Universitas Mercatorum

Università telematica delle Camere di Commercio Italiane

CORSO DI LAUREA IN
INGEGNERIA INFORMATICA

Prova Finale in
SICUREZZA DELLE RETI E CYBERSECURITY

Utilizzo della Blockchain per garantire la qualità e la trasparenza dell'olio di oliva: sviluppo di una *dApp* per il tracciamento e l'autenticità

RELATORE
Chiar.mo
Prof. Roberto Caldelli

CANDIDATO
LUCA RINALDI
MATR. 0082100037

Anno Accademico 2022/2023

INDICE

Introduzione

Il settore dell'agricoltura e della produzione alimentare è uno dei più importanti a livello globale, con un ruolo fondamentale per l'economia e per la sicurezza alimentare delle popolazioni. Tuttavia, il settore è soggetto a diverse sfide, come ad esempio la tracciabilità e l'autenticità dei prodotti, la sostenibilità e la qualità delle produzioni.

La tecnologia “blockchain” può rappresentare un'opportunità per lo sviluppo di soluzioni innovative che aiutino a superare queste sfide e a promuovere la qualità e la sostenibilità dei prodotti alimentari. In particolare, la possibilità di creare contratti smart sulla “blockchain” e di sviluppare “dApp” decentralizzate permette di automatizzare processi complessi e di eseguire operazioni in modo sicuro e tracciabile.

L'obiettivo di questa tesi è quello di presentare un approccio per lo sviluppo di una “dApp” per il tracciamento e l'autenticità dell'olio extravergine di oliva di qualità, utilizzando la piattaforma Ethereum e gli strumenti di sviluppo come ganache, truffle e nodejs, nonché solidity come linguaggio di programmazione per la scrittura dello “smart contract” che funge da base per l’intero progretto. L'olio extravergine di oliva è stato scelto come caso d'uso perché rappresenta un prodotto di qualità molto apprezzato a livello globale e perché è soggetto a numerose truffe e frodi commerciali che possono danneggiare il settore e ingannare i consumatori.

Contesto e obiettivi della tesi

In questa tesi, si studia il potenziale della “Blockchain” per lo sviluppo di una “dApp” (“Decentralized Application”) per il tracciamento e l'autenticità dell'olio extravergine di oliva di qualità. Questo settore è stato recentemente colpito da diversi casi di contraffazione e di mescolanza con oli di qualità inferiore, il che ha portato a perdite economiche per i produttori e ha minato la fiducia dei consumatori nella qualità del prodotto. La “dApp” sviluppata in questa tesi mira a risolvere questi problemi fornendo una piattaforma per il tracciamento delle fasi di produzione e distribuzione dell'olio di oliva, garantendo così l'autenticità del prodotto e la trasparenza della supply chain.

Argomenti trattati e struttura della tesi

La tesi è strutturata come segue:

- Nel capitolo 1 viene fornita una panoramica sulla “Blockchain”, sui suoi concetti di base e sui diversi tipi di reti disponibili. Inoltre, viene approfondita l'argomento degli “Smart Contracts” e della loro implementazione sulla piattaforma Ethereum e viene data una definizione di “dApp”.
- Nel capitolo 2 viene descritta la progettazione della “dApp”, compresi i requisiti e l'architettura della soluzione.
- Nel capitolo 3 ne viene presentata l'implementazione, con una descrizione degli strumenti utilizzati e del processo di sviluppo del “backend” e del “frontend”.
- Nel capitolo 4 viene eseguito un caso studio di test della “dApp” sviluppata con una breve sezione dedicata al controllo degli errori.

La tesi si conclude con i riferimenti bibliografici utilizzati per lo sviluppo e le conclusioni sul lavoro svolto.

1. Stato dell'Arte

La tecnologia "blockchain" rappresenta una delle principali innovazioni degli ultimi anni, con un impatto potenziale su diversi ambiti, come ad esempio le finanze, la supply chain e l'identità digitale.

1.1.1 Introduzione alla "Blockchain"

La "blockchain" è una tecnologia di registro distribuito che consente di registrare e verificare le transazioni in modo sicuro, senza la necessità di un'autorità centrale. Si tratta di un database che viene condiviso da una rete di nodi, ovvero computer che partecipano alla rete "blockchain". Ogni nodo mantiene una copia del database e, quando viene effettuata una nuova transazione, questa viene verificata e aggiunta alla catena di blocchi, che rappresenta la cronologia delle transazioni. Ciò rende la "blockchain" immune alle modifiche non autorizzate, poiché per effettuare una modifica sarebbe necessario agire sulla maggioranza dei nodi della rete, il che è molto difficile da fare.

"La blockchain (in italiano: blocchi concatenati) è una struttura dati che consiste in elenchi crescenti di record, denominati "blocchi", collegati tra loro in modo sicuro utilizzando la crittografia. Ogni blocco contiene un hash crittografico del blocco precedente, un timestamp e dati di transazione. Poiché ogni blocco contiene informazioni sul blocco precedente, questi formano effettivamente una catena con ogni blocco aggiuntivo che si collega a quelli precedenti."[1]

[1] Definizione di "Blockchain" – Wikipedia (https://it.wikipedia.org/wiki/Blockchain)

Infine, determinati tipi di “blockchain” sono trasparenti, poiché tutte le transazioni vengono registrate in modo pubblico e possono essere consultate da chiunque abbia accesso alla rete. Ciò la rende particolarmente adatta a sistemi che richiedono trasparenza, come ad esempio la supply chain, dove è importante conoscere l'origine e il percorso di un prodotto.

1.1.2 Caratteristiche della “Blockchain” e utilizzi potenziali

La tecnologia “blockchain” presenta diverse caratteristiche che la rendono adeguata per diverse applicazioni negli ambiti più disparati. Ecco alcune delle principali caratteristiche:

- Immutabilità: una volta che un blocco viene aggiunto alla “blockchain”, le informazioni in esso contenute diventano permanenti e non possono essere modificate.

- Decentralizzazione: la “blockchain” non dipende da un'autorità centrale, ma è gestita da una rete di nodi che collaborano tra loro.

- Tracciabilità: la “blockchain” consente di tracciare le transazioni effettuate e di verificarne l'autenticità, poiché le informazioni sono registrate in modo cronologico e sicuro.

- Sicurezza: la crittografia e la struttura della “blockchain” garantiscono un elevato livello di sicurezza e rendono difficile l'interferenza o la modifica dei dati da parte di terzi non autorizzati.

In base a queste caratteristiche può essere utilizzata per diverse applicazioni, come ad esempio:

- Tracciamento e autenticità dei prodotti di qualità: la “blockchain” può essere utilizzata per tracciare le fasi di produzione, trasporto e distribuzione dei prodotti e per verificarne l'origine e la qualità. Ad esempio, può essere utilizzata per garantire l'autenticità di prodotti alimentari di qualità, o di prodotti di lusso.

- Gestione della supply chain: la “blockchain” può essere utilizzata per automatizzare e rendere più efficiente la gestione della supply chain, consentendo di tracciare i prodotti in ogni fase della loro distribuzione e di verificare la sostenibilità e la qualità delle produzioni.

- Gestione dell'identità digitale: la “blockchain” può essere utilizzata per gestire l'identità digitale dei cittadini o degli utenti che accedono ai servizi online, consentendo di verificarla in modo sicuro e tracciabile.

- Gestione dei contratti: la “blockchain” può essere utilizzata per automatizzare la gestione dei contratti, utilizzando gli “smart contract”, ovvero, contratti digitali che vengono eseguiti automaticamente quando vengono soddisfatte determinate condizioni, senza la necessità di intermediari.

1.2 Elementi di crittografia

La crittografia è una tecnologia che consente di proteggere le informazioni mediante l'utilizzo di algoritmi di cifratura. Essa è utilizzata in molti ambiti, come ad esempio la sicurezza dei dati, la protezione della privacy e la creazione di sistemi di autenticazione.

1.2.1 Crittografia su “blockchain”

La crittografia è un elemento fondamentale della “blockchain”, poiché consente di garantire la sicurezza delle transazioni e l'immutabilità dei dati registrati sulla catena di blocchi. Ogni transazione sulla “blockchain” viene crittografata e viene assegnato un codice univoco, noto come "hash", che ne garantisce l'immutabilità. Inoltre, viene utilizzata per proteggere la privacy degli utenti, poiché consente di nascondere le informazioni sensibili dietro una "cifratura".

La crittografia è anche utilizzata per creare sistemi di autenticazione, come ad esempio le chiavi private e pubbliche, che vengono utilizzate per verificare l'identità degli utenti sulla “blockchain”. Le chiavi private sono utilizzate per firmare digitalmente le transazioni, mentre le chiavi pubbliche vengono utilizzate per verificare la firma e garantire che la transazione sia autentica.

Praticamente, chiave pubblica e privata, in un sistema basato sulla “blockchain”, a cosa corrispondono?

Supponiamo che tu voglia inviare una quantità di criptovaluta a un amico, innanzitutto usi la tua chiave privata per dimostrare che hai il permesso di farlo e per firmare digitalmente la transazione. La chiave pubblica del tuo amico viene utilizzata per sapere dove inviare la criptovaluta. La firma

digitale viene poi verificata utilizzando la tua chiave pubblica, per confermare che la transazione è legittima. Una volta che la transazione viene verificata, viene aggiunta alla blockchain e il tuo amico riceve la criptovaluta che hai inviato.

La chiave pubblica viene spesso utilizzata come indirizzo del “wallet” (cioè come una sorta di "conto bancario" virtuale). Tuttavia, la chiave pubblica non è la stessa cosa del “wallet”, che è un programma che ti permette di inviare, ricevere e conservare criptovalute. Solitamente, il “wallet” memorizza le tue chiavi private e pubbliche in modo sicuro e ti offre un'interfaccia grafica per gestire le tue criptovalute.

La chiave privata è un codice crittografico che ti viene assegnato quando crei un “wallet” per gestire le tue criptovalute. È simile alla chiave privata di una cassetta di sicurezza o di una serratura: serve a proteggere l'accesso alle tue criptovalute e solo tu dovresti conoscerla.

In questo modo, le queste due chiavi vengono utilizzate per garantire l'autenticità delle transazioni sulla “blockchain” e per proteggere gli utenti dalle frodi.

La crittografia viene anche utilizzata per proteggere la sicurezza dei nodi della rete “blockchain” in sistemi di consenso basati sulla "proof of stake" (PoS), come quello utilizzato sulla attuale “blockchain” di Ethereum. Nel PoS, i nodi che desiderano partecipare alla verifica delle transazioni devono dimostrare di avere una quantità significativa di Ether (la valuta nativa di Ethereum) come "stake" (investimento). Questo "stake" viene crittografato e immagazzinato in un indirizzo Ethereum, che viene utilizzato come prova del possesso dell'Ether da parte del nodo.

1.3 Ethereum e “Smart Contracts”

“Ethereum, lanciata nel 2015, è la seconda criptovaluta per capitalizzazione di mercato dopo il bitcoin, ma diversamente da esso non è stata creata come moneta digitale. I fondatori di Ethereum, invece, intendevano creare un nuovo tipo di piattaforma computazionale, globale e decentralizzata, che avesse lo stesso livello di sicurezza e apertura delle blockchain, estendendo questi attributi a un'ampia gamma di applicazioni.”[2]

Ethereum è una piattaforma open source basata sulla tecnologia “blockchain” che consente lo sviluppo di “dApp” (applicazioni decentralizzate) e di “smart contract” ovvero contratti automatizzati eseguiti sulla rete Ethereum utilizzando il linguaggio di programmazione Solidity.

“Il termine contratto intelligente è stato introdotto per la prima volta nel 1994 e si riferisce alla registrazione di contratti sotto forma di codice informatico. Quando vengono soddisfatte le condizioni preimpostate, il contratto viene automaticamente attivato. I contratti intelligenti consentono transazioni auto-eseguite, senza la necessità di intermediari come banche o altre istituzioni.”[3]

Gli “smart contract” hanno il vantaggio di essere trasparenti, sicuri e immutabili, poiché vengono eseguiti sulla “blockchain” e possono essere utilizzati per automatizzare molti processi che altrimenti richiederebbero l'intervento umano.
Dal momento che due parti (individui o organizzazioni) raggiungono un accordo utilizzando uno “smart contract”.

[2] Cos’è Ethereum – Coinbase.com (https://www.coinbase.com/it/learn/crypto-basics/what-is-ethereum)
[3] Origine del termine “contratto intelligente” - Ricercamy.com (https://ricercamy.com/2021/08/come-trovare-una-valido-solidity-developer)

- Nessuna parte può modificare da sola i termini del contratto.
- Tutte le azioni risultanti dal contratto intelligente sono automatiche e avvengono senza intermediari.
- Tutte le transazioni sono registrate sulla “blockchain” e sono irreversibili.
- Quando le condizioni preimpostate non sono soddisfatte, le transazioni non si verificano.

Le persone coinvolte non hanno nemmeno bisogno di fidarsi l’una dell’altra poiché i contratti vengono eseguiti solo quando i termini concordati sono rispettati, l’accordo viene eseguito sulla “blockchain” di Ethereum, il che significa che tutti i dettagli del contratto sono memorizzati su un libro mastro pubblico.

La corretta implementazione di uno “smart contract” sulla “blockchain” può essere verificata attraverso l'utilizzo di strumenti web come Etherscan o di strumenti da riga di comando (CLI) come Truffle, che consentono di verificare l'hash della transazione o l'indirizzo del contratto per verificarne la presenza sulla blockchain. Questo processo è importante per garantire l'integrità e la corretta implementazione degli smart contract sulla blockchain, ed è una delle attività di verifica e validazione necessarie nell'ambito della progettazione e sviluppo di applicazioni decentralizzate basate su blockchain.

1.4 Cos'è una “dApp”

Una “dApp” (“decentralized application”, applicazione decentralizzata) è un programma simile alle applicazioni tradizionali, ma invece di dipendere da un server centralizzato si basa su una piattaforma “blockchain” e sui suoi network distribuiti. Inoltre, gestisce i dati dell'utente in modo diverso rispetto alle app tradizionali: l'utente utilizza il proprio "account" “blockchain”, ovvero le proprie chiavi crittografiche, invece di fornire i suoi dati personali a un database centralizzato gestito dallo sviluppatore dell'applicazione. Uno dei vantaggi delle “dApp” è che sono resistenti ai tentativi di censura o di interruzione del servizio, poiché non dipendono da un'unica autorità o da un singolo server. Inoltre, possono offrire maggiore trasparenza e sicurezza rispetto alle app tradizionali, poiché i dati vengono registrati in modo immutabile.

2. Progettazione

2.1 Descrizione Progetto

Il progetto mira a costruire una piattaforma basata su “blockchain” per tracciare l'origine e l'autenticità dell'olio extravergine di oliva di qualità, garantendo trasparenza dei dati e sicurezza delle transazioni. Verrà utilizzata una “dApp” per garantire che i dati siano immutabili, consentendo di tracciare l'origine dell'olio in ogni fase della catena di fornitura.

I vantaggi della “dApp” sono:

- Trasparenza dei dati: la “blockchain” garantisce la trasparenza dei dati e la loro immutabilità, rendendo possibile il tracciamento dell'origine e della qualità dell'olio extravergine di oliva.
- Sicurezza: la “blockchain” offre un livello elevato di sicurezza dei dati, poiché i dati sono archiviati in modo decentralizzato e sono protetti da attacchi informatici.
- Qualità del prodotto: il tracciamento dell'origine dell'olio extravergine di oliva consente di garantirne la qualità e di soddisfare le esigenze dei consumatori finali.

I ruoli nella “supply chain” (Entità coinvolte)

Ora vengono descritti i ruoli e le entità coinvolte nella “supply chain” dell'olio extravergine di oliva e le loro responsabilità nella gestione della “dApp”:

- Amministratore (*Owner*): è l'entità responsabile dell'inserimento dei ruoli nella “supply chain” e della gestione della “dApp”. L'amministratore è il proprietario della “dApp” e ha il controllo totale su di essa, inoltre è l’entità che effettua il “deploy” dello “smart contract” sulla “blockchain”.

- Frantoio (Produttore): è l'entità responsabile della produzione dell'olio extravergine di oliva, dell'aggiunta di nuovi lotti di Olio nella “dApp”. Inoltre ha il ruolo di impostare le informazioni relative al prodotto descrivendo dettagliatamente caratteristiche organolettiche, uliveto di provenienza, date di raccolta e molitura, ecc.

- Distributore: è l'entità responsabile della distribuzione dell'olio ai rivenditori. Il distributore può accedere alla “dApp” per visualizzare i prodotti disponibili e per prendere in consegna un prodotto.

- Rivenditore: è l'entità responsabile della vendita ai consumatori finali. Il rivenditore può accedere alla “dApp” per visualizzare i prodotti disponibili e per impostare i prodotti come "in vendita" ed infine come "venduti" una volta che sono stati acquistati dai consumatori finali.

- Consumatore finale: rappresenta l'ultimo tassello della catena di distribuzione del prodotto. Grazie all'inserimento del codice o dell'ID del lotto nella sezione "Track", è in grado di accedere alla tracciabilità dell'intera filiera, ottenendo così la conferma dell'autenticità e della qualità del prodotto.

2.2 Analisi dei requisiti

L'analisi dei requisiti è una fase cruciale del processo di progettazione di un sistema informatico, poiché serve a definire il comportamento e le caratteristiche del sistema. In questo modo, si può progettare e implementare la soluzione in modo coerente con le esigenze del progetto, avendo un quadro chiaro di queste esigenze.

Ne vengono quindi analizzati e specificati i requisiti. In particolare, vengono individuati gli obiettivi del progetto e le funzionalità che dovrà avere per soddisfare questi obiettivi.

Gli obiettivi del progetto sono:

- Garantire l'autenticità dell'olio extravergine di oliva di qualità lungo tutta la “supply chain”, dalla produzione alla vendita al dettaglio.
- Fornire una tracciabilità completa del prodotto lungo tutta la “supply chain”, in modo che i consumatori finali possano scegliere con maggiore consapevolezza.
- Facilitare la collaborazione e il coordinamento tra le diverse entità coinvolte.

Le funzionalità che la “dApp” dovrà avere per soddisfare questi obiettivi sono:

- Consentire all'amministratore di inserire i ruoli delle altre entità nella “supply chain”.
- Consentire al frantoio di aggiungere nuovi lotti di Olio alla “dApp”.
- Consentire al distributore di marcare il prodotto come distribuito ai rivenditori.
- Consentire al rivenditore di marcare il prodotto come venduto ai consumatori finali.
- Consentire ai consumatori finali di tracciare il prodotto durante tutta la “supply chain” mediante l’utilizzo di un’interfaccia grafica.

2.3 Architettura della soluzione

La “dApp” progettata è costituita da diversi elementi che lavorano insieme per garantire il corretto funzionamento della soluzione.

Gli elementi che compongono la “dApp” sono:

- “Smart contract”: sono i contratti programmati sulla “blockchain” di Ethereum che contengono le regole e le logiche che governano il funzionamento della “dApp”. In particolare è stato implementato lo “smart contract” "SupplyChain.sol" che viene utilizzato per gestire il tracciamento e l'autenticità dei prodotti. Il sistema di controllo degli accessi dell'applicazione non è stato implementato come parte della logica di programmazione del lato web, ma invece è stato intrinsecamente integrato nello "smart contract" stesso. Ciò significa che, per poter effettivamente interagire con l'applicazione e apportare modifiche alla blockchain, gli indirizzi “Ethereum” che richiedono tali privilegi devono essere registrati preventivamente dall'owner. In particolare, ciò è stato realizzato attraverso la scrittura di codice “Solidity” specifico, che garantisce l'accesso solo ai soggetti previamente autorizzati e limita le operazioni eseguibili da utenti non autorizzati.
- Interfacce utente: sono le pagine web che consentono agli utenti di interagire con la “dApp” e di eseguire azioni come l'inserimento dei ruoli, l'aggiunta di nuovi prodotti, controllo della “supply chain” svolto unicamente dai rispettivi ruoli e il tracciamento dei prodotti. Le interfacce utente sono state sviluppate utilizzando il framework javascript “React”.

- Server: viene utilizzato per ospitare l'interfaccia utente e per gestire le richieste di connessione tra gli utenti e la “dApp”. Il server è stato sviluppato utilizzando “Node.js”.

Le interazioni tra questi elementi avvengono attraverso l'utilizzo della tecnologia Ethereum e dei suoi strumenti, come Solidity per la programmazione, Truffle per il test e il deploy degli “smart contracts”, Web3.js per l'interazione dell’interfaccia web con la “blockchain” e Metamask per consentire ai vari ruoli di interagire con la “dApp” attraverso i propri indirizzi Ethereum.

3. Implementazione

3.1 Strumenti Utilizzati

I capitoli precedenti hanno descritto la tecnologia “blockchain” e la piattaforma Ethereum come strumenti adeguati per lo sviluppo di soluzioni per il tracciamento e l'autenticità dei prodotti di qualità.

In questo capitolo verrà descritto in dettaglio il processo di sviluppo utilizzando gli strumenti e le tecnologie seguenti: solidity, ganache, truffle, nodejs, la libreria web3.js e metamask.

3.1.1 Solidity

Solidity è il linguaggio di programmazione utilizzato per scrivere gli “smart contract” di Ethereum. Si tratta di un linguaggio orientato agli oggetti, basato su JavaScript, con caratteristiche specifiche per la creazione di contratti intelligenti che possono essere eseguiti sulla “blockchain”. Permette di definire le variabili di stato, le funzioni e gli eventi all'interno di uno “smart contract”, nonché di specificare le regole per l'esecuzione e la modifica del contratto stesso.

3.1.2 Truffle

Truffle è un framework open-source per lo sviluppo di “smart contract” su blockchain Ethereum. Offre un ambiente di sviluppo integrato (IDE) che semplifica notevolmente il processo di creazione, compilazione e testing degli “smart contract”, quindi ne garantisce la corretta implementazione. Truffle utilizza il compilatore “Solidity” per convertire il codice sorgente dei contratti intelligenti in “bytecode”, il linguaggio di basso livello che viene eseguito sulla blockchain Ethereum. Tra gli altri strumenti, ci sono i test di unità che consentono di verificare il comportamento del contratto in diversi scenari, l'analisi statica del codice per identificare eventuali vulnerabilità e problemi di sicurezza, e l'utilizzo di strumenti come la “console” Truffle per verificare la presenza e la corretta esecuzione dello “smart contract”.

3.1.3 React App e Node Js

React è una libreria JavaScript per la creazione di interfacce utente, mentre Node Js è un ambiente di esecuzione JavaScript che permette di sviluppare “backend”. Insieme, questi strumenti sono stati utilizzati per creare l'interfaccia web che sarà utilizzata dagli utenti per interagire con la “blockchain”.

3.1.4 Ganache

Ganache è una piattaforma per lo sviluppo e il test di contratti intelligenti su Ethereum. Offre la possibilità di creare una rete privata con uno o più nodi di Ethereum, consentendo di effettuare test senza dover utilizzare la rete principale. Inoltre, fornisce diversi strumenti per il debugging e il testing di contratti intelligenti, come il monitoraggio delle transazioni e il controllo dell'esecuzione dei contratti.

3.1.5 Web3.js

Web3.js è una libreria JavaScript che fornisce un'interfaccia per interagire con la “blockchain” di Ethereum e con gli “smart contract” dall'interno delle “dApp” attraverso una connessione JSON-RPC. È stato utilizzato per connettersi alla rete e inviare transazioni agli “smart contract”. Web3.js permette anche di accedere ai dati presenti sulla “blockchain”, come ad esempio il contenuto degli “smart contract” e le informazioni sugli indirizzi Ethereum.

3.1.6 Metamask

Metamask è un portafoglio digitale basato su browser che permette di gestire indirizzi e di inviare transazioni sulla rete Ethereum. È stato utilizzato per interagire con lo “smart contract” dal browser.

3.2 Sviluppo del “backend”

3.2.1 Setup di Ganache

Il primo passo nello sviluppo del “backend” è stato quello di impostare Ganache, creando una “blockchain” locale con uno o più portafogli digitali (wallet) che verranno utilizzati per il testing dello “smart contract”. È stato necessario configurare le impostazioni della rete, come il numero di “wallet” e il tipo di consensus utilizzato.

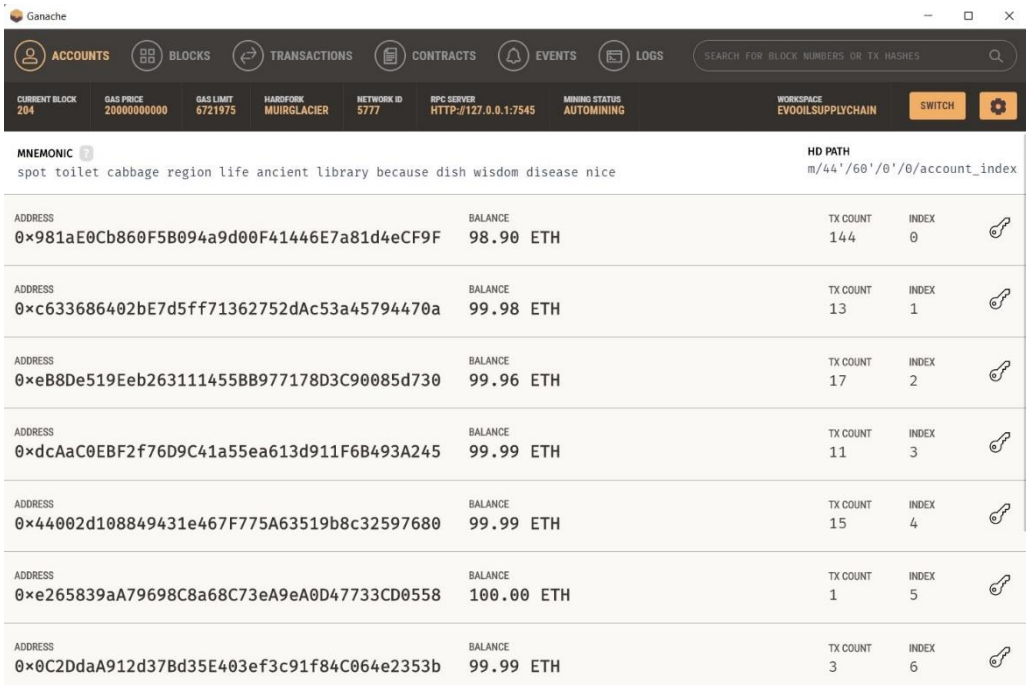

Figura 1- Pagina principale di Ganache che mostra gli indirizzi (Wallet) associati alla “blockchain locale”. Vengono mostrati anche Il Bilancio in “Eth” per ciascun indirizzo e il conteggio delle transazioni. Ognuno di questi indirizzi verrà usato da ogni singolo attore/ruolo della supply chain per interagire con la stessa, rendendo possibile in questo modo il passaggio di consegna del prodotto.

3.2.2 Il contratto “SupplyChain.sol”

Successivamente, è stato creato lo “smart contract” "SupplyChain.sol" utilizzando Solidity. Questo contratto rappresenta la logica di gestione della “supply chain” dell'olio extravergine di oliva, definendo le variabili di stato e le funzioni per il tracciamento e l'autenticità del prodotto. È stato inoltre definito l'insieme di regole per l'esecuzione e la modifica del contratto, specificando i permessi di accesso per ogni funzionalità in base al ruolo dell'utente nella “supply chain”.

```
pragma solidity >=0.4.22 <0.9.0;

contract SupplyChain {
    //Il proprietario dello Smart Contract sarà la persona che distribuisce
    //il contratto sulla blockchain, solo lui può autorizzare
    //vari ruoli come produttore, rivenditore, ecc
    address public Owner;

    //questo costruttore verrà chiamato quando lo smart contracrt
    //sarà registrato sulla blockchain
    constructor() public {
        Owner = msg.sender;
    }

    //questo modificatore serve ad essere sicuri che solo l'owner
    //userà le funzioni in cui sarà usato
    modifier onlyByOwner() {
        require(msg.sender == Owner);
        _;
    }

    //Ruoli (flusso della supply chain dell'Olio di Oliva)
    // Uliveto; //Qui è dove il Frantoio prenderà le Olive (materia prima)
    // Frantoio;  //Qui è dove le Olive vengono Molite per produrre Olio
    // Distributor; //Questo ruolo distribuisce l'Olio ai Rivenditori
    // Retailer; // Cliente finale che compra dal Rivenditore

    //I vari Step dell'Olio in EVO Oil supply chain
    enum STAGE {
        Frantoio,
        Distribution,
        Retail,
        sold
    }
    //using this we are going to track every single product the owner orders

    uint256 public productCtr = 0; //Conteggio Prodotti
    uint256 public fraCtr = 0; //Conteggio Frantoi
    uint256 public disCtr = 0; //Conteggio Distributori
    uint256 public retCtr = 0; //Conteggio Rivenditori
```

Figura 2- Codice sorgente dello "smart contract" [1 di 6]

Il contratto “SupplyChain.sol” include diverse funzioni per gestire la “supply chain” utilizzando la “blockchain”. Ecco una descrizione dettagliata di ciascuna di queste funzioni:

- constructor(): questa è una funzione speciale che viene eseguita automaticamente quando lo “smart contract” viene distribuito sulla “blockchain”. Nel caso del contratto SupplyChain, il costruttore imposta l'owner del contratto come la persona che ha distribuito il contratto.

- onlyByOwner(): questo è un modificatore che viene utilizzato per garantire che solo l'owner del contratto possa utilizzare determinate funzioni. Ad esempio, se una funzione è preceduta dal modificatore "onlyByOwner", significa che solo l'owner del contratto può chiamare tale funzione.

```
    //La struttura product per raccogliere le informazioni sul Prodotto
(Olio)
    struct product {
        uint256 id; //Id unico Prodotto
        string name; //Nome Prodotto
        string description; //Descrizione Prodotto
        uint256 FRAid; //Id del Frantoio utilizzato per questo particolare
Prodotto
        uint256 DISid; //Id del Distributore per questo particolare Prodotto
        uint256 RETid; //Id del Rivenditore per questo particolare Prodotto
        STAGE stage; //Step corrente del Prodotto
    }

    //Per salvare tutti i prodotti nella blockchain
    mapping(uint256 => product) public ProductStock;

    //Per mostrare lo status del prodotto nell'applicazione web
    function showStage(uint256 _productID)
        public
        view
        returns (string memory)
    {
        require(productCtr > 0);
        if (ProductStock[_productID].stage == STAGE.Frantoio)
            return "Frantoio";
        else if (ProductStock[_productID].stage == STAGE.Distribution)
            return "Distribuzione";
        else if (ProductStock[_productID].stage == STAGE.Retail)
            return "In Vendita";
        else if (ProductStock[_productID].stage == STAGE.sold)
            return "Venduto";
    }

    //La struttura frantoio per raccogliere le informazioni sul Frantoio
    struct frantoio {
        address addr;
        uint256 id; //Id del Frantoio
        string name; //Nome del Frantoio
        string place; //Luogo in cui si trova il Frantoio
    }

    //Per salvare tutti i Frantoi sulla blockchain
    mapping(uint256 => frantoio) public FRA;
```

Figura 3 - Codice sorgente dello "smart contract" [2 di 6]

- showStage(): questa funzione pubblica permette di visualizzare lo stato corrente di un prodotto nell'applicazione web. Riceve come input l'ID del prodotto e restituisce una stringa che descrive lo stato corrente del prodotto, come "Frantoio", "Distribuzione", "In Vendita" o "Venduto".

```
    //La struttura distributor per raccogliere le informazioni sul
Distributore
    struct distributor {
        address addr;
        uint256 id; //distributor id
        string name; //Name of the distributor
        string place; //Place the distributor is based in
    }

    //Per salvare tutti i Distributori sulla blockchain
    mapping(uint256 => distributor) public DIS;

    //La struttura retailer per raccogliere le informazioni sul Rivenditore
    struct retailer {
        address addr;
        uint256 id; //retailer id
        string name; //Name of the retailer
        string place; //Place the retailer is based in
    }

    //Per salvare tutti i Rivenditori sulla blockchain
    mapping(uint256 => retailer) public RET;

    //La funzione addFrantoio() serve ad aggiungere i Frantoi
    //e con la funzione onlyByOwner() dichiarata in precedenza,
    //solo l'owner dello smart contract può aggiungere un nuovo Frantoio
    function addFrantoio(
         address _address,
         string memory _name,
         string memory _place
     ) public onlyByOwner() {
        fraCtr++;
        FRA[fraCtr] = frantoio(_address, fraCtr, _name, _place);
    }

    //La funzione addDistributor() serve ad aggiungere i Distributori
    //e con la funzione onlyByOwner() dichiarata in precedenza,
    //solo l'owner dello smart contract può aggiungere un nuovo Distributore
    function addDistributor(
         address _address,
         string memory _name,
         string memory _place
     ) public onlyByOwner() {
        disCtr++;
        DIS[disCtr] = distributor(_address, disCtr, _name, _place);
    }
```

Figura 4 - Codice sorgente dello "smart contract" [3 di 6]

- addFrantoio(), addDistributor(), addRetailer(): queste funzioni pubbliche permettono all'owner del contratto di aggiungere un nuovo frantoio, distributore o rivenditore alla “supply chain”. Riceve come input l'indirizzo Ethereum, il nome e il luogo del partecipante, incrementa il contatore e assegna un ID univoco.

```
    //La funzione addRetailer() serve ad aggiungere i Rivenditori
    //e con la funzione onlyByOwner() dichiarata in precedenza,
    //solo l'owner dello smart contract può aggiungere un nuovo Rivenditore
    function addRetailer(
         address _address,
         string memory _name,
         string memory _place
     ) public onlyByOwner() {
        retCtr++;
        RET[retCtr] = retailer(_address, retCtr, _name, _place);
    }

    //per controllare se l'indirizzo del Frantoio che sta interagendo
    //con l'interfaccia web è stato registrato e autorizzato dell'owner nella
blockchain
    function findFRA(address _address) private view returns (uint256) {
        require(fraCtr > 0);
        for (uint256 i = 1; i <= fraCtr; i++) {
            if (FRA[i].addr == _address) return FRA[i].id;
        }
        return 0;
    }

    //Il prodotto passa dallo stato Frantoio allo stato Distributore.
    //Il Distributore si prende in carico del prodotto.
    function Distribute(uint256 _productID) public {
        require(_productID > 0 && _productID <= productCtr);
        uint256 _id = findDIS(msg.sender);
        require(_id > 0);
        require(ProductStock[_productID].stage == STAGE.Frantoio);
        ProductStock[_productID].DISid = _id;
        ProductStock[_productID].stage = STAGE.Distribution;
    }

    //per controllare se l'indirizzo del Distributore che sta interagendo
    //con l'interfaccia web è stato registrato e autorizzato dell'owner nella
blockchain
    function findDIS(address _address) private view returns (uint256) {
        require(disCtr > 0);
        for (uint256 i = 1; i <= disCtr; i++) {
            if (DIS[i].addr == _address) return DIS[i].id;
        }
        return 0;
    }
```

Figura 5 - Codice sorgente dello "smart contract" [4 di 6]

- findFRA(), findDIS() e findRET(): queste funzioni private permettono di controllare se l'indirizzo Ethereum dei vari ruoli (Frantoi, Distributori e Rivenditori) che sta interagendo con l'interfaccia web, è stato registrato e autorizzato dall'owner nella "blockchain".

```
    //Il prodotto passa dallo stato Distributore allo stato Rivenditore.
    //Il Rivenditore si prende in carico del prodotto.
    function Retail(uint256 _productID) public {
        require(_productID > 0 && _productID <= productCtr);
        uint256 _id = findRET(msg.sender);
        require(_id > 0);
        require(ProductStock[_productID].stage == STAGE.Distribution);
        ProductStock[_productID].RETid = _id;
        ProductStock[_productID].stage = STAGE.Retail;
    }

    //per controllare se l'indirizzo del Rivenditore che sta interagendo
    //con l'interfaccia web è stato registrato e autorizzato dell'owner nella
blockchain
    function findRET(address _address) private view returns (uint256) {
        require(retCtr > 0);
        for (uint256 i = 1; i <= retCtr; i++) {
            if (RET[i].addr == _address) return RET[i].id;
        }
        return 0;
    }

    //Il prodotto passa dallo stato Rivenditore allo stato Venduto.
    //Il prodotto ora è in mano al Consumatore
    function sold(uint256 _productID) public {
        require(_productID > 0 && _productID <= productCtr);
        uint256 _id = findRET(msg.sender);
        require(_id > 0);
        require(_id == ProductStock[_productID].RETid);
        //Solo i Rivenditori registrati dall'owner e autorizzati
        //possono segnare il prodotto come Venduto
        require(ProductStock[_productID].stage == STAGE.Retail);
        ProductStock[_productID].stage = STAGE.sold;
    }
```

Figura 6 - Codice sorgente dello "smart contract" [5 di 6]

- Distribute(), Retail() e sold(): queste funzioni pubbliche permettono di aggiornare lo stato di un lotto e tracciarne il movimento attraverso la catena di fornitura. Solo i rispettivi ruoli di competenza potranno usare queste funzioni e solo se lo step corrente del prodotto di cui si vuole aggiornare lo stato è lo step immediatamente precedente.

```
    //Per aggiungere nuovi prodotti nella lista dei prodotti commissionati
dall'owner
    //function addProduct(string memory _name, string memory _description)
public onlyByOwner()
    function addProduct(string memory _name, string memory _description)
public

    {
        require((fraCtr > 0) && (disCtr > 0) && (retCtr > 0));
        productCtr++;
        uint256 _id = findFRA(msg.sender); //con queste 2 righe solo un
Frantoio registrato
        require(_id > 0); //può inserire un prodotto
        ProductStock[productCtr] = product(
            productCtr,
            _name,
            _description,
            _id,
            0,
            0,
            STAGE.Frantoio
        );
    }
}
```

Figura 7 - Codice sorgente dello "smart contract" [6 di 6]

- addProduct() è una funzione pubblica che viene utilizzata per aggiungere un nuovo lotto alla catena di fornitura gestita dallo “smart contract”. Questa funzione prende alcuni parametri di input, come il nome del prodotto, la descrizione del prodotto e l'ID del frantoio lo ha prodotto. Utilizza questi dati per creare una nuova istanza della struttura product e assegnare un ID univoco al lotto. La funzione quindi aggiunge questo nuovo prodotto alla mappa ProductStock che viene utilizzata per tracciare tutti i lotti nella catena di fornitura. Anche se pubblica, questa è una funzione che richiamando la funzione findFRA() permette solo ai frantoi autorizzati precedentemente dall’owner del contratto di inserire nuovi lotti di Olio.

3.2.3 Compilazione e deploy dello “smart contract” con Truffle

Il passo successivo nello sviluppo del “backend” è la compilazione e il deploy dello “smart contract” sulla “blockchain” di test di Ethereum utilizzando Truffle.

Per compilare il contratto, è necessario aprire il terminale nella cartella del progetto e digitare il comando "*truffle compile*". Questo compilerà lo “smart contract” in un formato che può essere utilizzato dal motore di esecuzione della “blockchain” di Ethereum.

Una volta compilato il contratto, è possibile effettuarne il deploy sulla “blockchain” di test utilizzando il comando "*truffle migrate*". Questo processo caricherà il contratto sulla “blockchain” e lo renderà disponibile per l'utilizzo.

Durante il deploy, Truffle mostrerà l'indirizzo del contratto sulla “blockchain”, l’hash della transazione che sarà necessario utilizzare per interagire con esso nell'applicazione “frontend”. Inoltre, sarà necessario utilizzare il proprio account Ethereum (quello dell’owner) per firmare il deploy del contratto. Truffle utilizza il primo account/wallet creato in Ganache di default.

```
$ truffle migrate

Compiling your contracts...
===========================
> Compiling ./contracts/SupplyChain.sol
> Artifacts written to /build/contracts

Starting migrations...
======================
> Network name:    'ganache'
> Network id:      5777
> Block gas limit: 6721975


1_initial_migration.js
======================

   Deploying 'Migrations'
   ----------------------
   > transaction hash:    0x5471e2e69b5f1d2c8658a7a064a83d615c430293d29d9b7
   > Blocks: 0            Seconds: 0
   > contract address:    0x2aD7b406ebe2D1A95412eb0B13Df6cb19C85A455
   > block number:        1
   > block timestamp:     1650179128
   > account:             0x981aE0Cb860F5B094a9d00F41446E7a81d4eCF9F
   > balance:             99.9965324
   > gas used:            238381 (0x3a0cd)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00476762 ETH


   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:          0.00476762 ETH


Summary
=======
> Total deployments:   1
> Final cost:          0.00476762 ETH
```

Figura 8 - Output del comando "truffle migrate" che, tra le altre cose, mostra l'indirizzo del contratto, l'hash della transazione e l'account/wallet dell owner che effettua il deploy del contratto sulla blockchain locale.

3.3 Sviluppo del “frontend”

Il “frontend” dell'applicazione è stato sviluppato utilizzando React, un framework JavaScript per lo sviluppo di interfacce utente. La struttura della cartella del “frontend” include diverse sottocartelle per le diverse sezioni dell'applicazione.

3.3.1 Configurazione Server Web con Node js

Per iniziare a sviluppare il “frontend”, è stato necessario configurare un server web con Node js, un runtime JavaScript che permette di eseguire il codice JavaScript al di fuori del browser. È stato utilizzato per creare il server web e gestire le richieste HTTP per l'applicazione.

3.3.2 Web App con React – Sezioni del “frontend”

Una volta configurato il server web, è stato possibile iniziare a sviluppare le varie sezioni dell'applicazione con React.

```
import './App.css';
import AssignRoles from './AssignRoles';
import Home from './Home';
import AddProduct from './AddProduct';
import Supply from './Supply'
import Track from './Track'
import { BrowserRouter as Router, Switch, Route } from "react-router-dom"

function App() {
  return (
    <div className="App">
      <Router>
        <Switch>
          <Route path="/" exact component={Home} />
          <Route path="/roles" component={AssignRoles} />
          <Route path="/addproduct" component={AddProduct} />
          <Route path="/supply" component={Supply} />
          <Route path="/track" component={Track} />
        </Switch>
      </Router>
    </div>
  );
}

export default App;
```

Figura 9 - File "App.js" che gestisce le "Route" dell'applicativo.

Il file “App.js” rappresenta il componente principale dell'applicazione “React” in cui viene utilizzato il componente "Router" per gestire il rendering dei componenti corretti a seconda dell'URL corrente dell'applicazione.

Il componente "Switch" rende possibile specificare una serie di "Route" (che rappresentano le diverse pagine o sezioni dell'applicazione) e il componente "Route" specifica quali componenti verranno mostrati per ogni percorso specifico.

La "Route" con il percorso "/", mostrerà il componente "Home" per l'URL radice dell'applicazione, mentre gli altri percorsi mostreranno i componenti corrispondenti. Ad esempio, il percorso "/roles" mostrerà il componente "AssignRoles".

Le sezioni “frontend” dell'applicazione includono:

- Homepage (EVO Oli Supply Chain Flow) (/)

La homepage dell'applicazione presenta una panoramica del flusso di lavoro della “supply chain” dell'olio extravergine di oliva, dalla produzione al consumo finale.

- AssignRoles (/roles)

La pagina di registrazione permette all’owner del contratto di inserire i vari ruoli che hanno il diritto di partecipare alla “supply chain”.

- AddProduct (/addproduct)

La pagina dell’aggiunta prodotti permette ai vari frantoi regitrati di creare nuovi ordini di olio extravergine di oliva sotto forma di lotti. Richiede l'inserimento di informazioni come il tipo di lotto e la descrizione dettagliata, inoltre assegna un’ID univoco al lotto creato.

- Supply (/supply)

La pagina di controllo della “supply chain” permette ai partecipanti registrati di visualizzare e gestire gli ordini esistenti nella “supply chain”. Consente di aggiornare lo stato degli ordini in base all'avanzamento nella “supply chain” e solo se autorizzati nella pagina di registrazione dall’owner del contratto.

- Track (/track)

La pagina di tracciamento permette agli utenti finali, nonché a tutti i partecipanti, di verificare l'autenticità e la provenienza del lotto attraverso l’ID univoco assegnato in fase di creazione.

4. Test della “dApp”: Piattaforma per il tracciamento dell'Olio di qualità

Il quarto capitolo della tesi descrive il caso studio dell'utilizzo della “dApp” sviluppata per il tracciamento e l'autenticità dell'olio extravergine di oliva di qualità. In particolare viene descritto come è stata utilizzata l'applicazione per gestire il flusso di lavoro della “supply chain” dell'olio, dalla produzione al consumo finale.

Inizialmente viene avviato il server web in locale con “NodeJs” utilizzando il comando “npm start”

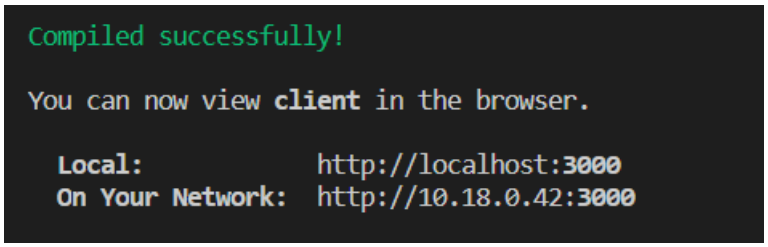

Figura 10 - Avvio server web in locale con “NodeJs”

Inserendo nel browser l’indirizzo http://localhost:3000 si apre la “Home” dove vengono mostrate e spiegate sinteticamente le varie sezioni della “dApp”. Ogni sezione è dedicata ai rispettivi partecipanti della filiera.

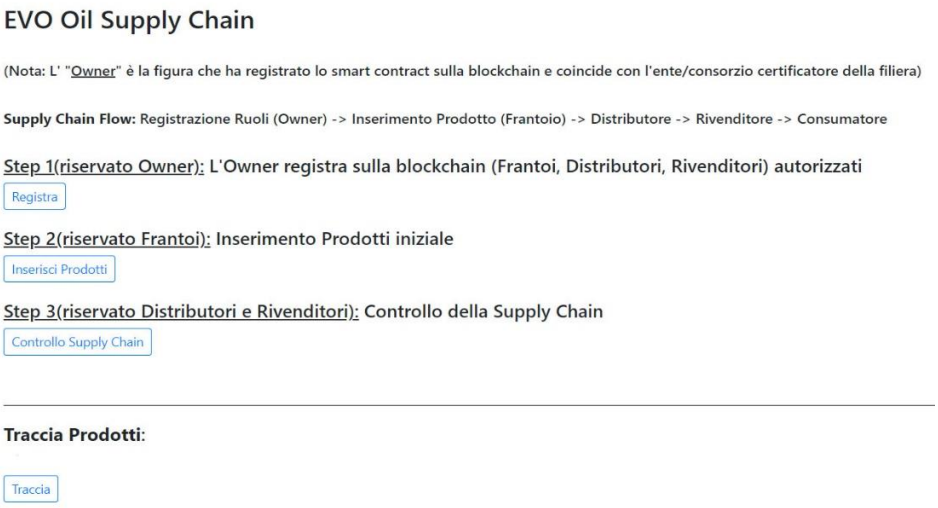

Figura 11 - Pagina principale dalla quale sono accessibili le varie sezioni della "dApp"

4.1 Inserimento dei ruoli da parte dell' "Owner" ("one time step")

Come primo passo, spostandoci nella pagina "AssignRoles", l'amministratore della "dApp" (l'owner) seleziona il proprio indirizzo Ethereum tramite Metamask e lo rende disponibile all'interfaccia web.

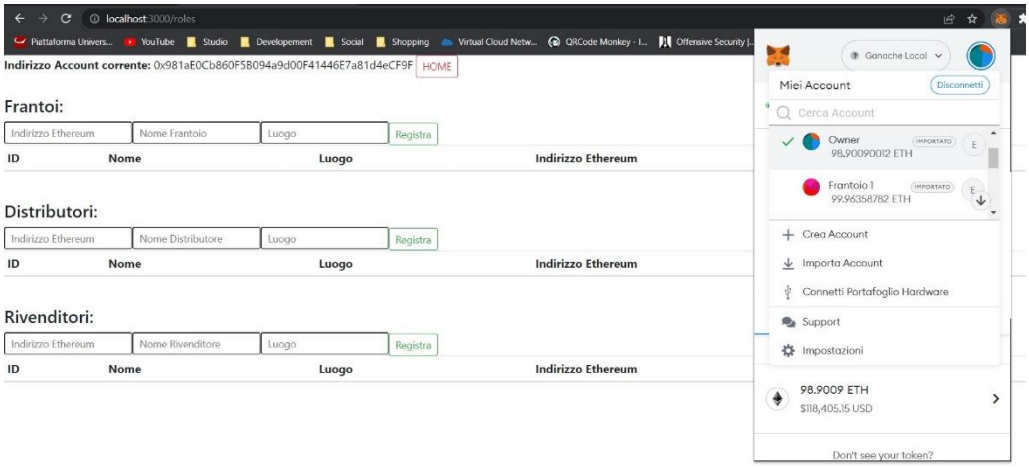

Figura 12 - Pagina dove l' "Owner" registra i ruoli autorizzati della "Supply Chain" (prima dell'inserimento)

Poi procede con l'inserimento dei diversi ruoli nella "supply chain" (frantoi, distributori, rivenditori) utilizzando l'applicazione. Questo è un passo necessario per impostare la struttura della "supply chain" e autorizzare i diversi ruoli.

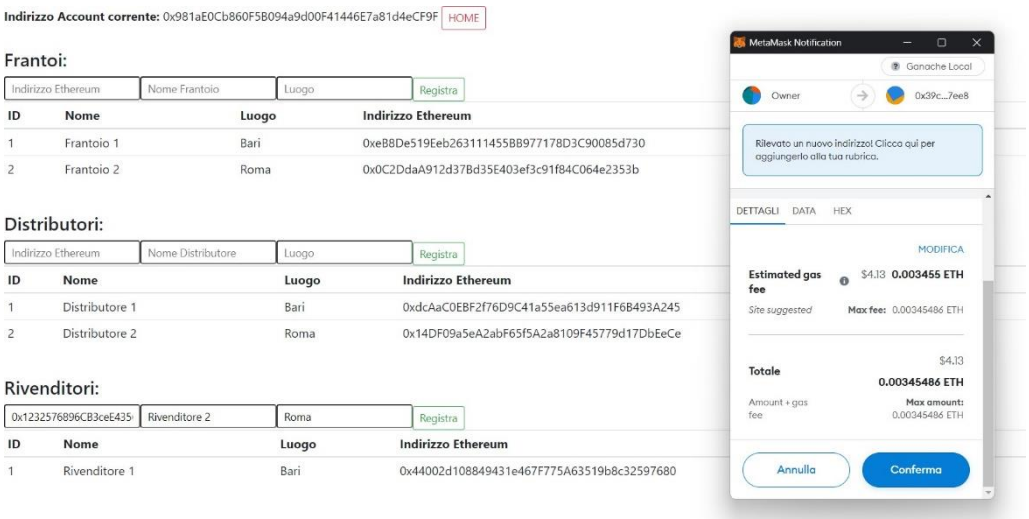

Figura 13 - Pagina dove l'Owner registra i ruoli autorizzati della "Supply Chain" (ruoli inseriti)

4.2 Aggiunta nuovo Prodotto (Solo Frantoio)

Il frantoio, nella pagina “AddProduct”, utilizza l'applicazione per aggiungere un nuovo prodotto sotto forma di lotto di olio extravergine di oliva alla “supply chain”. Inserisce informazioni come il tipo di olio, l’uliveto di provenienza, la data di raccolta e di molitura, la tipologia di olive utilizzate, l’acidità, ecc.

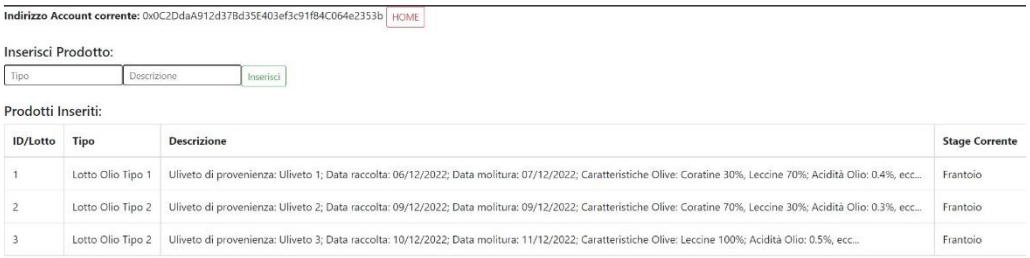

Figura 14 - Pagina in cui il Frantoio inserisce i nuovi lotti di olio

4.3 Distribuzione (Solo Distributore)

Il distributore invece, nella pagina “Supply”, utilizza l'applicazione per gestire la distribuzione del lotto ai rivenditori. Aggiornando lo stato dell'ordine per indicare che il prodotto è stato consegnato ai rivenditori.

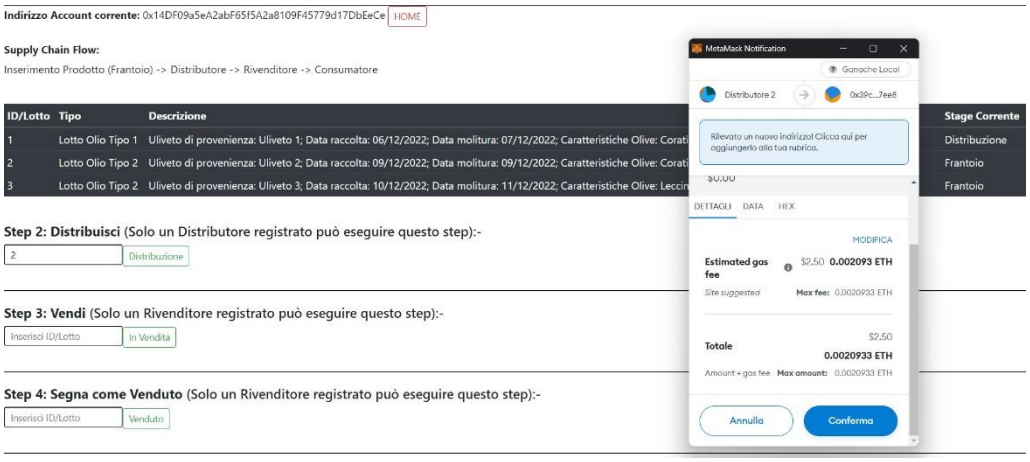

Figura 15 - Pagina “Supply” in cui Distributori e Rivenditori si prendono carico dei lotti con “Metamask” in evidenza per confermare la transazione.

4.4 Vendita (Solo Rivenditori)

I rivenditori gestiscono la vendita del prodotto ai consumatori finali. Aggiornano lo stato dell'ordine per indicare che il prodotto è disponibile per l'acquisto. Successivamente, aggiornano lo stato per indicare che il prodotto è stato venduto e fornito al consumatore.

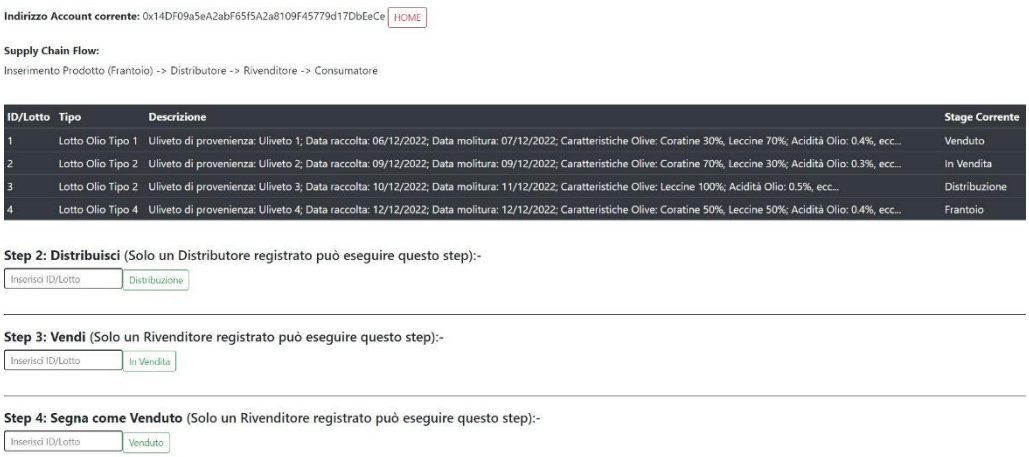

Figura 16 - Pagina "Supply" che il cambiamento il cambiamento dello stato di alcuni prodotti

4.5 Tracking del Prodotto (Consumatori)

I consumatori, nella pagina "Track", utilizzano il numero di tracciamento fornito dal rivenditore per verificare l'autenticità e la provenienza del prodotto acquistato potendo visualizzare le informazioni sulla "supply chain".

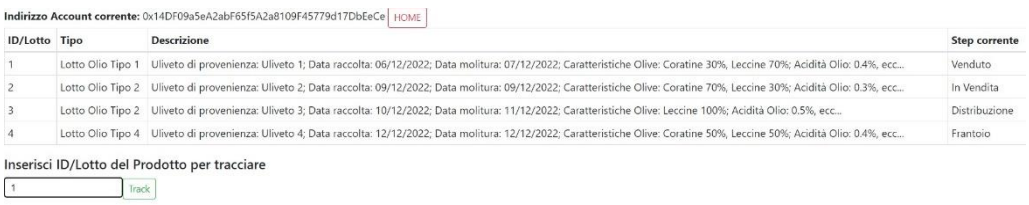

Figura 17 - Pagina "Track" che mostra descrizione e stato dei prodotti registrati sulla "blockchain"

Una volta premuto sul pulsante “Track” si apre una nuova pagina con i dettagli richiesti.

Figura 18- Pagina che mostra il dettaglio della tracciatura del prodotto selezionato.

4.6 Test Controllo Autorizzazioni

Il primo test da effettuare è assicurarsi che solo l'owner possa inserire i vari ruoli. Provando ad utilizzare un indirizzo Ethereum diverso da quello dell'owner si verifica l'errore.

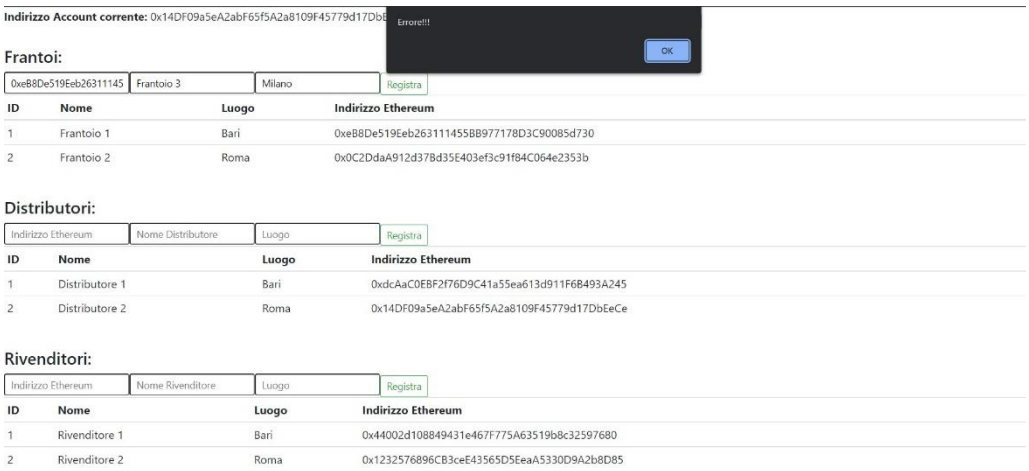

Figura 19 - Tentativo di registrazione di un ruolo da parte di un utente diverso dall'Owner.

Inoltre ci assicuriamo che solo un frantoio autorizzato possa inserire nuovi prodotti (nemmeno l'owner può farlo). Provando ad utilizzare un indirizzo Ethereum diverso da quello dei frantoi autorizzati si verifica l'errore.

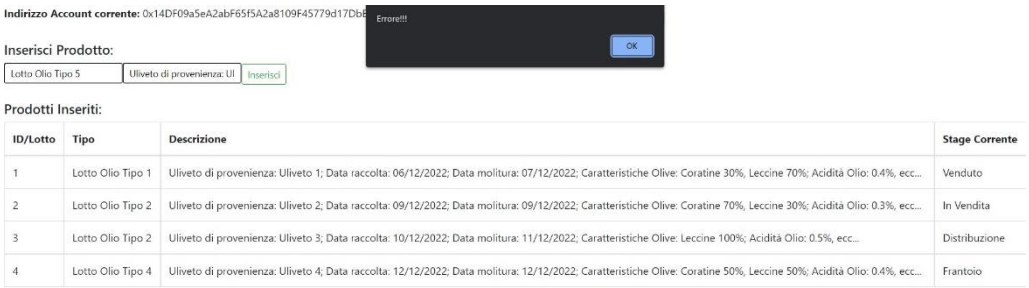

Figura 20 - Tentativo di registrazione di un nuovo prodotto da parte di un utente diverso da un Frantoio autorizzato.

Inoltre, solo un distributore autorizzato può distribuire i prodotti. Provando ad interagire con la "dApp" con un indirizzo Ethereum diverso da quello di un Distributore autorizzato si verifica l'errore.

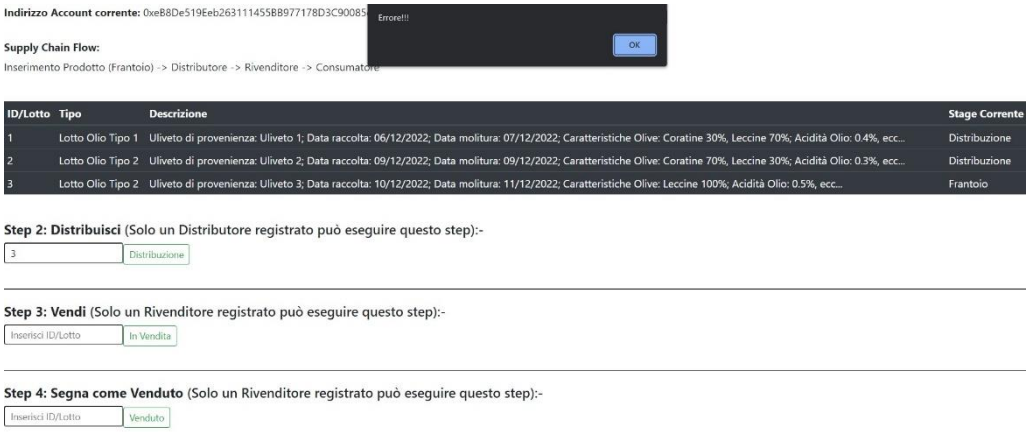

Figura 21 - Tentativo di assegnazione di un prodotto da parte di un utente diverso da un Rivenditore autorizzato.

Lo stesso vale per il Rivenditore. Se non si è autorizzati, si verifica l'errore.

Un ultimo controllo viene fatto quando un ruolo, ad esempio un Rivenditore, cerca di assegnarsi un prodotto che non si trova nello Stato immediatamente precedente al suo. Se un Rivenditore vuole prendere il controllo di un prodotto che si trova ancora assegnato ad un Frantoio per
metterlo nello stato "In Vendita", l'applicativo, a seguito di un controllo, non lo permette restituendo un errore.

Conclusione

In conclusione, lo sviluppo di una “dApp” per il tracciamento e l'autenticità dell'olio extravergine di oliva è stato possibile utilizzando la tecnologia “blockchain” e la piattaforma Ethereum, insieme agli strumenti di sviluppo ganache, truffle, nodejs e la libreria web3.js. Il processo di sviluppo è stato effettuato seguendo una metodologia iterativa, che ha previsto la definizione dei requisiti e dei casi d'uso, la progettazione dell'interfaccia utente e dello “smart contract”, il test della “dApp” e il deployment sulla “blockchain”. Grazie all'utilizzo di tale tecnologia, è stato possibile garantire la trasparenza e la sicurezza delle informazioni registrate sulla “dApp” e la verifica dell'autenticità del prodotto da parte dei consumatori finali.

La “dApp” ha permesso di gestire in modo trasparente e sicuro la “supply chain” dell'olio di qualità, consentendo ai diversi ruoli coinvolti (frantoio, distributore, rivenditore, consumatore) di interagire tra loro e di tracciare il percorso del prodotto dalla produzione alla vendita al dettaglio.

Durante lo sviluppo sono state affrontate diverse sfide, come la gestione dei ruoli nella “supply chain” e le autorizzazioni connesse. Tuttavia, l'utilizzo della tecnologia “Blockchain” ha reso possibile superare queste sfide e di offrire una soluzione sicura e trasparente per il tracciamento dell'olio di qualità.

Sitografia

1. Definizione di “Blockchain” – Wikipedia.org (https://it.wikipedia.org/wiki/Blockchain)

2. Cos’è Ethereum – Coinbase.com (https://www.coinbase.com/it/learn/crypto-basics/what-is-ethereum)

3. Definizione di “Permissionless Blockchain” – Foley.com (https://www.foley.com/en/insights/publications/2021/08/types-of-blockchain-public-private-between)

4. Origine del termine “contratto intelligente” – Ricercamy.com (https://ricercamy.com/2021/08/come-trovare-una-valido-solidity-developer)

Indice delle figure

Ringraziamenti

"Desidero esprimere la mia sincera gratitudine a tutti coloro che mi hanno supportato durante il mio percorso accademico e nella realizzazione di questa tesi.

In primo luogo, desidero ringraziare la mia famiglia per il loro sostegno, la loro pazienza e il loro incoraggiamento. Senza di loro, non sarei riuscito a raggiungere questo traguardo così importante nella mia vita.

Voglio poi ringraziare la mia compagna di vita per avermi sostenuto in ogni fase del mio percorso universitario, per avermi incoraggiato nei momenti difficili e per aver condiviso con me questa avventura.

Desidero inoltre ringraziare il mio relatore per la guida e la disponibilità durante la stesura della tesi.

Infine, desidero ringraziare tutte le persone che mi hanno offerto supporto morale e che mi hanno dato la forza di portare a termine questo progetto. Il vostro contributo è stato fondamentale per il mio successo.

Grazie di cuore a tutti voi."