说明文档

💡 * 代码公开在 Github：

TJU-Spring-2024-InfoSecurity/课程设计2：区块链设计 at master · MomoyamaSawa/TJU-Spring-2024-InfoSecurity (github.com)

* 作业笔记：
https://momoyamasawa.notion.site/cb75e684a803490a85217c6262e76a4b?pvs=4

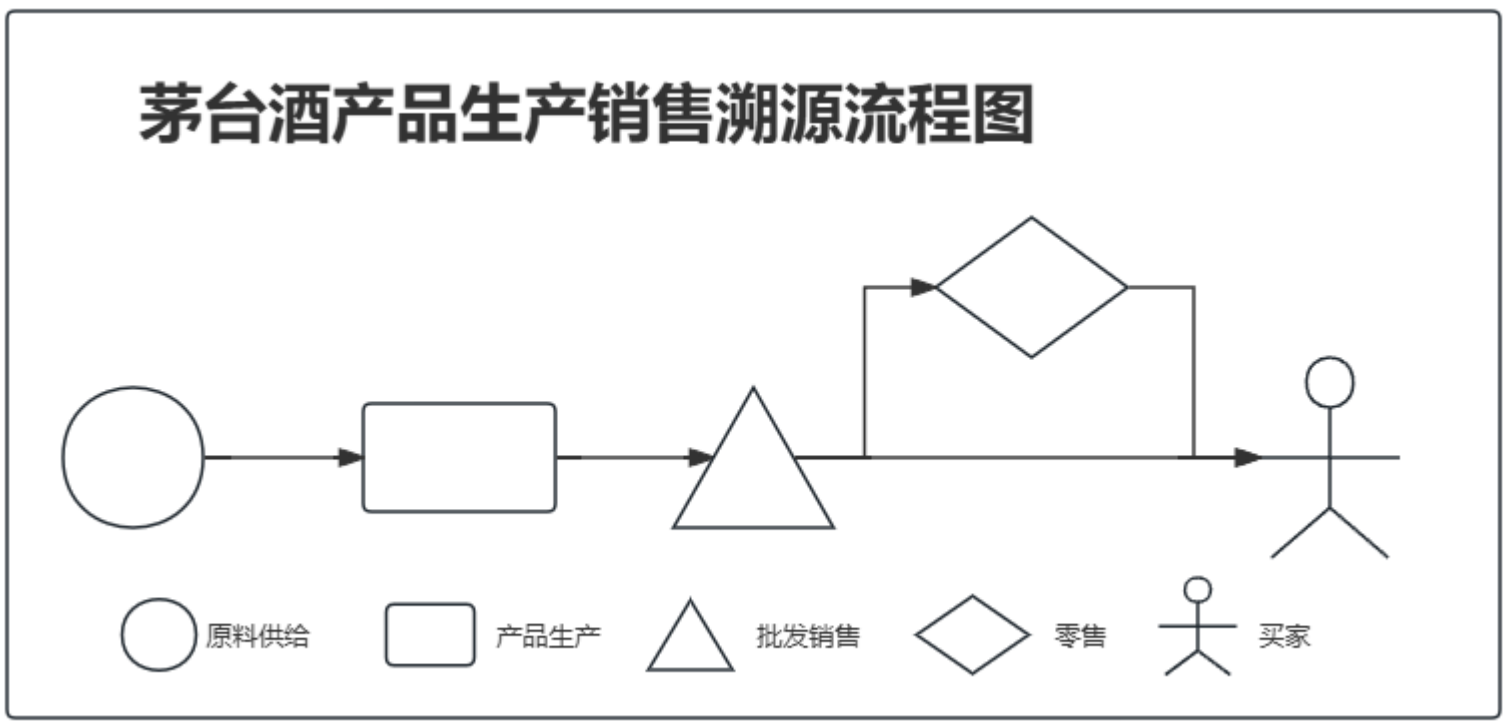
* 程序如何运行参考 README.md

0. 需求分析

目标：开发一个茅台酒溯源系统的演示版本，以确保产品从生产到销售的每个环节都能被追踪。

生产销售流程：

- **原料供给**：确定茅台酒的原料来源，确保原料的质量与可追溯性。
- **产品生产**：监控生产过程，记录关键生产环节和质量检验结果。
- **批发销售**：追踪批发环节，包括批发商信息和产品流向。
- **零售**：记录零售商信息和产品销售数据。
- **终端消费者**：确保消费者能够验证产品的真伪和溯源信息。



系统设计：

- 在本演示系统中，我们将模拟茅台酒的生产和销售流程可能包含四个阶段：原料供给、产品生产、批发销售和零售。
- 每个阶段的参与者将被视为一个独立的账户（节点），代表不同的厂家或供应商。
- **账户设置**：
  - 原料供给：2个账户
  - 产品生产：3个账户
  - 批发销售：2个账户

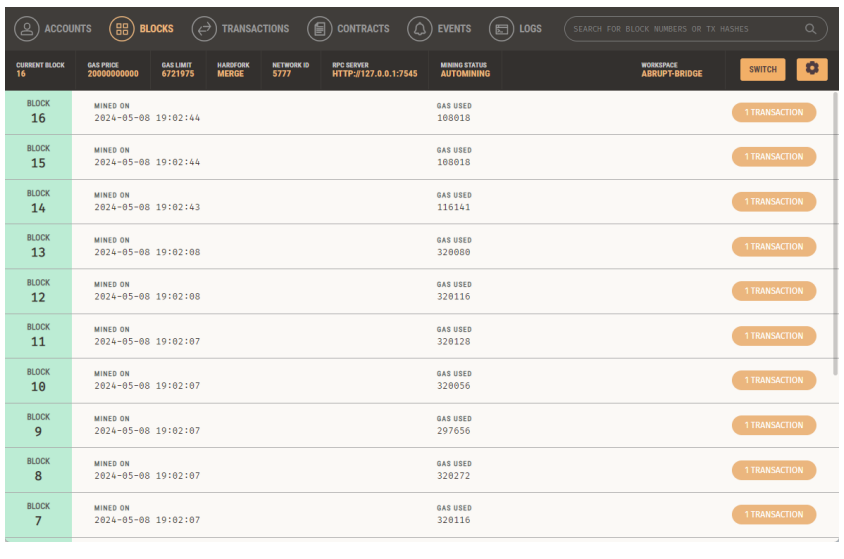
    - 零售：5个账户
- **产品追踪**：每瓶茅台酒将被赋予一个唯一的全程ID，以便在每个阶段记录关键信息，如时间戳、经手的账户（厂家信息）和验证人员信息。
- **账户信息**：每个账户将关联详细的厂家信息，包括：
    - 厂家名称
    - 联系方式
    - 类别
    - 店铺照片
    - 地址
    - 描述

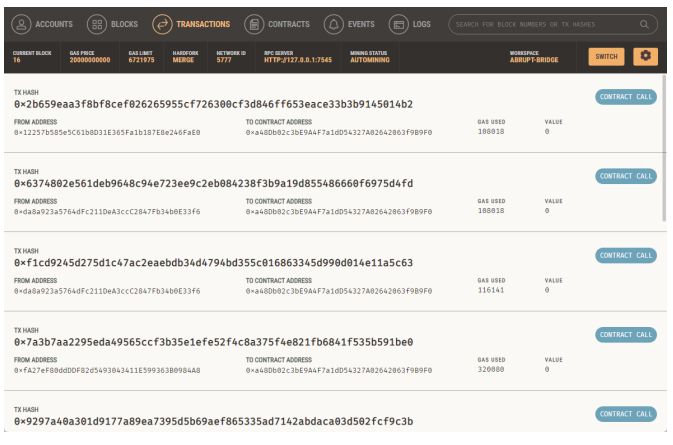
1. 以太坊区块链平台

为了构建一个高效且透明的区块链系统，我们选择了以太坊作为基础平台。以太坊的智能合约功能为我们提供了一个强大的环境，以实现自动化的交易和可信的记录保持。

本地模拟环境：

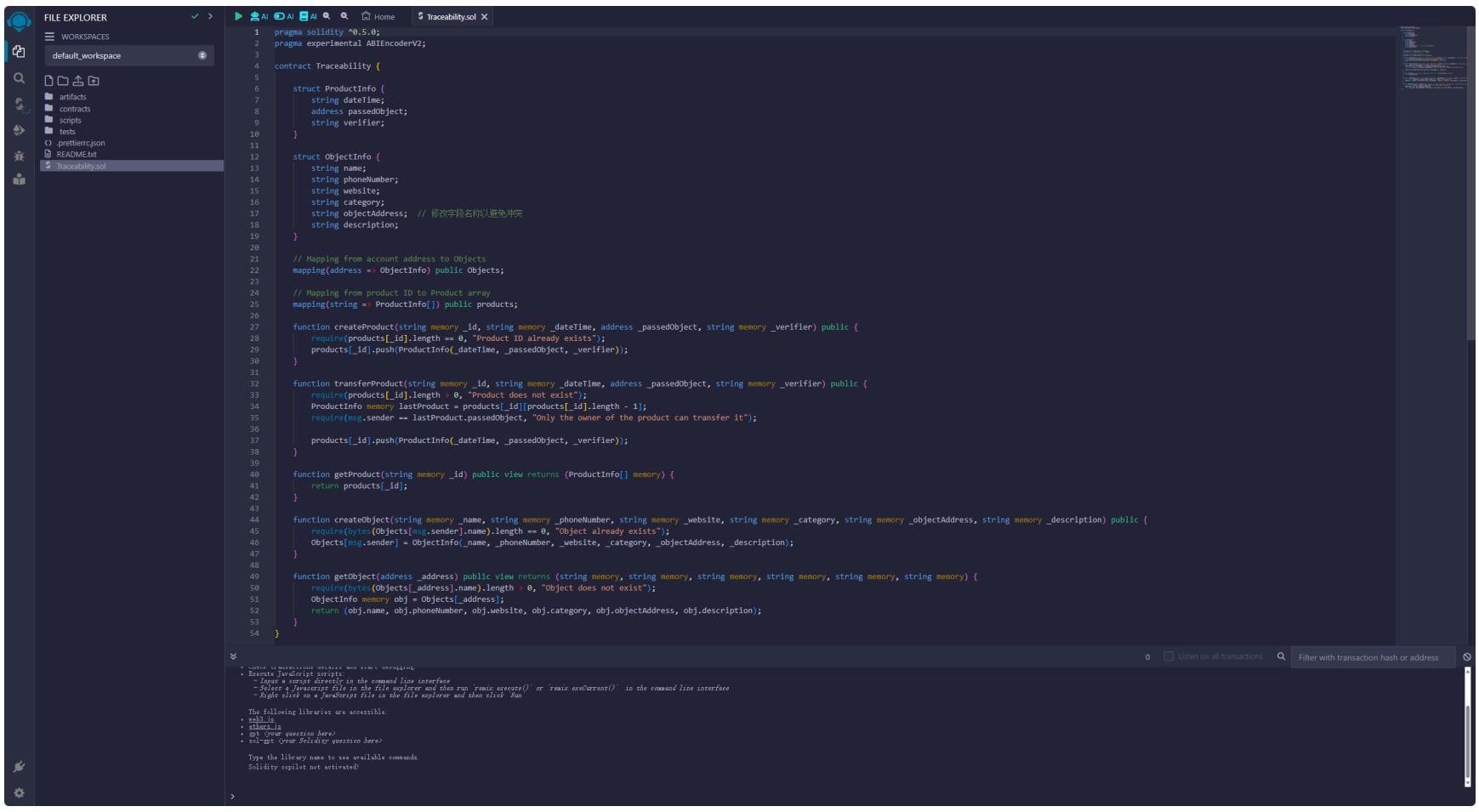
- 我们使用了**Ganache**，一个个人以太坊区块链，用于在本地开发环境中部署合约、开发应用程序和运行测试。
- Ganache提供了一个直观的界面，可以清晰地展示区块和交易信息，使得开发和调试过程更加高效。





智能合约开发：

- 合约代码的编写和测试是通过**Remix IDE**在线完成的，这是一个强大的开源工具，支持从编写到部署智能合约的整个生命周期。
- Remix的实时编译功能确保了代码的即时反馈和快速迭代，大大提高了开发效率。



系统特点：

- **可视化界面**：Ganache的用户界面提供了对区块链状态的即时可视化，包括最新的区块和详细的交易数据。
- **智能合约**：通过Remix IDE，我们能够确保智能合约的代码质量和安全性，同时也便于社区的审查和合作。

2. 智能合约设计

这个智能合约通过以下几个关键组件和主要功能实现：

1. **合约结构**：
   - `ProductInfo` 结构用于存储产品的时间戳、传递对象地址和验证者信息。
   - `ObjectInfo` 结构用于存储参与溯源的各个实体的信息，如名称、电话、网站等。

```
struct ProductInfo {
    string dateTime;
    address passedObject;
    string verifier;
}

struct ObjectInfo {
    string name;
    string phoneNumber;
    string website;
    string category;
    string objectAddress;
    string description;
}
```

2. **数据映射**：
   - `Objects` 映射将地址映射到 `ObjectInfo`，用于记录参与溯源的实体。
   - `products` 映射将产品ID映射到 `ProductInfo` 数组，用于记录产品的所有传递信息。

```
// Mapping from account address to Objects
mapping(address => ObjectInfo) public Objects;

// Mapping from product ID to Product array
mapping(string => ProductInfo[]) public products;
```

3. **核心功能**：
   - `createProduct` 函数用于创建新产品的溯源记录。
   - `transferProduct` 函数用于转移产品的所有权，并添加新的溯源记录。
   - `getProduct` 函数用于获取产品的所有溯源记录。
   - `createObject` 函数用于创建新的参与实体。
   - `getObject` 函数用于获取参与实体的信息。
4. **安全性和完整性**：
   - 使用 `require` 语句确保产品ID的唯一性和产品存在性。

```
// 新增路径前先检查茅台酒产品是否存在
require(products[_id].length == 0, "Product ID already exists");
```

   - 通过 `msg.sender` 验证产品转移的权限。

```
// 在茅台酒产品转移时检查被转移方是否有权力转移它
require(msg.sender == lastProduct.passedObject, "Only the owner of the product can tra
```

5. **扩展性和维护性**：
   - 代码中使用了 `pragma experimental ABIEncoderV2;` 以支持更复杂的数据类型，这有助于未来的扩展。
   - 合约中的注释和清晰的函数命名有助于维护和更新代码。
6. **用户交互**：
   - 提供了公共函数以便用户可以与合约交互，查询产品和实体信息。

整体设计的目标是确保茅台酒的每一步流转都被记录和验证，从而提供透明度和可追溯性。智能合约通过确保数据的不可篡改性和链上的透明度，为茅台酒的溯源提供了一个可靠的解决方案。此外，通过使用以太坊区块链，系统能够利用区块链的分布式和去中心化的特性，增加了整个系统的安全性和稳定性。

3. 后端程序设计

后端随机生成茅台酒产品并且追踪产品从原料到最终销售的整个过程，设计思路如下：

1. **智能合约部署**：
   - 使用python的Web3库与Ganache本地测试网络连接。
   - 读取智能合约的ABI和字节码，部署到Ganache。
2. **数据结构定义**：
   - 定义了`Peoples`数组来模拟不同的参与者。
   - `get_data`函数用于获取不同节点（供应商、生产商、批发商、零售商）的账户信息。

```
# 示例数据信息
# 随机质检员
Peoples = ["李晓婷", "张伟豪", "王雅娜", "刘军宇", "陈雪婷", "杨宇航", "赵晓梅"]
# 某个节点信息
tx_hash1 = contract.functions.createObject(
    "提供商A",  # 名称
    "1234567890",  # 电话号码
    "https://so1.360tres.com/t0126bab79cfb5f11bb.jpg",  # 图片
    "原料供给",  # 类别
    "山东省青岛市市南区香港中路123号",  # 地址
    "This is a description for Object 1",  # 描述
).transact({"from": w3.eth.accounts[0]})
w3.eth.wait_for_transaction_receipt(tx_hash1)
```

3. **产品生命周期管理**：

- `new_product`函数用于创建新产品，并随机分配原料供应商。
- 产品通过不同阶段（生产、批发、可能的零售）的转移，每个阶段都会记录时间戳和参与者。

```
# 随机产品生产过程
# 生成随机日期时间
date_time = generate_random_datatime(START_DATE, TIME_SPAN)
# 从生产者ID列表中随机选择一个ID
id1 = ids_list[1][random.randint(0, len(ids_list[1]) - 1)]
# 调用智能合约的transferProduct函数，传递产品ID，日期时间的时间戳，接收者账户，以及验证者信息
tx_hash1 = contract.functions.transferProduct(
    _id,  # 产品ID
    datetime_to_timestamp_string(date_time),  # 日期时间的时间戳
    w3.eth.accounts[id1],  # 接收者账户
    Peoples[random.randint(0, len(Peoples) - 1)],  # 验证者信息
).transact({"from": w3.eth.accounts[id0]})  # 发起交易的账户
# 等待交易收据
w3.eth.wait_for_transaction_receipt(tx_hash1)
```

4. **时间戳处理**：
   - `generate_random_datatime`和`datetime_to_timestamp_string`函数用于生成和处理时间戳。
5. **查询功能**：
   - `get_product`和`get_object`函数允许查询产品信息和参与者信息。（通过调用合约函数）
6. **随机性**：
   - 代码中使用了`random`库来模拟现实世界中的不确定性，如随机选择供应商节点、生产日期等。
7. **交易处理**：
   - 每个产品生命周期阶段的变化都通过智能合约的函数调用来实现，并等待交易收据确认。

这个设计利用了区块链的不可篡改性和透明度，确保了产品信息的真实性和可追溯性。每个产品的生命周期都被记录在区块链上，从原料到最终用户的每一步都可以被验证。

4. 前端设计

使用python的Gradio库构建的区块链茅台酒溯源系统的界面。它提供了一个用户友好的方式来与后端智能合约交互，允许用户生成新的产品ID，并查询产品和参与者的详细信息。设计的主要特点如下：

1. **界面布局**：
   - 使用`gr.Blocks`创建了一个灵活的布局，其中包含了图片、标签、按钮和文本框等元素。
   - 有两个主要的标签页：一个用于生成和显示产品溯源链，另一个用于查询溯源信息。
2. **功能实现**：
   - `generate_product`函数通过调用后端的`new_product`函数来生成新的产品ID，并获取产品的溯源链信息。
   - `show_info`函数允许用户输入一个地址，并查询该地址对应的参与者信息。
3. **用户交互**：
   - 提供了按钮来触发产品生成和信息查询的功能。
   - 使用文本框让用户输入查询的地址。
4. **信息展示**：
   - 产品ID和溯源链信息通过标签和Markdown组件展示给用户。
   - 参与者信息以Markdown格式展示，包括厂家名称、联系方式、类别、地址和描述。
5. **视觉元素**：
   - 使用图片来增强界面的视觉效果，例如标题图片和茅台酒生产销售流程图。

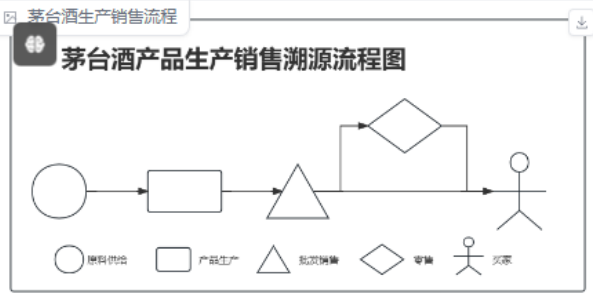
⭐区块链茅台酒溯源 DEMO

2151641王佳垚，先点击随机生成产品按钮，左侧会随机生成一瓶茅台酒产品和它的生产销售路径，从生产销售路径中选择一个溯源地址可以在右侧输入，然后查询到节点厂家商家的详细信息

1. 茅台酒产品溯源链

1. 茅台酒产品溯源链

茅台酒生产销售流程



茅台酒产品唯一ID

点击按钮生成一个茅台酒产品

在这里会显示茅台酒产品溯源链

随机生成产品

2. 溯源信息查询

2.溯源信息查询

溯源地址

请输入溯源地址

在这里会显示溯源信息

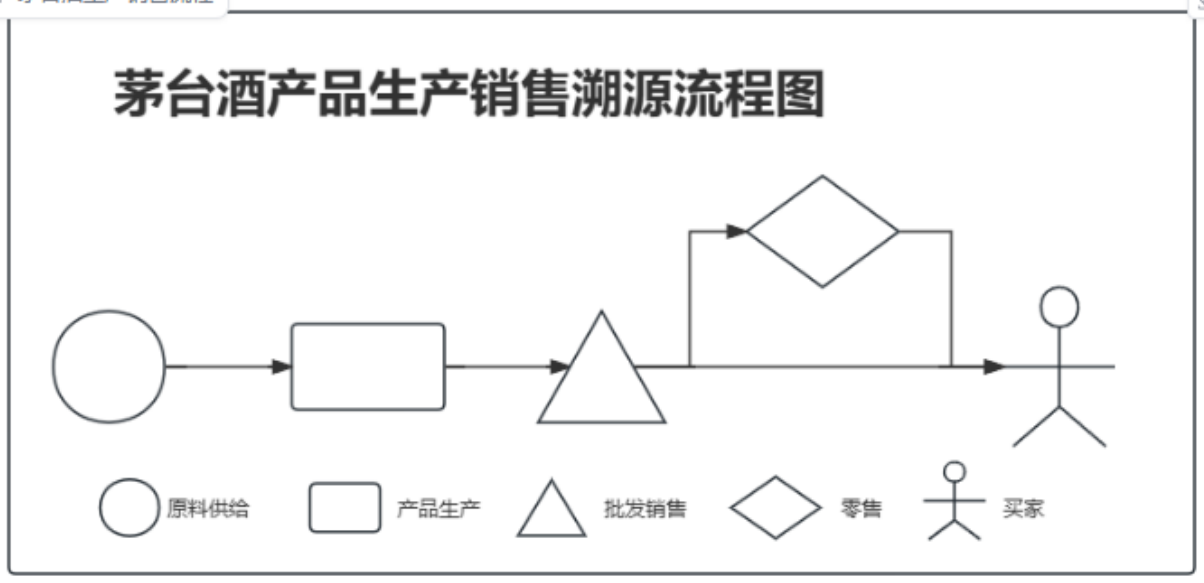
查询溯源信息

第一步，随机生成一个产品

1. 茅台酒产品溯源链

1. 茅台酒产品溯源链

茅台酒生产销售流程



茅台酒产品唯一ID

随机生成的茅台酒的唯一ID

26182250-e1e8-4107-ac09-ba3ff1d344f9

[第1站]
时间：2018年12月19日00时00分
溯源地址：0xda8a923a5764dFc211DeA3ccC2847Fb34b0E33f6
审查员：刘军宇

↓↓↓↓↓

[第2站]
时间：2019年05月11日00时00分
溯源地址：0x7B8F6982Fc9FC8ABf3fb5d7BbA320258887D2AE8
审查员：李晓婷

↓↓↓↓↓

[第3站]
时间：2020年06月28日00时00分
溯源地址：0x5ecB949dfcF2b216022017E4F91E409FcA1dc9A5
审查员：赵晓梅

↓↓↓↓↓

[第4站]
时间：2020年10月31日00时00分
溯源地址：0xe6040b4622B9a78F6f908948De6fb511426888fc
审查员：赵晓梅

溯源链条

溯源地址

↓↓↓↓↓

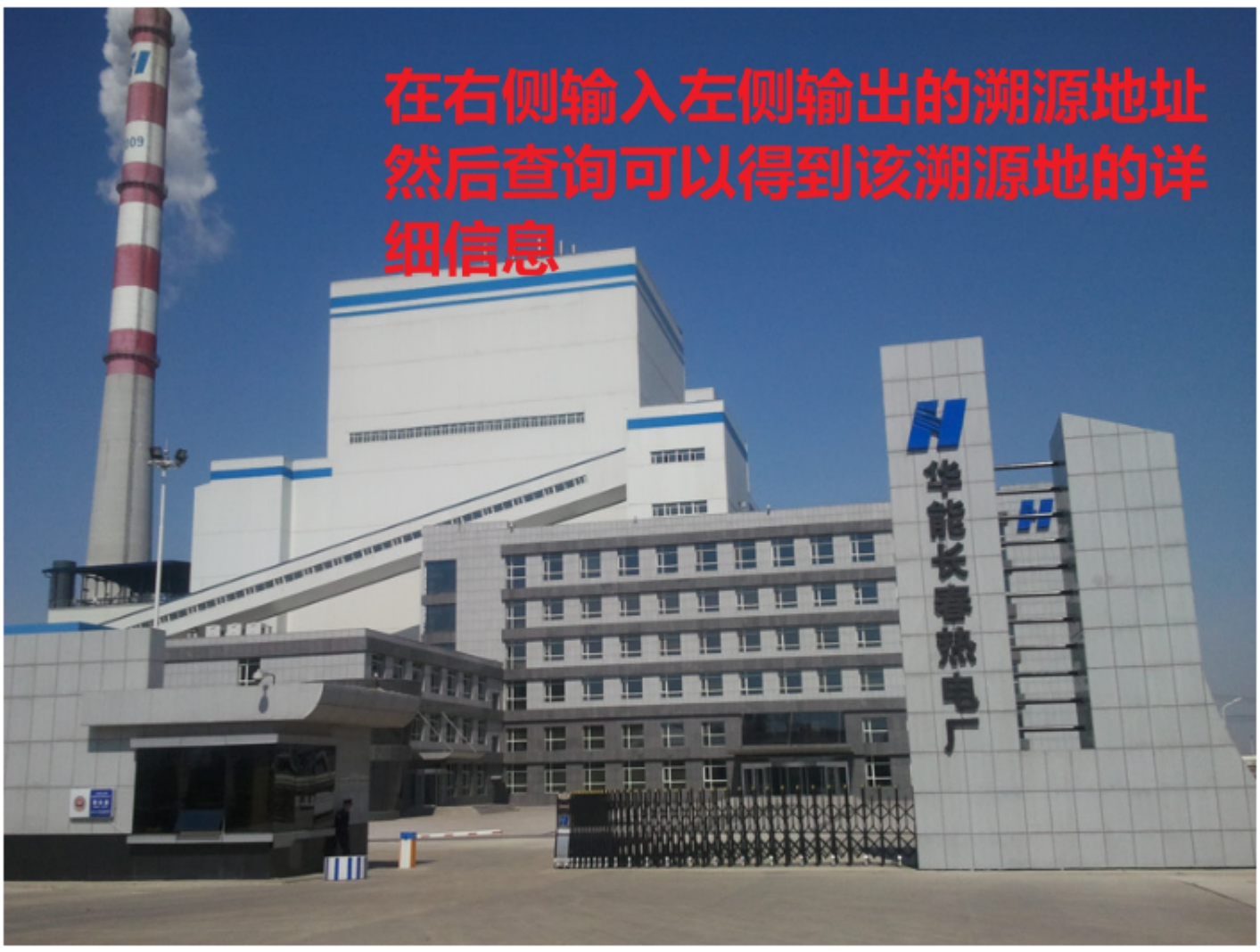
随机生成产品

2. 溯源信息查询

2.溯源信息查询

溯源地址

0xda8a923a5764dFc211DeA3ccC2847Fb34b0E33f6



厂家名称：提供商A
联系方式：1234567890
类别：原料供给
地址：山东省青岛市市南区香港中路123号
描述：This is a description for Object 1

查询溯源信息

注意：后端写了2个原料供给节点，3个产品生产节点，2个批发销售节点，5个零售节点，随机的茅台酒产品会在上面的节点按阶段顺序每个阶段随机选择一个节点信息生成溯源链条（其中可能会没有零售阶段，即买家直接从批发销售得到产品）