2024年“中银杯”甘肃省职业院校技能大赛

高职学生组电子与信息大类

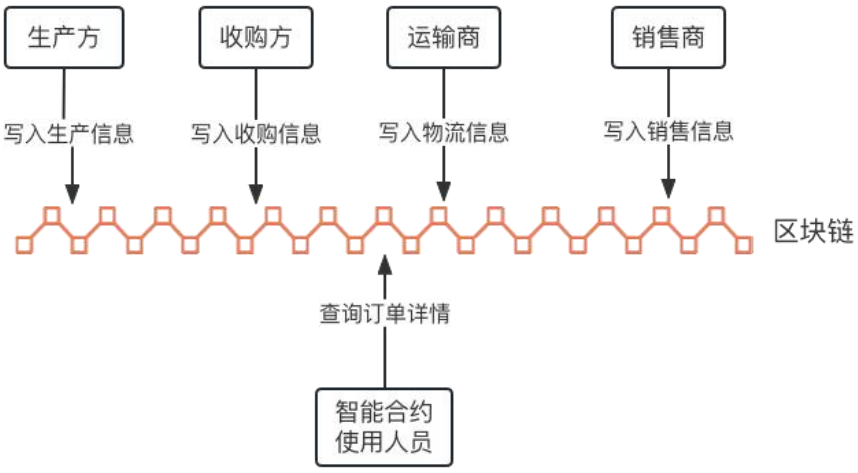
区块链技术应用赛项竞赛样题（2）

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：区块链部署与运维（35 分）

任务 1-1：区块链系统部署与运维（25 分）

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。最终利用业务需求规范，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 1-1-1： 搭建区块链系统并验证（8 分）

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 1-1-2：搭建区块链系统管理平台并验证（8 分）

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 1-1-3：区块链节点运维（5 分）

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 1-1-4：区块链网络运维（4 分）

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 1-2：区块链系统测试（10 分）

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。

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

4. 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：

（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 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 truffle 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）

任务 2-1：智能合约开发（20 分）

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 2-2：智能合约测试（10 分）

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 Ganache 软件，创建新的 Wordspace，配置对外访问的 RPC 接口为 7545，配置项目的 truffle-config.js 实现与新建 Workspace 的连接。

2.设置 producerId 和 sellederId 两个变量

基于 VSCODE 加载的 Truffle 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 Truffle 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 Truffle 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 Truffle 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）

任务 3-1：区块链应用前端功能开发（10 分）

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，清楚倒计时，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（20 分）

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_n
ame+"\","+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;
    }
```