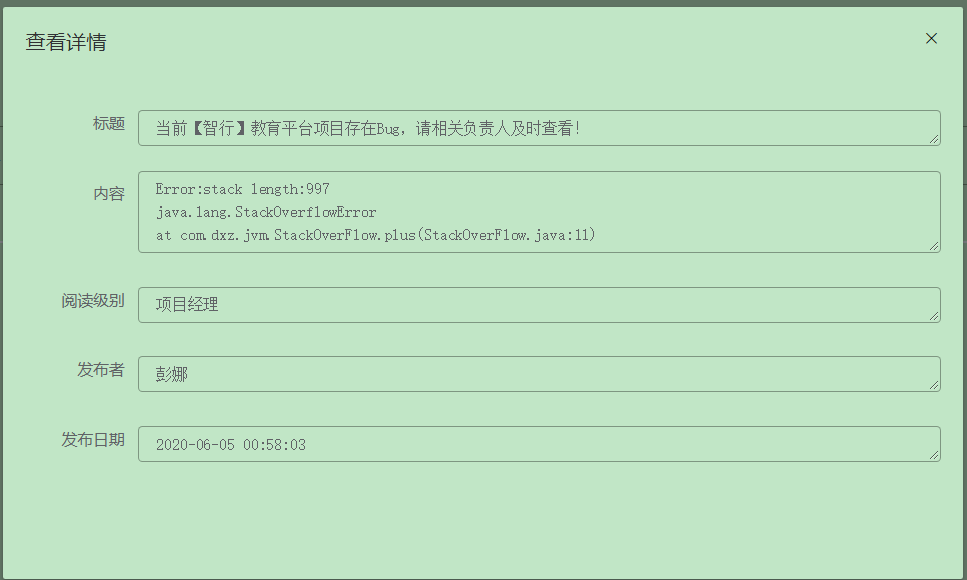


**1）管理员发布公告：权限 =》 项目经理极别及以上**









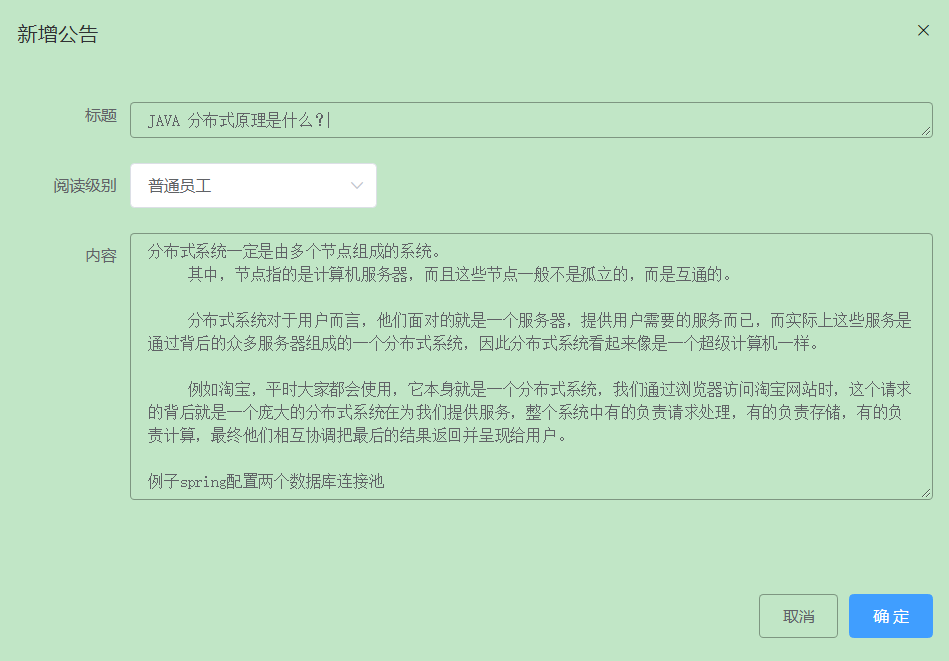
**我们登录普通员工账号，查看是否有该公告。**





**并没有刚才项目经理级别公告，测试成功**

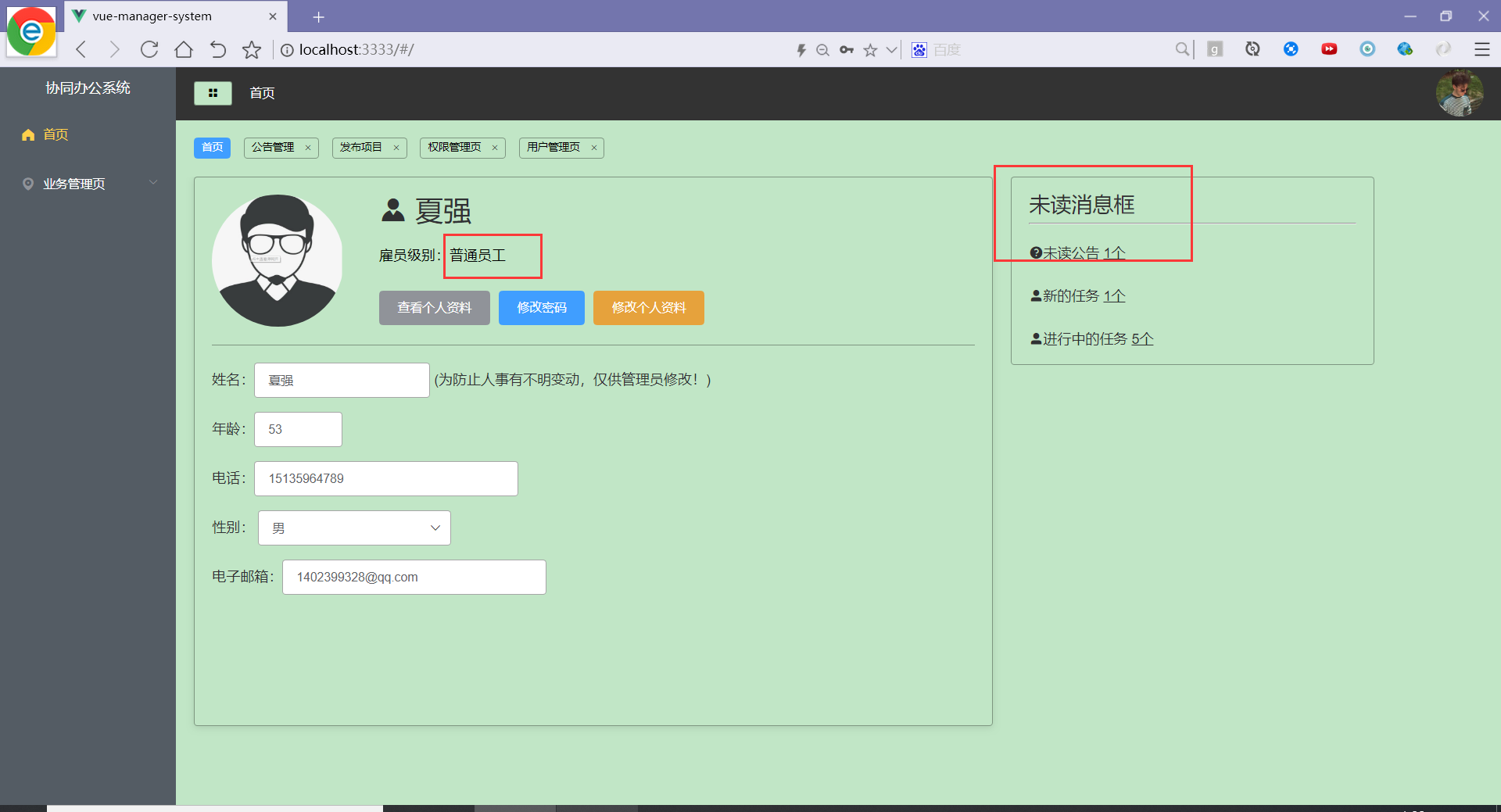
**2）项目经理发布公告： 权限=》 员工级别**





**项目经理可对自己发布的公告进行编辑和删除。**

**我们登录普通员工账号，查看**

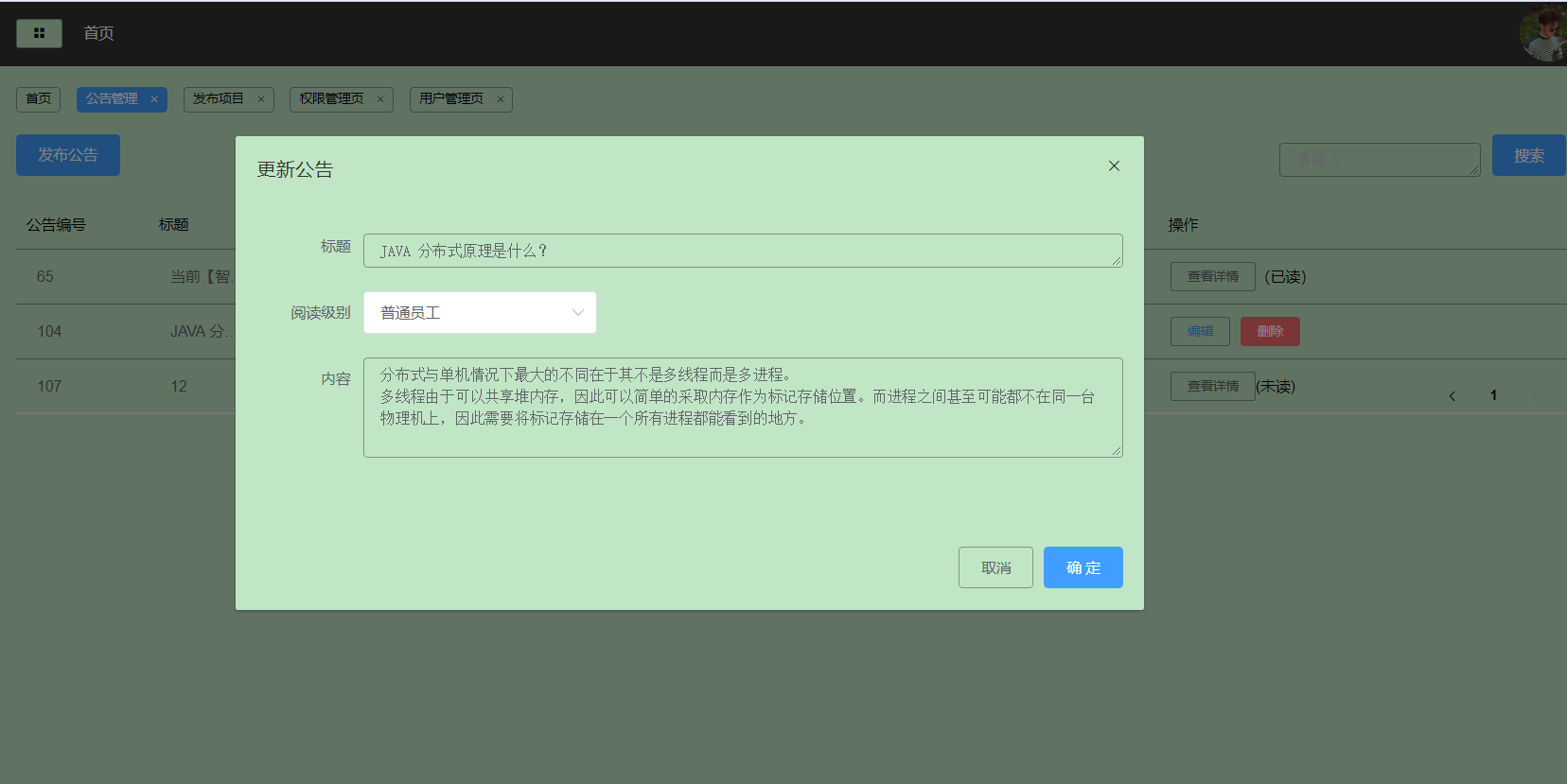




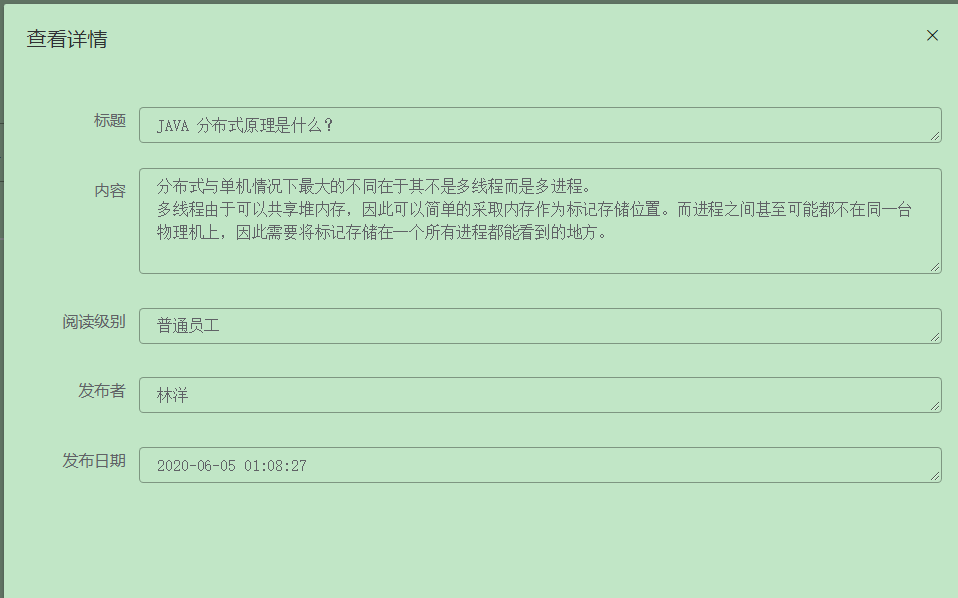


**每个成员只能访问一次该公告，可得到有效的阅读次数**

**3）更改公告，删除公告**



**登录普通员工账号查看：**



**（3）关键代码实现：**

**1）发布公告：**

//点击按钮，执行addUser方法，弹出表单等待用户输入内容

<el-button type="primary" @click="addUser">发布公告</el-button>

   addUser() {

      //创建一个数组存放用户输入的数据

      this.operateForm = {}

      this.operateType = 'add'

      //将输入的表单弹出来

      this.isShow = true

    }

if (this.operateType === 'add') {

        //将当前登录的用户当作发表者

        this.operateForm.ncreater = this.$session.get('rrr').data.uname

        this.operateForm.yourname = this.operateForm.ncreater

        //将存放数据的数组发送到Mock请求，等待下一步操作

        this.$http.post('/api/notice/add', this.operateForm).then(res => {

          console.log(res.data)

          this.isShow = false

          this.getList()

        })

      }

 addNotice: config => {

    const { ntitle, ncreater, ntime, nroot, ncontent, yourname } = JSON.parse(config.body)

    //获取当前时间

    const time = new Date(+new Date() + 8 \* 3600 \* 1000)

      .toISOString()

      .replace(/T/g, ' ')

      .replace(/\.[\d]{3}Z/, '')

    let nid = returnRand()

    console.log(JSON.parse(config.body), ntime)

    let root = exchangeRoot(nroot)

    //将当前展示的列表添加一行数据，无须刷新页面即可看到效果

    List.unshift({

      nid: nid,

      ntitle: ntitle,

      ncreater: ncreater,

      ntime: time,

      nreadcount: 0,

      nroot: root,

      ncontent: ncontent

    })

//发送请求到服务器，将其数据保存在数据库

    axios1

      .post(

        '/insertNotice',

        {

          nid: nid,

          ntitle: ntitle,

          ncreater: ncreater,

          ntime: time,

          nreadcount: 0,

          nroot: nroot,

          ncontent: ncontent,

          yourname: yourname

        },

        { headers: { 'Content-Type': 'application/json' } }

      )

      .then(res => {

        console.log(res)

      })

return {

//返回状态码及信息

      code: 20000,

      data: {

        message: '添加成功'

      }

    }

  },

**后台代码：**

@RequestMapping("/insertNotice")

public systemResult insertNotice(@RequestBody notice notice)  
{  
return noticeService.insertNotice(notice);  
}

@Override

@Transactional  
public systemResult insertNotice(notice notice) {  
 //添加公告到数据库  
 *anInt*=noticeMapper.insertNotice(notice);  
 //获取当前权限的所有成员  
 List<user> users=userMapper.selectByRoot(notice.getNroot()+"");  
 unread unread=new unread();  
 //遍历每个当前权限的成员，进行添加未读信息提示，通知所有级别成员  
 for(user user:users)  
 {

//如果是发布者本人就不要增加未读消息提示了  
 if(!notice.getNcreater().equals(user.getUname())) {  
 unread.setUnid(notice.getNid());  
 unread.setUname(user.getUname());  
 unread.setUstatus(0);  
 unreadMapper.insertNotification(unread);  
 }

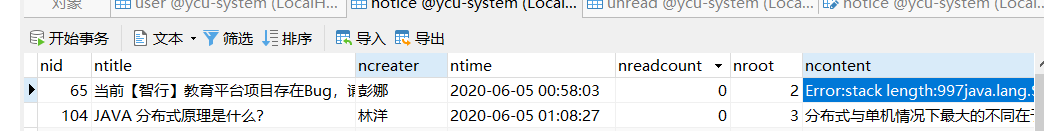
return *anInt*==1?(new

systemResult(systemStatus.*INSERT\_SUCCESS*,null)):( new systemResult(systemStatus.*FAIL*,null));  
}

**SQL语句：**

//发公告

@Insert("insert into notice (nid,ntitle,ncreater,ntime,nroot,ncontent) values(#{nid},#{ntitle},#{ncreater},#{" +  
 "ntime},#{nroot},#{ncontent})")  
int insertNotice(notice notice);



**2）阅读公告，阅读量+1 并在下一次阅读时不会添加阅读量**

**当用户点击查看详情按钮，执行handleSelect方法，最终当前公告阅读量会加1**

<el-button size="mini" @click="handleSelect(scope.row)" v-

if="name != scope.row.ncreater">查看详情</el-button>

    handleSelect(row) {

      this.$emit('select', row)

      //如果本人的权限与公告查看权限相符合

      if (exchangeRoot(row.nroot) == this.$session.get('rrr').data.uisRoot) {

        //发送请求，修改未读提示，改为已读

        this.$ajax.get('/updateUnreadStatus/' + this.$session.get('rrr').data.uname + '/' + row.nid)

      }

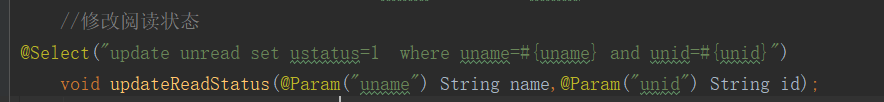
    }

**后台代码：**

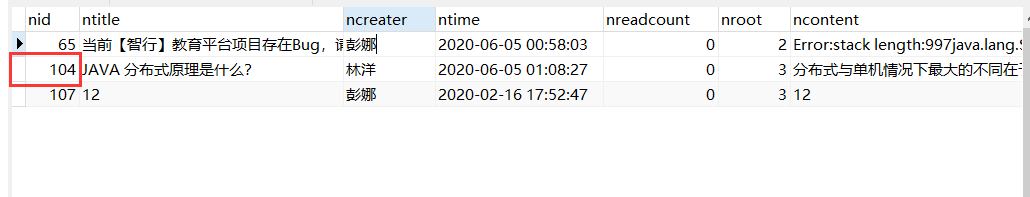
@RequestMapping("/updateUnreadStatus/{name}/{id}")  
public void updateStatus(@PathVariable("name") String name,  
 @PathVariable("id") String id)  
{  
 unreadService.updateReadStatus(name,id);  
}

@Override  
public void updateReadStatus(String name, String id) {  
 unreadMapper.updateReadStatus(name,id);  
}

**SQL代码：**



**数据库：**







**3)如果发送紧急公告，会发送紧急短信：**

@Override

@Transactional  
public systemResult insertNotice(notice notice) {  
 //添加公告到数据库  
 *anInt*=noticeMapper.insertNotice(notice);  
 //获取当前权限的所有成员  
 List<user> users=userMapper.selectByRoot(notice.getNroot()+"");  
 unread unread=new unread();

//创建发送短信对象  
 ZhenziSmsClient client = new ZhenziSmsClient("https://sms\_developer.zhenzikj.com", "103343", "4ae28a90-d1ed-4963-ba94-6d9c4cf9f1e2");  
 //遍历每个当前权限的成员，进行添加未读信息提示，通知所有级别成员  
 for(user user:users)  
 {  
 if(!notice.getNcreater().equals(user.getUname())) {  
 unread.setUnid(notice.getNid());  
 unread.setUname(user.getUname());  
 unread.setUstatus(0);  
 unreadMapper.insertNotification(unread);  
 }

//如果当前标题中有紧急两字出现  
 if(notice.getNtitle().indexOf("紧急") != -1 )  
 {  
 try {  
//编辑短信内容

String s= "尊敬的"+user.getUname()+"，当前有紧急公告，标题是:\" "+notice.getNtitle()+"\" 请及时查看！ 验证码 2500";

//发送  
 String result = client.send(user.getUbrith(),s,"dfee\_dfdw\_xdfd\_dfdfd");  
 System.*out*.println(result);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 return *anInt*==1?(new systemResult(systemStatus.*INSERT\_SUCCESS*,null)):( new systemResult(systemStatus.*FAIL*,null));  
}

**3、项目管理**

**（1）项目经理=》 发布项目 =》 发布任务=》 普通员工查看自己的任务**



**（1）前端实现关键代码：**

if (this.operateType === 'add') {

        //首先 项目创建者和项目负责人是登录的用户

        this.operateForm.pcreater = this.$session.get('rrr').data.uname

        this.operateForm.pcharger = this.operateForm.pcreater

        //this.operateForm为用户刚才输入的数据

        //然后发送Mock请求

        this.$http.post('/api/project/add', this.operateForm).then(res => {

          console.log(res.data)

          //发送请求完毕后，关闭表单框

          this.isShow = false

          //重新获取项目列表

          this.getList()

        })

      }

 addProject: config => {

    const { ptitle, pdes, pcreater, pcharger } = JSON.parse(config.body)

     //获取当前时间

    const time = new Date(+new Date() + 8 \* 3600 \* 1000)

      .toISOString()

      .replace(/T/g, ' ')

      .replace(/\.[\d]{3}Z/, '')

let pid = returnRand()

//将当前页面列表添加该数据

    List.unshift({

      pid: pid,

      ptitle: ptitle,

      pdes: pdes,

      pcreater: pcreater,

      pcharger: pcharger,

      ptime: time

    })

  //发送ajax请求到服务器

    axios1

      .post(

        '/insertProject',

        {

          pid: pid,

          ptitle: ptitle,

          pdes: pdes,

          pcreater: pcreater,

          pcharger: pcharger,

          ptime: time

        },

        { headers: { 'Content-Type': 'application/json' } }

      )

      .then(res => {

        console.log(res)

      })

    return {

      code: 20000,

      data: {

        message: '添加成功'

      }

    }

  }

**（2）后台实现关键代码：**

@RequestMapping("/insertProject")

public systemResult insertProject(@RequestBody project project)  
{  
 return projectService.insertProject(project);  
}

@Override

public systemResult insertProject(project project) {  
  
 *anInt*= projectMapper.insertProject(project);  
 return *anInt*==1?(new systemResult(systemStatus.*INSERT\_SUCCESS*,null)):( new systemResult(systemStatus.*FAIL*,null));  
}

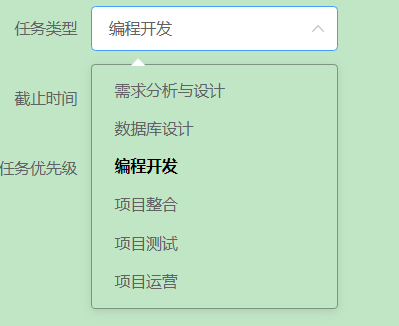
**SQL语句：**

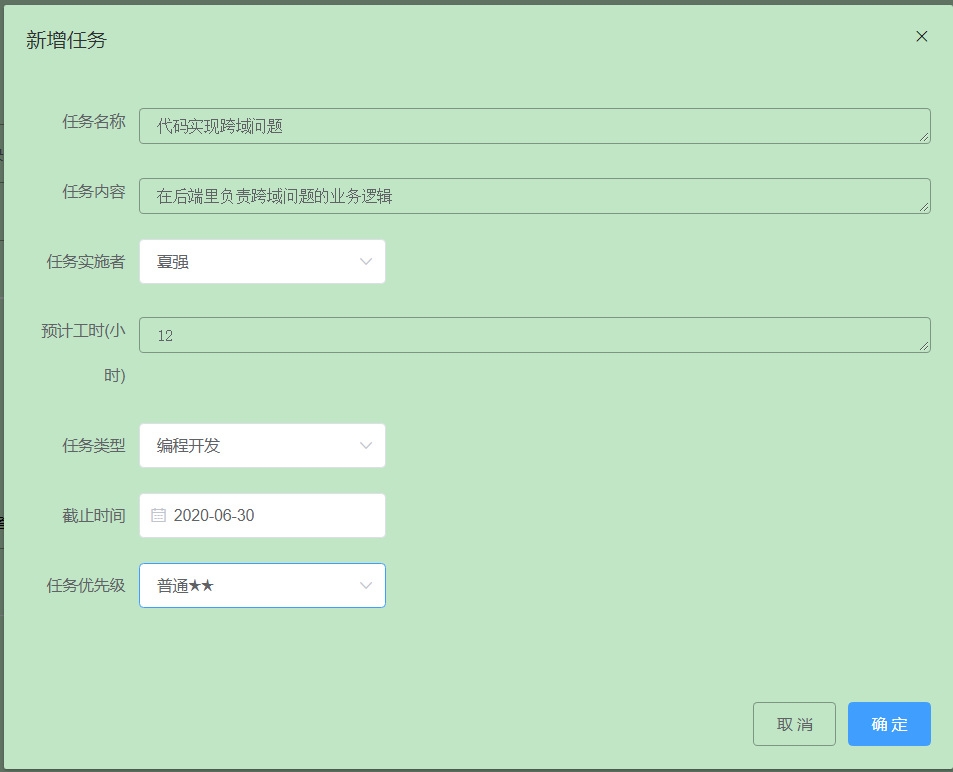
//创建项目  
@Insert("insert into project (pid,ptitle,pdes,pcreater,pcharger,ptime) values" +  
 "(#{pid},#{ptitle},#{pdes},#{pcreater},#{pcharger},#{ptime})")  
 int insertProject(project project);

**创建好项目后，我们分配一个任务给夏强：**

**选择普通员工：夏强**







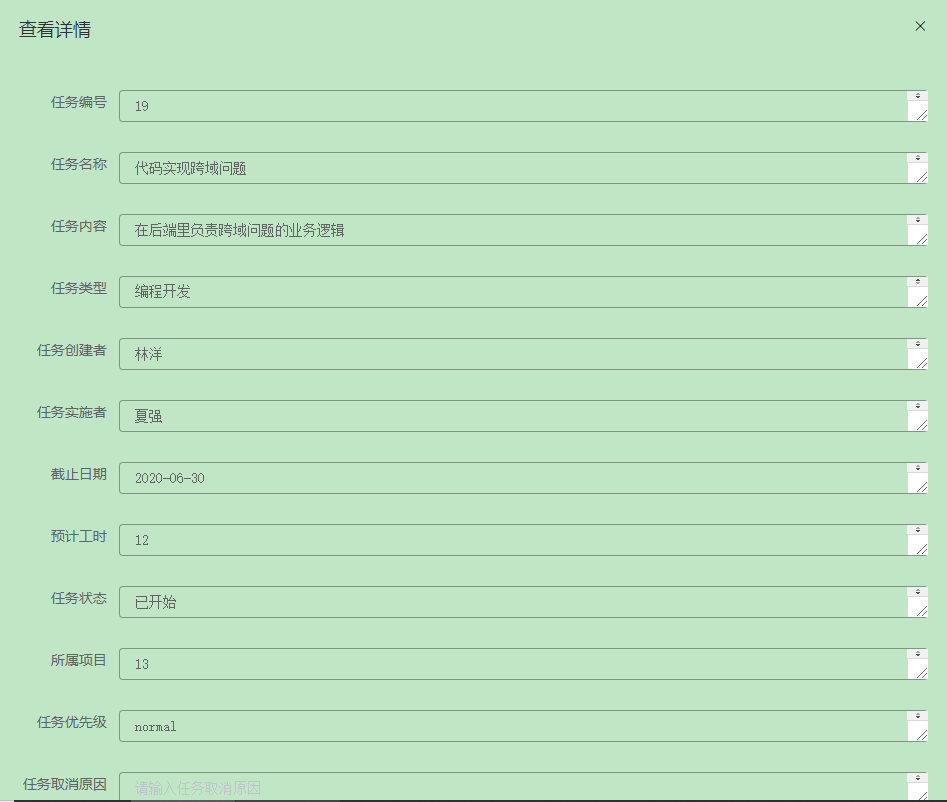
**我们先在项目经理任务界面查看：**



**我们进入夏强的账号界面查看：**







**前端关键代码：**

**当用户新建任务时，会执行以下方法：**

<el-button type="success" size="mini" @click="createTask(scope.row)" v-

if="scope.row.pcreater === username">新建任务</el-button>

   createTask(row) {

      //创建数组 接受新建任务的数据

      this.taskoperateForm = {}

      //监听器检测该类型是add 会执行以下方法

      this.taskoperateType = 'add'

      //弹窗展示新建任务表单

      this.taskisShow = true

      //将输入的数据传入进去

      this.operateForm = row

    }

**数据接受完毕后，发送ajax请求到后台服务器接口：**

   if (this.taskoperateType === 'add') {

        this.taskoperateForm.tcreater = this.$session.get('rrr').data.uname

        this.taskoperateForm.tstatus = 0

        this.taskoperateForm.tpid = this.operateForm.pid

        this.$ajax

          .post('/insertTask', {

            tname: this.taskoperateForm.tname,

            tcreater: this.taskoperateForm.tcreater,

            tputer: this.taskoperateForm.tputer,

            tendtime: this.taskoperateForm.tendtime,

            tworktime: this.taskoperateForm.tworktime,

            tstatus: this.taskoperateForm.tstatus,

            tpid: this.taskoperateForm.tpid,

            tcontent: this.taskoperateForm.tcontent,

            ttype: this.taskoperateForm.ttype,

            tpri: this.taskoperateForm.tpri,

            treason: this.taskoperateForm.treason,

            tquit: this.taskoperateForm.tquit

          })

          .then(res => {

            console.log('ok', res.data.code)

            this.taskisShow = false

            this.taskData = []

            return {

              code: 20000,

              message: '删除成功'

            }

          })

        this.$message({

          type: 'success',

          message: '添加任务成功!'

        })

      }

**后端关键代码：**

@RequestMapping("/insertTask")  
public systemResult insertTask(@RequestBody task task){  
 return taskService.insertTask(task);  
}

@Override

public systemResult insertTask(task task) {  
 *anInt*= taskMapper.insertTask(task);  
 return *anInt*==1?(new systemResult(systemStatus.*INSERT\_SUCCESS*,null)):( new systemResult(systemStatus.*FAIL*,null));  
}

**SQL语句：**

//创建任务

@Insert("insert into task (tname,tcreater,tputer,tendtime,tworktime,tstatus,tpid, tcontent,ttype,tpri, treason,tquit) values" +  
 "(#{tname},#{tcreater},#{tputer},#{tendtime},#{tworktime},#{tstatus},#{tpid},#{tcontent},#{ttype},#{tpri},#{treason},#{tquit})")  
int insertTask(task task);

**（2）项目经理取消普通员工的任务**





**再次进入到普通员工账号，查看是否还在进行任务？**



**前端关键代码：**

**当用户点击取消任务后，将operateType设置为del**

   <taskForm

        :formLabel=" operateType === 'del' ? caceloperateFormLabel : smalloperateFormLabel"

        :form="operateForm"

        ref="form"

      ></taskForm>

**//发送ajax请求，到后台进行取消任务**

if (this.operateType === 'del') {

        let id = this.operateForm.tid

        this.$ajax

          .post('/cacelTask/', {

            tid: id,

            treason: this.operateForm.treason,

            tstatus: 2,

            tquit: this.$session.get('rrr').data.uname

          })

          .then(res => {

            console.log(res.data)

            this.$message({

              type: 'success',

              message: '取消任务成功!'

            })

            this.taskData = ''

            this.isShow = false

          })

      }

**后端关键代码：**

@RequestMapping("/cacelTask")  
public systemResult cacelTask(@RequestBody task task)  
{  
 return taskService.cacelTask(task);  
}

@Override  
public systemResult cacelTask(task task) {  
 *anInt*= taskMapper.cacelTask(task);  
 return *anInt*==1?(new systemResult(systemStatus.*DELETE\_SUCCESS*,null)):( new systemResult(systemStatus.*FAIL*,null));  
}

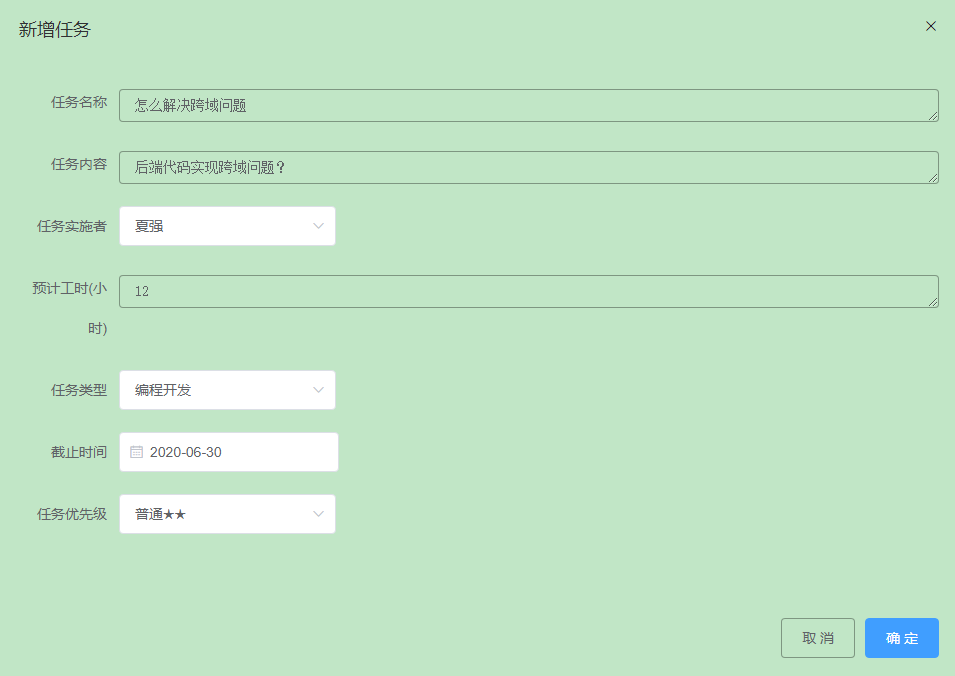
**SQL语句：**

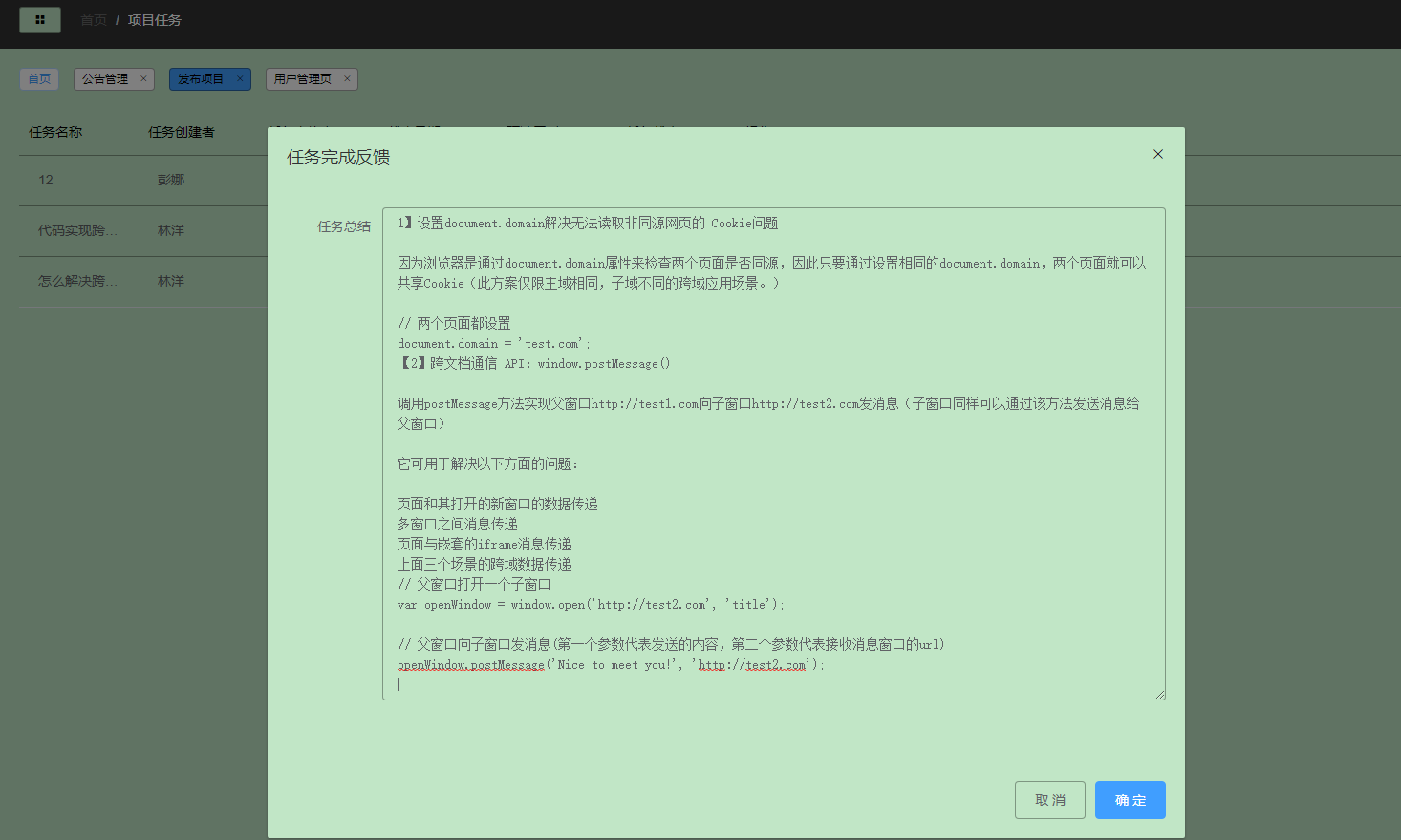
//取消任务

@Update("update task set tstatus=#{tstatus},treason=#{treason},tquit=#{tquit} where tid=#{tid}")  
int cacelTask(task task);

**（3）普通员工查看任务详情，提交任务**

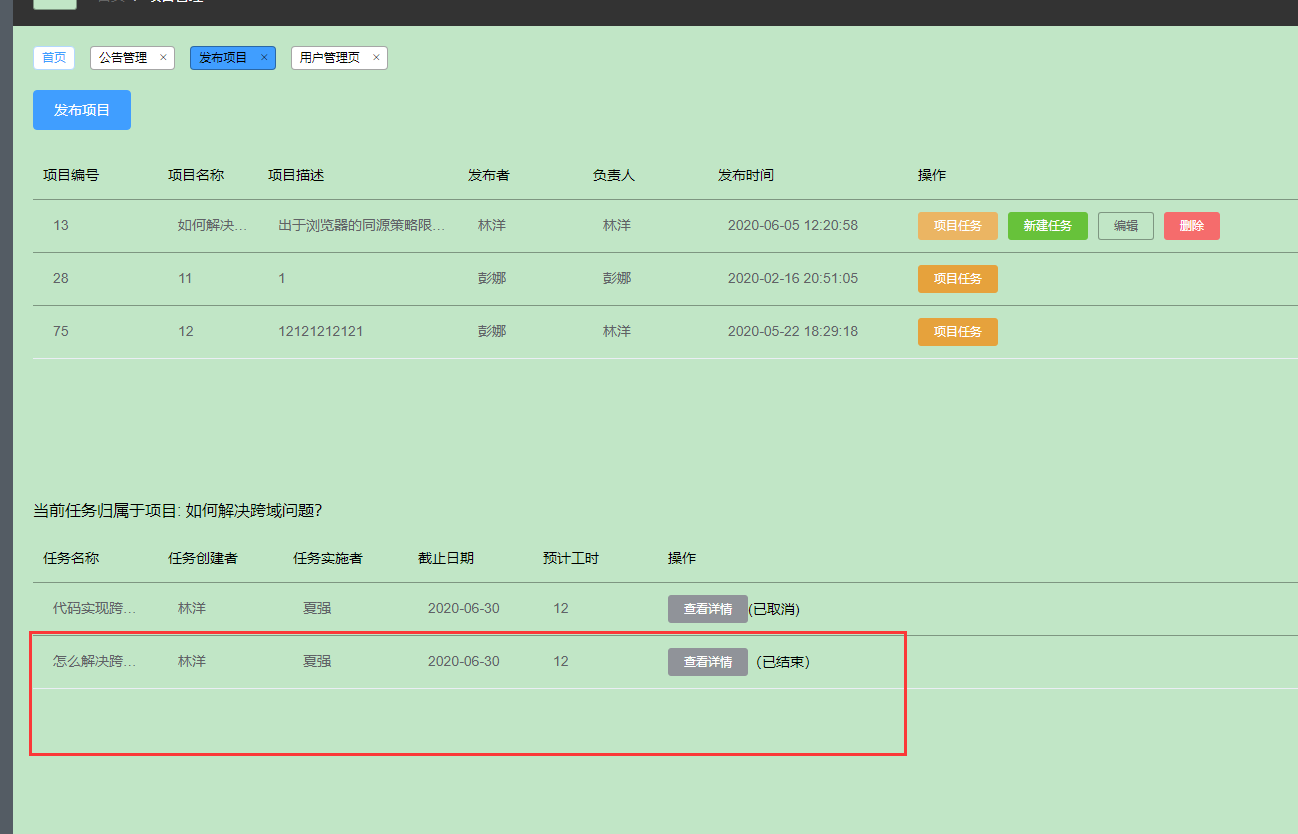
**我们重新添加任务，这次让员工提交成功查看**







**进入到项目经理，查看任务总结：**



前端关键代码：

当员工点击提示时，必须建立在实施者是自己，并且任务状态是已开始的情况下才能进行提交

 <el-

button size="mini" @click="handleEdit(scope.row)" v-if="scope.row.tputer === username && scope.row.tstatus === '已开始'">提交</el-button>

   handleEdit(row) {

      //将状态改为edit 使得监听器感知值变化

      this.operateType = 'edit'

      //将任务反馈表单展示给用户

      this.isShow = true

      this.operateForm = row

    }

整理好数据，发送请求到后台接口

  if (this.operateType === 'edit') {

        //提交总结

        this.$ajax

          .post('/submitTask', {

            tid: this.operateForm.tid,

            tstatus: '3',

            tsum: this.operateForm.tsum

          })

          .then(res => {

            console.log(res)

            window.location.reload()

            this.isShow = false

          })

      }

后端关键代码：

@RequestMapping("/submitTask")

public systemResult submitTask(@RequestBody task task)  
{  
 return taskService.submitTask(task);  
}

@Override  
public systemResult submitTask(task task) {  
 *anInt*= taskMapper.submitTask(task);  
 return *anInt*==1?(new systemResult(systemStatus.*DELETE\_SUCCESS*,null)):( new systemResult(systemStatus.*FAIL*,null));  
}

SQL语句：

@Update("update task set tstatus=#{tstatus},tsum=#{tsum}

where tid=#{tid}")  
int submitTask(task task);