核电科研项目设计说明书

目录

一、引言

本文档提供给开发技术人员使用，用于核电科研项目APP 的设计与开发。

1.1 编写目的

本文档的目的，旨在规范软件开发，推动项目有序正常的进行，使相关人员遵守统一的规范。节省制作相关文档的时间，降低系统实现的风险，加快项目实施进度，做到系统设计的规范性和全面性，以利于系统的设计、实现、测试、维护和版本升级。

1.2 项目范围

本文档用于软件设计阶段的概要设计。

软件概要设计的范围是：移动管理平台共三个模块：其中手机APP +Web端双端都有的两个模块，分别是人员培训管理模块、安全管理模块（包括区域检查记录登记录入、上传、检查结果消息推送、整改项查询等功能）。

单APP端拥有的移动对讲机模块（对讲、群里等功能）。下面为详细设计描述。

1.3 读者对象

系统设计人员、软件开发人员、客户方的系统设计人员和项目评审人员。

二、设计概述

2.1. 核电科研项目概括

本移动管理平台用于核电工程项目。

2.2. APP功能

1、人员在线培训功能

2、安全施工管理功能

3、移动对讲机功能

三、系统详细需求分析

3.1、人员培训管理模块：

培训种类分为三个种类：基础培训、技能培训、专项培训。

培训课件由Web端分配的人力资源管理权限（负责组织培训的主管）的人负责上传课件；

参与培训人员可以在线查看、学习，并在学习后参与考试功能；

新课件上传或者培训进度更新增添推送通知；

后台管理人员可查看参培人员学习记录列表。

3.2、安全管理模块

创建检查记录模板，并根据时间、分类、分版块对检查记录上传；需上传附件（图片）功能；

检查结果上传后推送给责任人，进行整改。

查询整改项信息，跟踪进度。

3.3、移动对讲机模块

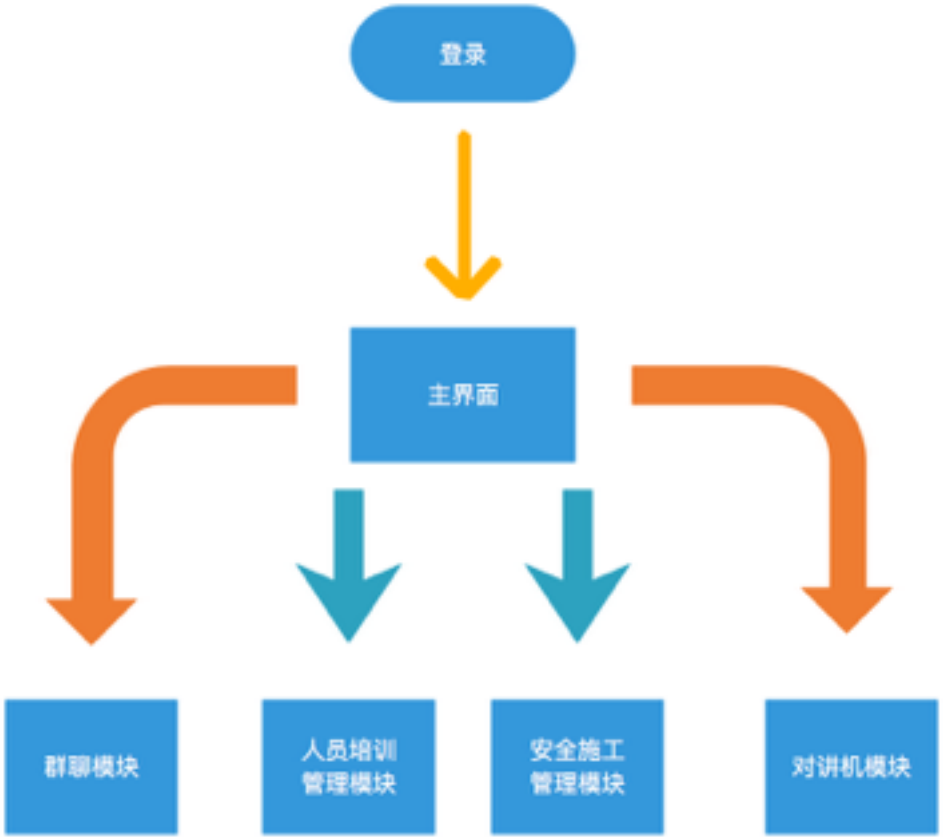
移动终端联系人实名显示，显示格式：部门+姓名；

群聊功能：可创建群聊、群聊发起人有权删除成员、群成员可以拉成员进群聊；

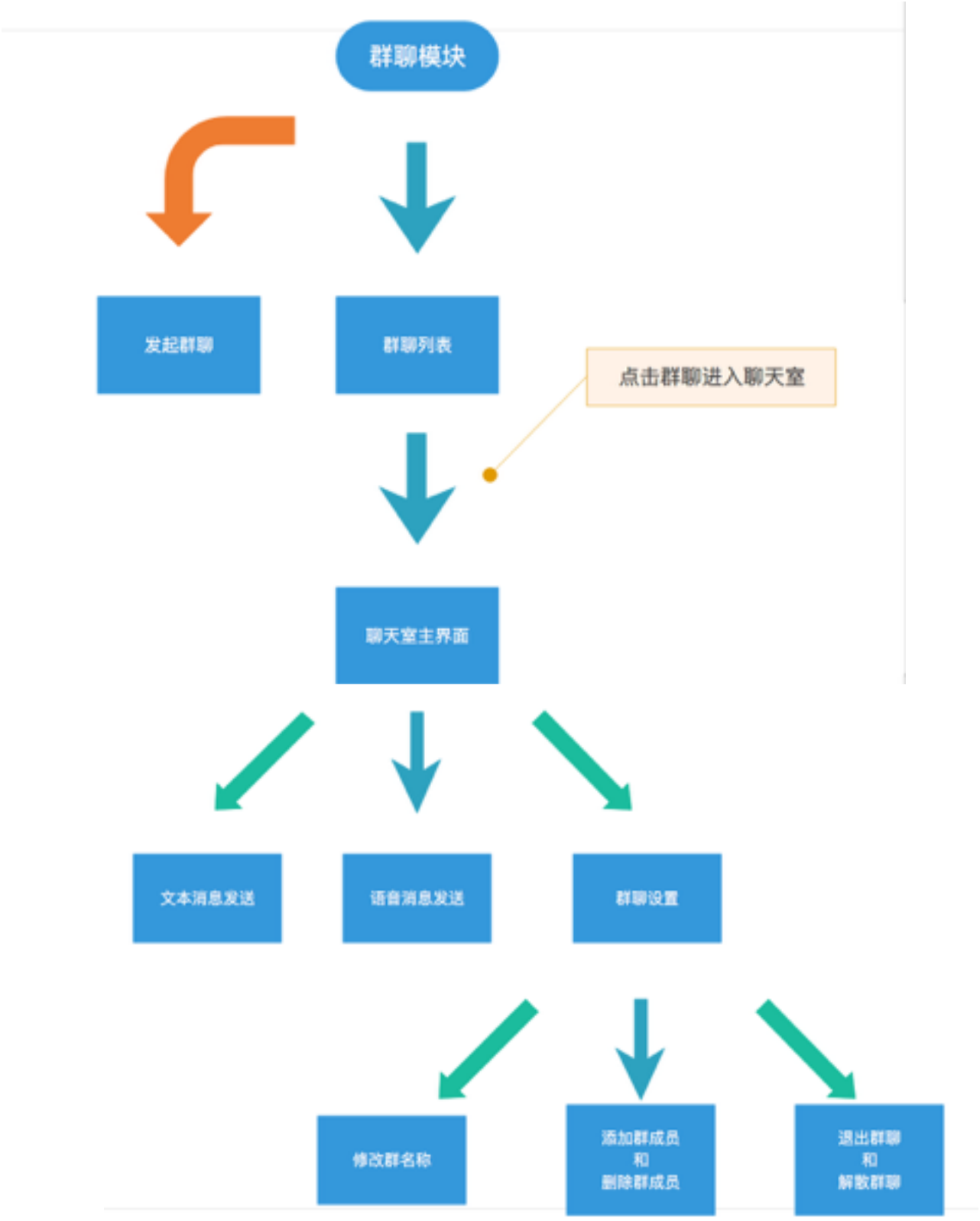
对讲功能：联系人可互相发送语音，联系人实时接收并选择接听。

四、 总体结构设计

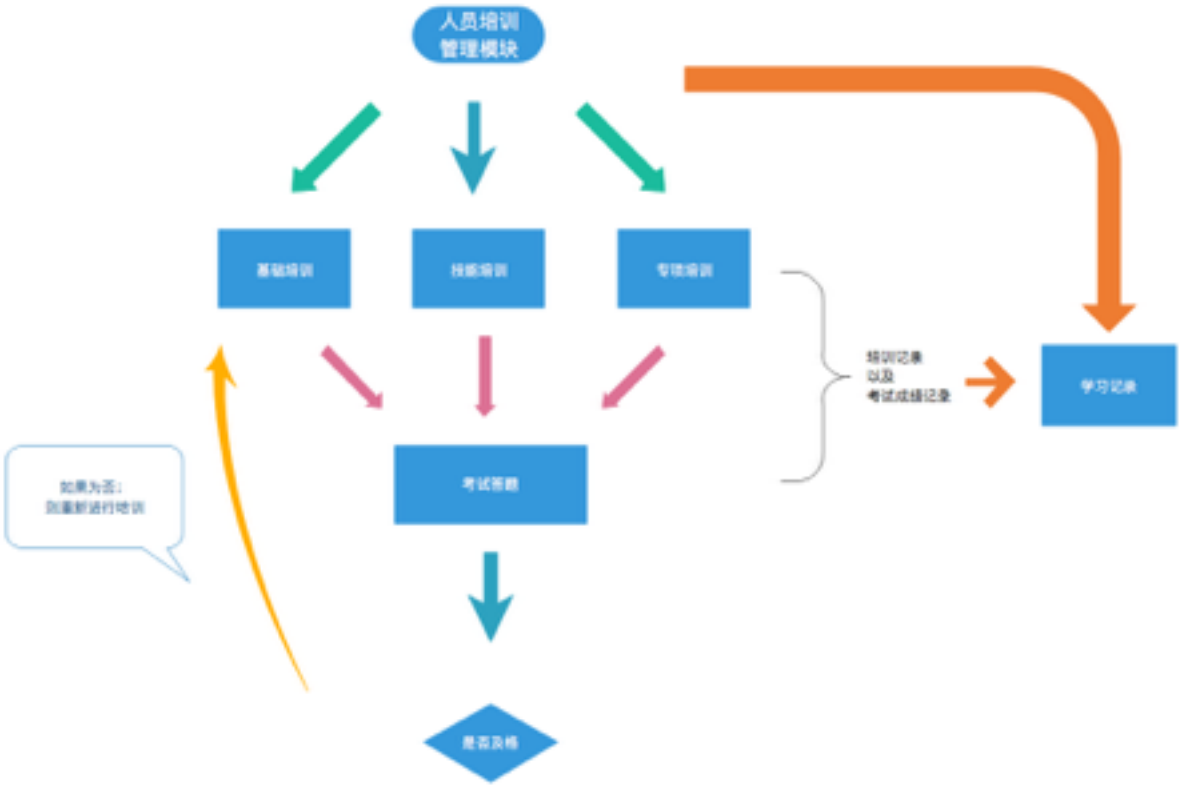
1）总体结构



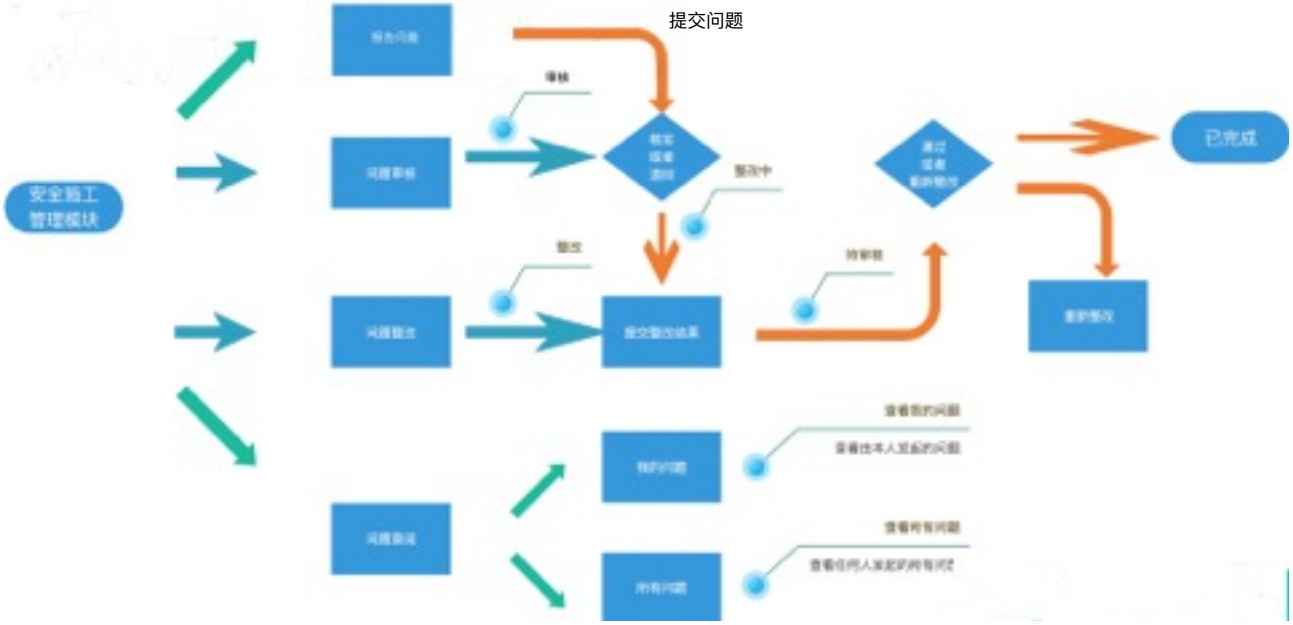
2）群聊模块


群聊模块
发起群聊
群聊列表
点击群聊进入聊天室
聊天室主界面
文本消息发送
语音消息发送
群聊设置
修改群名称
添加群成员
和
删除群成员
退出群聊
和
解散群聊

3）人员培训管理模块



4）安全施工管理模块



5）对讲机模块

五、 系统详细设计

5.1、主要页面及页面模块

1）、登录页面，无密码自定义登录

用户列表信息从后台配置，无登录设置

2)、群聊列表页面

聊天室列表页，获取群聊列表，并进行对应群聊操作。

3）、人员培训管理页面

培训模块按培训分类区分区域，分为基础培训、技能培训、专项培训三类功能区

4）、安全质量管理页面

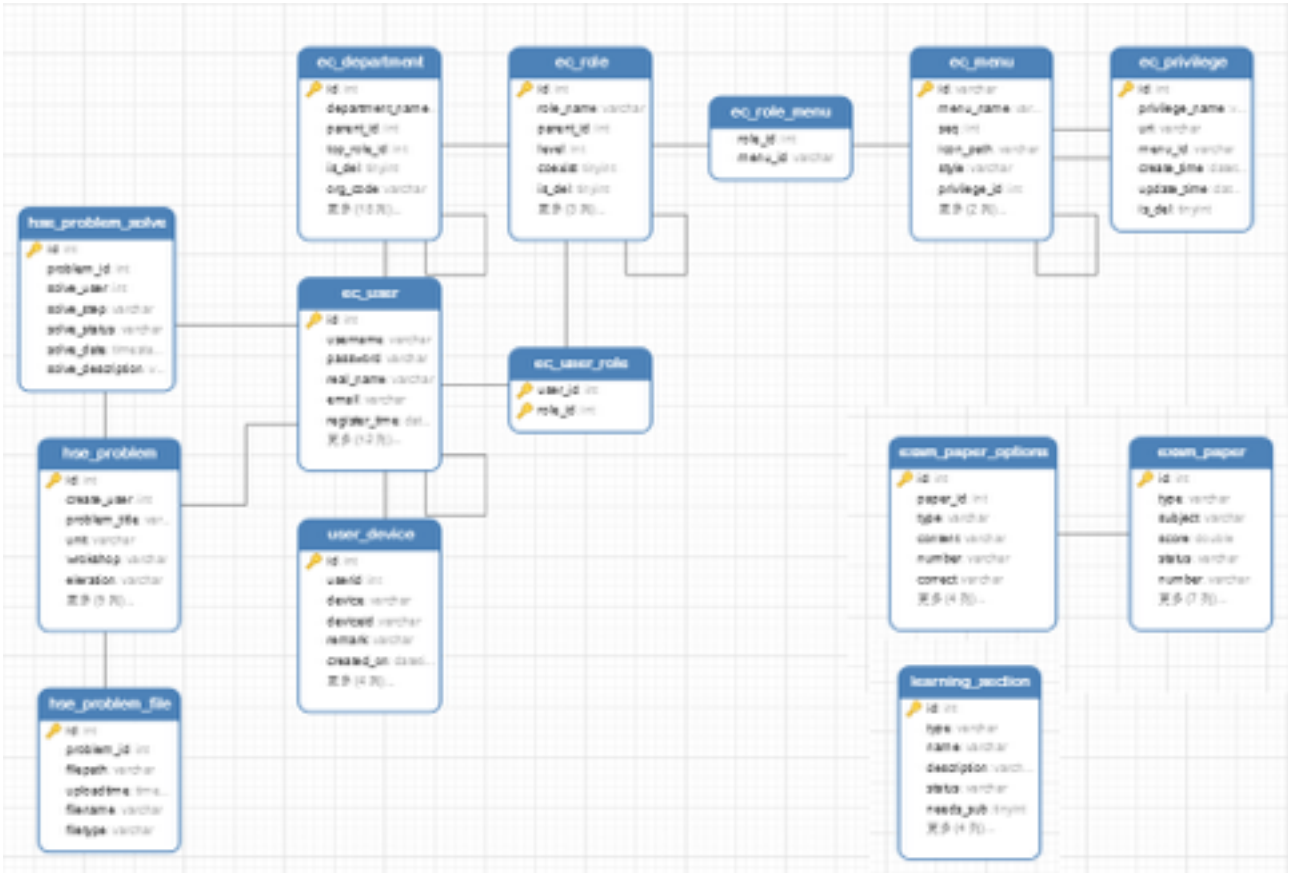
根据功能流程设计，区分为： 报告问题、问题审核、问题整改、问题查阅 四个功能区

5）、对讲机联系人页面

联系人列表，点击联系人进入对讲聊天室实时对讲

六、 数据库系统设计

6.1、数据库设计



7.2 数据库表单列表

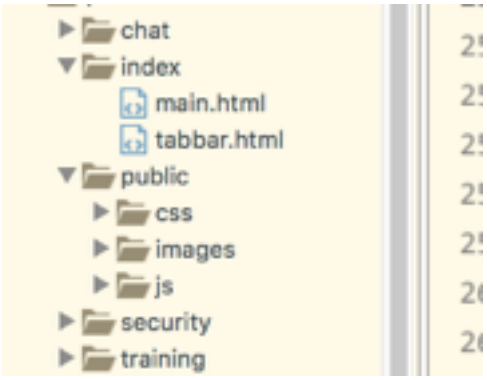
ec_department	部门表
ec_menu	web菜单表
ec_privilege	权限表
ec_role	岗位角色表
ec_role_menu	岗位角色菜单表
ec_user	人员表
ec_user_role	人员岗位关联表
exam_paper	试题表
exam_paper_options	试题选项表
exam_user_score	考试成绩表
hse_problem	安全问题表
hse_problem_file	安全问题附件表
hse_problem_solve	安全问题处理记录表
hse_unit_workshop	安全问题的机组和厂房记录表
learning	学习
learning_file	课程文件表
learning_fold	培训课程模块
learning_history	学习历史表
learning_section	培训课程分类
push_log	推送日志表
user_device	推送设备关联
variable_set	参数配置表

七、 信息编码设计

7.1、代码结构设计

1）App代码结构设计

代码结构按功能模块区分（如下图）：



主目录下分别为7个模块：

1、登录模块 ： 登录页

2、主页面模块：初始页、首页tabbar组件

tabbar页面 主要业务代码：

3、群聊以及对讲机模块

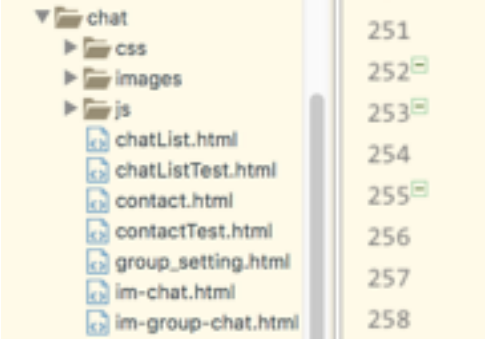
4、培训模块

5、安全施工模块

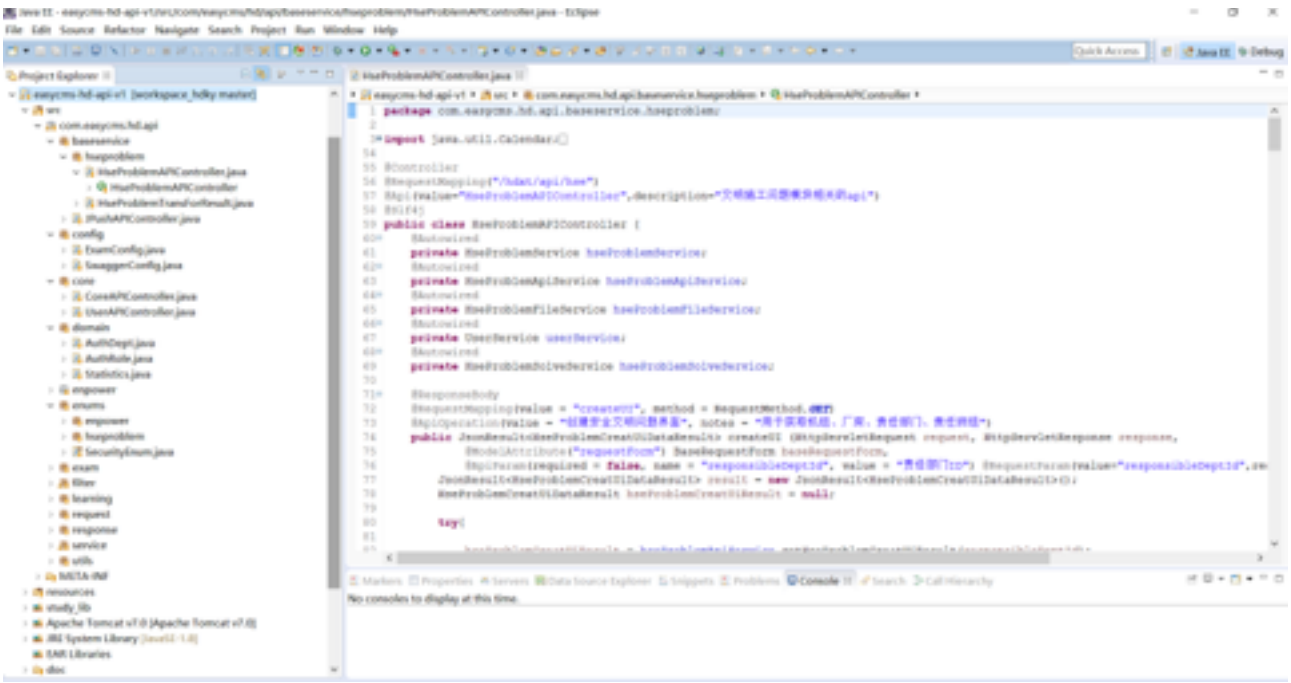
6、第三方公共类文件夹：Http 请求类、第三方选择框picker类、第三方全屏图片显示类、第三方页面刷新类、正则格式封装类

7、配置项文件夹：功能权限配置、版本信息、全局配置信息

第3、4、5 三个主功能模块 下都以css样式文件夹、images资源文件夹、js逻辑文件夹、html页面结构文件夹 四部分组成。详情如下图：



2）后台代码结构设计



后台代码结构按功能模块区分（如下图）：

主目录下分别为7个模块：

1、登录模块： 登录页

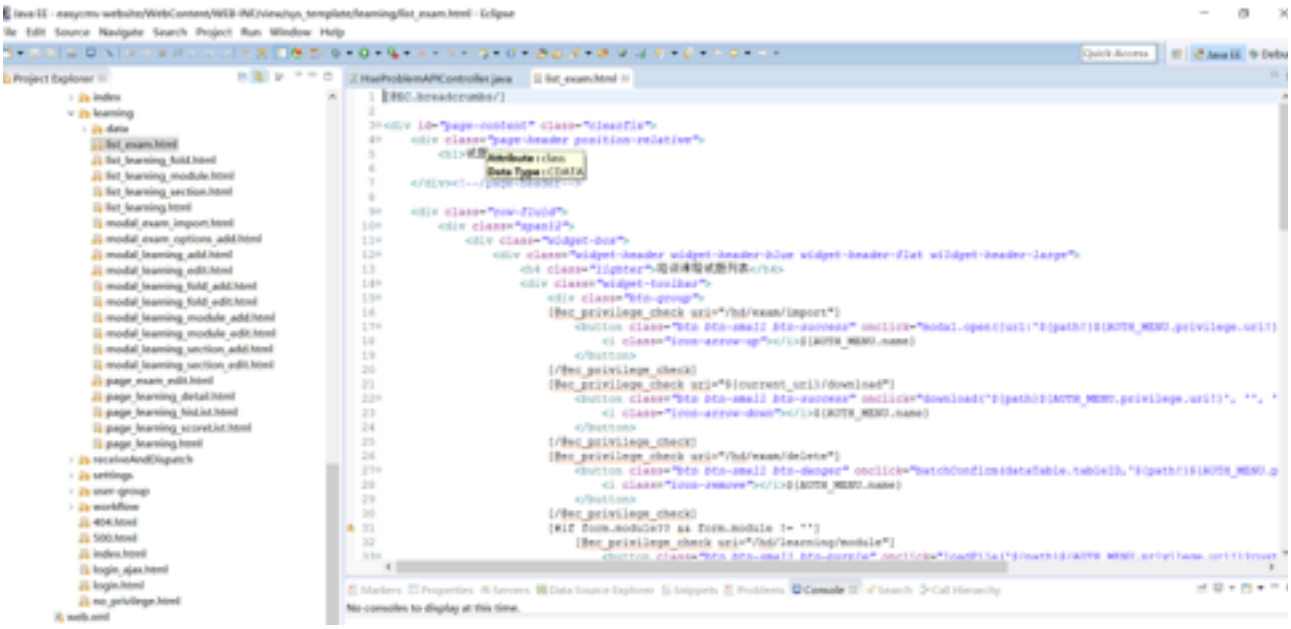
2、在线培训的数据管理：学习记录管理、培训管理、成绩列表

3、安全文明施工：创建安全问题、安全文明问题管理

4、用户管理

5、菜单管理

6、提供APP使用的接口



7.2 代码编制

1）APP端

tabbar页面代码：tabbar.html

```
<!DOCTYPE html>
<html>

	<head>
		<meta charset="utf-8">
		<title>Hello MUI</title>
		<meta name="viewport" content="width=device-width, initial-
scale=1,maximum-scale=1,user-scalable=no">
		<meta name="apple-mobile-web-app-capable" content="yes">
		<meta name="apple-mobile-web-app-status-bar-style" content="black">
		<!--标准mui.css-->
		<link rel="stylesheet" href="../../../public/css/mui.min.css">

		<style>

		</style>
	</head>

	<body>

		<nav class="mui-bar mui-bar-tab">

			<a id="defaultTab" class="mui-tab-item mui-active" href="../chat/
chatList.html">
				<span class="mui-icon mui-icon-chatboxes"></span>
				<span class="mui-tab-label">群聊</span>
			</a>

			<a class="mui-tab-item" href="../training/training_home.html">
				<span class="mui-icon mui-icon-compose"></span>
```

```
                        <span class="mui-tab-label">培训管理</span>
                    </a>

                    <a class="mui-tab-item" href="../security/security.html">
                        <span class="mui-icon mui-icon-flag"></span>
                        <span class="mui-tab-label">安全质量</span>
                    </a>
                    <a class="mui-tab-item" href="../chat/contact.html">
                        <span class="mui-icon mui-icon-speech"></span>
                        <span class="mui-tab-label">对讲机</span>
                    </a>
                </nav>

                <div class="mui-content">
                </div>

        </body>
        <script src="../../../public/js/mui.min.js"></script>
        <script src="../public/js/common.js"></script>
        <script type="text/javascript" src="../chat/js/TIMManager.js"></script>
        <script>
                var subpages = ["../chat/chatList.html", '../training/training_home.html',
'../security/security.html', '../chat/contact.html'];
                mui.init({
                        swipeBack: false, //启用右滑关闭功能

                });
                var subpage_style = {
                        top: '0px',
                        bottom: '51px'
                };

                var aniShow = {};

                function createLocalPushMsg() {

                        var options = {
                                cover: false
                        };
                        var str = "1欢迎使用Html5 Plus创建本地消息！";
                        plus.push.createMessage(str, "LocalMSG", options);
                        logAlert("创建本地消息成功！");

                }
                //创建子页面，首个选项卡页面显示，其它均隐藏；
                mui.plusReady(function() {
                        var clientId = localStorage.getItem("cid");
                        if(clientId) {
                                logAlert('hascid')
                        } else {
                                logAlert('nocid')
                                getPushInfo();
                        }


                        var self = plus.webview.currentWebview();

                        for(var i = 0; i < subpages.length; i++) {
```

```
                        var temp = {};
                        var sub = plus.webview.create(subpages[i], subpages[i],
subpage_style);
                        if(i > 0) {
                                sub.hide();
                        } else {
                                temp[subpages[i]] = "true";
                                mui.extend(aniShow, temp);
                        }
                        self.append(sub);
                }

                plus.webview.currentWebview().addEventListener('close',
                        function() {
                                unregisterListener('action_new_msg',
localStorage.newMsgListner + "");
                                unregisterListener('action_update_conversation',
localStorage.conversationUpdateLisnter + "");
                                localStorage.removeItem('newMsgListner');
                                localStorage.removeItem('conversationUpdateLisnter');
                                //unregister
                                logout();
                        });
        });

        function getPushInfo() {
                var t1 = window.setInterval(function() {
                        var info = plus.push.getClientInfo();
                        var cid = info.clientid;
                        var token = info.token;
                        if(cid && token) {

                                localStorage.setItem("cid", cid);
                                localStorage.setItem("pushtoken", token);
                                var cid1 = localStorage.getItem("cid");
                                var token1 = localStorage.getItem("pushtoken");
                                logAlert('cid1:' + cid1);
                                logAlert('token1:' + token1);
                                authUser(function() {
                                        window.clearInterval(t1);
                                }, function() {

                                        fail('认证失败');

                                });

                        }
                }, 1000);
        }
        var createIframe = function(el, opt) {
                var elContainer = document.querySelector(el);
                var wrapper = document.querySelector(".mui-iframe-wrapper");
                if(!wrapper) {
                        // 创建wrapper 和 iframe
                        wrapper = document.createElement('div');
                        wrapper.className = 'mui-iframe-wrapper';
                        for(var i in opt.style) {
                                wrapper.style[i] = opt.style[i];
                        }
                        var iframe = document.createElement('iframe');
```

```
                iframe.src = opt.url;
                iframe.id = opt.id || opt.url;
                iframe.name = opt.id;
                wrapper.appendChild(iframe);
                elContainer.appendChild(wrapper);
        } else {

                var iframe = wrapper.querySelector('iframe');
                iframe.src = opt.url;
                iframe.id = opt.id || opt.url;
                iframe.name = iframe.id;
        }
}

mui.ready(function() {

        if(!mui.os.plus) {
                var defaultTab = document.getElementById("defaultTab");
                //模拟首页点击
                mui.trigger(defaultTab, 'tap');
        }
});

//当前激活选项
var activeTab = subpages[0];
var title = document.getElementById("title");
//选项卡点击事件
mui('.mui-bar-tab').on('tap', 'a', function(e) {
        var targetTab = this.getAttribute('href');

        if(!mui.os.plus) {
                // 创建iframe代替子页面
                createIframe('.mui-content', {
                        url: targetTab,
                        style: {
                                top: '0px', //设置距离顶部的距离
                                bottom: '50px' //设置距离底部的距离
                        }
                });

        } else {
                if(targetTab == activeTab) {
                        return;
                }

                if(mui.os.ios || aniShow[targetTab]) {
                        plus.webview.show(targetTab);
                } else {
                        //否则，使用fade-in动画，且保存变量
                        var temp = {};
                        temp[targetTab] = "true";
                        mui.extend(aniShow, temp);
                        plus.webview.show(targetTab, "fade-in", 300);
                }
                //隐藏当前;
                plus.webview.hide(activeTab);
        }
        //显示目标选项卡
```

```
                //若为iOS平台或非首次显示，则直接显示

                //更改当前活跃的选项卡
                activeTab = targetTab;
        });

        //自定义事件
        document.addEventListener('group', function() {
                var defaultTab = document.getElementById("defaultTab");
                //模拟首页点击
                mui.trigger(defaultTab, 'tap');
                //切换选项卡高亮
                var current = document.querySelector(".mui-bar-tab>.mui-tab-
item.mui-active");
                if(defaultTab !== current) {
                        current.classList.remove('mui-active');
                        defaultTab.classList.add('mui-active');
                }
        });
    </script>
</html>
```

群聊页面部分业务代码：chatList.html

```
<!DOCTYPE html>
<html>

    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width,initial-scale=1,minimum-
scale=1,maximum-scale=1,user-scalable=no" />
        <title></title>

        <link href="../../../public/css/mui.min.css" rel="stylesheet" />
        <link rel="stylesheet" type="text/css" href="../../../public/css/
app.css" />
        <link rel="stylesheet" type="text/css" href="../public/css/common.css" />

        <style type="text/css">
            .conv-date {
                font-size: small;
                color: grey;
            }
        </style>
        <style type="text/css">
            .card-contact-name {
                font-size: large;
                color: black;
            }

            .mui-card,
            .mui-btn,
            .mui-btn-purple {
                margin-top: 20px;
                text-align: center;
            }

            #offCanvasSide {
```

```
                        width: 200px;
                        text-align: center;
                }

                .menu-user-card {
                        display: -webkit-box;
                        -webkit-box-align: center;
                        -webkit-box-orient: horizontal;
                        text-align: center;
                }
        </style>
</head>

<body>
        <header class="mui-bar mui-bar-nav">
                <span class="mui-action-back mui-icon mui-icon-left-nav mui-pull-
left"></span>
                <h1 id="title" class="mui-title">群聊</h1>
                <span id="create_group" class="mui-btn mui-btn-blue mui-btn-link
mui-pull-right">发起群聊</span>
        </header>
        <div class="mui-content">
                <div>
                        <ul class="mui-table-view" id="list">
                        </ul>
                        <div id="no_data" style="visibility: hidden; width:
100%;position: absolute;top: 114px;bottom: 0px;left: 0px;justify-content: center;-
webkit-justify-content: center;-webkit-align-items:center;align-items:center;
display:flex;display: -webkit-flex;">
                                <p class="" style="font-size: 20px;">没有数据</p>

                        </div>

                </div>
        </div>
</body>

</html>
<script src="../../../public/js/mui.min.js"></script>
<script src="../public/js/common.js"></script>
<script type="text/javascript" src="./js/TIMManager.js"></script>
<script>
        var chatPage = null;
        var conversationUpdateLisnter, newMsgListner;
        mui.init({
                swipeBack: true //启用右滑关闭功能
        });
        mui.plusReady(function() {
                var self = plus.webview.currentWebview();
                var param = {};
                param.action = "cmd_init";
                if(mui.os.ios) {
                        var paramios = {};
                        paramios.action = "cmd_init";
                        initIMSDK(paramios, function(data) {

                                self = this;
                                self.login(loginSuccess,
                                        function(result) {
```

```
                                        refreshChatListItems();

                                    }
                                );
                            },
                            function(result) {
                                logAlert(89);
                            });
                    } else {
                        plus.TIMManager.callIMCmd(JSON.stringify(param),
                            function(result) {
                                var self = this;
                                self.login(loginSuccess,
                                    function(result) {
                                        refreshChatListItems();

                                    }
                                );
                            },
                            function(result) {

                            }
                        );
                    }

                    plus.webview.currentWebview().addEventListener('close',
                        function() {
                            if (mui.os.ios) {
                                unregisterListener('action_new_msg',
localStorage.newMsgListner);
                            unregisterListener('action_update_conversation',
localStorage.conversationUpdateLisnter);
                            }else{
                                unregisterListener('action_new_msg', newMsgListner);
                            unregisterListener('action_update_conversation',
conversationUpdateLisnter);
                            }

                            //unregister
                            logout();
                        });


                        if(mui.os.ios) {

//
                            iOSRegisterIMListener('action_new_msg', function(data) {
//

refreshChatListItems();

                                                    },
                                                    function(result) {

                                                    }
                                                );
                            iOSRegisterIMListener('action_update_conversation',
function(data) {
```

```
refreshChatListItems();

                                                        },
                                                        function(result) {

                                                        }
                                                );
        } else {
                                                        newMsgListner =
registerListener('action_new_msg', function(data) {
                                refreshChatListItems();
                        });

                        conversationUpdateLisnter =
registerListener('action_update_conversation', function(data) {
                                refreshChatListItems();
                        });

                        localStorage.conversationUpdateLisnter = '' +
conversationUpdateLisnter;
                        localStorage.newMsgListner = '' + newMsgListner;
        }

        });

        document.getElementById('create_group').addEventListener('tap',
function(e) {


                mui.openWindow({
                        id: 'contact.html',
                        url: 'contact.html',
                        extras: {
                                createGroup: true,
                        }
                });

        });

        mui(".mui-content").on("tap", ".mui-table-view-cell", function() {

                //获取id

                var identifier = this.dataset.tag;
                var chat_type = this.dataset.type;

                //获得详情页面
                if(!chatPage) {
                        if(mui.os.plus) {
                                chatPage = plus.webview.getWebviewById('im-chat.html');
                        }
                }

                var contact = {
                        realname: this.dataset.name
```

```
	};

	if(chatPage) {
		//传值给详情页面，通知加载新数据
		mui.fire(chatPage, 'chatDetail', {
			identifier: identifier,
			chat_type: chat_type,
			contact: contact
		});
	}
	//打开详情

	mui.openWindow({
		id: 'im-chat.html',
		url: 'im-chat.html',
		extras: {
			identifier: identifier,
			chat_type: chat_type,
			contact: contact
		}
	});
	//打开详情

});

if(mui.os.ios) {

	initUI();
} else {
	initUI();
}

function loginSuccess(result) {
	setTimeout(function() {

		refreshChatListItems();

	}, 2000);

}


function sendPush() {
	var user = getCurrentUser();

	var param = {
		title: "12iOS推送测试",
		message: "推送的消息内容34",
		userId: user.id

	};

	post('/jpush/user', param, success, error);

	function success(data, param, dataStringify) {
```

```
                        //
                        logAlert("数据数据-----" + dataStringify);
                }

                function error(info) {
                        mui.alert("请求失败! " + info);
                }
        }

        function initUI() {
//                          logAlert(2);
                refreshChatListItems();
                fetchContacts();
        }

        function compare(property) {
                return function(a, b) {
                        var value1 = a[property];
                        var value2 = b[property];
                        return value2 - value1;
                }
        }

        function refreshChatListItems() {

                getConvList("group", function(data) {
                        mui.hideLoading();
                        if(data.length == 0) {
                                document.getElementById('list').innerHTML = '';
                                document.getElementById('list').style.visibility =
'hidden';
                                document.getElementById('no_data').style.visibility =
'visible';
                                return
                        }

                        data.sort(compare('timestamp'));

                        var htmlCells = '';

                        for(var i = 0; i < data.length; i++) {
                                var item = data[i];
                                var msg = item.lastmsg;
                                var content = '';
                                var unread = 0;
                                var timestamp = 0;
                                unread = item.unread;
                                if(msg) {

                                        content = msg.content;
                                        //
logAlert('msg'+msg);
                                        if(msg.type == 'sound') {
                                                content = '[语音消息]';
                                        } else if(msg.type == 'tips') {
                                                content = replaceUser(msg.content);
                                        }
                                }
```

```
                                var unreadStyle = 'style="display:none"';
                                if(unread > 0) {
                                        unreadStyle = '';
                                }

                                timestamp = formatChatDate(item.timestamp * 1000);
                                htmlCells += '<li data-name="' + item.name + '" data-
tag="' + item.identifier + '" data-type="' + item.type + '" class="mui-table-view-cell
mui-media">' +
                                                '                                       <a>' +
                                                '
<img  src="./images/group_chat_icon.png" class="mui-media-object mui-pull-left">' +
                                                '
<div class="mui-media-body">' +
                                                '
        <span class="conv-name">' + item.name + '</span> <span class="conv-date mui-pull-
right">' + timestamp + '</span>' +
                                                '
        <p class=\'mui-ellipsis \'>' +
                                                '
                <span class="conv-message">' + content + '</span>' +
                                                '
                <span class="mui-badge mui-badge-danger mui-pull-right" ' + unreadStyle +
' >' +
                                                unread +
                                                '                                       </span>' +
                                                '
        </p>' +
                                                '                                                       </
div>' +
                                                '                                               </a>' +
                                                '                                       </li>';
                        }
                        document.getElementById('list').innerHTML = htmlCells;
                        document.getElementById('list').style.visibility = 'visible';
                        document.getElementById('no_data').style.visibility =
'hidden';
                },
                function(result) {
                        mui.hideLoading();
                        document.getElementById('no_data').style.visibility =
'visible';
                }
        );

        }
</script>
```

人员培训管理页面代码：training_home.html

```
<!DOCTYPE html>
<html>

        <head>
                <meta charset="utf-8">
                <title>Hello MUI</title>
                <meta name="viewport" content="width=device-width, initial-
scale=1,maximum-scale=1,user-scalable=no">
```

```
            <meta name="apple-mobile-web-app-capable" content="yes">
            <meta name="apple-mobile-web-app-status-bar-style" content="black">

            <!--标准mui.css-->
            <link rel="stylesheet" href="../../../public/css/mui.min.css">
            <!--App自定义的css-->
            <link rel="stylesheet" type="text/css" href="../../../public/css/
app.css" />
            <style>
                  .item-box-icon {
                        display: inline-block;
                        width: 90px;
                        height: 60px;
                        margin-left: 5%;
                        position: relative;
                        display: -webkit-box;
                        display: -webkit-flex;
                        display: flex;
                        -webkit-box-pack: justify;
                        -webkit-box-align: center;
                        -webkit-align-items: center;
                        align-items: center;
                  }

                  .item-box-text {
                        margin-left: 5%;
                        font-size: 18px;
                  }

                  .mui-card-content {
                        width: 100%;
                        border: 0px;
                  }

                  .mui-card-box {
                        padding: 0px 0px;
                        margin: 0px;
                        position: relative;
                        display: -webkit-box;
                        display: -webkit-flex;
                        display: flex;
                        min-height: 44px;
                        -webkit-box-pack: justify;
                        -webkit-box-align: center;
                        -webkit-align-items: center;
                        align-items: center;
                  }

                  img {
                        width: 70px;
                        height: 70px;
                  }
            </style>
      </head>

      <body>
            <header class="mui-bar mui-bar-nav">
                  <span class="mui-action-back mui-icon mui-icon-left-nav mui-pull-
left"></span>
```

```
                <h1 class="mui-title">培训管理</h1>
            </header>
            <div class="mui-content">
                <div class="mui-card">
                    <button id="base" class="mui-card-content" data-type="JCPX"
href="training_skill_subpage.html">
                        <div class="mui-card-content-inner">
                            <div class="mui-card-box">
                                <div class="item-box-icon">
                                <img src="../public/images/basicIcon.png"
/>
                                </div>
                                <div class="item-box-text">
                                    <span>基础培训</span>
                                </div>
                            </div>

                        </div>
                    </button>
                </div>

                <div class="mui-card">
                    <button id="skill" class="mui-card-content" data-type="JNPX"
href="training_skill_subpage.html">
                        <div class="mui-card-content-inner">
                            <div class="mui-card-box">
                                <div class="item-box-icon">
                                <img src="../public/images/skillIcon.png"
/>
                                </div>
                                <div class="item-box-text">
                                    <span>技能培训</span>
                                </div>
                            </div>

                        </div>
                    </button>
                </div>

                <div class="mui-card">
                    <button id="special" class="mui-card-content" data-
type="ZXPX" href="training_skill_subpage.html">
                        <div class="mui-card-content-inner">
                            <div class="mui-card-box">
                                <div class="item-box-icon">
                                <img src="../public/images/
specialIcon.png" />
                                </div>
                                <div class="item-box-text">
                                    <span>专项培训</span>
                                </div>
                            </div>

                        </div>
                    </button>
                </div>

                <div class="mui-card">
```

```
                        <button id="record" class="mui-card-content"
href="learn_records.html">
                                <div class="mui-card-content-inner">
                                        <div class="mui-card-box">
                                                <div class="item-box-icon">
                                                <img src="../public/images/
recordIcon.png" />
                                                </div>
                                                <div class="item-box-text">
                                                        <span>学习记录</span>
                                                </div>
                                        </div>

                                </div>
                        </button>
                </div>

                <script src="../../../public/js/mui.min.js"></script>
                <script src="../public/js/common.js"></script>
                <script type="text/javascript" charset="utf-8">
                        mui.init({
                                swipeBack: true //启用右滑关闭功能
                        });

                        mui(".mui-content").on("tap", "button", function() {
                                var id = this.getAttribute("id");
                                var href = this.getAttribute("href");
                                var text = this.querySelector('.item-box-
text').innerText;

                                console.log('current Type:' +
this.dataset.type+";text="+text);
                                localStorage.training_type = this.dataset.type;
                                localStorage.training_title = text;
                                mui.openWindow({

                                        url: href,

                                        id: id,
                                        extras: {
                                                type: this.dataset.type,
                                                title:text
                                        }

                                });
                        });

                        var param = {};
                        get('/learning/section', param, success, error);

                        function success(data, param, dataStringify) {

                        }

                        function error(info) {
                                console.log("请求失败! " + info);
                        }
                </script>
        </body>
```

```
</html>
```

安全文明施工页面代码：security.html

```
<!DOCTYPE html>
<html>

	<head>
		<title>安全</title>
		<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
		<meta name="viewport" content="width=device-width, initial-
scale=1,maximum-scale=1,user-scalable=no">
		<meta name="apple-mobile-web-app-capable" content="yes">
		<meta name="apple-mobile-web-app-status-bar-style" content="black">
		<meta charset="utf-8" />
		<link href="../../../public/css/mui.min.css" rel="stylesheet" />
		<link rel="stylesheet" type="text/css" href="../../../public/css/
app.css" />
		<style type="text/css">
			html,
			img {
				border: 0;
			}

			ul {
				list-style: none outside none;
				margin: 0;
				padding: 0;
			}

			body {
				background-color: #eee;
			}

			body .mainmenu:after {
				clear: both;
				content: " ";
				display: block;
			}

			body .mainmenu li {
				float: left;
				margin-left: 2.5%;
				margin-top: 2.5%;
				width: 46%;
				border-radius: 3px;
				overflow: hidden;
			}

			body .mainmenu li img {
				margin: 15px auto 15px;
```

```
                        width: 50px;
                        height: 50px;
                }

                .item-box-text {
                        margin-left: 5%;
                        font-size: 18px;
                }

                .mui-card-content {
                        width: 100%;
                        border: 0px;
                }

                .mui-card{
                        margin: 0px;
                }


                #report{
                        background-color: #36A1DB!important;
                        solid #36A1DB!important;
                }

                #report:active {
                        color: #fff;
                        background-color: #36A1FB!important;
                }

                #look{
                        background-color: #84d018!important;
                        solid #84d018!important;
                }

                #look:active {
                        color: #fff;
                        background-color: #2AC845!important;
                }

        </style>
    </head>

    <body>
        <header class="mui-bar mui-bar-nav">
            <span class="mui-action-back mui-icon mui-icon-left-nav mui-pull-
left"></span>
            <h1 class="mui-title">安全管理</h1>
        </header>
        <div class="mui-content">
            <ul class="mainmenu">

                <li>
                    <div class="mui-card">

                        <button id="report" class="mui-card-content"
href="qc_report.html">
                    <div class="mui-card-content-inner">
                        <div class="mui-card-box">
                            <div class="item-box-icon">
```

```
                                        <img src="../public/images/
reportIcon.png" />
                                        </div>
                                        <div class="item-box-text">
                                              <span>报告问题</span>
                                        </div>
                                  </div>

                            </div>
                      </button>
                            </div>
                      </li>
                      <li>
                            <div class="mui-card">
                                  <button id="review" class="mui-card-content"
href="qc_review.html">
                            <div class="mui-card-content-inner">
                                  <div class="mui-card-box">
                                        <div class="item-box-icon">
                                        <img src="../public/images/checkIcon.png"
/>
                                        </div>
                                        <div class="item-box-text">
                                              <span>问题审核</span>
                                        </div>
                                  </div>

                            </div>
                      </button>
                            </div>
                      </li>
                      <li>
                            <div class="mui-card">
                                  <button id="reform" href="qc_reform.html"
class="mui-card-content">
                            <div class="mui-card-content-inner">
                                  <div class="mui-card-box">
                                        <div class="item-box-icon">
                                        <img src="../public/images/
correctionIcon.png" />
                                        </div>
                                        <div class="item-box-text">
                                              <span>问题整改</span>
                                        </div>
                                  </div>

                            </div>
                      </button>
                            </div>
                      </li>
                      <li>
                            <div class="mui-card">
                                  <button id="look" href="qc_inquire.html"
class="mui-card-content">
                            <div class="mui-card-content-inner">
                                  <div class="mui-card-box">
                                        <div class="item-box-icon">
                                        <img src="../public/images/
lookIcon.png" />
```

```
                                </div>
                                <div class="item-box-text">
                                        <span>问题查阅</span>
                                </div>
                        </div>

                </div>
        </button>
                </div>

        </li>

</ul>
</div>

<script src="../../../public/js/mui.min.js"></script>
<script src="../public/js/common.js"></script>
<script type="text/javascript" charset="utf-8">
        mui.init({
                swipeBack: true //启用右滑关闭功能
        });

        mui(".mui-content").on("tap", "button", function() {
                var id = this.getAttribute("id");
                var href = this.getAttribute("href");
                logAlert('id');
                if(id =='review'){
                 var isHse = isHSE();
                 if(isHse){
                                mui.openWindow({

                        url: href,

                        id: id

                });
                 }else{
                        mui.alert('当前用户不是HSE部门成员,无法执行该操作');
                 }

                }else if(id =='reform'){
                         var isReformUser = isReformTeam();
                 if(isReformUser){
                                mui.openWindow({

                        url: href,

                        id: id

                });
                 }else{
                        mui.alert('当前用户没有权限,无法执行该操作');
                 }


                }else{
                                mui.openWindow({

                        url: href,
```

```
                                id: id

                        });
                        }

                });
        </script>

</body>

</html>
```

请求公共类部分逻辑代码：common.js

```
function uploadFile(apiName, body, sucs, fail) {
        mui.showLoading("正在加载..", "div");

        var user = getCurrentUser();
        if(!user) {
                authUser(function() {
                        get(apiName, body, sucs, fail);
                }, function() {
                        fail('认证失败');
                });
                return;
        }

        mui.ajax(httpDomain + apiName+"?loginId="+user.id, {
                crossDomain: true,
                data: body,
                processData: false,  // 告诉jQuery不要去处理发送的数据
          contentType: false,   // 告诉jQuery不要去设置Content-Type请求头

                dataType: 'json', //服务器返回json格式数据
                type: 'POST', //HTTP请求类型
                timeout: 20000, //超时时间设置为20秒;
                success: function(data) {
                        //服务器返回响应，根据响应结果，分析是否登录成功;
                        mui.hideLoading();
                        var info = JSON.stringify(data);
                        console.log("返回请求地址->: " + apiName + "服务器内容-->" + info);
                        if(data.code == 1000) {
                                sucs(data, body, info);
                        } else if(data.code == -1002) { //没有认证
                                //
                        } else {
                                if(data.message) {
                                        fail(data.message);
                                        console.log('Upload request error:' + apiName +
":response Message=" + data.message);
                                } else {
                                        fail(info);
                                }
                        }

                },
                error: function(xhr, type, errorThrown) {
```

```
                    mui.hideLoading();

                    //异常处理；
                    fail("code=" + xhr.status + " error:" + type); //type错误描述，可取值："timeout", "error", "abort", "parsererror"、"null"

                    if(xhr.status == 0) {
                          networkWarning();
                          return
                    }
             }
       });
}

function mock(type, apiName, body, succ) {

       setTimeout(function() {
             callMock(type, apiName, body, succ);
             mui.hideLoading();
       }, 1000);

}

function post(apiName, body, sucs, fail) {
       if(!body.disableLoading) {
             mui.showLoading("正在加载..", "div");
       }

       var user = getCurrentUser();

       if(!user) {
             authUser(function() {
                    post(apiName, body, sucs, fail);
             }, function() {
                    fail('认证失败');
             });
             return;
       }

       body.loginId = user.id;

       if(mockData) {
             mock('post', apiName, body, sucs);
             return;
       }

       mui.ajax(httpDomain + apiName, {
             crossDomain: true,
             data: body,
             dataType: 'json', //服务器返回json格式数据
             type: 'POST', //HTTP请求类型
             timeout: 20000, //超时时间设置为20秒；
             success: function(data) {
                    //服务器返回响应，根据响应结果，分析是否登录成功；
                    mui.hideLoading();
                    var info = JSON.stringify(data);
//                  logAlert("请求地址和参数" + apiName + "参数" + JSON.stringify(data));
```

```
                        console.log("返回请求地址->: " + apiName + "服务器内容-->" + info);
                        if(data.code == 1000) {
                                sucs(data, body, info);
                        } else if(data.code == -1002) { //没有认证
                                //
                        } else {
                                if(data.message) {
                                        fail(data.message);
                                        console.log('Post request error:' + apiName +
":response Message=" + data.message);
                                } else {
                                        fail(info);
                                }
                        }

                },
                error: function(xhr, type, errorThrown) {
                        mui.hideLoading();

                        //异常处理;
                        fail("code=" + xhr.status + " error:" + type); //type错误描述，可取
值: "timeout", "error", "abort", "parsererror"、"null"
                        logAlert("code=" + xhr.status + " apiName=" + apiName + "
HttpRequestPostReusltErr:" + type);

                        if(xhr.status == 0 && !body.disableLoading) {
                                networkWarning();
                                return
                        }
                }
        });
}

function authUser(authSucc, authFailed) {
        //from login get userId,

        var cid = localStorage.getItem("cid");
var name = getLoginUserId();
        var body = {
                username: name,
                uuid:cid
        }
        if (mui.os.ios) {
                body.androidOrIOS = "IOS";
        } else{
                body.androidOrIOS = "android";
        }
        mui.ajax(httpDomain + '/getUserId', {
                crossDomain: true,
                data: body,
                dataType: 'json', //服务器返回json格式数据
                type: 'POST', //HTTP请求类型
                timeout: 20000, //超时时间设置为20秒;
                success: function(data) {
                        //服务器返回响应，根据响应结果，分析是否登录成功;
                        var info = JSON.stringify(data);
                        console.log("返回请求地址->: " + " auth " + "服务器内容-->" + info);
```

```
                    if(data.code == 1000) {

                            storeUserInfo(data.responseResult);
                            authSucc();
                    } else if(data.code == -1002) {
                            //user not exsit
                            authFailed();
                    } else {
                            authFailed();
                    }

            },
            error: function(xhr, type, errorThrown) {

                    mui.toast("code=" + xhr.status + " " + " 请求错误:" + type);
                    console.log("code=" + xhr.status + " " + "
HttpRequestAuthReusltErr:" + type);
                    authFailed();
                    if(xhr.status == 0) {
                            networkWarning();
                            return
                    }
            }
      });
}

function networkWarning() {
      mui.toast('请检查网络');
}

function getLoginUserId() {
      var myId = localStorage.getItem("usersname");
      return myId;
}

function storeUserInfo(userInfo) {
      localStorage.UserInfo = JSON.stringify(userInfo);
}

function getUserInfo() {
      var userStr = localStorage.UserInfo;

      if(!userStr) {
            return null;
      }
      var user = JSON.parse(userStr);

      return user;
}

function deleteCurrentUserInfo() {
      localStorage.removeItem('UserInfo');
}

function getCurrentUser() {
 //
      var user = getUserInfo();
      if(!user) {
            return null;
      }
```

```
	if(user.username != getLoginUserId()) {
		deleteCurrentUserInfo();
		return null;
	}
	return user;
}

function getStepNameWithProblemStatus(problemStatus){
	if(problemStatus == 'Need_Handle') {
	return "待处理";
} else if(problemStatus == 'Renovating') {
	return "整改中";
} else if(problemStatus == 'Need_Check') {
	return "待审核";
} else if(problemStatus == 'None') {
	return "不需处理";
} else {
	return "已完成";
}

}

function get(apiName, body, sucs, fail) {

	var user = getCurrentUser();

	if(!user) {
		authUser(function() {
			get(apiName, body, sucs, fail);
		}, function() {
			fail('认证失败');
		});
		return;
	}

	body.loginId = user.id;
//body.loginId = '514';
	var url = httpDomain + apiName;
	var param = "";
	for(var element in body) {
		param += element + "=" + body[element] + "&";
	}

	url = url + "?" + param;

	console.log("请求地址和参数" + url);
//	mui.alert('请求地址和参数:'+url);
	if(mockData) {
		mock('get', apiName, body, sucs);
		return;
	}

	mui.ajax(url, {
		crossDomain: true,
		dataType: 'json', //服务器返回json格式数据
		type: 'GET', //HTTP请求类型
		timeout: 20000, //超时时间设置为20秒;
```

```
		headers: {
			'Accept': 'application/json',
			'Content-Type': 'application/json'
		},
		success: function(data) {
			//服务器返回响应，根据响应结果，分析是否登录成功；
			var info = JSON.stringify(data);
			console.log("返回请求地址->: " + apiName + "服务器内容-->" + info);
			if(data.code == 1000) {
				sucs(data, body, info);
			} else {
				if(data.message) {
					fail(data.message);
					console.log('Get request error:' + apiName +
":response Message=" + data.message);
				} else {
					fail(info);
				}
			}

		},
		error: function(xhr, type, errorThrown) {
			//异常处理；

			fail("code=" + xhr.status + " error:" + type); //type错误描述，可取
值: "timeout", "error", "abort", "parsererror"、"null"
			console.log("code=" + xhr.status + " apiName=" + apiName + "
HttpRequestGetReusltErr:" + type);

			if(xhr.status == 0) {
				networkWarning();
				return
			}
		}
	});
}
```

2）Web后端

2.1 登陆安全认证接口：CoreAPIController.java

```
package com.easycms.hd.api.core;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
import java.util.Set;
import java.util.stream.Collectors;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;

import com.easycms.common.util.CommonUtility;
import com.easycms.hd.api.domain.AuthRole;
import com.easycms.hd.api.request.LoginRequestForm;
import com.easycms.hd.api.request.UserRequestForm;
import com.easycms.hd.api.request.base.BaseRequestForm;
import com.easycms.hd.api.response.AuthorizationResult;
import com.easycms.hd.api.response.ModulesResult;
import com.easycms.hd.api.response.TLSResult;
import com.easycms.hd.api.response.UserResult;
import com.easycms.hd.api.service.DepartmentApiService;
import com.easycms.hd.api.service.RoleApiService;
import com.easycms.hd.modules.domain.Modules;
import com.easycms.hd.modules.service.ModulesService;
import com.easycms.hd.response.JsonResult;
import com.easycms.hd.tls.TLSUtils;
import com.easycms.management.user.domain.User;
import com.easycms.management.user.domain.UserDevice;
import com.easycms.management.user.service.UserDeviceService;
import com.easycms.management.user.service.UserService;
import com.wordnik.swagger.annotations.Api;
import com.wordnik.swagger.annotations.ApiOperation;
import com.wordnik.swagger.annotations.ApiParam;

import lombok.extern.slf4j.Slf4j;

@Controller
@RequestMapping("/hdxt/api")
```

```
@Api(value = "CoreAPIController", description = "登录与主页相关api")
@Slf4j
public class CoreAPIController {

	@Autowired
	private UserService userService;
	@Autowired
	private UserDeviceService userDeviceService;
	@Autowired
	private RoleApiService roleApiService;
	@Autowired
	private DepartmentApiService departmentApiService;
	@Autowired
	private ModulesService modulesService;

	@Value("#{APP_SETTING['user_sig']}")
	private boolean userSig;

	@Value("#{APP_SETTING['file_base_path']}")
	private String basePath;

	@Value("#{APP_SETTING['windows_file_base_path']}")
	private String winBasePath;

	@Value("#{APP_SETTING['video_suffix']}")
	private String videoSuffix;

	/**
	 * 认证用户
	 *
	 * @param request
	 * @param response
	 * @param loginForm
	 * @return
	 */
	@ResponseBody
	@RequestMapping(value = "/authenticate", method = RequestMethod.POST)
	@ApiOperation(value = "Mobile端登录", notes = "用于移动端登录。")
	public JsonResult<AuthorizationResult> authenticate(HttpServletRequest request,
HttpServletResponse response,
			@ModelAttribute("requestForm") @Valid LoginRequestForm requestForm) {
		JsonResult<AuthorizationResult> result = new JsonResult<AuthorizationResult>();
		AuthorizationResult auth = new AuthorizationResult();
		String username = requestForm.getUsername();
		String password = requestForm.getPassword();
		String uuid = requestForm.getUuid();
		log.info("==> username： " + username);
		log.info("==> password： " + password);
		log.info("==> device： " + uuid);

		// LOGIN ID非空判断
		if (!CommonUtility.isNonEmpty(username)) {
			result.setCode("-1001");
			result.setMessage("用户名不能为空");
			return result;
		}

		// PASSWORD 非空判断
		if (!CommonUtility.isNonEmpty(password)) {
```

```
                result.setCode("-1001");
                result.setMessage("密码不能为空");
                return result;
        }

        // LOGIN ID有效判断
        User user = userService.findUser(username);
        if (user == null) {
                user = userService.findByNameIsDel1(username);
                if(user!=null){

                        result.setCode("-1001");
                        result.setMessage("用户" + username + "被冻结");
                        return result;
                }else{

                        result.setCode("-1001");
                        result.setMessage("用户" + username + "不存在");
                        return result;
                }
        }

        // 登录认证
        user = userService.userLogin(username, password, false);

        if (user == null) {
                result.setCode("-1002");
                result.setMessage("The cause: the password is wrong");
                return result;
        }

        if (CommonUtility.isNonEmpty(uuid)) {

                if (null == userDeviceService.getDeviceByDeviceIdAndUserId(user.getId(), uuid)) {
                        UserDevice userDevice = new UserDevice();
                        userDevice.setUser(user);
                        userDevice.setDeviceid(uuid);
                        userDevice.setCreatedOn(new Date());
                        userDevice.setCreatedBy("Core_API");
                        userDeviceService.add(userDevice);
                }
        }
        auth.setId(user.getId());
        auth.setRealname(user.getRealname());
        auth.setUsername(user.getName());

        if (user.getRoles() != null && user.getRoles().size() > 0) {
                Set<AuthRole> roles = roleApiService.getAuthRole(user);
                auth.setRoles(roles);
        }

//              //设置模块属性
//              List<Modules> modules = user.getModules();
//              if(modules!=null && modules.size()>0){
//                      Set<AuthModule> m = roleApiService.getAuthModule(user);
//                      auth.setModules(m);
//              }

        if (null != user.getDepartment()) {
```

```
auth.setDepartment(departmentApiService.departmentTransferToDepartmentResult(user.getDepartment()));
                }

                if (userSig) {
                        TLSResult tls = new TLSResult();

                        try {
                                tls = TLSUtils.getUserSig(user.getName());
                                auth.setTls(tls);
                        } catch (Exception e) {
                                e.printStackTrace();
                                log.error(e.getMessage());
                        }
                }

                result.setResponseResult(auth);
                return result;
        }

        @ResponseBody
        @RequestMapping(value = "/user", method = RequestMethod.GET)
        @ApiOperation(value = "Mobile端获取用户信息", notes = "用于移动端获取指定用户ID的用户信息。")

        public JsonResult<UserResult> user(HttpServletRequest request, HttpServletResponse response,
                        @ModelAttribute("form") UserRequestForm form) {
                JsonResult<UserResult> result = new JsonResult<UserResult>();
                UserResult auth = new UserResult();
                // LOGIN ID非空判断
                if (null == form.getUserId()) {
                        result.setCode("-1001");
                        result.setMessage("用户ID不能为空");
                        return result;
                }

                // 获取用户信息
                User user = userService.findUserById(form.getUserId());

                if (user == null) {
                        result.setCode("-1002");
                        result.setMessage("提定用户ID[" + form.getUserId() + "]的用户不存在。");
                        return result;
                }

                auth.setId(user.getId());
                auth.setRealname(user.getRealname());
                auth.setUsername(user.getName());

                if (user.getRoles() != null && user.getRoles().size() > 0) {
                        Set<AuthRole> roles = roleApiService.getAuthRole(user);
                        auth.setRoles(roles);
                }

                result.setResponseResult(auth);
                return result;

        }

        @ResponseBody
```

```
	@RequestMapping(value = "/getUserId", method = RequestMethod.POST)
	@ApiOperation(value = "Mobile端获取用户id", notes = "用于移动端获取指定用户的ID用户信息。")
	public JsonResult<UserResult> getUserId(HttpServletRequest request, HttpServletResponse
response,
			@ApiParam(required = true, name = "username", value =
"username")@RequestParam(value="username",required=true) String username,
			@ApiParam(required = false, name = "uuid", value =
"uuid")@RequestParam(value="uuid",required=false) String uuid,
			@ApiParam(required = false, name = "androidOrIOS", value =
"androidOrIOS")@RequestParam(value="androidOrIOS",required=false) String androidOrIOS) {

		JsonResult<UserResult> result = new JsonResult<UserResult>();
		UserResult auth = new UserResult();
		// LOGIN username非空判断
		if (null == username || "".equals(username.trim())) {
			result.setCode("-1001");
			result.setMessage("用户ID不能为空");
			return result;
		}

		// 获取用户信息
		User user = userService.findUser(username.trim());

		if (user == null) {
			result.setCode("-1002");
			result.setMessage("指定用户ID[" + username + "]的用户不存在。");
			return result;
		}

		if (CommonUtility.isNonEmpty(androidOrIOS) && CommonUtility.isNonEmpty(uuid)) {
			List<UserDevice> deviceList = userDeviceService.getDeviceByUserId(user.getId());
			if(deviceList==null || deviceList.size()==0){
				UserDevice userDevice = new UserDevice();
				userDevice.setUser(user);
				userDevice.setDeviceid(uuid);
				userDevice.setAndroidOrIOS(androidOrIOS);
				userDevice.setCreatedOn(new Date());
				userDevice.setCreatedBy("Core_API");
				userDeviceService.add(userDevice);
			}else{
				if(deviceList.size()==1){
					UserDevice userDevice = deviceList.get(0);
					userDevice.setDeviceid(uuid);
					userDevice.setAndroidOrIOS(androidOrIOS);
					userDevice.setUpdatedOn(Calendar.getInstance().getTime());
					userDeviceService.add(userDevice);
				}else if(deviceList.size()>1){
					//首记录更新使用，其它记录删除掉
					UserDevice userDevice = deviceList.get(0);
					deviceList.remove(0);
					userDevice.setDeviceid(uuid);
					userDevice.setAndroidOrIOS(androidOrIOS);
					userDevice.setUpdatedOn(Calendar.getInstance().getTime());
					userDeviceService.add(userDevice);

					deviceList.stream().forEach(d->userDeviceService.delete(d.getId()));
				}
			}
		}
```

```
//                  if (user == null) {
//                          result.setCode("-1003");
//                          result.setMessage("提定用户ID[" + username + "]的用户不存在。");
//                          return result;
//                  }

                    auth.setId(user.getId());
                    auth.setRealname(user.getRealname());
                    auth.setUsername(user.getName());

                    if (user.getRoles() != null && user.getRoles().size() > 0) {
                            Set<AuthRole> roles = roleApiService.getAuthRole(user);
                            auth.setRoles(roles);
                    }

                    if (null != user.getDepartment()) {

auth.setDepartment(departmentApiService.departmentTransferToDepartmentResult(user.getDepartment()));
                    }

                    result.setResponseResult(auth);
                    return result;

            }

            /**
             * 登出，清除clientid
             * @param request
             * @param response
             * @param username
             * @param uuid
             * @param androidOrIOS
             * @return
             */
            @ResponseBody
            @RequestMapping(value = "/logout", method = RequestMethod.POST)
            @ApiOperation(value = "Mobile端获取用户id", notes = "用于移动端获取指定用户的ID用户信息。")
            public JsonResult<UserResult> logout(HttpServletRequest request, HttpServletResponse response,
                            @ApiParam(required = true, name = "username", value =
"username")@RequestParam(value="username",required=true) String username) {

                    JsonResult<UserResult> result = new JsonResult<UserResult>();
                    UserResult auth = new UserResult();
                    if (null == username || "".equals(username.trim())) {
                            result.setCode("-1001");
                            result.setMessage("用户ID不能为空");
                            return result;
                    }

                    // 获取用户信息
                    User user = userService.findUser(username.trim());

                    if (user == null) {
                            result.setCode("-1002");
                            result.setMessage("指定用户ID[" + username + "]的用户不存在。");
                            return result;
                    }
```

```
                List<UserDevice> deviceList = userDeviceService.getDeviceByUserId(user.getId());
                if(deviceList==null || deviceList.size()==0){

                }else{
                        if(deviceList.size()==1){
                                UserDevice userDevice = deviceList.get(0);
                                userDevice.setDeviceid("");
                                userDevice.setAndroidOrIOS("");
                                userDevice.setUpdatedOn(Calendar.getInstance().getTime());
                                userDeviceService.add(userDevice);
                        }else if(deviceList.size()>1){
                                //首记录更新使用，其它记录删除掉
                                UserDevice userDevice = deviceList.get(0);
                                deviceList.remove(0);
                                userDevice.setDeviceid("");
                                userDevice.setAndroidOrIOS("");
                                userDevice.setUpdatedOn(Calendar.getInstance().getTime());
                                userDeviceService.add(userDevice);

                                deviceList.stream().forEach(d->userDeviceService.delete(d.getId()));
                        }
                }

                auth.setId(user.getId());
                auth.setRealname(user.getRealname());
                auth.setUsername(user.getName());

                if (user.getRoles() != null && user.getRoles().size() > 0) {
                        Set<AuthRole> roles = roleApiService.getAuthRole(user);
                        auth.setRoles(roles);
                }

                if (null != user.getDepartment()) {

auth.setDepartment(departmentApiService.departmentTransferToDepartmentResult(user.getDepartment()));
                }

                result.setResponseResult(auth);
                return result;

        }

        @ResponseBody
        @RequestMapping(value = "/module", method = RequestMethod.GET)
        @ApiOperation(value = "Mobile端module列表", notes = "在移动端登录之后主界面中间module列表")
        public JsonResult<List<ModulesResult>> getModules(HttpServletRequest request,
HttpServletResponse response,
                        @RequestParam(value = "loginId", required = true) @ApiParam(value = "用户登录之
后的用户ID", required = true) Long loginId) {
                JsonResult<List<ModulesResult>> result = new JsonResult<List<ModulesResult>>();

                List<ModulesResult> modulesResults = new ArrayList<ModulesResult>();

                List<Modules> modules = modulesService.findByStatus("ACTIVE");

                modulesResults = modules.stream().map(module -> {
                        ModulesResult modulesResult = new ModulesResult();
                        modulesResult.setName(module.getName());
                        modulesResult.setId(module.getId());
```

```
                modulesResult.setOrderNumber(module.getOrderNumber());
                modulesResult.setType(module.getType());
                return modulesResult;
            }).collect(Collectors.toList());

            result.setResponseResult(modulesResults);
            return result;

        }

        @ResponseBody
        @RequestMapping(value = "/tls", method = RequestMethod.POST)
        @ApiOperation(value = "Mobile端获取用户TLS", notes = "用于移动端获取指定用户名的TLS信息。")
        public JsonResult<TLSResult> tls(HttpServletRequest request, HttpServletResponse response,
                @ModelAttribute("form") BaseRequestForm form) {
            JsonResult<TLSResult> result = new JsonResult<TLSResult>();

            User user = userService.findUserById(form.getLoginId());

            if (user == null) {
                result.setCode("-1002");
                result.setMessage("提定用户ID[" + form.getLoginId() + "]的用户不存在。");
                return result;
            }

            TLSResult tls = new TLSResult();

            try {
                tls = TLSUtils.getUserSig(user.getName());
            } catch (Exception e) {
                e.printStackTrace();
                log.error(e.getMessage());
            }

            result.setResponseResult(tls);
            return result;
        }


        public static void main(String[] args) {
            File f = new File("D:\\tmp\\zhierxing.pdf");


            System.out.println(f.length());

        }

        @RequestMapping(value = {"/files/{filePath:.+}"}, method = RequestMethod.GET)
        @ApiOperation(value = "Mobile端得到文件", notes = "将uri跟在id上则可得到相应流文件")
        public void getConferenceFile(HttpServletRequest request, HttpServletResponse response,
                @PathVariable("filePath") @ApiParam(value = "文件路径，约定*代替为/", required =
true) String filePath)
                                        throws Exception {

            response.setDateHeader("Expires", 0);
            response.setHeader("Cache-Control", "no-store, no-cache, must-revalidate");
            response.addHeader("Cache-Control", "post-check=0, pre-check=0");
            response.setHeader("Pragma", "no-cache");
```

```
		InputStream in = null;
		OutputStream out = null;
		try {

			if( false ==
com.easycms.common.logic.context.ContextPath.Linux.equals(com.easycms.common.logic.context.Conte
xtPath.os_name)){
				basePath = winBasePath ;
			}

			String streamFilePath = basePath + filePath.replace("*", "/");
			String suffix = filePath.substring(filePath.lastIndexOf(".") + 1, filePath.length());
			log.debug("文件类型："+ suffix);
			List<String> videoSuffixList = Arrays.asList(videoSuffix.split(","));
			if(videoSuffixList.contains(suffix) || "mp4".equals(suffix)){//视频文件
				response.setContentType("video/x-msvideo");
				response.setContentType("APPLICATION/OCTET-STREAM");
				response.setHeader("Content-Type","video/x-msvideo");
			}

			File f = new File(streamFilePath);
			if(f!=null && f.exists()){
				response.setHeader("Content-Length", f.length()+"");
			}

			in = new FileInputStream(streamFilePath); // 获取文件的流


			int len = 0;
			byte buf[] = new byte[1024];// 缓存作用
			out = response.getOutputStream();// 输出流
			while ((len = in.read(buf)) > 0) {// 切忌这后面不能加 分号 ";“
				out.write(buf, 0, len);// 向客户端输出，实际是把数据存放在response中，然
后web服务器再去response中读取
			}
		} catch (Exception e) {
			e.printStackTrace();
		} finally {
			if (in != null) {
				try {
					in.close();
				} catch (IOException e) {
					e.printStackTrace();
				}
			}

			if (out != null) {
				try {
					out.close();
				} catch (IOException e) {
					e.printStackTrace();
				}
			}
		}
	}

}
```

2.2 安全文明施工接口：HseProblemAPIController.java

```
package com.easycms.hd.api.baseservice.hseproblem;

import java.util.Calendar;
import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.multipart.commons.CommonsMultipartFile;

import com.easycms.common.logic.context.ContextPath;
import com.easycms.common.upload.FileInfo;
import com.easycms.common.upload.UploadUtil;
import com.easycms.common.util.CommonUtility;
import com.easycms.common.util.FileUtils;
import com.easycms.core.util.Page;
import com.easycms.hd.api.enums.hseproblem.HseProbelmTypeEnum;
import com.easycms.hd.api.enums.hseproblem.HseProblemSolveStatusEnum;
import com.easycms.hd.api.enums.hseproblem.HseProblemStatusEnum;
import com.easycms.hd.api.request.base.BaseRequestForm;
import com.easycms.hd.api.request.hseproblem.HseProblemAssignRequestForm;
import com.easycms.hd.api.request.hseproblem.HseProblemListRequestFormByStatus;
import com.easycms.hd.api.request.hseproblem.HseProblemRenovateForm;
import com.easycms.hd.api.request.hseproblem.HseProblemRequestForm;
import com.easycms.hd.api.response.hseproblem.HseProblemCreatUiDataResult;
import com.easycms.hd.api.response.hseproblem.HseProblemPageDataResult;
import com.easycms.hd.api.response.hseproblem.HseProblemResult;
import com.easycms.hd.api.service.HseProblemApiService;
import com.easycms.hd.hseproblem.domain.HseProblem;
import com.easycms.hd.hseproblem.domain.HseProblemFile;
import com.easycms.hd.hseproblem.domain.HseProblemFileType;
import com.easycms.hd.hseproblem.domain.HseProblemSolve;
import com.easycms.hd.hseproblem.domain.HseProblemSolveStep;
import com.easycms.hd.hseproblem.domain.HseProblemStep;
import com.easycms.hd.hseproblem.service.HseProblemFileService;
import com.easycms.hd.hseproblem.service.HseProblemService;
import com.easycms.hd.hseproblem.service.HseProblemSolveService;
import com.easycms.hd.response.JsonResult;
import com.easycms.management.user.domain.User;
import com.easycms.management.user.service.UserService;
import com.wordnik.swagger.annotations.Api;
import com.wordnik.swagger.annotations.ApiOperation;
import com.wordnik.swagger.annotations.ApiParam;

import lombok.extern.slf4j.Slf4j;

@Controller
@RequestMapping("/hdxt/api/hse")
@Api(value="HseProblemAPIController",description="文明施工问题模块相关的api")
@Slf4j
```

```
public class HseProblemAPIController {
	@Autowired
	private HseProblemService hseProblemService;
	@Autowired
	private HseProblemApiService hseProblemApiService;
	@Autowired
	private HseProblemFileService hseProblemFileService;
	@Autowired
	private UserService userService;
	@Autowired
	private HseProblemSolveService hseProblemSolveService;

	@ResponseBody
	@RequestMapping(value = "createUI", method = RequestMethod.GET)
	@ApiOperation(value = "创建安全文明问题界面", notes = "用于获取机组、厂房、责任部门、责任班
组")
	public JsonResult<HseProblemCreatUiDataResult> createUI (HttpServletRequest request,
HttpServletResponse response,
				@ModelAttribute("requestForm") BaseRequestForm baseRequestForm,
				@ApiParam(required = false, name = "responsibleDeptId", value = "责任部门ID")
@RequestParam(value="responsibleDeptId",required=false) Integer responsibleDeptId){
		JsonResult<HseProblemCreatUiDataResult> result = new
JsonResult<HseProblemCreatUiDataResult>();
		HseProblemCreatUiDataResult hseProblemCreatUiResult = null;

		try{

			hseProblemCreatUiResult =
hseProblemApiService.getHseProblemCreatUiResult(responsibleDeptId);
		}catch(Exception e){
			e.printStackTrace();
			result.setMessage("操作异常:"+e.getMessage());
			result.setResponseResult(hseProblemCreatUiResult);
			return result;
		}

		result.setResponseResult(hseProblemCreatUiResult);
		return result;

	}

	@ResponseBody
	@RequestMapping(value = "create", method = RequestMethod.POST)
	@ApiOperation(value = "创建安全文明问题", notes = "用于创建安全文明问题")
	public JsonResult<Boolean> create(HttpServletRequest request, HttpServletResponse response,
			@ModelAttribute("requestForm") HseProblemRequestForm
hseProblemRequestForm,@RequestParam(value="file",required=false) CommonsMultipartFile[] files){
		JsonResult<Boolean> result = new JsonResult<Boolean>();
		Boolean status = false ;
		Integer loginId = hseProblemRequestForm.getLoginId();
		if (null == loginId || loginId.intValue()==0) {
			result.setCode("-1001");
			result.setMessage("Error/s occurred, loginId is not available");
			return result;
		}
		if(!CommonUtility.isNonEmpty(hseProblemRequestForm.getUnit())){
			result.setCode("-1001");
			result.setMessage("Error/s occurred, Unit is not available");
			return result;
```

```
		}
		if(!CommonUtility.isNonEmpty(hseProblemRequestForm.getWrokshop())){
			result.setCode("-1001");
			result.setMessage("Error/s occurred, Wrokshop is not available");
			return result;
		}
		if(!CommonUtility.isNonEmpty(hseProblemRequestForm.getEleration())){
			result.setCode("-1001");
			result.setMessage("Error/s occurred, Eleration is not available");
			return result;
		}
		if(!CommonUtility.isNonEmpty(hseProblemRequestForm.getRoomno())){
			result.setCode("-1001");
			result.setMessage("Error/s occurred, Roomno is not available");
			return result;
		}
		try{
			status = hseProblemApiService.createHseProblem(hseProblemRequestForm,
files,request);

		}catch(Exception e){

			e.printStackTrace();
			result.setMessage("操作异常:"+e.getMessage());
			result.setResponseResult(false);
			return result;
		}

		if(!status){
			result.setCode("-1001");
			result.setMessage("create problem error!");
			return result;
		}
		result.setResponseResult(status);
		return result;
	}

	@ResponseBody
	@RequestMapping(value = "createFile", method = RequestMethod.POST)
	@ApiOperation(value = "创建安全文明问题", notes = "用于创建安全文明问题")
	public JsonResult<Boolean> createFile(HttpServletRequest request, HttpServletResponse
response,
			@RequestParam(value="file") CommonsMultipartFile[] files){
		JsonResult<Boolean> result = new JsonResult<Boolean>();
		Boolean status = false ;

		if(files==null){
			result.setCode("-1001");
			result.setMessage("Error/s occurred,files is not available");
			return result;
		}
		try{
			if(files!=null) {
				testmoreUpload(files,HseProblemFileType.before.toString());
			}

		}catch(Exception e){

			e.printStackTrace();
			result.setMessage("操作异常:"+e.getMessage());
```

```
                result.setResponseResult(false);
                return result;
            }
            result.setResponseResult(status);
            return result;
        }
        /**
         * 文件上传
         * @param problemId
         * @param files
         * @param request
         * @return
         */
        private boolean testmoreUpload(CommonsMultipartFile files[], String type) {
            // 上传位置  设定文件保存的目录
            String filePath = ContextPath.fileBasePath+FileUtils.questionFilePath;
            List<FileInfo> fileInfos = UploadUtil.moreUpload(filePath, "utf-8", true, files);
            if (fileInfos != null && fileInfos.size() > 0) {
                for (FileInfo fileInfo : fileInfos) {
                    HseProblemFile file = new HseProblemFile();
                    file.setFilepath(fileInfo.getNewFilename());
                    file.setUploadtime(Calendar.getInstance().getTime());
                    file.setProblemId(1);
                    if(fileInfo.getFilename()!=null && fileInfo.getFilename().length()>50){
                        file.setFilename(fileInfo.getFilename().substring(0, 50));
                    }else{
                        file.setFilename(fileInfo.getFilename());
                    }
                    file.setFiletype(type);
                    hseProblemFileService.add(file);
                }
            }
            return true;
        }


        @ResponseBody
        @RequestMapping(value = "problemList", method = RequestMethod.GET)
        @ApiOperation(value = "查询安全文明问题分页列表", notes = "用于查询各状态安全文明问题列表,
问题进度状态：need_handle:待处理;renovating:整改中;need_check:待审查;finish:已完成;none:不需要处理")
        public JsonResult<HseProblemPageDataResult> problemList(HttpServletRequest request,
HttpServletResponse response,
                @ModelAttribute("requestForm") HseProblemListRequestFormByStatus
hseProblemRequestForm){
            JsonResult<HseProblemPageDataResult> result = new
JsonResult<HseProblemPageDataResult>();
            HseProblemPageDataResult hseProblemPageDataResult = new
HseProblemPageDataResult();
            Page<HseProblem> page = new Page<HseProblem>();
            String problemStatus = null;
            String solveStatus = null;
            String problemType = null;
            //问题类型判断
            if(hseProblemRequestForm.getProbelmType()!=null){
                problemType = hseProblemRequestForm.getProbelmType().toString();
            }
            //问题状态判断
            if(hseProblemRequestForm.getProblemStatus()!=null){
                problemStatus = hseProblemRequestForm.getProblemStatus().toString();
```

```
                }
                //判断处理情况是否为空
                if(hseProblemRequestForm.getProblemSolveStatus()!=null){
                        solveStatus = hseProblemRequestForm.getProblemSolveStatus().toString();
                }

                if (null != hseProblemRequestForm.getPagenum() && null !=
hseProblemRequestForm.getPagesize()) {
                        page.setPageNum(hseProblemRequestForm.getPagenum());
                        page.setPagesize(hseProblemRequestForm.getPagesize());
                }
                try{
                        if(hseProblemRequestForm.getLoginId()!=null){
                                User user =
userService.findUserById(hseProblemRequestForm.getLoginId());
                                if(user == null){
                                        result.setCode("-1001");
                                        result.setMessage("用户不存在！");
                                        return result;
                                }

                                if(HseProbelmTypeEnum.mine.toString().equals(problemType)){
                                        hseProblemPageDataResult =
hseProblemApiService.getHseProblemResultByUser(page, problemStatus, user);
                                }else{
                                        hseProblemPageDataResult =
hseProblemApiService.getHseProblemPageDataResult(page, problemStatus,solveStatus,user);
                                }

                        }
                }catch(Exception e){
                        e.printStackTrace();
                        result.setCode("-1001");
                        result.setMessage("操作异常："+e.getMessage());
                }
                result.setResponseResult(hseProblemPageDataResult);
                return result;
        }

        @ResponseBody
        @RequestMapping(value = "problem/{id}", method = RequestMethod.GET)
        @ApiOperation(value = "查询安全文明问题", notes = "用于查询单个安全文明问题")
        public JsonResult<HseProblemResult> problemListOne(HttpServletRequest request,
HttpServletResponse response,
                        @ModelAttribute("requestForm") BaseRequestForm
hpRequestForm,@PathVariable("id") Integer id) throws Exception {
                JsonResult<HseProblemResult> result = new JsonResult<HseProblemResult>();
                HseProblemResult hpResult = null;
                HseProblem hp = hseProblemService.findById(id);
                if(hp != null){
                        hpResult = hseProblemApiService.transferForList(hp, null);
                        result.setResponseResult(hpResult);
                }else{
                        result.setCode("-1001");
                        result.setMessage("查询的安全问题不存在！");
                }

                return result;
        }
```

```
	@ResponseBody
	@RequestMapping(value = "assign", method = RequestMethod.POST)
	@ApiOperation(value = "HSE执行分派", notes = "用于HSE分派整改任务,responsibleTeamId责任班组id，problemId问题id")
	public JsonResult<Boolean> assign (HttpServletRequest request, HttpServletResponse response,
			@ModelAttribute("requestForm") HseProblemAssignRequestForm hseProblemAssignRequestForm){
		JsonResult<Boolean> result = new JsonResult<Boolean>();
		boolean status = false;
		Integer loginId = null;
		if(hseProblemAssignRequestForm.getLoginId() != null){
			loginId = hseProblemAssignRequestForm.getLoginId();
		}
		User loginUser = userService.findUserById(loginId);
		if(loginUser == null){
			result.setCode("-1000");
			result.setMessage("当前用户不存在！");
			return result;
		}
		try{
			//登录用户是否是HSE部门成员
			if(!userService.isDepartmentOf(loginUser, "HSEDepartment")){
				result.setCode("-1000");
				result.setMessage("当前用户不是HSE部门成员，无法执行该操作！");
				return result;
			}
		}catch(Exception e){
			log.error("something error");
			e.printStackTrace();
			result.setMessage("操作异常:"+e.getMessage());
			result.setResponseResult(false);
			return result;
		}
		if(hseProblemAssignRequestForm.getProblemId() == null){
			result.setCode("-1000");
			result.setMessage("必要参数problemID为空！");
			return result;
		}

		status = hseProblemApiService.hseAssignTasks(hseProblemAssignRequestForm);
		if(!status){
			result.setCode("-1001");
			result.setMessage("执行分派出错！");
		}
		result.setResponseResult(status);
		return result;

	}

	@ResponseBody
	@RequestMapping(value = "unAssign", method = RequestMethod.POST)
	@ApiOperation(value = "HSE执行不需要处理问题", notes = "用于HSE将问题设为不需要处理状态，problemId问题id")
	public JsonResult<Boolean> unAssign(HttpServletRequest request, HttpServletResponse response,
```

```
                @ApiParam(required = true, name = "problemId", value = "文明施工问题ID")
@RequestParam(value="problemId",required=true) Integer problemId,@ModelAttribute("requestForm")
BaseRequestForm baseRequestForm){
        JsonResult<Boolean> result = new JsonResult<Boolean>();
        boolean status = false;
        try{
                Integer loginId = null;
                if(baseRequestForm.getLoginId() != null){
                        loginId = baseRequestForm.getLoginId();
                }
                User loginUser = userService.findUserById(loginId);
                if(loginUser == null){
                        result.setCode("-1000");
                        result.setMessage("当前用户不存在！");
                        return result;
                }
                //登录用户是否是HSE部门成员
                if(!userService.isDepartmentOf(loginUser, "HSEDepartment")){
                        result.setCode("-1000");
                        result.setMessage("当前用户不是HSE部门成员，无法执行该操作！");
                        return result;
                }

                //将问题状态改为不需要处理状态
                HseProblem hseProblem = hseProblemService.findById(problemId);
                boolean isNeedHandle = false;
                if(hseProblem != null){
                        isNeedHandle =
hseProblem.getProblemStatus().equals(HseProblemStatusEnum.Need_Handle.toString());

                }else{
                        result.setCode("-1001");
                        result.setMessage("当前问题不存在！");
                        return result;
                }
                if(!isNeedHandle){
                        result.setCode("-1001");
                        result.setMessage("当前状态不允许关闭问题！");
                        return result;
                }
                hseProblem.setProblemStatus(HseProblemStatusEnum.None.toString());
                hseProblem.setProblemStep(HseProblemStep.isFinished.toString());//问题关闭
                hseProblem.setFinishDate(Calendar.getInstance().getTime());//问题关闭时间
                hseProblemService.update(hseProblem);

                //将HSE处理情况更新为已处理
                List<HseProblemSolve> hseProblemSolves
=hseProblemSolveService.findByProblenIdAndSolveStep(problemId,
HseProblemSolveStep.hseConfirm.toString());
                if(hseProblemSolves != null && !hseProblemSolves.isEmpty()){
                        HseProblemSolve hseProblemSolve = hseProblemSolves.get(0);
                        hseProblemSolve.setSolveUser(baseRequestForm.getLoginId());
                        hseProblemSolve.setSolveDate(Calendar.getInstance().getTime());

hseProblemSolve.setSolveStatus(HseProblemSolveStatusEnum.Done.toString());
                        hseProblemSolveService.update(hseProblemSolve);
                }
                status=true;
```

```
		}catch(Exception e){
			log.error("something error");
			e.getMessage();
		}
		if(!status){
			result.setCode("-1001");
			result.setMessage("操作失败！");
		}
		result.setResponseResult(status);
		return result;
	}

	@ResponseBody
	@RequestMapping(value = "submitRenovateResult", method = RequestMethod.POST)
	@ApiOperation(value = "提交整改结果", notes = "用于整改责任部门提交整改结果，problemId问题
id")
	public JsonResult<Boolean> submitRenovateResult(HttpServletRequest request,
HttpServletResponse response,
			@ModelAttribute("requestForm") HseProblemRenovateForm form,
			@RequestParam(value="file",required=false) CommonsMultipartFile[] files){
		JsonResult<Boolean> result = new JsonResult<Boolean>();
		boolean status = false;

		status = hseProblemApiService.processRenovateResult(form, files,request);
		if(!status){
			result.setCode("-1001");
			result.setMessage("提交结果失败！");
		}
		result.setResponseResult(status);
		return result;
	}

	@ResponseBody
	@RequestMapping(value = "checkRenovateResult", method = RequestMethod.POST)
	@ApiOperation(value = "HSE审查整改结果", notes = "用于HSE审查整改结果，problemId问题id,
checkResult：通过或者重新整改")
	public JsonResult<Boolean> checkRenovateResult(HttpServletRequest request,
HttpServletResponse response,
			@ApiParam(required = true, name = "problemId", value = "文明施工问题ID")
@RequestParam(value="problemId",required=true) Integer problemId,
			@ApiParam(required = true, name = "checkResult", value = "审查结果，1-通过，2-
不通过") @RequestParam(value="checkResult",required=true) Integer checkResult,
			@ModelAttribute("requestForm") BaseRequestForm baseRequestForm){
		JsonResult<Boolean> result = new JsonResult<Boolean>();
		Integer loginId = null;
		if(baseRequestForm.getLoginId() != null){
			loginId = baseRequestForm.getLoginId();
		}
		User loginUser = userService.findUserById(loginId);
		if(loginUser == null){
			result.setCode("-1000");
			result.setMessage("当前用户不存在！");
			return result;
		}
		//登录用户是否是HSE部门成员
		if(!userService.isDepartmentOf(loginUser, "HSEDepartment")){
			result.setCode("-1000");
			result.setMessage("当前用户不是HSE部门成员，无法执行该操作！");
```

```
                return result;
            }

            boolean status =
hseProblemApiService.checkRenovateResult(baseRequestForm,problemId,checkResult);
            if(!status){
                result.setCode("-1001");
                result.setMessage("操作失败！");
            }

            result.setResponseResult(status);
            return result;
        }


}
```

2.3 培训考试答题获取接口：ExamAPIController.java

```
package com.easycms.hd.api.exam;

import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import com.easycms.common.util.CommonUtility;
import com.easycms.hd.api.exam.request.ExamPaperRequestForm;
import com.easycms.hd.api.exam.response.ExamPaperResult;
import com.easycms.hd.api.exam.response.ExamScoreResult;
import com.easycms.hd.api.request.base.BasePagenationRequestForm;
import com.easycms.hd.api.service.ExamApiService;
import com.easycms.hd.exam.domain.ExamUserScore;
import com.easycms.hd.exam.enums.ExamStatus;
import com.easycms.hd.exam.service.ExamUserScoreService;
import com.easycms.hd.learning.domain.LearningFold;
import com.easycms.hd.learning.service.LearningFoldService;
import com.easycms.hd.response.JsonResult;
import com.wordnik.swagger.annotations.Api;
import com.wordnik.swagger.annotations.ApiOperation;

import lombok.extern.slf4j.Slf4j;

@Slf4j
@Controller
```

```
@RequestMapping("/hdxt/api/exam")
@Api(value = "ExamAPIController", description = "考试相关的api")
public class ExamAPIController {

	@Autowired
	private ExamApiService examApiService;
	@Autowired
	private ExamUserScoreService examUserScoreService;
	@Autowired
	private LearningFoldService learningFoldService;

	/**
	 * @param request
	 * @param response
	 * @return
	 * @throws Exception
	 */
	@ResponseBody
	@RequestMapping(value = "", method = RequestMethod.GET)
	@ApiOperation(value = "获取考试题目", notes = "获取考试题目")
	public JsonResult<ExamPaperResult> getTestPaper(HttpServletRequest request,
HttpServletResponse response,
			@ModelAttribute("form") ExamPaperRequestForm form){
		JsonResult<ExamPaperResult> result = new JsonResult<ExamPaperResult>();
		boolean status = false;
		if(!CommonUtility.isNonEmpty(form.getSection())){
			result.setCode("-1001");
			result.setMessage("Error/s occurred, Section is not available");
			return result;
		}
		if("JNPX".equals(form.getSection()) && !CommonUtility.isNonEmpty(form.getModule())){
			result.setCode("-1001");
			result.setMessage("Error/s occurred, Module is not available");
			return result;
		}
		Map<String,Object> res = new HashMap<>();
		ExamPaperResult examPaperResult = null;
		String msg = "";
		try {
			if("ZXPX".equals(form.getSection())){
				if(!CommonUtility.isNonEmpty(form.getModule())){
					result.setCode("-1001");
					result.setMessage("Error/s occurred, Module is not available");
					return result;
				}
				examPaperResult = examApiService.getAllExamPaper(form.getSection(),
form.getModule());
			}else{
				res = examApiService.generateTestPaper(form);
				if(res.isEmpty()){
					result.setCode("-1000");
					result.setMessage("服务器繁忙，稍后再试");
					return result;
				}
				examPaperResult = (ExamPaperResult) res.get("data");
				msg = (String) res.get("msg");
			}
		} catch (Exception e) {
			log.error(e.getMessage());
			e.printStackTrace();
```

```
                result.setCode("-1004");
                result.setMessage("服务器繁忙，稍后再试");
                return result;
            }

            if(examPaperResult != null){
                status = true;
            }else{
                result.setCode("-1000");
                result.setMessage(msg);
                return result;
            }

            //获取试题，考试开始，并记录考试基础信息，用户交卷时考试结束
            List<ExamUserScore> examing =
examUserScoreService.findByUserAndExamStatus(form.getLoginId(), ExamStatus.DOING.toString());
            ExamUserScore examUserScore = null;

            if(examing.isEmpty() || examing.size()==0){
                examUserScore = new ExamUserScore();
                examUserScore.setExamStatus(ExamStatus.DOING.toString());
            }else if(examing != null && examing.size() < 2){
                examUserScore = examing.get(0);
            }else{
                status = false;
            }

            if(status){
                examUserScore.setExamStartDate(new Date());
                examUserScore.setExamType(form.getSection());
                examUserScore.setUserId(form.getLoginId());
                LearningFold learningFold = null;
                if(form.getFold() != null){
                    Integer foldId = Integer.parseInt(form.getFold());
                    learningFold = learningFoldService.findById(foldId);
                }
                examUserScore.setLearningFold(learningFold);
                examUserScoreService.add(examUserScore);

            }else{
                result.setCode("-1000");
                result.setMessage("服务器繁忙，稍后再试");
                return result;
            }

            result.setResponseResult(examPaperResult);
            return result;
        }


        @ResponseBody
        @RequestMapping(value = "/examScore", method = RequestMethod.GET)
        @ApiOperation(value = "获取考试成绩", notes = "获取用户所有考试成绩")
        public JsonResult<ExamScoreResult> getTestScore(HttpServletRequest request,
HttpServletResponse response,
                @ModelAttribute("form") BasePagenationRequestForm form) throws Exception {
            JsonResult<ExamScoreResult> result = new JsonResult<ExamScoreResult>();
            if(form.getLoginId() == null){
                result.setCode("-1001");
```

```
                result.setMessage("Error/s occurred, LoginId is not available");
                return result;
            }
            ExamScoreResult examScoreResult = examApiService.getExamScore(form);
            if(examScoreResult != null){
                result.setResponseResult(examScoreResult);
            }else{
                result.setCode("-1000");
                result.setMessage("执行操作失败！");
            }
            return result;
        }
}
```

2.4 培训课件学习接口：LearningAPIController.java

```
package com.easycms.hd.api.learning;

import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import com.easycms.hd.api.learning.response.LearningResult;
import com.easycms.hd.api.learning.response.SectionDetailResult;
import com.easycms.hd.api.learning.response.SectionResult;
import com.easycms.hd.api.learning.service.LearningApiService;
import com.easycms.hd.api.request.base.BaseRequestForm;
import com.easycms.hd.learning.domain.LearningSection;
import com.easycms.hd.learning.service.LearningSectionService;
import com.easycms.hd.response.JsonResult;
import com.wordnik.swagger.annotations.Api;
import com.wordnik.swagger.annotations.ApiOperation;
import com.wordnik.swagger.annotations.ApiParam;

@Controller
@RequestMapping("/hdxt/api/learning")
@Api(value = "LearningAPIController", description = "培训管理相关的api")
public class LearningAPIController {
```

```
	@Autowired
	private LearningSectionService learningSectionService;

	@Autowired
	private LearningApiService learningApiService;

	/**
	 * @param request
	 * @param response
	 * @return
	 * @throws Exception
	 */
	@ResponseBody
	@RequestMapping(value = "/section", method = RequestMethod.GET)
	@ApiOperation(value = "培训页面主模块", notes = "培训管理页面主模块的获取")
	public JsonResult<List<SectionResult>> module(HttpServletRequest request, HttpServletResponse
response,
			@ModelAttribute("form") BaseRequestForm form) throws Exception {
		JsonResult<List<SectionResult>> result = new JsonResult<List<SectionResult>>();

		List<SectionResult> sectionResults = new ArrayList<SectionResult>();

		List<LearningSection> learningSections = learningSectionService.findByStatus("ACTIVE");

		sectionResults = learningSections.stream().map(sections -> {
			SectionResult sectionResult = new SectionResult();
			sectionResult.setName(sections.getName());
			sectionResult.setId(sections.getId());
			sectionResult.setType(sections.getType());
			return sectionResult;
		}).collect(Collectors.toList());

		result.setResponseResult(sectionResults);
		return result;
	}

	/**
	 * @param request
	 * @param response
	 * @return
	 * @throws Exception
	 */
	@ResponseBody
	@RequestMapping(value = "/section/{section}", method = RequestMethod.GET)
	@ApiOperation(value = "培训页面主模块下的分支", notes = "培训管理页面主模块分支的获取")
	public JsonResult<SectionDetailResult> section(HttpServletRequest request, HttpServletResponse
response,
			@ApiParam(value = "例如基础培训则传入JCPX",required=true)
@PathVariable("section") String section,
			@ModelAttribute("form") BaseRequestForm form) throws Exception {
		JsonResult<SectionDetailResult> result = new JsonResult<SectionDetailResult>();

		result.setResponseResult(learningApiService.getSection(section, form.getLoginId()));
		return result;
	}

	/**
	 * @param request
	 * @param response
	 * @return
```

```
     * @throws Exception
     */
    @ResponseBody
    @RequestMapping(value = "/detail/{id}", method = RequestMethod.GET)
    @ApiOperation(value = "培训详细信息", notes = "培训管理详细页面的获取")
    public JsonResult<LearningResult> detail(HttpServletRequest request, HttpServletResponse
response,
                                             @ApiParam(value = "培训课程的ID",required=true) @PathVariable("id") Integer id,
                                             @ModelAttribute("form") BaseRequestForm form) throws Exception {
        JsonResult<LearningResult> result = new JsonResult<LearningResult>();

        result.setResponseResult(learningApiService.getDetail(id));
        return result;
    }
}
```

3）代码接口编制规则

安全施工问题状态：

problemStatus	字段
‘Need_Handle'	待处理
‘Renovating'	整改中
‘Need_Check'	待审核
‘None'	不需处理
‘Finish’	已完成

用户身份判断：

roleType	字段
‘monitor'	班长
‘team’	组长
‘captain’	队长
'member'	成员

八、 系统配置

8.1、配置原则

能良好地为移动管理平台系统服务，便于开发。

1） 支持系统中人员培训与安全培训记录 <= 1万份，所包含主信息总数 <= 20万条

2）单条数据文件附件上传 <=2M。

3）人员基础信息<= 1万条。

8.2、硬件配置

后台服务器：

1）16G内存

2）2T 硬盘

8.3、软件配置

1）HBuilder 移动端开发工具

2）SourceTree 版本控制系统工具

3）Eclipse 后台开发工具

4）Navicat Premium 数据库管理工具

5）MySQL 关系型数据库管理系统

九、 关键技术

9.1 关键技术的提出

1）混合开发

2）即时聊天

3）推送

4） 原生与H5之间的数据通信

9.2 关键技术的实现方案

1）混合开发

1.利用HBuilder 开发移动App项目；

2.使用MUI框架开发UI

HBuilder是DCloud（数字天堂）推出一款支持HTML5的Web开发IDE。“快，是HBuilder的最大优势，通过完整的语法提示和代码输入法、代码块及很多配套，HBuilder能大幅提升HTML、js、css的开发效率。

2）即时聊天

1.集成腾讯云通信IM

云通信（Instant Messaging）承载亿级 QQ 用户即时通信技术，数十年技术积累，腾讯云为您提供超乎寻常即时通信聊天服务。针对开发者的不同阶段需求及不同场景，云通信提供了一系列解决方案，包括： Android/iOS/Windows/Web 的 SDK 组件、服务端集成接口、第三方回调接口等，利用这些组件，可以在应用中构建自己的即时通信产品，解决开发者面临的高并发、高可用性的一系列问题。

3）推送

集成个推开放平台；

个推推送提供移动智能终端消息推送技术方案，通过高效稳定推送SDK，使APP快速集成云推送功能，免去自行开发成本，有效提升产品活跃度和用户黏性。

个推推送核心技术于2012年10月正式对外开放，并于2014年5月率先推出“智能精准推送“方案，提供基于用户属性的标签分析，以及A/B分组测试推送功能，帮助找到APP用户中最精准的人群。

2015年4月，个推推送发布 “应景推送”，可根据大数据分析人群属性，同时利用LBS地理围栏技术，实现消息的精准触发。

4） 原生与H5之间的数据通信

在编写扩展插件的JS时需要调用Javascript Plugin Bridge的API用来完成对Native层代码的调用和运行结果的返回。编写JS API 处理Native与H5之间的数据通信。

十、 实现后的功能流程

详情图示功能流程，请参考应用操作图文手册；

10.1 账户登录流程

步骤1：登录页点击账户选择框 —> 选择登录账户 —> 确定

10.2 群聊对讲机功能流程

1）创建群聊：

步骤1：群聊列表页—> 点击发起群聊 —> 选择添加初始群聊联系人 —> 点击完成 —>输入群名称— > 确定

2）修改群名称：

步骤2：点击进入群聊聊天室 —> 选择右上角齿轮设置按钮 —> 进入群聊设置页面—>点击右上角编辑 —>修改群名称— > 点击右上角保存 —> 修改成功

3）删除群成员：

步骤3：点击进入群聊聊天室 —> 选择右上角齿轮设置按钮 —> 进入群聊设置页面—>点击右上角编辑 —>点击红色减号删除群成员— > 点击确认

4）解散群聊或者退出群聊：

两种情况：

1、当前用户为群聊发起人时，退出群聊将会解散群聊。

2、当前用户为普通群聊成员时，退出群聊为当前用户退出该群，群聊不会解散。

步骤4：点击进入群聊聊天室 —> 选择右上角齿轮设置按钮—> 进入群聊设置页面—>退出群聊

5）对讲机：

步骤5：对讲机联系人列表 —> 选择联系人—> 进入对讲语音框 —>长按 “按住说话”按钮进行语音录制 —> 松开“按住说话”按钮即发送语音对讲

ps：点击对讲语音框中历史语音记录可重听语音。

10.3 人员培训管理流程

1）学习培训流程：

步骤1：选择基础培训或技能培训或者专项培训，三选其一 —> 选择学习的培训课件 —> 进入培训课件详情进行学习—> 视频播放或图文阅读—> 培训完成

2）考试答题流程：

步骤2：选择基础培训或技能培训或者专项培训，三选其一 —>选择要进行考试的类目（高亮为选择类目，默认选择为第一项） —> 点击右上角考试 —> 显示要考试的目录— > 开始考试 — > 进行考试答题 —> 右上角交卷 —>重新考试或再次培训

3）学习记录查询：

步骤3：学习记录 —>选择查看培训记录 —> 选择查看考试成绩

10.4 安全施工管理流程

附上人员施工流程图:

流程图：

详情步骤如下：

1）报告问题：

问题发起人（任何人）报告问题

步骤1：报告问题 —> 填写问题详情—> 确认提交问题

2）问题审核：

HSE部门权限账户才可进入问题审核模块,否则将出现权限不足提示

步骤2：问题审核 —> 待处理状态下 问题审核项—> 问题详情查看 —> 核实或者退回 —> 提交成功

ps:提交成功后问题出现在问题整改模块

3）问题整改：

拥有作业部门领导（作业班长）权限账户才可进入问题整改模块,否则将出现权限不足提示

步骤3：问题整改 —> 待处理状态下 问题—> 问题详情查看 —> 整改描述输入以及整改照片附近添加 —> 提交整改结果

ps:提交成功后问题重新出现在问题审核模块，待HSE部门人员验证

4）整改验证审核：

HSE部门权限账户才可进入问题审核模块,否则将出现权限不足提示

步骤4：问题审核 —> 待处理状态下 整改验证项—> 问题详情查看 —> 通过或者重新整改 —> 审核成功

ps:重新整改后问题重新出现在问题整改模块，状态变更为整改中；
通过后问题状态变更为已完成，可在问题查阅模块查询，HSE部门人员也可在问题审核模块已完成项查询

5）问题查阅：

HSE部门权限账户可进入问题审核模块查阅，任何人都可在问题查阅模块下查阅

步骤5：问题查阅 —> 我的问题或者查看所有问题—> 问题详情查看

ps: 我的问题只出现发起人为自己的问题，所有问题中按时间顺序显示出现所有问题