2023年全国职业院校技能大赛

高职组

“区块链技术应用”

赛项赛卷（3卷）

任

务

书

参赛队编号：

**背景描述**

新能源作为新兴领域，产业呈现碎片化与复杂化的特性，逐渐出现管理困难、供应链金融、可信监管与数据安全等一系列产业发展的阻碍。为了实现高质量发展和能源消纳利用，新能源产业发展需要与产业数字化趋势相融合。作为数字经济时代的前沿产物，区块链以其分布式存储、不可伪造和防篡改、透明可信、高可靠性和可追溯等优势，区块链技术为破解能源业存在的问题和创新发展模式提供了新的机遇，满足了新能源和分布式能源数字化的需求。

区块链与太阳能的融合，可以让新能源治理变得更加高效，让能源流转到急需的领域，尤其在能源交易过程中实现原子化交易。太阳能管理系统中除了资产与能源外，还包括了不同主体的管理，资产与太阳能源需要和用户深度绑定。基于区块链的太阳能管理系统可有效提升能源利用效率，同时也可有效降低能源浪费的现象。



## 模块一：区块链产品方案设计及系统运维（35分）

选手完成本模块的任务后，将任务中设计结果、运行代码、运行结果等截图粘贴至客户端桌面【区块链技术应用赛\重命名为工位号\模块一提交结果.docx】中对应的任务序号下。

### 任务1-1：区块链产品需求分析与方案设计

本任务需要依据项目背景完成需求分析与方案设计，具体要求如下:

1. 根据项目背景描述，将系统分为业务平台和支撑平台两个部分，使用工具展现项目系统架构图；
2. 根据项目背景描述内容，设计总体功能结构图；
3. 根据项目背景描述内容，设计并形成“系统概要设计说明书.doc”，具体要求如下：
   1. 做好概要设计，包括总体设计、基本设计概念和处理流程、结构等基本内容；
   2. 完善说明书中接口说明部分内容。

完成后提交“新能源管理系统概要设计说明书.doc”文档至【区块链技术应用赛\重命名为工位号】下。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 新能源管理系统概要设计说明书1．引言1.1编写目的 本概要说明书按照《需求规格说明书》的功能需求，设计本系统的软件结构，供后续软件过程(详细设计等)做依据，预期读者为后续软件过程的设计、开发人员。 2. 总体设计2.1需求规定 按照新能源系统的需求规定，用户可以通过本系统实现能源管理与交易、新能源资产管理与交易、用户管理等功能。 2.1.2输入输出要求 参照需求文档的说明 2.2运行环境 本系统必须运行在局域网或互联网中 2.2.1硬件设备 Web服务器:数量1台  数据库服务器：数量1台  客户端PC机：数量若干台，按需要增加 2.2.2支持架构 客户端  服务器端  后台数据库 2.3基本设计概念和处理流程 本系统软件部分可划分为浏览器页面、服务器端。用户通过浏览器访问相关页面，输入业务请求，发送到服务器端，调用业务系统接口处理后，把业务处理应答发送给客户端用户。  //请将任务1-1第1小题的答案复制至此处; 2.3.1 登录业务功能模块 //在这里补充 2.3.2 能源交易功能模块 //在这里补充 2.4结构2.4.1 总体框图 //请将任务1-1第2小题的答案复制至此处;     2.4.2 功能模块接口及说明 由于在后续设计中，对各功能模块的每个子功能都要详细的加以描述，因此恰当的标识符将有助于文档的阅读和代码的编写。  命名规则：  数据库表名命名规则  由前缀和实际名字组成。前缀：使用小写字母tb，表示表。实际名字采用其英文缩写，且首字母要大写。  数据库表中字段命名规则  根据其中文名称直接翻译成英文，其首字母要大写；若为多个词的组合，则每个词的首字母都要大写。  模块标识符命名规则  应本着便于理解和尽量简洁的原则来命名标识符。在本系统中，各模块均以每个词英文的前几个字母来命名，模块下的子功能也以其英文简写来命名,且每个词首字母均为大写。  程序中变量命名规则  每个变量都要以其所在的模块的为前缀，该前缀可以是该模块标识符中大写字母的组合，前缀后面加下划线“\_”。后面的字母小写，并且不能包含JAVA的关键字，如果变量是两个单词组成，第二个单词的首字母要大写。  程序中函数命名规则  每个函数名都要以其所在的模块的为前缀，该前缀可以是该模块标识符中大写字母的组合，前缀后面加下划线“\_”。首字母小写，多个单词的函数名，从第二个单词开始单词首字母大写。  //在下方各接口功能处补充完整请求参数、相应参数内容 2.4.2.1 用户注册 1.功能描述   |  |  | | --- | --- | | 标识符 | Register | | 功能 | 使用用户地址、角色、密码注册 |     2. 接口功能   |  |  |  | | --- | --- | --- | | 请求路由 | /Register | | | 请求方法 | POST | | | 请求参数 | | | | 参数名 | 类型 | 说明 | |  |  |  | |  |  |  | | 响应参数 | | | | 参数名 | 类型 | 说明 | |  |  |  | |  |  |  |    2.4.2.2 用户信息查询 1.功能描述   |  |  | | --- | --- | | 标识符 | userInfo | | 功能 | 通过指定用户地址获取详细信息 |     2. 接口功能   |  |  |  | | --- | --- | --- | | 请求路由 | /userInfo | | | 请求方法 | POST | | | 请求参数 | | | | 参数名 | 类型 | 说明 | |  |  |  | |  |  |  | | 响应参数 | | | | 参数名 | 类型 | 说明 | |  |  |  | |  |  |  |    2.4.2.3 创建资产 1. 功能描述   |  |  | | --- | --- | | 标识符 | SPU\_insert | | 功能 | 新增资产 |     2. 接口功能   |  |  |  | | --- | --- | --- | | 请求路由 | /SPU\_insert | | | 请求方法 | POST | | | 请求参数 | | | | 参数名 | 类型 | 说明 | |  |  |  | |  |  |  | | 响应参数 | | | | 参数名 | 类型 | 说明 | |  |  |  | |  |  |  |    2.4.2.4 资产交易 1. 功能描述   |  |  | | --- | --- | | 标识符 | SPU\_transfer | | 功能 | 资产交易 |   2. 接口功能   |  |  |  | | --- | --- | --- | | 请求路由 | distribute | | | 请求方法 | POST | | | 请求参数 | | | | 参数名 | 类型 | 说明 | |  |  |  | |  |  |  | | 响应参数 | | | | 参数名 | 类型 | 说明 | |  |  |  | |  |  |  |  2.4.2.5 能源交易 1. 功能描述   |  |  | | --- | --- | | 标识符 | Energy\_transfer | | 功能 | 能源交易 |   2. 接口功能   |  |  |  | | --- | --- | --- | | 请求路由 | /Energy\_transfer | | | 请求方法 | POST | | | 请求参数 | | | | 参数名 | 类型 | 说明 | |  |  |  | |  |  |  | | 响应参数 | | | | 参数名 | 类型 | 说明 | |  |  |  | |  |  |  |  3．系统出错处理设计3.1出错信息 服务器端可能有Java的异常产生，可用Java中的异常处理方法try…catch处理,并记录日志，以备事后分析查找原因。   3.2补救措施 采用双服务器热备份的方法，一旦服务器崩溃，立即切换到另一台服务器运行。 3.3系统维护设计 保留本系统的一系列文档，加适当注释，以备维护者维护。每次修改/维护，都做记录。 |

### 任务1-2：区块链系统部署与运维

围绕新能源管理区块链平台部署与运维需求，进行项目相关系统、节点以及管理工具的部署工作。通过监控工具完成对网络、节点服务的监控。最终基于业务需求规范，完成系统日志、网络参数、节点服务等系统结构的维护，具体要求如下：

1. 根据参数与端口设置要求，部署区块链系统并验证；
2. 根据参数与端口设置要求，部署区块链网络管理平台并验证；
3. 基于区块链系统相关管理平台，按照任务指南实施系统运维工作并验证；
4. 基于区块链系统相关监管工具，按照任务指南对区块链系统进行监管。

**子任务1-2-1： 搭建区块链系统并验证**

区块链系统中包括3个独立组织以及6个节点，网络拓扑为星型网络，并在联盟链中需实现2个数据隔离场景（group1、group2），节点可根据具体业务加入不同的业务场景。具体工作内容如下：

1. 使用给定工具（地址“/root/tools”），配置区块链系统；
2. 启动区块链系统全部节点并验证启动情况；
3. 检查节点的连接状态以及所属场景的共识状态。

**子任务1-2-2：搭建区块链系统管理平台并验证**

基于给定服务器环境、软件以及工具（地址“/root/tools”），构建区块链系统相关控制台，并通过控制台实现相关系统管理，具体工作内容如下：

1. 使用给定工具，配置控制台证书；
2. 使用给定工具，以pem加密方式创建3个区块链账户（Account1、Account2、Account3）；
3. 使用给定工具，通过Account1登录控制台，并查看区块高度；
4. 使用给定工具，通过Account2登录控制台，并查看共识状态；
5. 使用控制台部署HelloWorld.sol智能合约并调用set和get方法；
6. 使用控制台查看区块链中的情况高度。

**子任务1-2-3：区块链节点运维**

基于已完成的区块链系统与管理平台搭建工作，开展区块链节点的加入与退出运维工作，具体内容如下：

1. 获取node3的节点ID；
2. 启动控制台，在group1中移除node3节点；
3. 检查node3节点共识状态日志。

**子任务1-2-4：区块链网络运维**

基于已完成的区块链系统与管理平台搭建工作，开展节点运维工作，具体内容如下：

1. 生成新节点(node4)，启动并检查；
2. 修改新节点配置，并查看节点的nodeid；
3. 将新节点作为观察节点加入group1当中，并检查是否加入成功；
4. 将区块打包最大交易数量设为2000笔。

### 任务1-3：区块链系统测试

设计对区块链系统的测试流程；结合实际业务需求，调用部署的智能合约中进行系统测试、性能测试等；根据业务需求，分析并且修复给定智能合约中的安全漏洞。利用模拟业务和测试工具来完成对区块链系统服务数据的测试。

1. 使用命令启动区块链系统验证启动情况；
2. 创建user3用户，部署新能源管理合约，调用合约注册功能（account\_register），将user3进行注册为“用户”并获取正确返回；
3. 调用新能源管理合约judgeUserRole功能，验证注册情况；
4. 参照工程项目（地址：“/root/projects”）使用测试工具对新能源管理系统智能合约太阳能板能源信息新增(Energy\_insert)功能进行压力测试。具体要求如下：
   1. 提供核心测试代码；
   2. 设置txNumber=10，tps=1，所有测试通过率为100%。
5. 智能合约安全漏洞测试

有如下问题智能合约：

pragma solidity ^0.8.3;

contract EtherGame {

uint public targetAmount = 7 ether;

address public winner;

function deposit() public payable {

require(msg.value == 1 ether, "You can only send 1 Ether");

uint balance = address(this).balance;

require(balance <= targetAmount, "Game is over");

if (balance == targetAmount) {

winner = msg.sender;

}

}

function claimReward() public {

require(msg.sender == winner, "Not winner");

(bool sent, ) = msg.sender.call{value: address(this).balance}("");

require(sent, "Failed to send Ether");

}

}

contract Attack {

EtherGame etherGame;

constructor(EtherGame \_etherGame) {

etherGame = EtherGame(\_etherGame);

}

function attack() public payable {

address payable addr = payable(address(etherGame));

selfdestruct(addr);

}

}

* 1. 分析智能合约中存在问题，并说明危害；
  2. 根据代码文件，编写测试用例，复现智能合约中存在的漏洞；
  3. 创建新的智能合约，修复其中问题，说明修复内容并测试。

## 模块二：智能合约开发与测试（30分）

选手完成本模块的任务后，将任务中设计结果、运行代码、运行结果等截图粘贴至客户端桌面【区块链技术应用赛\重命名为工位号\模块二提交结果.docx】中对应的任务序号下。

### 任务2-1：智能合约设计

**子任务2-1-1 编写新能源智能合约功能需求文档。**

根据区块链新能源管理项目背景和需求分析，编写该区块链新能源管理的智能合约功能需求文档。

**子任务2-1-2 完成区块链新能源管理智能合约的设计。**

1. 设计区块链新能源管理智能合约接口，画出区块链新能源管理智能合约的角色UML用例图；

2. 以图文结合的方式描述智能合约各参与实体间的关系。

### 任务2-2：智能合约开发

使用Solidity语言完成智能合约开发、部署和调用，要求如下：

**子任务2-2-1：太阳能板管理接口编码**

1. 根据文档要求，编写太阳能板新增接口功能，必须将新增太阳能板数据存入指定表中，在存储完成后需触发后事件并返回存储与否的标识；

2. 根据文档要求，编写太阳能板修改接口，必须通过指定表修改完成数据更新，在完成更新后需触发事件并返回更新与否的标识。

**子任务2-2-2：能源管理接口编码**

1. 根据文档要求，编写能源新增接口功能，必须将新增能源数据存入指定表中，在存储完成后需触发后事件并返回存储与否的标识；

2. 根据文档要求，编写能源修改接口，必须通过指定表修改完成数据更新，在完成更新后需触发事件并返回更新与否的标识。

**子任务2-2-3：合约部署和调用**

1. 解决代码错误和警告，正确编译并部署合约，成功获取部署的合约地址和ABI；

2. 调用太阳能板查询合约接口，完整验证业务流程；

3. 调用能源查询合约接口，完整验证业务流程。

### 任务2-3：智能合约测试

根据已完成的智能合约，针对开发功能开展相关合约测试工作，具体工作内容如下：

1. 调用太阳能板查询合约接口，完整验证业务流程；

2. 调用能源查询合约接口，完整验证业务流程。

## 模块三：区块链应用系统开发（30分）

选手完成本模块的任务后，将任务中设计结果、运行代码、运行结果等截图粘贴至客户端桌面【区块链技术应用赛\重命名为工位号\模块三提交结果.docx】中对应的任务序号下。

### 任务3-1：区块链应用前端功能开发

1.请基于前端系统的开发模板，在注册组件Register.vue、中添加对应的逻辑代码，实现对前端的角色选择功能，并测试功能完整性，示例页面如下：



本任务的具体要求如下：

1. 需要填写的项有地址、角色、密码；
2. 地址项输入框默认内容为“请输入0x开头的用户地址”，若不满足则报错；
3. 角色使用单选按钮组件，可选项为“用户”和“电力公司”；
4. 密码项需验证长度大于6位，输入框默认内容为“请输入6位以上的密码”；
5. 页面需要有“返回”按钮，可以返回登录页面；
6. 注册成功后返回登录页面Login.js。

代码片段1：

<el-row style="height: 100%;">

<el-col :span="8" :offset="8" class="register-box">

<el-row>

<el-col :span="16" :offset="4">

<el-form :model="queryForm" ref="queryForm" label-width="100px">

<h2>注册页面</h2>

<el-form-item label="地址:" prop="address">

<input type="primary" 选手填写部分v-model=选手填写部分/>

</el-form-item>

<el-form-item label="角色:" prop="role">

选手填写部分

</el-form-item>

<el-form-item label="密码:" prop="password">

<input type="primary" 选手填写部分 v-model=选手填写部分

show-password/>

</el-form-item>

</el-form>

</el-col>

</el-row>

<el-row style="margin-bottom: 20px">

<el-col :span="16" :offset="4">

<el-button type="primary" 选手填写部分round>注册</el-button>

<el-button type="primary" 选手填写部分round>返回</el-button>

</el-col>

</el-row>

</el-col>

代码片段2：

goback: function () {

选手填写部分

}

代码片段3：

register: function() {

// 表单信息检查

选手填写部分 this.axios.post('/account/register', this.registerdata).then((response) => {

if (response.data.code == 选手填写部分) {

alert(选手填写部分)

this.$router.push('/login')

}else {

alert(`选手填写部分, ${response.data.data}`)

}

})

}

2.基于前端系统的开发模板，在登录组件Login.vue文件中添加对应的登录逻辑代码，实现对后端系统的登录功能，并测试功能完整性。

本任务的具体要求如下：

1. 需要填写的项有地址、密码，且密码以密文形式展现；
2. 地址填写栏默认内容为“请输入0x开头的用户地址”；
3. 页面需要有“注册”按钮，可以跳转注册页面；
4. 点击“登录”按钮时需要验证，地址栏内容需以”0x”开头，密码需大于6位；
5. 对登录密码进行简单加密，传输内容为原内容后增加字符串“pd”
6. 登录成功后进入首页，参考界面如下：



代码片段1：

<el-row style="height: 100%;">

<el-col :span="8" :offset="8" class="login-box">

<el-row >

<el-col :span="16" :offset="4">

<el-form label-width="100px">

<h2>电力能源交易系统登录</h2>

<el-form-item label="地址:">

<input type="primary" 选手填写部分v-model="logindata.address"/>

</el-form-item>

<el-form-item label="密码:">

<input 选手填写部分v-model=选手填写部分/>

</el-form-item>

</el-form>

</el-col>

</el-row>

<el-row style="height: 50px">

<el-col :span="16" :offset="4">

<el-button type="primary" 选手填写部分round>登录</el-button>

<el-button type="primary" 选手填写部分 round>注册</el-button>

</el-col>

</el-row>

代码片段2：

loginfun() {

// 表单信息检查

选手填写部分

console.log(this.logindata.password )

this.axios.post('/account/login', this.logindata).then((response) => {

if (response.data.code == 选手填写部分) {

this.$cookies.set("useraddress", response.data.data)

console.log(this.$cookies.get('useraddress'))

// this.uaddress = response.data.data

选手填写部分

}

})

}

代码片段3：

register: function () {

选手填写部分

}

3.基于前端系统的开发模板，编写个人产权组件MyAsset.vue，实现对个人能源的管理功能，并测试功能完整性

本任务具体要求如下：

1. 界面有明确的资产相关提示语；
2. 界面需要展现当前用户设备列表，无设备时要给出提示；
3. 对于每个列出的设备在操作栏增加“出售”按钮，可添加出售订单，点击后应有弹出框提示用户填写相关信息（设备编号、所属权、价格）用于出售，该弹出框有“确认”按钮，点击该按钮有结果提示。

代码片段1：

<el-row gutter="20" type="flex" justify="space-between">

<el-col style="margin-bottom:0px">

<el-page-header content="资产列表" @back="goBack"/>

</el-col>

<el-divider direction="horizontal " />

<el-col :span="2" style="margin-bottom:10px">

<el-button primary 选手填写部分>新增设备</el-button>

</el-col>

</el-row>

<el-row gutter="20" type="flex" justify="space-between">

<el-col>

<el-table :data="tableData" stripe border style="width: 100%">

<el-table-column type="selection" width="55"></el-table-column>

<el-table-column fixed prop="numid" label=选手填写部分width="150" show-overflow-tooltip />

<el-table-column prop="name" label=选手填写部分width="150" />

<el-table-column prop="actual\_Power" label=选手填写部分width="150" />

<el-table-column prop="rated\_Power" label=选手填写部分width="150" />

<el-table-column prop="input\_Time" label=选手填写部分width="100" />

<el-table-column prop="position" label=选手填写部分width="150" />

<el-table-column prop="price" label=选手填写部分width="120" />

<el-table-column prop="Ownership" label=选手填写部分 show-overflow-tooltip width="170" />

<el-table-column fixed="right" label=选手填写部分width="150">

<template #default>

<el-button link type="primary" size="small" 选手填写部分>出售</el-button>

</template>

</el-table-column>

</el-table>

代码片段2：

<el-dialog v-model="dialogFormVisible2" 选手填写部分>

<el-form>

<el-form-item label="设备编号">

<input v-model=选手填写部分/>

</el-form-item>

<el-form-item label="所属权">

<input v-model=选手填写部分/>

</el-form-item>

<el-form-item label="价格">

<el-input v-model=选手填写部分/>

</el-form-item>

</el-form>

<template #footer>

<span class="dialog-footer">

<el-button 选手填写部分>离开</el-button>

<el-button type="primary" 选手填写部分>确认</el-button>

</span>

</template>

</el-dialog>

代码片段3：

saller() {

this.axios.post(选手填写部分, this.createOrderPrice).then((res) => {

if (res.data.code == 选手填写部分){

alert('添加成功')

this.dialogFormVisible2 = 选手填写部分

this.listinfo()

}else('添加失败')

})

}

}

代码片段4：

goBack() {

this.$router.push('/')

}

// 新增

add() {

选手填写部分

}

handleClick() {

选手填写部分

}

getAssert(){

let \_Address = 选手填写部分

this.axios.get(`/get\_Address\_Assert?\_Address=${\_Address}`).then((res) => {

if (res.data.code == 200){

选手填写部分

}else{

alert(`失败,$(res.data.data)`)

}

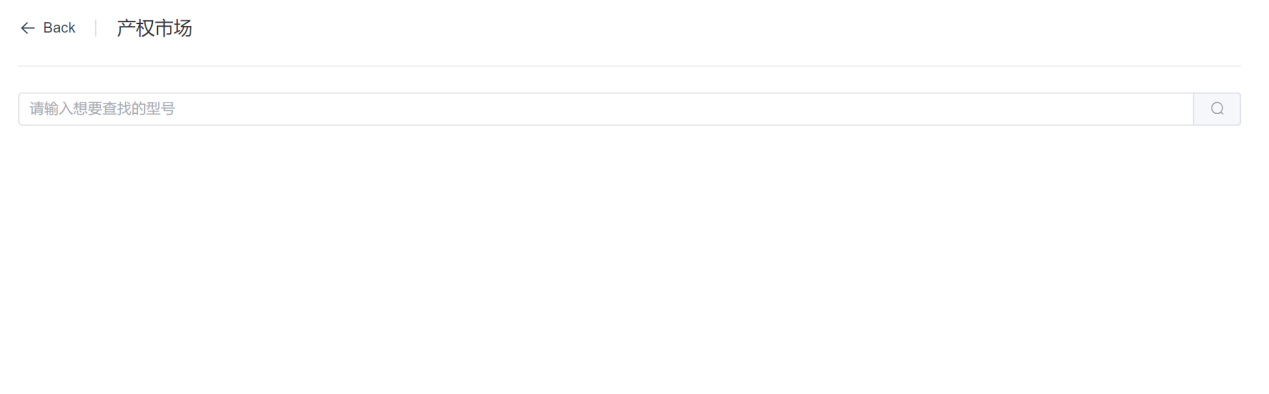
})

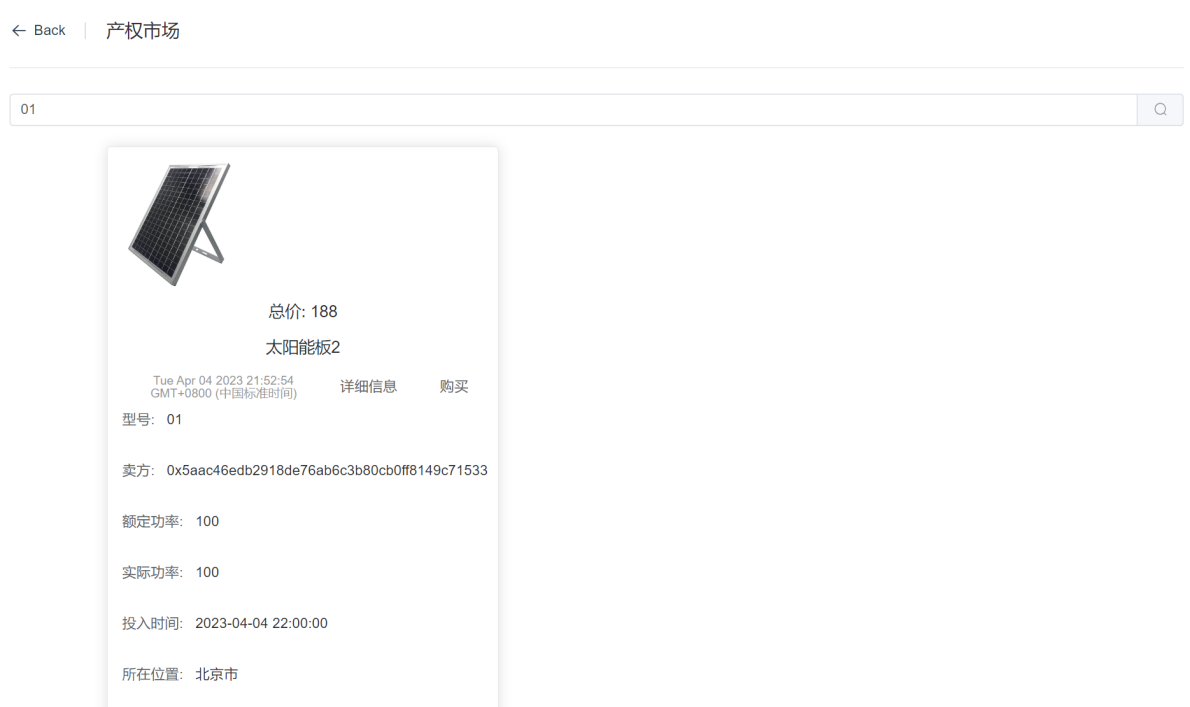
}

4.基于前端系统的开发模板，编写产权市场组件Market.vue文件，实现产权的查询和交易功能，并测试功能完整性。

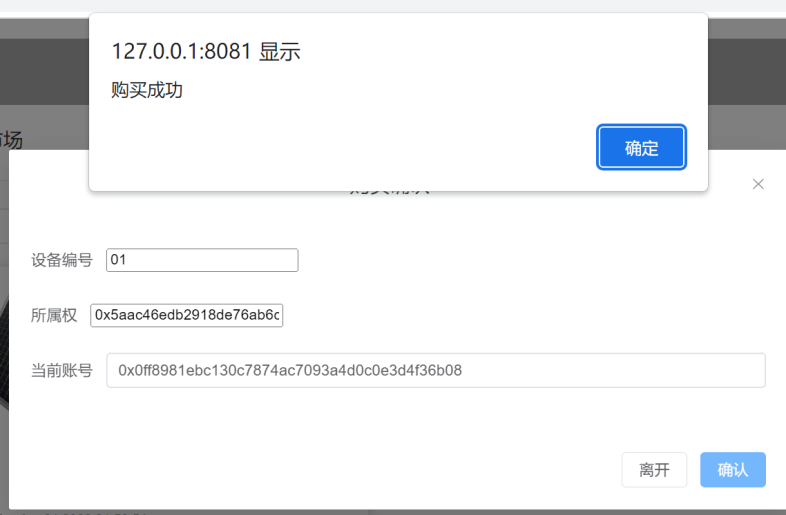
本任务具体要求如下：

1. 界面有明确的产权市场相关提示语；
2. 界面有输入框，提示用户输入产权型号，有搜索按钮可供搜索；
3. 搜索结果展示页面展示该产权的具体信息，包括型号、卖方、额定功率、实际功率、投入时间、所在位置，有“购买”按钮可进行购买操作；
4. 点击“购买按钮”，提示用户输入设备编号、所属权、当前账号，该弹出框有“确认”按钮，点击该按钮有结果提示，参考界面如下：









代码片段1：

<el-container style="height: 100% ">

<el-aside width="auto"><Aside></Aside></el-aside>

<el-container>

<el-header><Header></Header></el-header>

<el-main>

<el-row><el-col>

<el-page-header content=选手填写部分@back="goBack"/>

</el-col></el-row>

<el-divider direction="horizontal " />

代码片段2：

<el-row>

<el-input placeholder=选手填写部分 v-model="searchs" style="margin-bottom:20px">

<template #append>

<el-button 选手填写部分>

<el-icon><Search /></el-icon>

</el-button>

</template>

</el-input>

代码片段3：

<el-card :body-style="{ padding: '0px' }" style="margin-bottom: 20px">

<img

v-for="item2 in imgs"

:key="item2"

:src="item2.img"

class="image"/>

<span>

总价:

</span>

<span>

选手填写部分

</span>

<div style="padding: 14px">

<span>选手填写部分</span>

<div class="bottom">

<time class="time">{{ currentDate }}</time>

<el-button text @click="drawer = true">详细信息</el-button>

<el-button text 选手填写部分>购买</el-button>

</div>

<el-form size="mini">

<el-form-item label="型号:">

选手填写部分

</el-form-item>

<el-form-item label="卖方:">

选手填写部分

</el-form-item>

<el-form-item label="额定功率:">

选手填写部分

</el-form-item>

<el-form-item label="实际功率:">

选手填写部分 </el-form-item>

<el-form-item label="投入时间:">

选手填写部分

</el-form-item>

<el-form-item label="所在位置:">

选手填写部分

</el-form-item>

</el-form>

代码片段4：

<el-dialog v-model="transferorder" title="购买确认">

<el-form>

<el-form-item label="设备编号">

<input v-model=选手填写部分/>

</el-form-item>

<el-form-item label="所属权">

<input v-model=选手填写部分/>

</el-form-item>

<el-form-item label="当前账号">

<el-input v-model=选手填写部分/>

</el-form-item>

</el-form>

<template #footer>

<span class="dialog-footer">

<el-button @click="transferorder = false">离开</el-button>

<el-button type="primary" 选手填写部分>确认</el-button>

</span>

</template>

</el-dialog>

</div>

</el-card>

代码片段5：

transfer() {

this.axios.post(`/SSPU\_transfer`, this.owner).then((res) => {

if (res.data.code == 200){

alert(选手填写部分)

this.transferorder = false

}else{

alert(选手填写部分)

}

})

}

代码片段6：

getData() {

let \_numid = this.searchs

this.axios.get(`/get\_numid\_Spu?\_numid=${\_numid}`).then((res) => {

console.log(res.data.data)

if(res.data.code == 200 ){

选手填写部分

console.log(res.data.data)

}else{

alert(`失败,$(res.data.data)`)

}

})

}

}

5.请基于前端系统的开发模板，编写管理员的区块链浏览器组件Browser.vue文件，实现底层合约的管理，并测试功能完整性。

本任务具体要求如下：

1. 界面有明确的区块链浏览器相关提示语；
2. 展示页面展示合约名称、合约地址、部署时间、创建时间、修改时间；
3. 后端代码完成后截图。

参考界面如下：



代码片段1：

<el-page-header content=选手填写部分 @back="goBack" />

</el-col>

<el-divider direction="horizontal " />

<el-col>

<el-table

:data="contracts"

style="width: 100%">

<el-table-column

prop="合约名称"

label=选手填写部分

width="180">

</el-table-column>

<el-table-column

prop="合约地址"

label=选手填写部分

width="180">

</el-table-column>

<el-table-column

prop="部署时间"

label=选手填写部分>

</el-table-column>

<el-table-column

prop="创建时间"

label=选手填写部分>

</el-table-column>

<el-table-column

prop="修改时间"

label=选手填写部分>

</el-table-column>

代码片段2：

created() {

this.axios.get(`/list`).then((response) => {

if(response.data.result){

this.contracts = 选手填写部分

}

})

}

### 任务3-2：区块链应用后端功能开发

1.编写注册相关文件，实现注册接口，并测试功能完整性。具体要求如下：

1. 开发注册文件RegisterBO.java实现注册功能实体的定义包含登录页面所传递的三个变量：地址、角色、密码，并补充get、set方法；
2. 开发账户管理文件AccountService.java中注册部分，再次验证前端传值，合约的交互逻辑，返回值封装成为Result结果类；
3. 开发账户管理入口文件AccountController.java中注册部分，要求接口地址为/account/register。

RegisterBO.java：

代码片段1：

public class RegisterBO {

String 选手填写部分;

String 选手填写部分;

String 选手填写部分;

public String getAddress() {

选手填写部分

}

public void setAddress(选手填写部分) {

选手填写部分

}

public String getRole() {

选手填写部分

}

public void setRole(选手填写部分) {

选手填写部分

}

public String getPassword() {

选手填写部分

}

public void setPassword(选手填写部分) {

选手填写部分

}

}

AccountService.java：

代码片段1：

public Result<String> RegisterService(RegisterBO registerBO) {

List funcParam = new ArrayList();

funcParam.add选手填写部分;

funcParam.add选手填写部分;

funcParam.add选手填写部分;

Dict result = 选手填写部分;

JSONObject respBody = JSONUtil.parseObj(result.get("result"));

String data = (String)respBody.get("message");

if (data.equals("Success")) {

return Result.success("ok");

}else {

return Result.error(ResultVO.PARAM\_EMPTY);

}

}

AccountController.java：

代码片段1：

@Autowired

AccountService accountService;

@RequestMapping(选手填写部分, method = RequestMethod.POST)

public Result<String> register(@RequestBody RegisterBO registerBO) {

return 选手填写部分;

}

2.编写登录相关文件，实现登录接口，并测试功能完整性。本任务具体要求如下：

1. 开发注册文件实现功能实体的定义,包含登录页面所传递的两个变量：地址、密码并补充get、set方法；
2. 开发账户管理文件登录部分，实现与合约的交互逻辑，返回值封装成为Result结果类；
3. 开发账户管理登录部分，要求接口地址为/account/login。

LoginBO.java：

代码片段1：

public class LoginBO {

String 选手填写部分;

String 选手填写部分;

public String getAddress() {

选手填写部分

}

public void setAddress(选手填写部分) {

选手填写部分

}

public String getPassword() {

选手填写部分

}

public void setPassword(选手填写部分) {

选手填写部分

}

}

AccountService.java：

代码片段1：

public Result LoginService(LoginBO loginBO) {

List funcParam = new ArrayList();

funcParam.add(选手填写部分);

funcParam.add(选手填写部分);

Dict result = 选手填写部分;

JSONArray respBody = JSONUtil.parseArray(result.get("result"));

String data = (String)respBody.get(0);

if (data.equals("true")){

return Result.success(loginBO.getAddress());

}else {

return Result.error(ResultVO.PARAM\_EMPTY);

}

}

AccountController.java：

@RequestMapping(选手填写部分, method = RequestMethod.POST)

public Result<String> login(@RequestBody LoginBO loginBO) {

选手填写部分;

}

3.编写产权管理相关文件，实现产权管理的相关接口，并测试功能完整性。具体要求如下：

1. 开发实现功能实体的定义,包含变量用户地址、价格和产权编号,并补充get、set方法；
2. 开发产权出售部分，对前端的数据进行二次验证，要求用户地址为0x开头的数据串，实现与合约的交互逻辑，返回值封装成为Result结果类；
3. 开发产权出售部分，要求接口地址为/SPU\_saller，接口可接受请求方法为POST。

SPU\_sallerBO.java:

public class SPU\_sallerBO {

String 选手填写部分;

String 选手填写部分;

String 选手填写部分;

public String get\_numid() {

return 选手填写部分;

}

public void set\_numid(选手填写部分) {

选手填写部分;

}

public String get\_price() {

选手填写部分;

}

public void set\_price(选手填写部分) {

选手填写部分;

}

public String getAddress() {

选手填写部分;

}

public void setAddress(选手填写部分) {

选手填写部分;

}

}

NewEnergyService.java：

public Result<String> SPU\_sallerService(SPU\_sallerBO spu\_sallerBO) {

List funcParam = new ArrayList();

if(!spu\_sallerBO.getAddress().startsWith("0x")){

return Result.error(ResultVO.ADDRESS\_INVALID);

}

选手填写部分

Dict result = 选手填写部分;

JSONObject resBody = JSONUtil.parseObj(result.get("result"));

String data = (String) resBody.get("message");

if (data.equals("Success")) {

return Result.success("ok");

}else {

return Result.success("failed");

}

}

NewEnergyController.java：

@RequestMapping(选手填写部分, method = RequestMethod.POST)

public Result<String> SPU\_saller(@RequestBody SPU\_sallerBO spu\_sallerBO) {

return 选手填写部分;

}

4.编写产权交易相关文件，实现产权交易的相关接口，并测试功能完整性。具体要求如下：

1. 开发实现功能实体的定义,包含变量：设备编号、设备名称、实际功率、额定功率、结束时间、工作地点、单价、所属权,并补充get、set方法；
2. 开发实现功能实体的定义,包含变量：产权编号、价格、用户地址,并补充get、set方法；
3. 开发产权交易、查询部分，实现与合约的交互逻辑，返回值封装成为Result结果类；
4. 开发产权交易，要求接口地址为/SSPU\_transfer，请求方法为POST；产权查询部分，要求接口为/get\_numid\_Spu\_，请求方法为GET。

SpuVO.java:

public class SpuVO {

String 选手填写部分;

String 选手填写部分;

String 选手填写部分;

String 选手填写部分;

String 选手填写部分;

String 选手填写部分;

String 选手填写部分

String 选手填写部分;

public String get\_numid() {

选手填写部分

}

public void set\_numid( 选手填写部分) {

选手填写部分

}

public String getName() {

选手填写部分

}

public void setName( 选手填写部分) {

选手填写部分

}

public String getActual\_Power() {

选手填写部分

}

public void setActual\_Power( 选手填写部分) {

选手填写部分

}

public String getRated\_Power() {

选手填写部分

}

public void setRated\_Power( 选手填写部分) {

选手填写部分

}

public String getInput\_Time() {

选手填写部分

}

public void setInput\_Time( 选手填写部分) {

选手填写部分

}

public String getPosition() {

选手填写部分

}

public void setPosition( 选手填写部分) {

选手填写部分

}

public String getPrice() {

选手填写部分

}

public void setPrice( 选手填写部分) {

选手填写部分

}

public String getAddress() {

选手填写部分

}

public void setAddress( 选手填写部分) {

选手填写部分

}

}

NewEnergyService.java：

public Result<String> SPU\_transferService(SPU\_transferBO spu\_transferBO) {

List funcParam = new ArrayList();

选手填写部分;

JSONObject resBody = JSONUtil.parseObj(result.get("result"));

String data = (String) resBody.get("message");

if (data.equals("Success")) {

return Result.success("ok");

}else {

return Result.success("failed");

}

}

public Result get\_numid\_SpuService(String \_numid) {

List funcParam = new ArrayList();

选手填写部分;

JSONArray resBody = JSONUtil.parseArray(result.get("result"));

JSONArray res = JSONUtil.parseArray(resBody.get(0));

JSONArray array = new JSONArray();

选手填写部分

return Result.success(array);

}

NewEnergyController.java:

@RequestMapping( 选手填写部分)

public Result<String> SSPU\_transfer(@RequestBody SPU\_transferBO spu\_transferBO) {

return 选手填写部分);

}

@RequestMapping( 选手填写部分)

public Result get\_numid\_Spu(@RequestParam("\_numid") String \_numid) {

return 选手填写部分;

}