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（1卷）

任

务

书

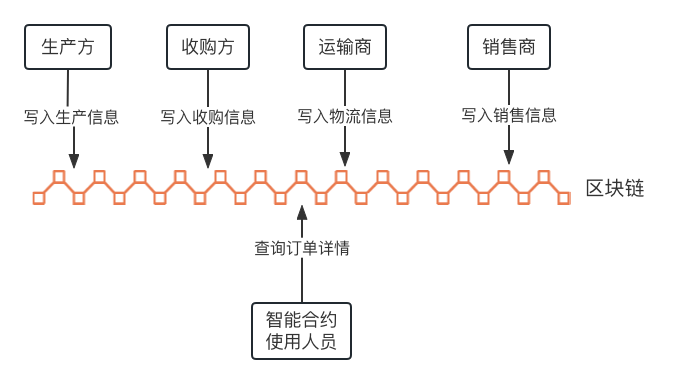
参赛队编号：

**背景描述**

随着消费需求的不断变化，消费者对食品安全的关注度越来越高，希望能参与食品供应链管理，让每个环节都透明化。但传统的供应链管理依靠纸张记录，保存数据具有随意性，消费者无法确认其真实性。此外，传统管理模式中心化，多数环节间信息流通不畅，影响供应链管理效率。因此，供应链管理面临效率和安全透明的双重挑战，迫切需要有效变革，促进食品供应链更高效、透明和安全。

从技术层面来看，区块链技术具有去中心化、公开透明、不可篡改等优点，可解决食品供应链短板，与现行管理相结合，不仅可提升透明度，还可提升管理效率。

通过构建基于区块链技术的食品安全溯源平台，有效将包括生产日期、生产产地、生产商、流通企业等食品安全溯源相关信息通过区块链去中心化的方式存储，有效保证了数据真实以及不可篡改。另一方面，借助区块链智能合约技术，灵活设计食品安全溯源相关业务，在确保数据安全的前提下实现透明公开，在此基础上引入监管机制有效保证业务良性开展。



## 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. 依据给定区块链食品溯源系统的业务架构图，对考题进行业务分析，尽可能全面的考虑一个业务系统所需要的模块，使用Visio或思维导图工具展现本系统的基本设计概念和处理流程，要求分为区块链食品溯源业务平台和支撑平台两个部分；

2. 根据食品溯源系统设计，在系统中分别包括用户管理以及订单追溯功能，在订单追溯中需要包括生产商、分销商、经销商订单的追溯管理功能。根据描述，设计区块链系统的总体功能概览图；

3. 整合上述设计内容，完善“区块链食品溯源系统概要设计说明书.doc”，具体工作内容如下：

（1）将设计内容根据说明书中模块需求进行补充，并完善概要设计说明书中的“需求概要”；

（2）完善说明书中接口说明部分内容。

完成后提交“区块链食品溯源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。

# 区块链食品溯源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

### 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。

系统框架图如下：

//将任务1-1第1小题结果复制至此；

## 2.3.1 登录业务功能模块

//在这里补充

#### 2.3.2 食品订单溯源功能模块

//在这里补充

## 2.4结构

### 2.4.1 总体框图

//将任务1-1第2小题结果复制至此；

### 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详细的描述，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：

1．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。

2．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。

3．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。

4．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。

5．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。

//补充填写请求参数、相应参数内容

#### 2.4.2.1 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 |
|  |  |  |
|  |  |  |

#### 2.4.2.2 生产订单创建

1. 功能描述

|  |  |
| --- | --- |
| 标识符 | produce |
| 功能 | 生产商创建食品溯源中生产订单相关信息 |

2. 接口功能

|  |  |  |
| --- | --- | --- |
| 请求路由 | /produce | |
| 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 分销订单创建

1. 功能描述

|  |  |
| --- | --- |
| 标识符 | ditribute |
| 功能 | 分销商创建食品溯源中分销订单相关信息 |

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 |
|  |  |  |
|  |  |  |

#### 2.4.2.4 销售订单创建

1. 功能描述

|  |  |
| --- | --- |
| 标识符 | retail |
| 功能 | 分销商创建食品溯源中分销订单相关信息。 |

2. 接口功能

|  |  |  |
| --- | --- | --- |
| 请求路由 | retail | |
| 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 食品信息追溯查询

1. 功能描述

|  |  |
| --- | --- |
| 标识符 | trace |
| 功能 | 通过指定订单号追溯食品信息 |

2. 接口功能

|  |  |  |
| --- | --- | --- |
| 请求路由 | trace | |
| 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．系统数据库设计

通过中心化的方式存储系统中用户信息，包括用户名(username)、密码(password)、角色(role)、地址(address)内容。

## 3.1 User表（user）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 字段类型 | 含义 | 是否可为NULL | 是否Primary Key |
| username | varchar | 用户名 | 否 | 是 |
| password | varchar | 密码 | 否 |  |
| role | Integer | 用户角色 | 否 |  |
| address | varchar | 区块链地址 |  |  |

# 4．系统出错处理设计

## 4.1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。

## 4.2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。

## 4.3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

围绕食品安全溯源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。最终根据业务需求规范，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**

基于给定服务器环境以及软件（地址“/root/tools”），搭建一条4节点的区块链系统并验证，具体工作内容如下：

（1）采用默认配置搭建区块链网络；

（2）通过命令验证区块链节点进程运行状况；

（3）通过命令验证区块链连接状态和共识状态日志输出。

**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以及软件（地址“/root/tools”），搭建区块链控制台并开展相关运维工作，具体工作内容如下：

（1）配置控制台，管理相关证书并启动；

（2）使用控制台安装HelloWorld智能合约；

（3）使用控制台完成HelloWorld智能合约的set与get操作；

（4）使用控制台查看区块链中区块高度。

**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）获取指定区块链节点输出等级为警告级，并设置日志存储阈值位100MB并验证；

（2）通过给定工具（地址/root/tools）完成新节点（node4）配置；

（3）启动新节点加入区块链系统并验证。

**子任务1-2-4：区块链网络运维**

根据任务描述要求，完成网络配置与管理运维操作，具体内容如下：

（1）设置区块链系统黑名单，将node3设为黑名单禁止连接，并验证；

（2）设置系统中区块打包最大交易数量设为2000；

（3）验证区块最大打包交易数量情况。

### 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可视化一体平台并验证启动情况；
2. 通过可视化平台生成包括生产商(Producer)、经销商(distributor)、零售商(retailer)账户，并将账户以p12加密形式导出后倒入指定前置可视化平台，验证地址一致性；
3. 使用Postman对上述功能接口进行验证，并将验证结果截图提交工程文档。对食品溯源系统服务端“添加食品”(/produce)功能接口进行验证。

|  |  |
| --- | --- |
| 请求路由: | /produce |
| 请求方法： | POST |
| 输入项说明： | |  |  |  | | --- | --- | --- | | **输入项** | **类型** | **说明** | | traceNumber | String | 追踪编号 | | foodName | String | 食品名称 | | traceName | String | 存证人地址 | | quality | Integer | 食品质量 | |
| 输出项说明： | |  |  |  | | --- | --- | --- | | **输出项** | **类型** | **说明** | | ret | Integer | 返回值 | | msg | String | 返回消息，"Success"表示操作成功 | |

1. 参照工程项目（地址：“/root/projects”）使用Caliper测试工具对食品安全溯源系统智能合约生成新食品(newFood)功能进行压力测试。具体要求如下：
2. 提供核心测试代码；
3. 设置txNumber=10，tps=1，所有测试通过率为100%。
4. 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 EtherStore {

mapping(address => uint) public balances;

function deposit() public payable {

balances[msg.sender] += msg.value;

emit Balance(balances[msg.sender]);

}

function withdraw() public {

uint bal = balances[msg.sender];

require(bal > 0);

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

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

balances[msg.sender] = 0;

}

// Helper function to check the balance of this contract

function getBalance() public view returns (uint) {

return address(this).balance;

}

}

contract Attack {

EtherStore public etherStore;

constructor(address \_etherStoreAddress) {

etherStore = EtherStore(\_etherStoreAddress);

}

// Fallback is called when EtherStore sends Ether to this contract.

fallback() external payable {

if (address(etherStore).balance >= 1) {

etherStore.withdraw();

}

}

function attack() external payable {

require(msg.value >= 1);

etherStore.deposit{value: 1}();

etherStore.withdraw();

}

// Helper function to check the balance of this contract

function getBalance() public view returns (uint) {

return address(this).balance;

}

}

1. 分析智能合约中存在问题，并说明危害；
2. 根据测试工具中的代码文件，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。

完成本任务后将相关命令、代码以及运行结果截图填写至工程文档并提交。

## 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

根据区块链食品溯源应用需求分析和方案设计文档的描述，编写该区块链产品的智能合约功能需求文档，以及设计该智能合约UML图。具体要求如下：

1.编写区块链食品溯源智能合约功能需求文档

根据区块链食品溯源产品项目背景和需求分析，编写该区块链食品溯源产品的智能合约功能需求文档。

2.完成区块链食品溯源智能合约的设计图

设计区块链食品溯源智能合约接口，画出区块链食品溯源智能合约的角色UML用例图，以及画出区块链食品溯源各智能合约关系的时序图。

### 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：

1.食品信息（FoodInfoItem）的接口编码

（1）编写食品信息实体的接口，完成可溯源食品信息初始化，实现可追溯的原始生产商食品信息上链功能；

表2-2-1 FoodInfoItem实体说明

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| \_currentTraceName | string | 当前用户名 |
| \_name | string | 食品名称 |
| \_owner | address | 合约的创建者 |
| \_quality | uint8 | 质量 |
| \_status | uint8 | 状态 |
| \_traceName | string[] | 用户名 |
| \_timestamp | uint[] | 流转时间戳 |
| \_traceAddress | address[] | 用户地址 |
| \_traceQuality | uint8[] | 食品质量 |

contract FoodInfoItem{

//①保存食品流转过程中各个阶段的时间戳

//②保存食品流转过程各个阶段的用户名

//③保存食品流转过程各个阶段的用户地址信息（和用户一一对应）

//④保存食品流转过程中各个阶段的质量

//⑤食品名称

//⑥当前用户名称

//⑦质量（0=优质 1=合格 2=不合格）

//⑧状态（0:生产 1:分销 2:出售）

//⑨初始化owner

（2）编写分销商食品上链信息接口，根据食品溯源智能合约地址获取分销商上链食品的信息；

function addTraceInfoByDistributor(①, uint8 quality) public returns(bool) {

require(\_status == 0 , "status must be producing");

//②

\_timestamp.push(now);

\_traceName.push(traceName);

\_currentTraceName = traceName;

//③

//④

\_traceQuality.push(\_quality);

\_status = 1;

return true;

}

（3）编写超市进行食品上链信息的接口，根据食品溯源智能合约地址获取超市上链食品信息。

function addTraceInfoByRetailer(①, uint8 quality) public returns(bool) {

require(\_status == 1 , "status must be distributing");

//②

\_timestamp.push(now);

\_traceName.push(traceName);

\_currentTraceName = traceName;

//③

//④

\_traceQuality.push(\_quality);

\_status = 2;

return true;

}

2.食品溯源(Trace)的接口编码

（1）编写食品溯源智能合约生产商Producer添加食品接口，必须生产商才能添加可溯源的食品，实现溯源功能；

function newFood(①, string traceName, uint8 quality)

public ② returns(③)

{

//④

//⑤

//⑥

//⑦

//⑧

}

（2）编写食品溯源智能合约分销商Distributor添加食品接口，必须分销商才能添加可溯源的食品，实现溯源功能；

function addTraceInfoByDistributor(①, uint8 quality)

public ② returns(bool) {

//③

return FoodInfoItem(foods[traceNumber]).④, quality);

}

（3）编写食品溯源智能合约超市Retailer添加食品接口，必须超市才能添加可溯源的食品，实现溯源功能。

function addTraceInfoByRetailer(①, uint8 quality)

public ② returns(bool) {

require(③, "traceNumber does not exist");

return FoodInfoItem(foods[traceNumber]).④, quality);

}

3.角色（Role）管理的接口编码

（1）编写食品溯源增加角色接口，必须是未增加的角色才能被添加，实现添加角色的功能；

function add(①, address account) ② {

require(!③, "Roles: account already has role");

role.④ = true;

}

（2）编写食品溯源移除角色接口，必须是已增加的角色才能被移除，实现移除角色的功能；

function remove(①, address account) ② {

require(③, "Roles: account does not have role");

role.④ = false;

}

（3）编写食品溯源角色授权接口，必须是授权的角色地址，实现角色权限管理功能。

function has(①, address account) ② returns (bool) {

require(③, "Roles: account is the zero address");

return role.④;

}

4.合约编译、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）调用食品溯源智能合约的接口，完整验证业务流程。

### 任务2-3：智能合约测试

编写智能合约单元测试代码并完成合约功能测试、性能测试，具体要求如下：

1.配置区块链网络

启动区块链网络，创建新的Workspace，配置对外访问的RPC接口为7545，配置项目的配置文件config.js实现与新建Workspace的连接。

2.设置producerId和sellederId两个变量

基于VSCODE加载测试项目，补全位于test文件夹中foodTraceNew.js文件预操作的方法。在测试文件中添加预定义的方法（在其他方法启动前使用），在方法中分别设置producerId和sellederId两个变量，具体要求如下：

（1）producerId设置为1；

（2）sellderId设置为4。

3.补全createMember和getMember方法

基于VSCODE加载的测试项目，补全位于test文件夹中foodTraceNew.js文件，添加测试用例，测试智能合约的createMember和getMember方法。

4.测试createOrder和getOrder方法

基于VSCODE加载的测试项目，补全位于test文件夹中foodTraceNew.js文件，添加测试用例，测试智能合约的createOrder和getOrder方法。

5.测试createFood和getFood方法

基于VSCODE加载的测试项目，补全位于test文件夹中foodTraceNew.js文件，添加测试用例，测试智能合约的createFood和getFood方法。

## 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，在登录组件login.js、组件管理文件components.js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：



具体要求如下：

（1）有明确的提示，提示用户选择角色；

（2）用户可看到四个不同的角色可选（生产商、中间商、超市、消费者）；

（3）每个用户所对应的组件请在components中找寻并填入；

（4）页面顶部要有食品溯源平台的网站标题和logo。

Login.js:

代码片段1：

template: `

<div class="login">

<!-- 角色选择 -->

<h3 v-if="currentUser === null">选手填写部分</h3>

<el-row :gutter="80" v-if="currentUser === null">

<el-col :span="6" v-for="选手填写部分" :key="index">

<div @click="选手填写部分">选手填写部分</div>

</el-col>

</el-row>

代码片段2：

// 用户身份

users: [

{

name: 选手填写部分,

userName: 'producer',

component: 选手填写部分,

},

{

name: 选手填写部分,

userName: 'distributor',

component: 选手填写部分,

},

{

name: 选手填写部分,

userName: 'retailer',

component: 选手填写部分,

},

{

name: 选手填写部分,

userName: 'consumer',

component: 选手填写部分,

},

],

currentUser: 选手填写部分, // 当前用户

components.js:

代码片段3：

// 头部组件

const Header = {

// 接受传入的登录状态、用户信息

props: ['login', 'user'],

template: `

<div class="header">

<img src="选手填写部分" />

<h3>选手填写部分</h3>

<span v-if="login" class="user-name">{{ 选手填写部分 }}</span>

</div>

`

}

2．请基于前端系统的开发模板，在登录组件login.js、组件管理文件components.js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：





具体要求如下：

（1）点击角色进入相应角色登录页面；

（2）登录界面提示用户的地址（消费者不显示），有登录操作的相关提示；

（3）登录界面有5秒倒计时；

（4）登录界面有“直接登录”按钮，点击可直接跳转到相应角色首页。

login.js:

代码片段1：

<!-- 角色登录 -->

<div v-else class="is-login">

<h3>登录中......(倒计时：{{ 选手填写部分}} 秒)</h3>

<div>角色:

<span>{{ 选手填写部分}}</span>

</div>

<!-- 非消费者则显示角色地址 -->

<div v-if="选手填写部分">角色地址:

<span>{{ 选手填写部分}}</span>

</div>

<!-- 直接登录按钮 -->

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

</div>

代码片段2：

// 登录时有个5秒的倒计时，这里是在点击直接登录时，清除倒计时，直接跳到相关页面

clearTimer() {

clearInterval(选手填写部分);

this.$emit(选手填写部分, {

component: this.loginItem.component,

user: this.loginItem.name,

});

},

// 倒计时

countdownInterval({ component, name: user }) {

this.timer = setInterval(() => {

if(this.countdown <= 0){

选手填写部分;

}

选手填写部分;

}, 选手填写部分);

},

代码片段3：

// 点击用户登录，获取用户地址

handleClick(item) {

this.loginItem = item;

// 处理消费者角色，其他三个角色都有一个角色地址

if (item.userName !== 选手填写部分) {

axios({

method: 'get',

url: `/userinfo?userName=${item.userName}`,

})

.then(ret => {

this.address = 选手填写部分;

this.currentUser = 选手填写部分;

this.countdownInterval(选手填写部分);

})

.catch(err => {

console.log(err)

})

} else {

this.currentUser = item.name;

this.countdownInterval(item);

}

}

### 任务3-2：区块链应用后端功能开发

1．请基于已有的项目，开发完善IndexController类，编写添加食品生产信息的方法，实现食品信息的添加功能，并测试功能完整性。

本任务具体要求如下：

（1）开发文件IndexController.java中的produce方法，请求接口为/produce；

（2）开发文件IndexController.java中的produce方法，要求对前端传入的参数进行二次验证；

（3）开发文件IndexController.java中的produce方法，要求封装返回值为String，但不返回视图页面。

produce方法:

/\*\*

\* 添加食品生产信息

\* traceNumber: 食品溯源id，食品溯源过程中的标识符

\* foodName: 食物名称

\* traceName: 用户名，食品流转过程各个阶段的用户名

\* quality: 当前食品质量（0=优质 1=合格 2=不合格）

\* @return：添加食品生产信息结果

\*/

@选手填写部分

@PostMapping(选手填写部分, produces=MediaType.APPLICATION\_JSON\_VALUE)

public String produce(@RequestBody JSONObject jsonParam) {

//声明返回对象

JSONObject \_outPutObj = new JSONObject();

//生产商生产食品

if(jsonParam == null){

选手填写部分

}

int trace\_number = 选手填写部分;

String food\_name = 选手填写部分;

String trace\_name = 选手填写部分;

int quality = 选手填写部分;

JSONArray params = JSONArray.parseArray("[\""+food\_name+"\","+trace\_number+",\""+trace\_name+"\","+quality+"]");

JSONObject \_jsonObj = new JSONObject();

\_jsonObj.put("contractName",CONTRACT\_NAME);

\_jsonObj.put("contractAddress",CONTRACT\_ADDRESS);

\_jsonObj.put("contractAbi",JSONArray.parseArray(CONTRACT\_ABI));

\_jsonObj.put("user",PRODUCER\_ADDRESS);

\_jsonObj.put("funcName",选手填写部分);

\_jsonObj.put("funcParam",选手填写部分);

String responseStr = httpPost(URL,选手填写部分);

JSONObject responseJsonObj = JSON.parseObject(responseStr);

String msg = responseJsonObj.getString("message");

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

\_outPutObj.put("ret",选手填写部分);

\_outPutObj.put("msg",msg);

}else{

\_outPutObj.put("ret",选手填写部分);

\_outPutObj.put("msg",msg);

}

return 选手填写部分;

}

2．开发完善IndexController类，编写中间商添加食品流转信息的方法，实现中间商添加食品流转信息的功能，并测试功能完整性。

具体要求如下：

（1）开发文件IndexController.java中的add\_trace\_by\_distrubutor方法，请求接口为/adddistribution；

（2）开发文件IndexController.java中的add\_trace\_by\_distrubutor方法，要求对前端传入的参数进行二次验证；

（3）开发文件IndexController.java中的add\_trace\_by\_distrubutor方法，要求封装返回值为String，但不返回视图页面；

add\_trace\_by\_distrubutor方法:

/\*\*

\* 中间商添加食品流转信息

\* traceNumber: 食品溯源id，食品溯源过程中的标识符

\* traceName: 用户名，食品流转过程各个阶段的用户名

\* quality: 当前食品质量（0=优质 1=合格 2=不合格）

\* @return：中间商添加食品流转信息结果

\*/

@选手填写部分

@PostMapping(选手填写部分, produces=MediaType.APPLICATION\_JSON\_VALUE)

public String add\_trace\_by\_distrubutor(@RequestBody JSONObject jsonParam) {

//声明返回对象

JSONObject \_outPutObj = new JSONObject();

if(jsonParam == null){

选手填写部分

}

String trace\_number = 选手填写部分;

String trace\_name = 选手填写部分;

int quality = 选手填写部分;

JSONArray params = JSONArray.parseArray("["+trace\_number+",\""+trace\_name+"\","+quality+"]");

JSONObject \_jsonObj = new JSONObject();

\_jsonObj.put("contractName",CONTRACT\_NAME);

\_jsonObj.put("contractAddress",CONTRACT\_ADDRESS);

\_jsonObj.put("contractAbi",JSONArray.parseArray(CONTRACT\_ABI));

\_jsonObj.put("user",DISTRIBUTOR\_ADDRESS);

\_jsonObj.put("funcName",选手填写部分);

\_jsonObj.put("funcParam",选手填写部分);

String responseStr = httpPost(URL,选手填写部分);

JSONObject responseJsonObj = JSON.parseObject(responseStr);

String msg = responseJsonObj.getString("message");

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

\_outPutObj.put("ret",选手填写部分);

\_outPutObj.put("msg",msg);

}else{

\_outPutObj.put("ret",选手填写部分);

\_outPutObj.put("msg",msg);

}

return 选手填写部分;

}

3．请基于已有的项目，开发完善IndexController类，编写获取某个食品的溯源信息的方法，实现获取某个食品的溯源信息的功能，并测试功能完整性。

具体要求如下：

（1）开发文件IndexController.java中的trace方法，请求接口为/trace，该接口调用私有方法get\_trace，不直接与合约交互，提高系统的安全性；

（2）开发文件IndexController.java中的trace方法，对传入数据进行二次验证；

（3）开发文件IndexController.java中的get\_trace方法，要求通过合约进行溯源信息的查询，且外部无法直接调用；

（4）开发文件IndexController.java中的trace方法，要求封装返回值为String，但不返回视图页面。

trace方法:

/\*\*

\* 获取某个食品的溯源信息

\* @param traceNumber 食品溯源id，食品溯源过程中的标识符

\* @return 对应食品的溯源信息

\*/

@选手填写部分

@GetMapping(选手填写部分, produces=MediaType.APPLICATION\_JSON\_VALUE)

public String trace(String traceNumber){

JSONObject \_outPut = new JSONObject();

if (Integer.parseInt(traceNumber) <= 0){

选手填写部分

}

List res = get\_trace(traceNumber);

JSONArray o = new JSONArray(res);

return 选手填写部分;

}

get\_trace方法：

/\*\*

\* 从链上获取某个食品的溯源信息

\* @param traceNumber 食品溯源id，食品溯源过程中的标识符

\* @return 对应食品的溯源信息

\*/

选手填写部分JSONArray get\_trace(String traceNumber){

//获取食品基本信息

JSONArray params = JSONArray.parseArray("["+traceNumber+"]");

JSONObject \_jsonObj = new JSONObject();

\_jsonObj.put("contractName",CONTRACT\_NAME);

\_jsonObj.put("contractAddress",CONTRACT\_ADDRESS);

\_jsonObj.put("contractAbi",JSONArray.parseArray(CONTRACT\_ABI));

\_jsonObj.put("user","");

\_jsonObj.put("funcName",选手填写部分);

\_jsonObj.put("funcParam",选手填写部分);

String responseStr = httpPost(URL,选手填写部分);

JSONArray food = JSON.parseArray(responseStr);

//获取食品溯源信息

JSONObject \_jsonObj2 = new JSONObject();

\_jsonObj2.put("contractName",CONTRACT\_NAME);

\_jsonObj2.put("contractAddress",CONTRACT\_ADDRESS);

\_jsonObj2.put("contractAbi",JSONArray.parseArray(CONTRACT\_ABI));

\_jsonObj2.put("user","");

\_jsonObj2.put("funcName",选手填写部分);

\_jsonObj2.put("funcParam",选手填写部分);

String responseStr2 = httpPost(URL,选手填写部分);

JSONArray traceInfoList = JSON.parseArray(responseStr2);

JSONArray time\_list = 选手填写部分;

JSONArray name\_list = 选手填写部分;

JSONArray address\_list = 选手填写部分;

JSONArray quality\_list = 选手填写部分;

JSONArray \_outPut = new JSONArray();

for (int i=0;i<time\_list.size();i++){

if (i==0){

JSONObject \_outPutObj = new JSONObject();

\_outPutObj.put("traceNumber",选手填写部分);

\_outPutObj.put("name",选手填写部分);

\_outPutObj.put("produce\_time",选手填写部分);

\_outPutObj.put("timestamp",选手填写部分);

\_outPutObj.put("from",选手填写部分);

\_outPutObj.put("quality",选手填写部分);

\_outPutObj.put("from\_address",选手填写部分);

\_outPut.add(\_outPutObj);

}else{

JSONObject \_outPutObj = new JSONObject();

\_outPutObj.put("traceNumber",选手填写部分);

\_outPutObj.put("name",选手填写部分);

\_outPutObj.put("produce\_time",选手填写部分);

\_outPutObj.put("timestamp",选手填写部分);

\_outPutObj.put("from",选手填写部分);

\_outPutObj.put("to",选手填写部分);

\_outPutObj.put("quality",选手填写部分);

\_outPutObj.put("from\_address",选手填写部分);

\_outPutObj.put("to\_address",选手填写部分);

\_outPut.add(\_outPutObj);

}

}

return \_outPut;

}