UNIVERSITA' POLITECNICA DELLE MARCHE

FACOLTA' DI INGEGNERIA

Corso di Laurea magistrale in Ingegneria Informatica e dell'Automazione

Blockchain: Reputazione e Trust in una filiera alimentare

Trust and reputation in a food supply chain by means DLTs

Relatore: Chiar.mo

Prof. Spalazzi Luca

Correlatore:

Ing. Pirani Massimiliano

Tesi di Laurea di:

Alessio Cacopardo

A.A. 2022 / 2023

Indice

Capitolo 1

Introduzione

Questa tesi ha l'obiettivo di fornire nuove idee sull'utilizzo dei recenti progressi della tecnologia dei token non fungibili (NFT) e Soulbound, che stanno diventando sempre più importanti nel contesto delle blockchain. In particolare, si discute come queste tecnologie possano consentire un ruolo rinnovato e rafforzato del concetto di Internet of Everything nei processi complessi di Industria 5.0, dove le dimensioni sociali, societarie e tecniche si fondono in un contesto applicativo irriducibile.[1]

In questo ambito diventa di fondamentale importanza il concetto di reputazione e trust (fiducia), il quale è strettamente correlato ad un altro concetto più ampio che è quello della Cybersecurity, argomento sempre di grande attualità. Senza dimenticare un altro importante tassello che è quello della privacy, altra materia molto in auge. Le tecnologie delle blockchain, o più in generale delle DLT (Distributed Ledger Technology), rappresentano validi strumenti per realizzare determinati obiettivi saldamente legati a tali concetti che diventano ancora più peculiari in una supply chain (catena di approvvigionamento).

L'utilizzo di queste tecnologie può migliorare la reputazione e la fiducia all'interno di una catena di approvvigionamento. Le blockchain possono fornire una maggiore trasparenza e tracciabilità delle informazioni, consentendo ai consumatori di conoscere meglio la provenienza dei prodotti che acquistano. Ciò può aumentare la fiducia dei consumatori nella filiera alimentare e migliorare la reputazione delle aziende coinvolte. In effetti, anche la reputazione e la fiducia di tali attori possono essere tracciate ed automatizzate, rendendo l'intero processo più affidabile, sicuro e dinamico. Le potenzialità di questo approccio sono illustrate attraverso un semplice ma significativo progetto su un caso di studio industriale riguardante la catena di approvvigionamento alimentare, che attualmente è oggetto di ricerca nell'ambito di un'attività di ricerca collaborativa europea.[2]

1.1 Progetto Enough

Il progetto europeo Enough è un'iniziativa finanziata dal programma Horizon 2020 dell'Unione Europea che mira a ridurre le emissioni di gas serra del settore alimentare, in linea con la strategia Farm to Fork dell'UE. Il progetto coinvolge 30 partner di 9 paesi dell'UE, Regno Unito, Norvegia e Turchia, coordinati da SINTEF Ocean (Norvegia).[1] [2]

Il progetto ha come obiettivi principali:

- Ridurre le emissioni di gas serra del settore alimentare di almeno il 50% entro il 2050 e raggiungere la neutralità carbonica entro il 2050;
- Ridurre il consumo di energia e aumentare l'efficienza energetica lungo la catena alimentare, in particolare nei processi di refrigerazione che sono responsabili del 60% degli alimenti e del 70% delle emissioni;
- Migliorare la sostenibilità globale integrata dei sistemi alimentari, tenendo conto degli aspetti sociali, sanitari, ambientali ed economici;
- Sensibilizzare i responsabili politici, le imprese, gli investitori, gli imprenditori, le istituzioni, le parti interessate e i cittadini alle soluzioni sistemiche innovative selezionate e al loro potenziale di adozione su scala dell'UE.

Per raggiungere questi obiettivi, il progetto prevede di:

- Fornire informazioni e stabilire una baseline sulle emissioni del settore alimentare nel 1990 e nel 2020 e prevedere le emissioni nel 2030 e nel 2050, considerando diversi fattori chiave;
- Definire tabelle di marcia per descrivere l'intera catena di approvvigionamento alimentare, dalla raccolta al consumo, e stabilire nuove pratiche per ciascun anello della catena alimentare (lavorazione, confezionamento, stoccaggio, trasporto, ecc.);
- Dimostrare soluzioni tecnologiche promettenti applicate nei sette settori alimentari: carne, pesce, frutta/verdura, latticini, trasporti, vendita al dettaglio e domestico;
- Coinvolgere diversi settori della catena alimentare, tra cui la produzione primaria, la trasformazione, il confezionamento, lo stoccaggio, il trasporto, la vendita al dettaglio e il consumo domestico;
- Utilizzare una metodologia multidisciplinare che integra aspetti tecnici, economici, sociali e ambientali.

[1] https://cordis.europa.eu/project/id/101036588
[2] https://www.zerosottozero.it/Enough

Il progetto ENOUGH si propone quindi di contribuire a una trasformazione sostenibile e verde del sistema alimentare europeo, in linea con il Green Deal europeo e gli obiettivi globali di sviluppo sostenibile.

La Commissione europea intende utilizzare le innovazioni delle tecnologie blockchain per contribuire alla lotta al cambiamento climatico. Le blockchain sono strumenti potenti che possono migliorare significativamente la trasparenza, la responsabilità e la tracciabilità delle emissioni di gas serra. Aiuta le aziende a fornire dati più precisi, affidabili, standardizzati e prontamente disponibili sulle emissioni di carbonio.

Le blockchain possono essere utilizzate attraverso gli smart contract per calcolare, tracciare e comunicare meglio la riduzione dell'impronta di carbonio lungo l'intera catena. Può fornire autenticazione istantanea, verifica dei dati in tempo reale e registrazioni chiare dei dati.

Le tecnologie blockchain possono trasformare gli sforzi individuali delle aziende in uno sforzo di rete. Inoltre, possono individuare chiaramente i contributi che i singoli attori apportano per ridurre l'impronta di carbonio. Lo spirito di concorrenza e gli incentivi basati sul mercato creano una situazione vantaggiosa per tutti.

Le startup della tecnologia pulita svolgono un ruolo fondamentale in questo processo. Sviluppano piattaforme abilitate alle blockchain che riuniscono tutte le parti interessate, tra cui aziende, governi e cittadini.

L'approccio decentralizzato delle blockchain offre sia ampiezza che profondità. Coinvolge e consente a tutti di partecipare al calcolo. Consente di tracciare e comunicare le riduzioni delle emissioni di gas serra lungo l'intera catena di approvvigionamento, compresi produttori, fornitori, distributori e consumatori.

Le innovazioni nelle tecnologie blockchain sono potenti strumenti per l'azione collettiva nella lotta al cambiamento climatico. Riconoscere il valore unico delle startup delle tecnologie pulite in questo processo è di grande importanza. Gli investitori pubblici e privati stanno iniziando a prendere atto del loro valore unico.[3]

[3]https://digital-strategy.ec.europa.eu/en/policies/blockchain-climate-action

1.2 IoE e Industria 5.0

L'Internet of Everything (IoE) può essere definito come "*l'interconnessione discontinua e il coordinamento autonomo di un numero enorme di elementi informatici e sensori, entità inanimate e viventi, persone, processi e dati attraverso l'infrastruttura di Internet*"[4].

Industria 5.0[5], invece, deve andare "*oltre l'efficienza e la produttività come unici obiettivi mettendo la ricerca e l'innovazione al servizio della transizione verso un'industria sostenibile, incentrata sull'uomo e resiliente*".[3].
Poiché l'Industria 5.0 integra e amplia le caratteristiche di Industria 4.0, tutti i modelli di distribuzione e decentralizzazione esistenti devono essere progettati per essere sostenibili.

La progettazione ingegneristica sistemica deve considerare l'importanza della connessione in rete di persone, processi, dati e cose, ovvero IoE per Industria 5.0. [4]. Inoltre, la progettazione deve includere l'etica e i valori come fattori chiave per consentire la simbiosi uomo-macchina in Industria 5.0 [5].

Per un Industria 5.0 sostenibile, dove ci sono molti attori indipendenti e con interessi diversi, è importante che ci sia "fiducia" tra tutti gli attori coinvolti. Le tecnologie dei registri distribuiti (DLT) e in particolare le blockchain possono aiutare a creare questa "fiducia". La natura distribuita delle DLT significa che nessun attore può essere superiore agli altri. La maggior parte dei lavori che trattano le DLT si concentra sulla fiducia dei dati e delle comunicazioni, grazie all'immutabilità e la tracciabilità delle transazioni garantite dalle blockchain (ad esempio, si veda [6–8]). Recentemente, alcuni lavori propongono di utilizzare le DLT per migliorare la fiducia e la conformità dei processi aziendali (ad esempio, per l'orchestrazione e la collaborazione si veda [9–11], per la coreografia si veda [12, 13]). Fino ad oggi, solo pochi lavori si sono concentrati sulla gestione della fiducia nel caso di società eterogenee di attori umani ed entità artificiali che vengono intenzionalmente riunite in un processo complesso.

1.3 Blockchain e Smart Contract

Possiamo definire una blockchain come un libro contabile condiviso, immutabile e facile da usare, che all'interno di una rete commerciale si rivela utile per registrare transazioni e tracciare i beni. I beni possono essere materiali come una casa, un terreno o un'auto o immateriali come un brevetto, un marchio o il diritto d'autore su un libro. Qualsiasi bene può essere tracciato e scambiato (venduto o acquistato) su una rete blockchain con una notevole riduzione dei rischi e dei costi di tutte le parti coinvolte. Con le blockchain si ottengono informazioni immediate e accurate, sono condivise, trasparenti e memorizzate

[4]https://ioe.org/
[5]https://research-and-innovation.ec.europa.eu/

su un libro mastro che non può essere manomesso a cui possono accedere solo partecipanti autorizzati. Attraverso le reti blockchain è possibile tracciare ordini, pagamenti, conti, produzione ed è possibile verificare ogni transazione nel dettaglio.[6]

Per "blockchain", si intende un insieme di dati (o transazioni) protetti da crittografia, assimilabili appunto ad una "catena di blocchi". In ogni blocco della catena sono contenute informazioni, un timestamp, un hash del blocco e un "puntatore" al blocco precedente. Quest'ultimo serve a collegare due blocchi tra loro e a impedire qualsiasi manomissione del libro contabile o dell'ordine delle transazioni.

Ci sono moltissime compagnie che stanno già adottando le blockchain e nei settori più diversi. Ecco alcuni esempi di ambiti in cui le blockchain stanno funzionando:

- **Blockchain nell'industria alimentare**: Con l'uso delle blockchain i marchi del settore alimentare possono tracciare il percorso di un prodotto alimentare dalla sua origine alla sua consegna. In questo caso il vantaggio del conoscere l'intera filiera agroalimentare è rappresentato dal fatto che è possibile valutare e verificare in ogni passaggio lo stato di conservazione dell'alimento. Difatti questa tesi tratterà proprio questa casistica;

- **Blockchain nel settore delle banche e finanza**: Le blockchain nelle banche apportano notevoli vantaggi come la registrazione in tempo reale delle transazioni indipendentemente dalle festività o dall'orario. Ciò comporta per gli istituti bancari un risparmio dei costi e la diminuzione dei rischi;

- **Blockchain per le criptovalute**: Se c'è un settore che ha maggiormente beneficiato delle blockchain questo riguarda le criptovalute e i Bitcoin. Distribuendo le operazioni su una rete di computer, le blockchain consenteno al Bitcoin e alle altre criptovalute di operare senza la necessità di un'autorità centrale con una notevole riduzione dei rischi e il risparmio su costi di commissione e sui tempi di elaborazione delle transazioni.

- **Blockchain per l'assistenza sanitaria**: Le cartelle cliniche possono essere archiviate nelle blockchain in maniera sicura e senza che possano essere modificate. Restano accessibili solo a persone autorizzate.

- **Blockchain per la registrazione dei diritti di proprietà**: Con le blockchain è possibile eliminare la scansione dei documenti o dei file fisici necessari per la registrazione di una proprietà presso gli enti preposti. Quando l'acquisizione/vendita di una proprietà è registrata sulle blockchain questo rappresenta un atto sicuro, verificabile e registrato in maniera permanente.

[6] https://www.agendadigitale.eu/tag/blockchain/

Parlando di blockchain si deve necessariamente introdurre il concetto di **smart contract**. Gli smart contract sono gli elementi fondamentali delle blockchain. Sono programmi informatici memorizzati nelle blockchain e consentono di convertire i contratti tradizionali in paralleli digitali. I contratti intelligenti sono molto logici e seguono una struttura "if this then that". Ciò significa che sono "programmati" e non possono essere modificati una volta "deployati" in una blockchain.

I contratti sono a tutti gli effetti degli accordi. Cioè, qualsiasi forma di accordo può essere racchiusa nelle condizioni di un contratto. Gli accordi verbali o i contratti scritti con carta e penna sono accettabili per molte cose, ma non sono privi di difetti. Uno dei maggiori problemi di un contratto tradizionale è la necessità di persone fidate che seguano i risultati del contratto. Gli smart contract digitalizzano i contratti trasformando i termini di un accordo in codice informatico che viene eseguito automaticamente quando i termini del contratto sono soddisfatti. Una semplice metafora di uno smart contract è un distributore automatico, che funziona in modo simile a un contratto intelligente: input specifici garantiscono output predeterminati.

Il distributore automatico erogherà il prodotto desiderato solo dopo aver soddisfatto tutti i requisiti. Se non si seleziona un prodotto o non si inserisce un importo sufficiente, il distributore automatico non erogherà il prodotto.

Uno dei vantaggi più significativi degli smart contract rispetto ai contratti normali è che il risultato viene eseguito automaticamente quando le condizioni del contratto si realizzano. Non è necessario attendere che un essere umano esegua il risultato. In altre parole, i contratti intelligenti eliminano la necessità di fiducia.

Il fattore umano è uno dei maggiori punti di fallimento dei contratti tradizionali. Ad esempio, due giudici possono interpretare un contratto tradizionale in modo diverso. Le loro interpretazioni potrebbero portare a decisioni diverse e a risultati disparati. Gli smart contract eliminano la possibilità di interpretazioni diverse. Al contrario, i contratti intelligenti vengono eseguiti con precisione in base alle condizioni scritte nel codice del contratto. Questa precisione significa che, a parità di circostanze, lo smart contract produrrà lo stesso risultato.

Gli smart contract sono utili anche per le verifiche e il monitoraggio. Poiché negli smart contract di una blockchain pubblica, chiunque può tracciare istantaneamente i trasferimenti di asset e altre informazioni correlate. Ad esempio, è possibile verificare se qualcuno ha inviato denaro al proprio indirizzo.

Gli smart contract possono anche proteggere la privacy. Poiché le blockchain sono, in generale, reti pseudonime, le transazioni sono collegate pubblicamente ad un indirizzo crittografico unico e non sono legate all'identità.

Infine, come per i contratti, è possibile verificare il contenuto di uno smart contract prima di firmarlo (o di interagire con esso in altro modo). In altre parole: la trasparenza pubblica dei termini del contratto significa che chiunque può esaminarlo.[7]

Ricapitolando: le blockchain, sono caratterizzate da trasparenza, tracciabilità e sicurezza. Queste caratteristiche rendono l'adozione delle blockchain una buona soluzione per migliorare la sicurezza delle informazioni, la privacy e l'affidabilità in contesti molto diversi. In particolare, diventa di fondamentale importanza l'utilizzo delle blockchain nel contesto di sistemi distribuiti di gestione della fiducia e della reputazione (DTRMS - Distributed Trust and Reputation Management Systems).[14]

1.4 Trust e Reputation

In generale, la trust e la reputazione sono due concetti strettamente correlati. Trust si riferisce alla fiducia che una persona ha in un'altra persona o in un'organizzazione. La reputazione si riferisce alla percezione che le persone hanno di un'organizzazione o di una persona. La reputazione può essere influenzata da molti fattori, come la qualità del prodotto o del servizio offerto, la comunicazione e l'immagine dell'organizzazione o della persona.

I sistemi di reputazione sono programmi o algoritmi che consentono agli utenti di valutarsi a vicenda nelle comunità online al fine di creare fiducia attraverso la reputazione. Questi sistemi di reputazione rappresentano una tendenza significativa nel "supporto decisionale per la fornitura di servizi". Il ruolo dei sistemi di reputazione, al contrario, è quello di raccogliere un'opinione collettiva al fine di creare trust tra gli utenti di una comunità online.[8]

Un sistema di reputazione raccoglie informazioni sugli attori di un sistema per creare un punteggio di reputazione. I sistemi di reputazione devono fornire i seguenti tre elementi fondamentali:

[7] https://ethereum.org/en/smart-contracts/
[8] https://en.wikipedia.org/wiki/Reputation_system

- **il processo di rating** ovvero una procedura che consente ad un valutatore di fornire un feedback sulla loro esperienza durante l'utilizzo o l'interazione con l'elemento della reputazione;
- **un processo di query** che consente agli utenti di indagare sulla reputazione di un certo elemento;
- **una funzione di reputazione** che calcola un punteggio di reputazione.

Un sistema trusted è un sistema su cui si fa affidamento in una certa misura per applicare una politica di sicurezza specifica. Ciò equivale a dire che un sistema attendibile è un sistema il cui fallimento interromperebbe una politica di sicurezza (se esiste una politica per cui il sistema viene definito trusted).

La parola "fiducia" è fondamentale, poiché non ha il significato che ci si potrebbe aspettare nell'uso quotidiano. Un sistema attendibile è un sistema che l'utente si sente sicuro di utilizzare e nel quale si "fida" di eseguire attività, senza che vengano eseguiti (segretamente) programmi dannosi o non autorizzati. Il trusted computing si riferisce al fatto che i programmi possano fidarsi che la piattaforma non venga modificata rispetto a quanto previsto e se tali programmi siano o meno innocenti o dannosi o se eseguano attività indesiderate dall'utente.

Un sistema attendibile può anche essere visto come un sistema di sicurezza basato su livelli in cui la protezione viene fornita e gestita in base a diversi livelli. Questo si trova comunemente nelle forze armate, dove le informazioni sono classificate come non classificate (U), riservate (C), segrete (S), top secret (TS) e oltre. Questi applicano anche le politiche no read up-e no write-down.[9]

L'idea di base del progetto è quello di legare questa caratteristica intrinseca chiamata trust ad una particolare entità tramite Soulbound Token (SBT). Mentre i prodotti (alimentari) e la loro relativa impronta ecologica saranno legati agli NFT.

1.5 NFT e SBT

Un token non fungibile (Not-fungible Token) è un identificatore unico digitale registrato in una blockchain e utilizzato per certificare la proprietà e l'autenticità. Non può essere copiato, sostituito o suddiviso. La proprietà di un NFT è registrata in una blockchain e può essere trasferita dal proprietario, consentendo agli NFT di essere venduti e scambiati.

[9] https://en.wikipedia.org/wiki/Trusted_system

Gli NFT possono essere creati da chiunque e richiedono poche o nessuna competenza di codifica per essere creati. Gli NFT contengono tipicamente riferimenti a file digitali come opere d'arte, foto, video e audio. Poiché gli NFT sono identificabili in modo univoco, si differenziano dalle criptovalute, che sono fungibili.[10]

I token Soulbound, o SBT, sono token di identità digitale che rappresentano le caratteristiche, le peculiarità, i tratti e i risultati di una persona o di un'entità. Costruito sulle tecnologie blockchain, portano le applicazioni NFT a un livello superiore.

Gli NFT sono collegati a una blockchain tramite il loro codice di identificazione unico e non possono essere falsificati o replicati, ma possono essere venduti o trasferiti. Tuttavia, quando un NFT viene coniato per fungere da gettone Soulbound, non può mai essere trasferito. L'obiettivo di SBT è quello di trasformare il concetto di NFT in qualcosa che vada oltre il denaro e i diritti di proprietà in un token unico e non trasferibile.

I token Soulbound sono quindi NFT non trasferibili che rappresentano un'identità. Sono legati ad una rete blockchain e contengono le informazioni identificative uniche di una persona o di un'entità, che comprendono dati personali e storia, come età, titolo di studio, istruzione, documentazione sanitaria e risultati lavorativi.[11]

1.6 Lavori collegati

La nozione di "trust" implica convinzioni e aspettative, con un certo livello di confidenza, sull'affidabilità (in particolare affidabilità e sicurezza), sulla competenza e su altre caratteristiche di un'entità o di dati [15]. La nozione di "reputazione" è legata a quella di "trust" e può essere considerata un'opinione sulla fiducia. Esistono diversi lavori sui sistemi di gestione della trust (Trust Management System o TMS) [16], che possono essere suddivisi grossolanamente in TMS basati sulle policy e TMS basati sulla reputazione. I modelli basati su policy utilizzano regole di policy che determinano se fidarsi di un'entità o di un dato.[15, 17] In base a questo approccio, la fiducia è una sorta di valore booleano: un valore "vero" per chi rispetta le politiche e un valore "falso" per chi non le rispetta. I modelli di basati sulla reputazione utilizzano il comportamento passato (soprattutto durante le interazioni) e i dati raccolti da altre fonti (incluse le raccomandazioni di altre entità) per determinare se fidarsi di un'entità o di un dato [15, 17]. Sulla base di questo approccio, la fiducia viene misurata con un punteggio in base a una determinata scala.

[10] https://en.wikipedia.org/wiki/Non-fungible_token
[11] https://www.leewayhertz.com/soulbound-tokens

Grazie alla loro capacità di garantire l'immutabilità e la tracciabilità delle transazioni e ai loro meccanismi di consenso, le DLT hanno trovato ampia applicazione per la gestione della fiducia basata sulle policy (ad esempio, si veda [6, 7, 18]). Al contrario, le DLT hanno trovato finora solo poche applicazioni dell'approccio basato sulla reputazione [17, 19–23]. La maggior parte di essi si occupa della fiducia dei dati, soprattutto in uno scenario IoT, e dei sensori che li producono [17, 20, 21, 24]. Alcuni di loro propongono anche alcuni meccanismi di penalizzazione [24]. Invece, Wu e Zhang [23] si occupano della fiducia degli attori.

I lavori presentati da Malik et al. [19] e da Putra et al. [22] sono molto vicini a questa tesi per l'intenzione e il tipo di modello di fiducia adottato. Infatti, secondo questi lavori, la reputazione si basa sulla combinazione di due aspetti: i dati dei sensori e le valutazioni degli attori umani. I dati dei sensori sono utilizzati come indicatori della qualità dei prodotti alimentari e la reputazione di una merce è tracciata lungo tutta la catena di fornitura. Le valutazioni attribuite dagli acquirenti e dalle autorità di regolamentazione ai venditori, la reputazione dei prodotti e la storia della reputazione vengono utilizzate per calcolare la reputazione attuale di un venditore. In [22] si tiene conto anche del fatto che le merci possono essere utilizzate per produrre nuove merci. Inoltre, entrambi i lavori misurano la reputazione come un punteggio complessivo che nasconde le diverse caratteristiche che contribuiscono a formare la reputazione e non considerano i meccanismi di penalizzazione.

In sintesi, nessuno dei lavori citati considera un meccanismo di valutazione multidimensionale che tenga conto contemporaneamente della sostenibilità ambientale, sociale, economica e tecnica. Questo meccanismo dovrebbe considerare le emissioni totali di gas serra di un processo insieme alla fiducia e alla qualità in ambito di salute, luogo di lavoro, sicurezza e molti altri aspetti della sostenibilità.[12] In aggiunta, nessuno dei lavori considera l'utilizzo dei token non fungibili (NFT) come meccanismo di registrazione della reputazione e/o della fiducia. La reputazione e la fiducia in un sistema DLT potrebbero essere rappresentate da token non fungibili (NFT) e utilizzate come incentivo per promuovere un comportamento virtuoso che estrae la fiducia da un processo distribuito e complesso come le catene di approvvigionamento. Il lavoro progettuale di questa tesi si basa in gran parte sul lavoro trattato in «A Soulbound Token-based Reputation System in Sustainable Supply Chains»[1], approfondendone alcuni contenuti e presentando altre considerazioni e nuovi spunti di riflessione.

1.7 Contenuti originali

In questa tesi si esplorano le possibilità delle tecnologie blockchain nel contesto dei processi distribuiti associati ad attori di qualsiasi scala, ma con uguale capacità di azione e

[12] https://www.undp.org/sustainable-development-goals

contributo come nel paradigma IoE. Inoltre, gli attori che partecipano al processo globale sono di natura eterogenea, ma unificati in una stretta simbiosi tra partecipanti umani e automazioni, che costituiscono il tipico contesto socio-tecnico e complesso generato dagli obiettivi dell'Industria 5.0.

Le blockchain sono tecnologie interessanti per condividere le informazioni agroalimentari in un ambiente affidabile: nel caso della catena di approvvigionamento alimentare mondiale, infatti, tutti gli operatori (coltivatori, fornitori, trasformatori, distributori, rivenditori, legislatori e consumatori) possono ottenere il permesso di accedere al database dei blocchi e poter così avere la garanzia di conoscere dati affidabili sull'origine e lo stato degli alimenti per effettuare le loro transazione. In particolare, nel documento si indaga sulle possibilità delle tecnologie blockchain nell'integrare le molteplici dimensioni informative create in un processo di filiera alimentare. Il lavoro è infatti legato alle attività di ricerca condotte nell'ambito del progetto europeo Enough [2], in cui si cerca di supportare la digitalizzazione per gli obiettivi di sostenibilità ambientale insieme agli incentivi sociali e morali di reputazione della varietà di partecipanti ad un processo di filiera nel settore alimentare. I risultati e le metodologie possono essere facilmente estesi ad altri settori dell'industria e dell'ingegneria quando la dimensione sociale incontra gli obiettivi di sostenibilità ambientale e produttività.

Nel progetto specifico sulla filiera alimentare sostenibile, l'esigenza principale è il trattamento delle emissioni di gas serra (in equivalenti di CO2) e la reputazione degli elementi che forniscono questo tipo di dati nella filiera. Tuttavia, è importante evitare un approccio centralizzato per lasciare che i nodi partecipanti agiscano in modo autonomo e privato. Senza dimenticare un riconoscimento della loro reputazione rispetto agli obiettivi di qualità delle loro produzioni e al raggiungimento degli obiettivi di sostenibilità. Le blockchain possono essere la soluzione per decentralizzare e distribuire le informazioni in modo da soddisfare queste esigenze.

Il contributo originale di questo lavoro consiste nell'esplorare un uso appropriato della tecnologia *token*, nel contesto delle blockchain, per veicolare le molteplici dimensioni dell'informazione necessarie a un processo complesso e affidabile in cui le fonti e i destinatari dell'informazione sono i tipici attori (eventualmente autonomi) di natura eterogenea, come quelli previsti dal paradigma IoE (caratterizzato dal contesto sociale). In particolare, per esplorare le varie alternative, in questo contesto, del token non fungibile (NFT) e non trasferibile (SBT)[13], il cui uso e la cui ampia adozione sono ancora oggetto di interessanti indagini.

[13] https://wiki.rugdoc.io/docs/introduction-to-soulbound-tokens/

Un secondo contributo di questo lavoro è quello di fornire un resoconto sia generale che pratico sul calcolo di un valore di reputazione per mezzo di un'opportuna combinazione di proprietà NFT e SBT in quanto conformi, in un complesso scenario di processo industriale distribuito, rispettivamente alle proprietà locali di un partecipante e a un consenso globale.

Le possibilità dei sistemi contrattuali blockchain di implementare specifiche di prodotto generiche, valutazioni di qualità e misure di sostenibilità, insieme alla reputazione, sono discusse attraverso un'implementazione e mostrate attraverso un test pratico.

1.8 Struttura dell'elaborato

Il documento è strutturato come segue:
Nel Capitolo 2 viene esposta l'architettura della soluzione proposta. Nel Capitolo 3 viene illustrata nel dettaglio l'implementazione della soluzione attraverso gli smart contract, seguita da un esempio di funzionamento del progetto nel Capitolo 4. L'ultimo capitolo, il 5, è dedicato alle conclusioni e alle prospettive future. In Appendice A viene riportato il codice sorgente completo, scritto in linguaggio Solidity, degli smart contract presenti nel progetto. Infine, in Appendice B si può trovare il codice sorgente completo, scritto in linguaggio PHP, dell'applicativo di esempio.

Capitolo 2

Architettura e Workflow

2.1 Architettura generale

Per semplicità, ma senza perdita di generalità, illustriamo l'architettura prevista attraverso un'istanza di essa con un tipico insieme di partecipanti alla catena di approvvigionamento alimentare.

Nella Figura 2.1 sono rappresentati i principali componenti dell'architettura di un'applicazione che ha lo scopo di utilizzare e integrare le tecnologie NFT e SBT per un consorzio di attori che diventa fiduciario avendo la connessione IoE come principale mezzo di comunicazione. IoE unifica gli attori umani e automatizzati in una fonte di dati per il rilevamento e la misurazione delle prestazioni. Nella figura sono indicati come *sensori*, ma devono essere intesi come un processo sistematico di misurazione della qualità di un processo derivante da statistiche su dati grezzi e in tempo reale. Di solito questo tipo di misurazioni diventano indicatori di performance e possono essere unificati, normalizzati e costruiti in molti modi [25].

Gli indicatori di performance valutano l'efficacia (verso qualche obiettivo concordato collettivamente) di un processo produttivo che si conclude con un prodotto, intendendo la "produzione" nel suo senso più ampio (in alcuni casi potrebbe essere anche un semplice servizio). Le cifre ottenute con gli indicatori costituiscono la parte che può essere contabilizzata in un NFT immutabile, che successivamente può essere trasferito attraverso le molteplici fasi dell'intero processo e dei partecipanti.

Nella stessa Figura 2.1, viene mostrato un elenco di partecipanti classici al processo complessivo, ovvero: un agricoltore, un produttore, un trasportatore e un venditore, al fine di rappresentare in modo minimale quattro fasi fondamentali della catena di approvvigionamento, rispettivamente la produzione primaria e secondaria, il trasporto e la

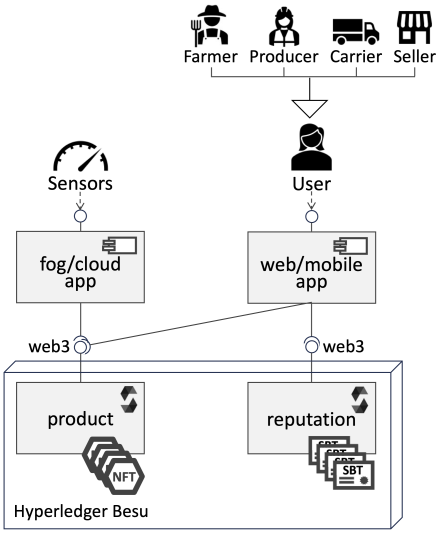

Figura 2.1: Architettura nella filiera alimentare[1]

vendita al dettaglio. A questi utenti viene poi associato il SBT. L'informazione ad essi legata è una combinazione di misurazioni sistematiche delle prestazioni dovute al commercio e alla produzione sul lato NFT, e un voto di consenso più soggettivo che ciascuno dei partecipanti può esprimere sulla base di criteri non ben formalizzati ma condivisibili, che alla fine concorrono al valore di reputazione di un partecipante.

2.2 Workflow

La precedente struttura deve essere accompagnata da una descrizione del flusso di informazioni per evidenziare meglio la dinamica dell'interazione tra le parti, come fatto nella Figura 2.2. In questa figura si può dedurre meglio il ciclo di trasferimento delle informazioni di un NFT e il ruolo del SBT. Anche se in questa figura, per motivi di semplificazione, viene mostrata una struttura in serie, sono possibili altre topologie e cicli di informazioni più distribuiti e in rete. Tuttavia, qui viene mostrato come il giudizio sistematico sulle prestazioni di un prodotto sia convogliato in un insieme di NFT, che riportano molteplici dimensioni di prestazione come l'impronta di CO_2 e la qualità per una determinata produzione. Queste informazioni sono abbinate a un feedback fornito dall'attore ricevente che è quindi uno stakeholder nella catena dei processi interessati. L'accoppiamento di valutazioni sistematiche delle prestazioni e di informazioni generate socialmente costituirà il contenuto del SBT che sarà per sempre associato a un partecipante.

Dopo una qualsiasi fase di produzione, un NFT può essere trasferito ad una fase di

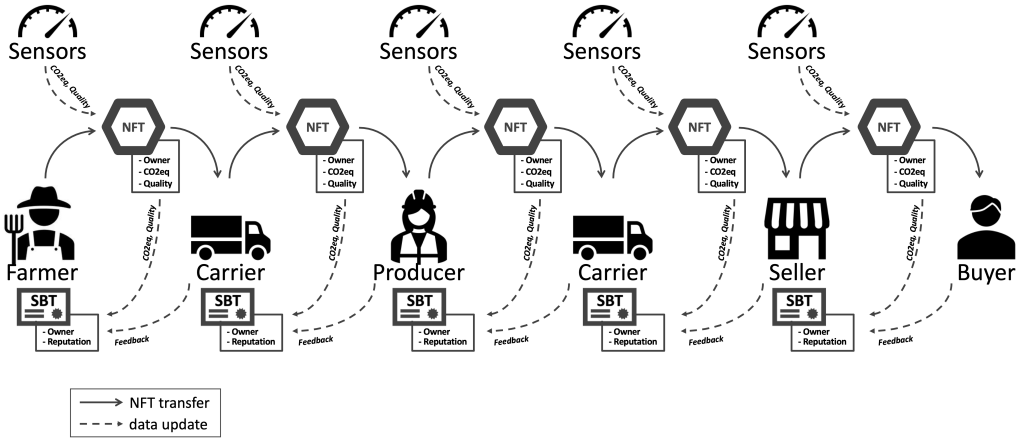

Figura 2.2: Workflow nella filiera alimentare[1]

produzione successiva, ed il valore del NFT viene in qualche modo ereditato dal proprietario della fase successiva, fino all'utente finale. Questo crea un grande legame di fiducia tra i partecipanti, che sono interessati a migliorare la propria reputazione e quindi ad influenzare positivamente tutte le fasi della catena di fornitura, promuovendo, quindi, un circolo virtuoso.[1]

2.3 Architettura nella Blockchain

Nello schema 2.3 si vedono quattro tipi di attori(o ruoli), ognuno con il suo account nella blockchain. Tale schematizzazione, non esclude in alcun modo, che un attore possa avere, con lo stesso account, più ruoli. Gli attori sono: amministratore, produttore, commerciante (trasformatore o acquirente) e sensore. Come si vede nella figura 2.2 gli attori nella filiera alimentare possono essere molteplici, ma generalmente, possono essere trattati con i quattro ruoli enunciati prima. Ad esempio, un agricoltore è assimilabile al ruolo di produttore (ma solo di materie prime), mentre i rivenditori, i trasportatori e gli acquirenti possono corrispondere, nell'architettura precedente, al ruolo di commerciante (trader)2.3.

L'amministratore, in generale, si occupa di istanziare il **Quality Contract** (trattati in seguito) e di assegnare i ruoli. Esso è il deployer dei vari contract, questa è solo una possibile soluzione, ma si possono applicare anche soluzioni dove i deployer sono tutti gli attori.

Il produttore si occupa della creazione del prodotto, utilizzando il **Product Contract**,

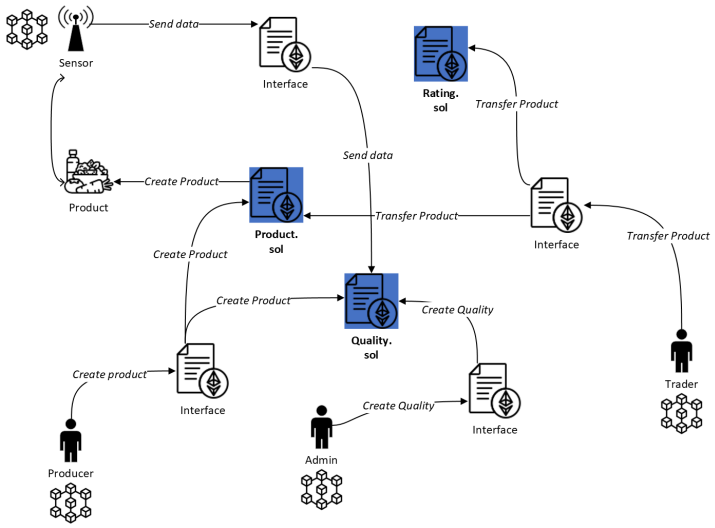

Figura 2.3: Architettura nella Blockchain

a cui associa una specifica istanza del Quality Contract e un sensore.

Il sensore verifica la temperatura (o altra grandezza fisica) del prodotto ed invia i dati acquisiti in blockchain.

Infine il commerciante acquisisce la proprietà del prodotto ed esprime un voto di rating tramite il **Rating Contract**. Sarebbe raccomandabile prevedere, in questo caso, un meccanismo di penalizzazione onde evitare comportamenti opportunistici da parte del commerciante.

Si potrebbe prevedere anche un quinto attore (non riportato nello schema), un ente regolatore, che dopo opportuni audit, esprima a sua volta un voto di rating ad certo produttore (o commerciante).

Il dati del sensore, il rating del commerciante e, se previsto, il rating dell'ente regolatore concorrono (con pesi arbitrari) al valore della reputazione di un certo produttore (o commerciante), e di conseguenza al suo valore di Trust.

Si comprende, quindi, che il fulcro dello schema sono i tre smart contract: Product,

Quality e Rating. In particolare, si è cercato di **generalizzare** la loro implementazione, spostando la logica application-specific nelle interfacce (o smart contract intermediari). Gli attori esercitano le loro funzioni attraverso gli smart contract d'interfaccia, tali interfaccie possono anche essere implementate out-chain.

Product Contract: fornisce vari metodi di estrazione dati e un metodo per la creazione di prodotti (o materie prime). In particolare, ad ogni prodotto, viene associato un **token NFT** che inizialmente appartiene al produttore e cambia di proprietario in base alla filiera. Insieme al token viene registrato un valore CHG relativo all'impronta ecologica del prodotto, vengono registrate le lavorazione (e il loro peso nel CHG) ed eventuali altri prodotti usati per la realizzazione del prodotto finito. Implementa anche un metodo per il trasferimento ad un altro proprietario.

Quality Contract: fornisce vari metodi di estrazione dati e un metodo per la creazione di uno o più vincoli di qualità, relativo ad una certa tipologia di prodotto. Per ogni istanza vengono associati un insieme di vincoli (ad esempio i vincoli di temperatura) che non devono essere superati per mantenere il prodotto di qualità. Infine, fornisce un metodo che in base al valore inviato (ad esempio dal sensore), restituisce un risultato booleano che indica l'eventuale violazione del vincolo.

Rating Contract: fornisce vari metodi di estrazione dati, memorizza i rating, le reputazioni e la trust degli account. Fornisce un metodo per l'invio del rating (array), tale metodo calcola anche reputazione e trust in base alle funzioni definite nel deploying del contract e alla storia dei precedenti valori di reputazione e trust. Questo favorisce una buona versatilità del contratto e quindi permette di lasciare arbitrario, in base all'applicazione, il calcolo della reputazione e trust, ovvero la loro dipendenza funzionale. Anche la trust di un certo attore viene associata ad un token NFT ma di tipo **soulbound** (SBT), dato che non si può trasferire la propria certificazione di trust.

Naturalmente l'architettura che è stata mostrata è volutamente semplificata per scopi divulgativi. La sua flessibilità permette facilmente di aumentare in maniera arbitraria la complessità ed aggiungere, quindi, uno o più layer a monte.

Si potrebbe, ad esempio, pensare che il deploying dei vari contract segua un pattern multisign, in modo che ogni attore debba esplicitamente validare i vari smart contract. Tale obiettivo è facilmente realizzabile creando un nuovo contract che accentri il deploy dei core contract (Product, Quality, Rating) implementando la logica del pattern multisign al suo interno.

Per mantenere l'esempio applicativo facilmente comprensibile, sono stati implementati solo due contract (*Main.sol* e *ProductManagement.sol*) che deployano gli altri ma non è stata implementata una logica multisign, anzi si è optato per un solo owner/amministratore del contract come si può facilmente consultare nel codice sorgente in Appendice A.

2.4 Software Cybersecurity

Nell'architettura del software, quindi nella scrittura degli smart contract, è stata data un'enorme importanza alla software cybersecurity, di cui alcuni aspetti sono, di default, intrinsecamente adottati nella blockchain, ma in particolare, si è cercato di seguire le linee guida per la stesura del software redatte dalla OWASP.[1]

Le OWASP Secure Coding Practices sono delle linee guida per scrivere codice sicuro e robusto, che possono essere integrate nel ciclo di vita dello sviluppo software. L'obiettivo è di prevenire le vulnerabilità più comuni e mitigare i rischi di attacchi informatici. Le pratiche coprono diversi aspetti della sicurezza, tra cui:

- La validazione e la sanificazione degli input;
- La codifica degli output;
- L'autenticazione e la gestione delle password;
- La gestione delle sessioni;
- Il controllo degli accessi;
- Le pratiche crittografiche;
- La gestione degli errori e dei log;
- La protezione dei dati;
- La sicurezza delle comunicazioni;
- La gestione dei file;
- La gestione della memoria;
- Le pratiche generali di codifica.

[1] https://owasp.org/www-project-secure-coding-practices-quick-reference-guide

Le OWASP Secure Coding Practices sono indipendenti dalla tecnologia usata e sono presentate in un formato di checklist, facile da consultare e applicare.

Nel Capitolo 3, che segue, si parlerà nello specifico di alcuni punti focali dell'implementazione degli smart contract, accompagnati da alcuni snippet di codice, si passerà poi nello specifico alla trattazione di alcuni componenti utilizzati per la creazione del progetto.

Capitolo 3

Implementazione

3.1 Implementazione tramite Solidity

Per implementare gli smart contract che consentono di realizzare gli elementi dell'architettura presentata in 2.3 è stata utilizzata la libreria, che è uno standard de-facto, OpenZeppelin[1].

Di seguito vengono riportati alcuni frammenti di codice relativi allo smart contract *Product.sol* ed altri parti basilari del codice in Solidity che rappresentano una chiave di lettura per la nostra trattazione. Il primo nel quale ci focalizzeremo è:

```
contract Product is ERC721,
                    ERC721URIStorage,
                    Ownable
```

Il codice Solidity precedente fa sì che il contratto erediti da *ERC721* e *ERC721URIStorage* per generare token NFT (Non-Fungible Token), ed inoltre, eredita da *Ownable* in modo da rendere alcuni metodi utilizzabili solo dal proprietario del contratto, e così aumentare la sicurezza complessiva.

Nel seguente frammento di codice viene mostrato come la struttura Resource rappresenti il prodotto con i suoi componenti. Una risorsa è quindi una descrizione delle attività e dei prodotti necessari per comporre un determinato prodotto in una determinata fase della catena, ossia:

- name rappresenta il nome della risorsa;
- activityList rappresenta l'elenco descrittivo delle attività per creare la risorsa;

[1] https://www.openzeppelin.com/

- activityGHGList rappresenta l'elenco delle emissioni di gas serra relative alle attività per la creazione della risorsa;
- otherResourceList Altri ID di risorse utilizzate (se utilizzate) per creare la risorsa;
- GHG Emissione totale di gas serra per la risorsa;

```
struct Resource
{
  string name;
  string[] activityList;
  uint[] activityGHGList;
  uint[] otherResourceList;
  uint GHG;
}
```

La risorsa precedentemente descritta è strettamente legata, tramite un mapping, ad un NFT, identificato da uno specifico id (*tokenId*), ottenuto da un contatore della classe *Counter*Counter di OpenZeppelin. Sotto si può vedere l'implementazione della funzione atta alla creazione della risorsa e quindi del token.

```
using Counters for Counters.Counter;
Counters.Counter private _tokenIds;

-------------------------------------------------------------

mapping(uint=>Resource) private resources; // resources mapping

constructor()
   ERC721("Product", "PDC"){
}

function create(Resource calldata resource, address owner)
external onlyOwner returns(uint)
{
    _tokenIds.increment();
    uint tokenId = _tokenIds.current();
    _safeMint(owner, tokenId);
    resources[tokenId] = resource;
    return tokenId;
}
```

Il trasferimento del prodotto, e di conseguenza del token NFT, è consentito solo al proprietario del contratto (attraverso il modificatore OnlyOwner). In questo modo l'applicazione può controllare lo scambio di prodotti:

```
function _beforeTokenTransfer(
    address from, address to,
    uint firstTokenId, uint batchSize
) internal virtual override onlyOwner
{
  super._beforeTokenTransfer(
    from, to,
    firstTokenId, batchSize
  );
}
```

Anche lo smart contract *Quality.sol* eredita da *Ownable*. Esso contiene la struttura *QualityBounds* che rappresenta i valori di vincolo (ad esempio, valori minimi e massimi in un intervallo di temperatura) e un nome per descrivere il vincolo. Una generica funzione per la verifica dei limiti è richiesta come parametro nel costruttore (*checkBounds*), questa funzione riceverà come parametro i vincoli definiti e i valori da confrontare.

```
using Counters for Counters.Counter;
Counters.Counter private _ids;

function(uint[] memory, uint[] memory)
external pure returns(bool) _checkBounds;

struct QualityBounds
{
  string name;
  uint[] bounds;
}

mapping(uint=>QualityBounds) private _qualitiesBounds;

constructor(
   function(uint[] memory, uint[] memory)
   external pure returns(bool) checkBounds
)
{
  _checkBounds = checkBounds;
}
```

Di seguito si può vedere la funzione di creazione di un istanza del contratto di qualità. Mentre la funzione definita nel costruttore viene richiamata in *executeCheckBounds*.

```
function create(QualityBounds calldata qualityBounds)
external onlyOwner returns(uint)
{
   uint[] calldata bounds = qualityBounds.bounds;
   require(bounds.length > 0, "Inser at least one bound!");

   _ids.increment();
   uint id = _ids.current();
   _qualitiesBounds[id] = qualityBounds;
   return id;
}
---------------------------------------

function executeCheckBounds(uint id, uint[] memory values)
external validId(id) view returns(bool)
{
return _checkBounds(
        values, _qualitiesBounds[id].bounds
    );
}
```

Il terzo pilastro dell'architettura presentata nel Capitolo 2, è lo smart contract *Rating.sol*. Anche questo contratto eredita da *ERC721* e *ERC721URIStorage* per generare token NFT (qui Soulbound) e da *Ownable*.

```
contract Rating is ERC721,
                   ERC721URIStorage,
                   Ownable
```

In questo caso, una funzione è dedicata alla valutazione della reputazione (*reputationCalc*) e un'altra alla valutazione della fiducia (*trustCalc*). Le funzioni sono richieste come parametri nel costruttore.

```
using Counters for Counters.Counter;
Counters.Counter private _tokenIds;
```

```
function(uint[] memory, uint[] memory, uint[] memory)
   external pure returns(uint) _reputationCalc;
function(uint[] memory, uint[] memory, uint[] memory)
   external pure returns(uint) _trustCalc;

mapping(address => uint) private _accountSBT;
mapping(address => uint[][]) private _ratings;
mapping(address => uint[]) private _reputations;
mapping(address => uint[]) private _trusts;

constructor(
  function(
    uint[] memory, uint[] memory, uint[] memory
  ) external pure returns(uint32) reputationCalc,
  function(
    uint[] memory, uint[] memory, uint[] memory
  ) external pure returns(uint) trustCalc
) ERC721("Trust", "TRS")
{
  _reputationCalc = reputationCalc;
  _trustCalc = trustCalc;
}
```

Una certificazione di trust viene assegnata ad un determinato attore sotto forma di token SBT, che non è quindi trasferibile e, in questo caso specifico, non è nemmeno bruciabile:

```
function _beforeTokenTransfer(
  address from,
  address to,
  uint tokenId,
  uint batchSize
) internal override(ERC721)
{
  require(from == address(0),
          "Token not transferable!");
  super._beforeTokenTransfer(
    from, to, tokenId, batchSize);
}

function _burn(
  uint tokenId
) internal override(ERC721, ERC721URIStorage)
```

```
{
  require(tokenId == 0, "Token not burnable!");
  super._burn(tokenId);
}

function createSBT(address account)
external onlyOwner
{
   uint tokenId = _accountSBT[account];
   if (tokenId <= 0) {
        _tokenIds.increment();
       tokenId = _tokenIds.current();
       _safeMint(account, tokenId);
       _accountSBT[account] = tokenId;
   }
}
```

Infine, ma non in ordine di importanza, le definizioni di *setReputation* e *setTrust*, che, come si può vedere, utilizzano i vari mapping e le due funzioni definite nel costruttore.

```
function setReputation(
    address account,
    uint[] memory ratings,
    uint[] memory otherValues
) external onlyOwner returns(uint)
{
    uint reputation = _reputationCalc(
        ratings,
        _reputations[account],
        otherValues
    );
    _ratings[account].push(ratings);
    _reputations[account].push(reputation);
    return reputation;
}

function setTrust(
   address account,
   uint[] memory otherValues
) external onlyOwner returns(uint)
{
    uint trust = _trustCalc(
```

```
        _reputations[account],
        _trusts[account],
        otherValues
    );
    _trusts[account].push(trust);
    return trust;
}
```

I tre smart contract, che come già detto in precedenza, rappresentano il core del progetto, sono completamente disaccoppiati e indipendenti, possono essere usati standalone oppure legati direttamente (o indirettamente) da altri smart contract. Inoltre, non si esclude che questi contract possano essere usati in contesti anche molto differenti da una filiera alimentare. In Appendice A si può trovare il codice sorgente completo degli smart contract.

L'algoritmo di calcolo della reputazione basato sul rating, può quindi essere arbitrariamente definito in base al contesto dell'applicativo. Dovrà essere basato sui punteggi ricevuti da tutte le parti interessate, di cui potrebbero essere disponibili le serie temporali. Inoltre, deve dipendere dalle prestazioni di un insieme di sistemi o processi di proprietà del partecipante. Questi indicatori (che nel seguito chiameremo *ratings*) possono essere resi pubblici anche agli altri partecipanti, in modo che possano valutare l'affidabilità dei loro valori e i metodi con cui sono stati ottenuti e misurati. La valutazione può essere effettuata in molti modi, in quanto si tratta in generale di un indicatore di performance (ad esempio, efficienza, tasso, efficacia).

Nella sua forma più generica, si può lasciare che il calcolo della reputazione sia un funzionale statistico (ad esempio, che sia una deviazione standard su record di valori di prestazioni o il risultato di un processo di apprendimento automatico federato). Tuttavia, dovrebbe in generale dipendere anche dall'evoluzione dinamica (storia) dei voti di rating ricevuti.[1]

Un'altra importante considerazione da tenere a mente, è che nel meccanismo di calcolo della reputazione, e quindi nei rating, devono essere inclusi i feedback dei partecipanti che per la loro "natura" non possono essere considerati affidabili. Un partecipante potrebbe, per scopi personali, dare dei feedback molto positivi o molto negativi. Per evitare un comportamento opportunistico di questo tipo si potrebbe pensare ad ulteriori attori umani, come ad esempio dei moderatori che intervengano sulla base di report statistici. Oppure un'altra soluzione potrebbe essere quella di uso di algoritmi di penalizzazione, di tipo automatico, legati statisticamente allo storico dei feedback in-

seriti dal partecipante o ricevuti da chi viene valutato. Se ne riporta nell'algoritmo 1 un possibile esempio. Questo algoritmo verrà effettivamente usato nell'applicativo.

Algorithm 1: Un algoritmo per bannare un votante "oppurtunista"

```
Data: storico_rating[votato];
media(storico_rating[votato]);
deviazione_standard(storico_rating[votato]);
Result: contatore_feedback_anomalo[votante];
soglia_contatore;
foreach votazione do
    if valore_assoluto(voto − media(storico_rating[votato])) >
     deviazione_standard(storico_rating[votato]) then
        contatore_feedback_anomalo[votante] + +;
        if contatore_feedback_anomalo[votante] > soglia_contatore then
            bannare il votante
        end
    end
end
```

Una volta definiti gli smart contract bisogna scegliere dove deployarli. Dato che il Early Adopters Programme del European Blockchain Services Infrastructure (EBSI) prevede tra i requisiti l'utilizzo di una rete Hyperledger Besu si è optato per sviluppare l'applicativo per questo tipo di blockchain.

3.2 Hyperledger Besu

Hyperledger Besu è un client Ethereum progettato per essere adatto alle imprese per casi d'uso di reti permissioned pubbliche e private. Può anche essere eseguito su reti di prova come Sepolia e Görli. Hyperledger Besu include diversi algoritmi di consenso, tra cui Proof of Stake, Proof of Work e Proof of Authority (IBFT 2.0, QBFT e Clique). I suoi schemi di autorizzazione completi sono progettati specificamente per l'uso in un ambiente consortile.[2]

Hyperledger Besu include una interfaccia a riga di comando e delle API jsonRPC per eseguire, mantenere, debuggare e monitorare i nodi in una rete Ethereum. Le API jsonRPC permettono di comunicare con i nodi tramite richieste e risposte in formato JSON, usando protocolli come HTTP o WebSocket. L'API jsonRPC supporta anche il meccanismo Pub/Sub. Con jsonRPC è possibile interagire con la blockchain per leggere i dati, inviare transazioni, sottoscriversi a eventi e molto altro. Difatti jsonRPC è un protocollo che viene usato nelle blockchain come standard nelle comunicazioni con e tra i nodi di

[2] https://www.hyperledger.org/use/besu

una rete, tramite richieste e risposte in formato JSON.[3]

Hyperledger Besu supporta anche lo sviluppo, il deployment e l'utilizzo di smart contract e dapp, usando strumenti come Truffle, Remix e web3js. Il client supporta i metodi comuni delle API jsonRPC, come eth, net, web3, debug e miner. Hyperledger Besu non supporta la gestione delle chiavi all'interno del client. Per accedere al keystore e firmare le transazioni è possibile usare Ethsigner con Besu, di cui si parlerà in seguito.

Si è optato per un tipo di rete privata con protocollo di consenso QBTF (Quorum Byzantine Fault Tolerant). La Byzantine Fault Tolerant si riferisce alla capacità di una rete o di un sistema di continuare a funzionare anche quando alcuni componenti sono difettosi o si sono guastati. Con un sistema BFT, le reti blockchain continuano a funzionare o ad attuare le azioni pianificate finché la maggior parte dei partecipanti alla rete è affidabile e genuina. In particolare, nel protocollo QBFT, gli account approvati, noti come validatori, convalidano le transazioni e i blocchi. I validatori creano a turno il blocco successivo. Prima di inserire il blocco nella catena, una super-maggioranza (maggiore o uguale a 1/2) di validatori deve firmare il blocco. Difatti, la principale differenza tra le rete Besu e una rete ConsenSys Quorum[4] sta nell'obbligo di firmare le transazioni.

3.3 EthSigner

Un modo per firmare le transazioni è l'utilizzo di EthSigner, che agisce come servizio proxy inoltrando le richieste al client Ethereum. Quando EthSigner riceve una transazione, genera una firma utilizzando la chiave privata memorizzata e inoltra la transazione firmata al client Ethereum.[5] EthSigner può firmare transazioni con chiavi memorizzate in:

- Un file keystore V3 memorizzato su un file system accessibile dall'host;
- HashiCorp Vault;
- Azure Key Vault.

Il processo di transazione con Ethsigner viene evidenziato in figura 3.1.

Per semplicità di esposizione nell'esempio applicativo non viene fatto uso di questa soluzione, ma rimane una valida alternativa per la firma delle transazioni.

[3] https://ethereum.org
[4] https://consensys.net/quorum/
[5] https://docs.ethsigner.consensys.net/

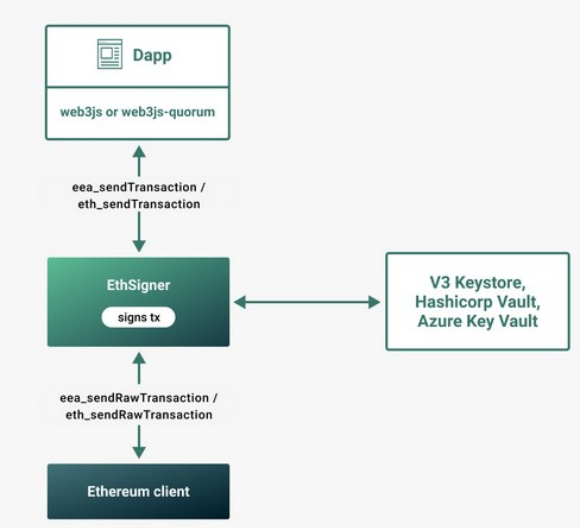

Figura 3.1: Transazione con Ethsigner

3.4 Apache Web Server

Apache web server è nato nel 1995 come un progetto collaborativo di sviluppo software basato sul codice sorgente del server HTTP pubblico sviluppato da Rob McCool al National Center for Supercomputing Applications (NCSA)[6]

Apache web server è un software libero e open source che permette di gestire siti web e applicazioni su internet. Apache è probabilmente il web server più usato al mondo e offre diverse funzionalità, tra cui:

- Supportare diversi linguaggi di programmazione, come PHP, Python, Perl e Java;
- Configurare il web server in base alle proprie esigenze, grazie a un sistema modulare e flessibile;
- Garantire la sicurezza e la privacy dei dati, grazie a protocolli come HTTPS e SSL;
- La possibilità di scalare le prestazioni del web server, grazie a tecniche come il load balancing e il caching.

Apache web server è un progetto della Apache Software Foundation, una comunità di sviluppatori volontari che collaborano per migliorare il software ed è distribuito con licenza Apache 2.0, che consente di usare, modificare e distribuire il software liberamente. Apache è compatibile con diversi sistemi operativi, tra cui Windows, Linux e MacOS ed

[6]https://httpd.apache.org/

è un web server potente, versatile e affidabile, che può soddisfare le esigenze di qualsiasi tipo di sito web o applicazione.

Apache offre una struttura modulare e delle API per una maggiore estensibilità, una gestione della memoria basata su pool, e un modello di processo pre-forking adattivo.

Per tutte queste ragioni, l'applicativo è stato sviluppato su Apache Web Server tramite linguaggio di programmazione PHP.

3.5 PHP

PHP[7] è un linguaggio di programmazione che si usa per creare siti web dinamici e interattivi. PHP sta per Hypertext Preprocessor, ma in realtà è un acronimo ricorsivo che significa PHP: Hypertext Preprocessor. Nato nel 1994 come una semplice collezione di script per il web, ma poi è diventato uno dei linguaggi più diffusi e potenti del web. Ha diverse caratteristiche che lo rendono unico e divertente da usare, come ad esempio la sua facilità di apprendimento e di uso, grazie anche ad una sintassi semplice e flessibile. Inoltre, ha la possibilità di:

- Integrarsi con diversi database, come MySQL, PostgreSQL, Oracle e altri;
- Incorporare codice PHP all'interno di documenti HTML, usando il simbolo <?php ... ?>;
- Estendere le funzionalità di PHP, grazie a un sistema modulare e a una vasta collezione di estensioni disponibili;
- Creare applicazioni web scalabili e performanti, grazie al supporto di tecniche come l'object-oriented programming, il caching e il testing.

PHP è un linguaggio che si adatta a diverse esigenze e situazioni, dallo sviluppo di semplici pagine web personali alla realizzazione di complessi sistemi web aziendali. PHP è anche un linguaggio che si evolve costantemente, seguendo le tendenze e le innovazioni del web. L'ultima major version stabile di PHP è la 8, rilasciata nel novembre 2020, che introduce diverse novità, tra cui:

- I tipi *union*, che permettono di specificare più tipi possibili per una variabile o un parametro;
- Gli attributi, che permettono di aggiungere metadati al codice PHP, usando una sintassi simile alle annotazioni in Java o ai decoratori in Python;

[7] https://www.php.net/

- Il supporto al *match expression*, che permette di scrivere codice più conciso ed elegante per gestire le condizioni multiple;
- I *named arguments*, che permettono di passare i parametri alle funzioni usando i loro nomi invece che la loro posizione;
- La *constructor property promotion*, che permette di dichiarare le proprietà di una classe direttamente nel costruttore, risparmiando codice.

Difatti è stata utilizzata la versione 8.1 di PHP per la realizzazione del software. In particolare è stata utilizzata una libreria[8] per l'interazione con gli smart contract, questa libreria è relativamente giovane ma si presta molto bene per lo scopo.

Tale libreria sfrutta le API jsonRPC per la comunicazione con la blockchain e quindi per il deploying e l'utilizzo degli smart contract. Naturalmente per un'applicazione web non possono mancare *HTML5*[9], *CSS3*[10] e *Javascript*[11].

3.6 HTML, CSS3 e Javascript

HTML5, CSS3 e Javascript sono i tre linguaggi fondamentali per creare siti e applicazioni web dinamiche, interattive e responsive. Vediamo brevemente cosa sono e come si usano.

HTML (HyperText Markup Language) è il linguaggio che definisce la struttura e il contenuto di una pagina web. Usa dei tag per delimitare gli elementi della pagina, come titoli, paragrafi, immagini, link, form, ecc. Ogni tag ha un nome e può avere degli attributi che specificano ulteriori informazioni sull'elemento.

HTML5 è la versione più recente di HTML, che introduce nuovi elementi semantici (come *<section>*, *<article>*, *<nav>*, ecc.), nuovi attributi (come *placeholder*, *required*, *data-*, ecc.), nuove API (come *Canvas*, *Geolocation*, *Web Storage*, ecc.) e nuove funzionalità multimediali (come *<video>* e *<audio>*).

CSS (Cascading Style Sheets) è il linguaggio che definisce lo stile e la presentazione di una pagina web. CSS usa delle regole per applicare dei stili agli elementi HTML, come colore, dimensione, posizione, sfondo, bordo, animazione, ecc. Ogni regola ha un selettore che indica a quali elementi si applica e una dichiarazione che indica quali proprietà si modificano.

[8] https://github.com/drlecks/Simple-Web3-Php
[9] https://it.wikipedia.org/wiki/HTML
[10] https://it.wikipedia.org/wiki/CSS
[11] https://it.wikipedia.org/wiki/JavaScript

CSS3 è la versione più recente di CSS, che introduce nuove proprietà (come *border-radius*, *box-shadow*, *transform*, ecc.), nuovi selettori (come *:nth-child()*, *::before*, ecc.), nuove unità di misura (come *rem*, *vw*, *vh*, ecc.) e nuove funzionalità (come media query, gradienti, transizioni, animazioni, ecc.).

Javascript è il linguaggio che definisce il comportamento e l'interattività di una pagina web. Usa delle istruzioni per manipolare gli elementi HTML e CSS, reagire agli eventi dell'utente o del browser, comunicare con il server o con altre pagine, creare effetti dinamici e animazioni, ecc. Ogni istruzione ha una sintassi che segue le regole del linguaggio e può usare delle variabili, delle funzioni, degli oggetti o degli operatori.

Javascript è un linguaggio molto flessibile e potente, che può essere usato anche per creare applicazioni web complesse, usando dei framework (come React, Angular, Vue, ecc.) o dei moduli (come ESM, CommonJS, AMD, ecc.). Inoltre, può essere usato anche per creare applicazioni mobile (usando strumenti come Cordova, React Native, NativeScript, ecc.) o applicazioni desktop (usando strumenti come Electron, NW.js, Proton Native, ecc.).

3.7 Esempio applicativo

Per concretizzare l'argomentazione con un esempio minimo, immaginiamo lo scenario nel contesto della Figura 2.2, un caso di studio di una catena di approvvigionamento alimentare. L'esempio applicativo prevede l'utilizzo degli smart contract di cui si è parlato in precedenza, più altri smart contract creati appositamente per la dimostrazione pratica, il progetto definitivo è stato chiamato Enoughchain.

Enoughchain è un applicativo basato su Blockchain che utilizza smart contract per tracciare le emissioni di gas serra. Questa applicazione consente agli stakeholder di monitorare e gestire le loro emissioni di gas serra in modo trasparente e affidabile. Utilizza anche un sistema decentralizzato per garantire che le informazioni sulle emissioni siano accurate e verificabili. L'esempio utilizza un approccio con policy centralizzata sulla reputazione e sulla trust. Nel caso più generale, essa può essere stabilita dal risultato di una decisione di consenso, che può essere evoluta dinamicamente nel tempo.

In Enoughchain un prodotto alimentare viene tracciato dalla sua origine sino al consumatore finale. Supponiamo che si tratti di un pacchetto di caffè proveniente da una piantagione sostenibile in Colombia. Il produttore può registrare sulla blockchain le in-

formazioni relative alla qualità, alla quantità e alla data di raccolta del caffè, nonché le sue pratiche ambientali e sociali. Queste informazioni possono, ad esempio, essere associate a un codice QR che viene stampato sull'etichetta del prodotto. Il trasportatore può verificare la provenienza del caffè e aggiungere sulla blockchain i dati relativi al percorso, al tempo e alle condizioni di trasporto. Il distributore può controllare la conformità del caffè ai requisiti di sicurezza e qualità e registrare sulla blockchain la data e il luogo di consegna al rivenditore. Il rivenditore può monitorare lo stato del caffè e gestire le scorte in modo efficiente. Infine, il consumatore può scansionare il codice QR con il suo smartphone e accedere alla storia completa del caffè, dalla piantagione alla tazza. In questo modo, il consumatore può avere fiducia nell'autenticità e nella sostenibilità del prodotto che acquista.

Enoughchain mostra una soluzione innovativa per la tracciabilità e la trasparenza della catena alimentare, basata sulla tecnologia blockchain. Questa tecnologia permette di creare un registro condiviso, immutabile e autorizzato delle informazioni relative ai prodotti alimentari, che possono essere verificate da tutti gli attori coinvolti nella rete. Inoltre, utilizza un sistema di reputazione e trust basato su SBT per incentivare i comportamenti virtuosi e scoraggiare le frodi e le contraffazioni. Enoughchain rappresenta quindi un esempio concreto di come la blockchain possa contribuire a creare un ecosistema alimentare più sicuro, intelligente e sostenibile.

Tuttavia, l'implementazione di Enoughchain e di altre soluzioni basate sulla blockchain nella catena alimentare non è priva di sfide e difficoltà. Alcune delle principali sfide sono le seguenti:

- La complessità tecnica e l'interoperabilità tra diverse piattaforme e standard blockchain, che richiedono una forte collaborazione tra gli attori della rete e una definizione chiara dei requisiti funzionali e non funzionali;
- La qualità e l'integrità dei dati inseriti nella blockchain, che devono essere accurati, completi, coerenti e verificabili, altrimenti si rischia di compromettere la validità e l'affidabilità delle informazioni tracciate;
- L'incertezza legale e normativa, che riguarda la definizione dei diritti e delle responsabilità dei partecipanti alla blockchain, la protezione dei dati personali e sensibili, la conformità alle norme vigenti in materia di sicurezza alimentare e di etichettatura dei prodotti;
- La resistenza organizzativa e culturale, che implica il superamento delle barriere psicologiche, comportamentali e relazionali che possono ostacolare l'adozione della blockchain da parte degli operatori della catena alimentare, come la mancanza di fiducia, di consapevolezza o di competenze.

Queste sfide richiedono un approccio olistico e multidisciplinare per affrontare le diverse dimensioni tecniche, organizzative, sociali ed economiche coinvolte nell'innovazione basata sulla blockchain nella catena alimentare. Inoltre, richiedono una maggiore ricerca e sperimentazione per valutare l'impatto e il valore aggiunto della blockchain rispetto ad altre tecnologie esistenti o emergenti.

Nel successivo capitolo, il 4, si vedrà un esempio pratico con Enoughchain, corredato da alcuni screenshot.

Capitolo 4

Esempio con Enoughchain

Immaginiamo il seguente possibile scenario:

Un prodotto la cui qualità è influenzata dalla temperatura, rilevata da un sensore, che non deve violare un limite inferiore di 5 e superiore a 20. Il numero di volte in cui viene superato il limite influenza negativamente il rating del sensore.
I valori di rating variano da un minimo di 0 a un massimo di 10, mentre il valore di reputazione è calcolato con la seguente formula:

$$Rp(t) = ceil(\frac{3 \cdot Rt_{reg}(t) + 3 \cdot Rt_{sens}(t) + 4 \cdot Rt_{trad}(t) + \sum_{k=1}^{t-1} Rp(t-k)}{k+1})$$

dove $Rp(t)$ è la reputazione al tempo t, $Rt_{reg}(t)$ è la valutazione del regolatore, $Rt_{sens}(t)$ è la valutazione del sensore, $Rt_{trad}(t)$ è il rating del trader e *ceil* è la parte intera superiore.

Il valore di trust di un attore viene, inizialmente, fissato a 30 nel momento dell'assegnazione del ruolo da parte dell'amministratore. Se il valore di Trust diventa inferiore a quello iniziale l'attore non può più effettuare operazioni nella rete. La formula per il calcolo del valore di Trust è la seguente:

$$T(t) = ceil(\frac{Rp(t) - ceil(\frac{GHG}{100}) + \sum_{k=1}^{t-1} T(t-k)}{k+1})$$

dove $T(t)$ è la fiducia al tempo t, GHG è il valore relativo ai gas serra del prodotto scambiato, che nell'esempio applicativo va da un minimo di 0 a un massimo di 1000.

4.1 Configurazione applicativo

L'applicativo è stato testato in una macchina con sistema operativo linux Ubuntu 22.04[1] con Web Server Apache, configurando la rete Besu per l'utilizzo di quattro nodi. Il file di configurazione *genesis.json* utilizzato è il seguente:

[1] https://www.ubuntu-it.org/

```
{
  "config" : {
    "chainId" : 1337,
    "contractSizeLimit": 2147483647,
    "berlinBlock" : 0,
    "qbft" : {
      "blockperiodseconds" : 2,
      "epochlength" : 30000,
      "requesttimeoutseconds" : 4
    }
  },
  "nonce" : "0x0",
  "timestamp" : "0x58ee40ba",
  "gasLimit" : "0x1fffffffffffff",
  "difficulty" : "0x1",
  "mixHash" :
  "0x63746963616c2062797a616e74696e65206661756c7420746f6c6572616e6365",
  "coinbase" : "0x0000000000000000000000000000000000000000",
  ----omissis----
}
```

Mentre, per l'implementazione dell'applicativo, è stato utilizzato il linguaggio PHP con l'orchestratore di pacchetti composer[2]. Il codice sorgente è stato inserito in Appendice B. Di seguito i pacchetti PHP utilizzati nell'applicativo con le loro versioni, indicati nel file *composer.json*:

```
{
    "require": {
        "drlecks/simple-web3-php": "^0.10.0",
        "ext-bcmath": "*",
        "ext-curl": "*",
        "ext-fileinfo": "*",
        "ext-gmp": "*",
        "ext-intl": "*",
        "ext-xml": "*",
        "ext-zip": "*",
        "monolog/monolog": "^3.4",
        "smarty/smarty": "^4.3"
    }
}
```

Gli smart contract scritti in linguaggio solidity sono invece stati compilati con il pacchetto apt *solc*[3]. Utilizzando il comando *composer install* vengono installati i pacchetti e le loro

[2]https://getcomposer.org/
[3]https://docs.soliditylang.org/

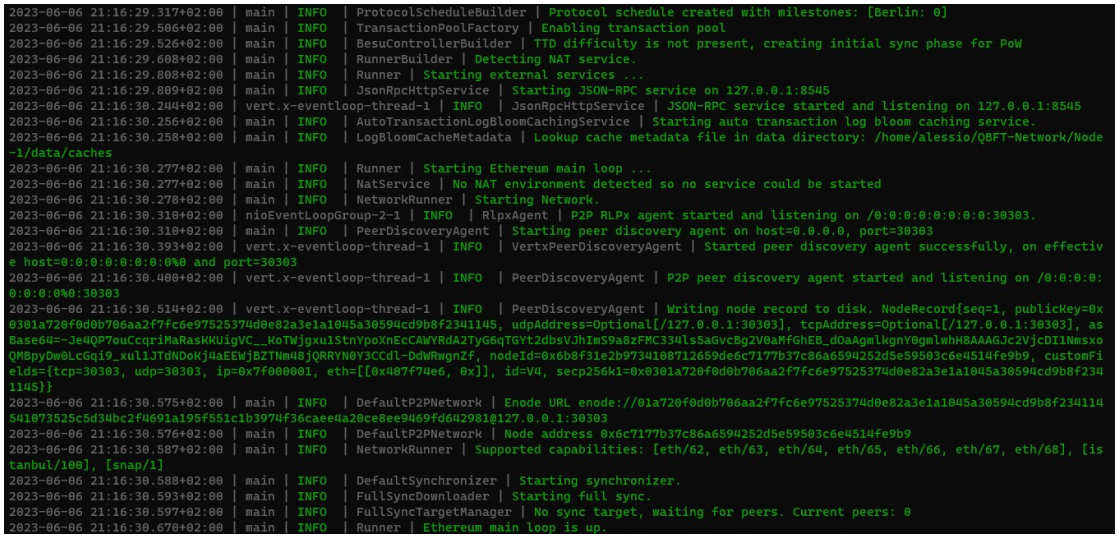

Figura 4.1: Besu Node 1

relative dipendenze, a questo punto si possono avviare la rete Besu e l'applicativo.

4.2 Esecuzione

Per avviare il primo nodo Besu (nodo master) è stato utilizzato il seguente comando:

```
besu-22.10.3/bin/besu --data-path=/home/[user]/QBFT-Network/Node-1/data
--genesis-file=/home/[user]/QBFT-Network/genesis.json --rpc-http-enabled
--rpc-http-api=ETH,NET,QBFT --host-allowlist="*" --rpc-http-cors-origins="all"
--min-gas-price=0
```

Ottenendo un risultato simile all'immagine 4.1 Analoghi comandi sono stati usati per gli altri tre nodi, con l'aggiunta del parametro *–bootnodes* valorizzato con l'indirizzo *enode URL* del nodo master. Dopo aver avviato la rete Hyperledger Besu, si può avviare l'applicazione tramite browser web. In 4.2 si può vedere uno screenshot dell'applicativo.

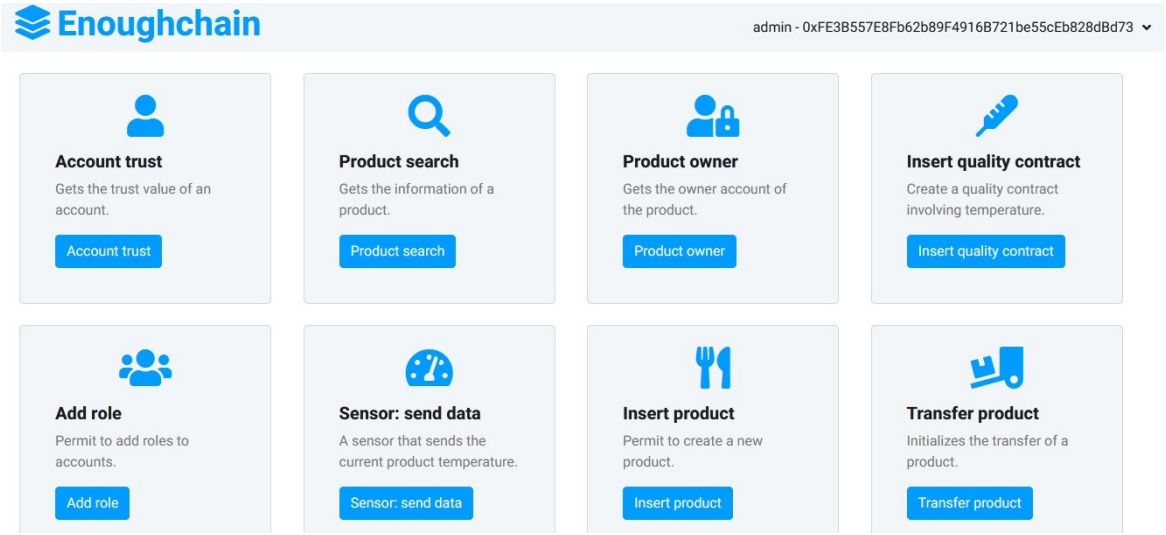

Figura 4.2: Enoughchain

4.3 Violazione del Quality contract

1. Fare il login con l'account amministratore;

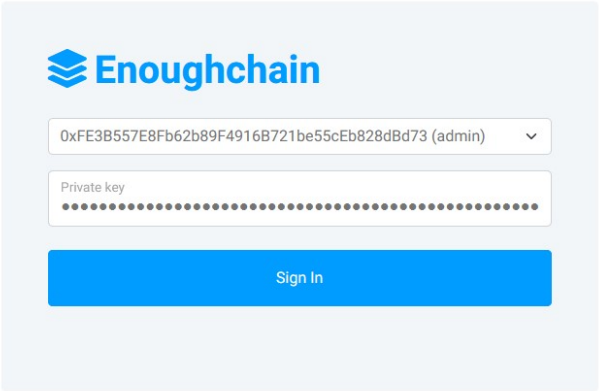

Figura 4.3: Enoughchain - Login

2. Assegnare i ruoli di sensor, producer e trader a degli account a scelta (con trader e producer diversi) con la funzione *Add role*;

3. Creare un quality contract con la funzione *Insert quality contract*, assegnare un nome arbitrario ed inserire 5 e 20, rispettivamente, come valori di temperatura minimo e massimo;

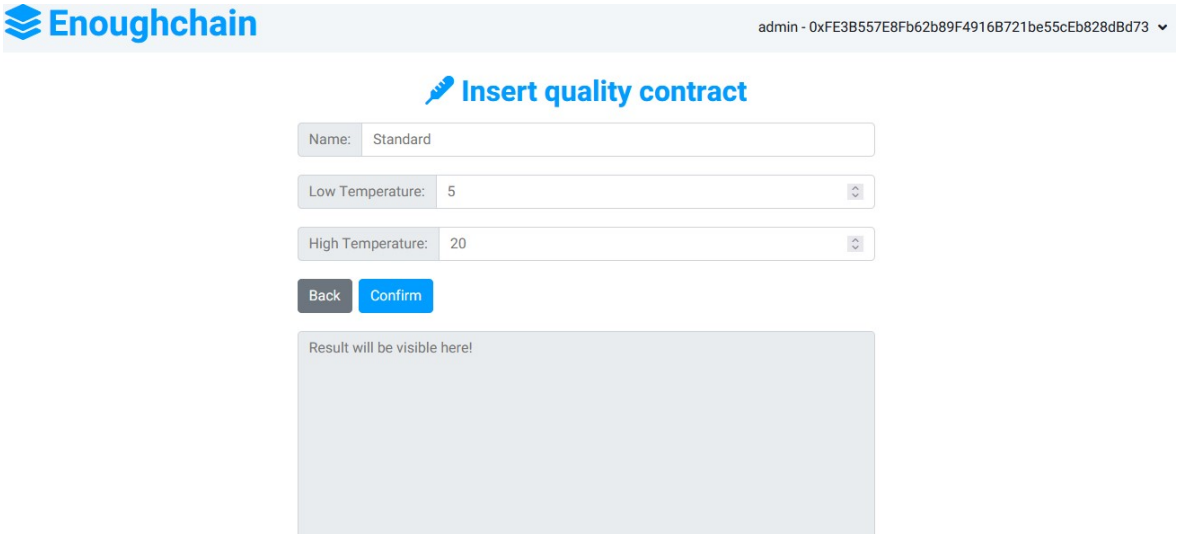

Figura 4.4: Enoughchain - Insert quality contract

4. Fare il logout e poi loggarsi con l'account del produttore scelto al punto 2;
5. Tramite la funzione *Insert product* creare un nuovo prodotto con nome arbitrario e GHG value pari a 100, scegliere poi il quality contract generato al punto 3 e il sensore creato al punto 2;
6. Fare di nuovo il logout e poi loggarsi con l'account del sensore scelto al punto 2;
7. Utilizzare la funzione *Sensor: send data* e scegliere un valore di temperatura compreso nell'ntervallo [5,20], non verrà in questo caso generato alcun evento *QualityWarning*;
8. Ripetere il punto 7 inserendo un valore al di fuori dell'intervallo [5,20], in questo caso verrà generato un evento *QualityWarning*.
9. Fare il logout;

Nome Variabile	Valore passo 7	Valore passo 8	Valore passo 9
QualityWarning	No	No	Sì
productQualityExceed	0	0	1
sensorValue	-	8	21
sensorRating	10	10	9

Tabella 4.1: Violazione del Quality Contract

4.4 Rating, Reputazione e Trust

Partendo dalla situazione precedente.

1. Fare di nuovo il login con l'account amministratore;
2. Usare la funzione *Account trust* per consultare il valore di trust del produttore creato in precedenza, il valore dovrebbe corrispondere a 30;
3. Usare la funzione *Transfer Product* e trasferire il prodotto creato prima all'account del trader;
4. Fare logout e selezionare l'account del trader;
5. Con la funzione *Complete Transfer* selezionare il prodotto di prima ed inserire un rating di 10;

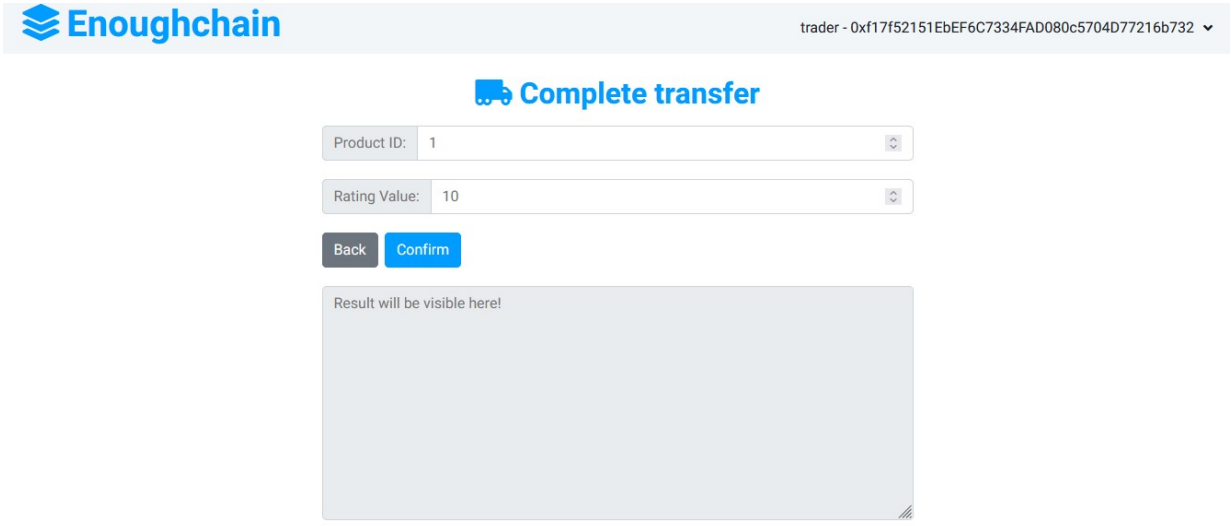

Figura 4.5: Enoughchain - Complete transfer

6. Infine, usare di nuovo la funzione *Account trust* per consultare il nuovo valore di trust del produttore che dovrebbe corrispondere a 58;

7. Chiudere il browser web e la rete Besu.

Nome Variabile	Valore passo 5	Valore passo 6
regulatorRating	10	10
sensorRating	9	9
transferRating	-	10
accountReputation	-	87
accountTrust	30	58

Tabella 4.2: Rating, Reputazione e Trust

Seguono nel Capitolo 5 le conclusioni della tesi.

Capitolo 5

Conclusioni e sviluppi futuri

Questa tesi ha quindi dimostrato come i token NFT e SBT possano essere utilizzati per implementare una soluzione basata sulle blockchain per la tracciabilità e la sostenibilità della catena alimentare. Si è discusso di come i token forniscano una solida base tecnologica per le nuove sfide di sostenibilità implicate in Industria 5.0. In particolare, il legame tra le blockchain e Industria 5.0 si basa sulla multidimensionalità delle informazioni elaborate in Internet of Everything.

Prendendo come riferimento il contesto della filiera alimentare, sono state mostrate le caratteristiche di un'architettura e un esempio di implementazione che aiuta a comprendere meglio le potenzialità della tecnologia *token* oggetto di studio.

Nell'implementazione del progetto si è anche cercato di soddisfare i requisiti della European Blockchain Services Infrastructure (EBSI), tenendo quindi aperta la possibilità che questo lavoro possa diventare in futuro uno degli use cases trattati.[1]

Il risultato ottenuto con il lavoro è ancora circoscritto ed ha ampi margini di miglioramento. Ad esempio, l'indagine sul calcolo della reputazione può essere approfondita in molti modi, per prendere in considerazione altri approcci matematici o di apprendimento automatico.

Come gran parte degli applicativi sviluppati con l'utilizzo delle blockchain, rimane da risolvere un noto problema, il cosiddetto, "problema dell'oracolo", ovvero l'impossibilità delle blockchain di accedere ai dati esterni.[26] Al fine di bilanciare al meglio le partizioni off-chain e on-chain di una rete blockchain, a maggior ragione nel contesto IoE.

[1] https://ec.europa.eu/digital-building-blocks/wikis/display/EBSI/Home

Questo problema limita la capacità delle blockchain di interagire con il mondo reale e di verificare le informazioni provenienti da fonti diverse. Per risolvere questo problema, sono necessari degli intermediari chiamati oracoli, che fungono da ponte tra le blockchain e le fonti di dati esterne. Gli oracoli devono essere affidabili, sicuri e decentralizzati, altrimenti si rischia di compromettere la validità e l'integrità dei dati tracciati sulla blockchain.[2].

5.1 Problema dell'oracolo

Il problema dell'oracolo della blockchain è una delle barriere più importanti da superare se si vuole che gli smart contract intelligenti su reti come Ethereum vengano adottati in massa in un'ampia varietà di mercati e casi d'uso.

Gli smart contract eseguiti su blockchain offrono un immenso potenziale per ridefinire il modo in cui le entità indipendenti si impegnano in accordi contrattuali e scambiano valore. Separata dall'economia degli smart contract è l'economia digitale non-blockchain, molto più ampia, costituita da tutti i dispositivi connessi a Internet che operano online. Un sottoprodotto dell'infrastruttura digitale che è un serbatoio in continua espansione di dati e API, che forniscono approfondimenti sul funzionamento del mondo; ad esempio, i risultati delle ricerche su Internet che presentano argomenti di discussione popolari nella società o i sensori IoT che mostrano modelli di traffico comuni.

Gli smart contract basati su blockchain e le economie tradizionali di dati e API hanno un immenso potenziale per combinarsi in smart contract ibridi e formare la futura architettura dell'automazione guidata dai dati, ma la domanda è: come si collegano questi due mondi? Questo è il nocciolo del "problema dell'oracolo".

Esistono diverse soluzioni proposte per implementare degli oracoli decentralizzati, tra cui Chainlink, che è uno standard per le reti di oracoli decentralizzati. Chainlink permette di collegare le blockchain con qualsiasi tipo di dato o API esterna, garantendo la sicurezza e la scalabilità della soluzione. Un possibile sviluppo futuro di questa tesi potrebbe essere quello di integrare Chainlink con Enoughchain per realizzare una soluzione completa e robusta per la tracciabilità e la sostenibilità della catena alimentare basata sulla blockchain.[3]

[2]https://blockpit.io/en/blog/how-to-solve-the-oracle-problem
[3]https://chain.link/education-hub/oracle-problem

5.2 Meccanismi di premialità e penalizzazione

Un partecipante alla filiera alimentare in una blockchain è un soggetto che registra le informazioni relative a un prodotto o a una fase della sua produzione, trasformazione o distribuzione. Questo comportamento può aumentare la trasparenza e la tracciabilità del prodotto e garantire ai consumatori la sua qualità e sicurezza.

Per incentivare questo comportamento, esistono dei meccanismi di premialità che possono essere applicati ai partecipanti alla filiera alimentare in una blockchain. Ad esempio, si possono prevedere dei token fungibili o delle criptovalute che vengono assegnati ai partecipanti che condividono i dati in modo verificabile e affidabile. Questi token o criptovalute possono essere usati per accedere a servizi o a sconti nella filiera alimentare o in altri settori.

Per scoraggiare i comportamenti scorretti o fraudolenti nella filiera alimentare in una blockchain, esistono dei meccanismi di penalizzazione che possono essere applicati ai partecipanti che forniscono dati falsi o incompleti. Ad esempio, si possono prevedere delle sanzioni o delle multe che vengono sottratte dai token fungibili o dalle criptovalute dei partecipanti che violano le regole della blockchain. Queste sanzioni o multe possono essere usate per compensare i danni subiti dagli altri partecipanti o dai consumatori.

Il machine learning potrebbe essere utilizzato per migliorare i modelli di analisi e di condivisione dei dati nella filiera alimentare in una blockchain, rendendoli più intelligenti e personalizzati. Il machine learning è una branca dell'intelligenza artificiale che permette ai computer di apprendere dai dati e di fare previsioni o decisioni. Si potrebbe usare per:

- Riconoscere le immagini dei prodotti e verificarne l'origine, la qualità e la freschezza. Ad esempio, si possono usare algoritmi di visione artificiale per scansionare i codici a barre o i QR code dei prodotti e confrontarli con i dati registrati nella blockchain;
- Ottimizzare i processi logistici e di distribuzione dei prodotti nella filiera alimentare. Ad esempio, si possono usare algoritmi di ottimizzazione o di apprendimento per rinforzo per calcolare i percorsi più efficienti, ridurre gli sprechi e le emissioni, e aumentare la soddisfazione dei clienti;
- Personalizzare le offerte e le raccomandazioni dei prodotti ai consumatori in base alle loro preferenze e ai loro bisogni. Ad esempio, si possono usare algoritmi di filtraggio collaborativo o di apprendimento profondo per analizzare i dati dei consumatori e suggerire loro i prodotti più adatti o più sostenibili;
- Creare dei modelli statici e/o dinamici di meccanismi di premialità e penalizzazione ed eventualmente migliorare tali modelli con l'evoluzione dinamica dei sistemi.

5.3 Sviluppi futuri

Enoughchain è un progetto creato a scopo esemplificativo ma nulla vieta che in futuro possa essere modificato, in maniera opportuna, per rappresentare una catena di approvvigionamento alimentare del mondo reale. Inoltre, ci sono altri possibili sviluppi che potrebbero rendere la soluzione basata sulla blockchain più efficace e scalabile per la catena alimentare. Alcuni di questi sono:

- Esplorare altre tipologie di token oltre agli NFT e agli SBT, come i token fungibili o i token ibridi, che potrebbero offrire vantaggi in termini di flessibilità, liquidità e interoperabilità tra diverse piattaforme blockchain;
- Sviluppare applicazioni user-friendly e intuitive per facilitare l'accesso e l'utilizzo dei dati tracciati sulla blockchain da parte dei consumatori e degli altri stakeholder della catena alimentare, come ad esempio delle app per smartphone o dei portali web;
- Valutare l'impatto economico, sociale e ambientale della soluzione basata sulle blockchain per la catena alimentare, confrontandola con le soluzioni tradizionali o alternative, e misurando gli indicatori di performance e di sostenibilità;
- Ampliare il campo di applicazione della soluzione basata sulle blockchain ad altre filiere produttive oltre a quella alimentare, come ad esempio quella tessile, quella farmaceutica o quella energetica, che presentano sfide simili in termini di tracciabilità e sostenibilità;
- Ridurre l'impatto energetico delle blockchain e quindi la loro occupazione in termini di storage (si veda ad esempio [27]).

Appendice A

Smart contract

Segue il codice sorgente, in linguaggio Solidity, degli smart contract utilizzati per l'esempio di implementazione, rispettivamente: *Account.sol*, *Main.sol*, *Product.sol*, *ProductManagement.sol*, *Quality.sol*, *Rating.sol*, *Utils.sol*.

Account.sol

```
// SPDX-License-Identifier: MIT

pragma solidity 0.8.12;
//pragma experimental SMTChecker;

import "./Utils.sol";
import "@openzeppelin/contracts@4.8.3/access/Ownable.sol";

contract Account is Ownable {

    mapping(address=>uint) private _accountPenalties;
    mapping(address=>string[]) private _accountsRoles;

    uint public constant _MAX_HISTORY_RATING = 10;
    uint public constant _MAX_SUSPECT_RATING = 1;
    string public constant _PRODUCER_ROLE = "producer";
    string public constant _SENSOR_ROLE = "sensor";
    string public constant _TRADER_ROLE = "trader";
    string[] private _roles = [_PRODUCER_ROLE, _SENSOR_ROLE, _TRADER_ROLE];

    function addRole(string calldata role, address account)
    external onlyOwner validRole(role) {
        require(
            !hasRole(role, account),
```

```
            "Account already has this role!!!"
            );
        _accountsRoles[account].push(role);
    }

    function getAccountRoles(address account)
    external view returns(string[] memory) {
        return _accountsRoles[account];
    }

    function getRoles()
    external view returns(string[] memory) {
        return _roles;
    }

    function hasRole(address account)
    external view returns(bool) {
        return _accountsRoles[account].length > 0;
    }

    function hasRole(string memory role, address account)
    public view validRole(role) returns(bool)
    {
        return Utils.stringListExists(role, _accountsRoles[account]);
    }

    function hasRoleProducer(address account)
    external view returns(bool)
    {
        return hasRole(_PRODUCER_ROLE, account);
    }

    function hasRoleSensor(address account)
    external view returns(bool)
    {
        return hasRole(_SENSOR_ROLE, account);
    }

    function hasRoleTrader(address account)
    external view returns(bool)
    {
        return hasRole(_TRADER_ROLE, account);
    }
```

```
function isValidRole(string memory role)
public view returns(bool) {
    return Utils.stringListExists(role, _roles);
}

function removeRolefromAccount(string calldata role, address account)
external onlyOwner validRole(role) {
    Utils.stringListRemove(role, _accountsRoles[account]);
}

function setPenalties(address account, uint dataIndex, uint[][] memory data)
external onlyOwner returns(bool) {
    if (data.length > 1) {
        uint[] memory population = new uint[](data.length);
        uint value = data[data.length-1][dataIndex];
        uint i = data.length;
        uint count = 0;
        while(count < _MAX_HISTORY_RATING && i > 0) {
            count +=1;
            i -=1;
            population[i] = data[i][dataIndex];
        }
        (bool result, uint average, uint stdDev) =
        Utils.calcStocastic(population, _MAX_HISTORY_RATING);

        unchecked {
            if (stdDev > 0 && result &&
                Utils.abs(int(average - value)) > stdDev ) {
                _accountPenalties[account]+=1;
                return true;
            }
        }
        if (_accountPenalties[account] >= _MAX_SUSPECT_RATING) {
            _accountPenalties[account] = 0;
            Utils.stringListRemove(_PRODUCER_ROLE, _accountsRoles[account]);
            Utils.stringListRemove(_TRADER_ROLE, _accountsRoles[account]);
            return true;
        }
    }
    return false;
}
```

```
    modifier validRole(string memory role) {
        require(isValidRole(role),"Not valid role!");
        _;
    }
}
```

Main.sol

```
// SPDX-License-Identifier: MIT

pragma solidity 0.8.12;
//pragma experimental SMTChecker;

import "./Account.sol";
import "./ProductManagement.sol";
import "./Rating.sol";
import "@openzeppelin/contracts@4.8.3/utils/Strings.sol";

contract Main is Ownable {

    event BannedAccount(
        address trader
    );

    event QualityWarning(
        address productOwner,
        address sensor,
        uint productId,
        uint qualityId,
        uint timestamp,
        uint[] values
    );

    event UncompletedTransferWarning(
        address productBuyer,
        address productSeller,
        uint productId,
        uint timestamp
    );

    uint private _minTrust;

    Account private _account = new Account();
```

```
ProductManagement private _productManagement;
Rating private _rating = new Rating(this.reputationCalc, this.trustCalc);

constructor(
    uint maxActivities, uint maxAggregateProducts, uint minTrust
) {
  require(minTrust > 0 && minTrust < 100, "Not valid min trust value!");
  _productManagement = new ProductManagement(
    maxActivities, maxAggregateProducts);
  _minTrust = minTrust;
}

function addRole(string memory role, address account)
external onlyOwner
{
    if (!_account.hasRole(account)) {
        _rating.createSBT(account);
        _rating.resetTrust(account, _minTrust);
    }
    _account.addRole(role, account);
}

function createQuality(
  string calldata name, uint[] calldata bounds
) external onlyOwner returns(uint)
{
    return _productManagement.createQuality(name, bounds);
}

function createProduct(
    string calldata name, string[] memory activities,
    uint[] memory activitiesGHG, uint[] memory products,
    uint qualityId, address sensor
) external validSensor(sensor) returns(uint) {
    require(this.isTrustedProducer(msg.sender),
    "Only trusted producer could create material!");
    return _productManagement.createProduct(msg.sender, name,
           activities, activitiesGHG, products,
           qualityId, sensor);
}

function getProductLastId() external view returns(uint) {
    return _productManagement.getProductLastId();
```

```
    }

    function getProductOwner(uint productId)
    external view returns(address) {
        return _productManagement.getProductOwner(productId);
    }

    function getProductResource(uint productId)
    external view returns(Product.Resource memory) {
        return _productManagement.getProductResource(productId);
    }

    function getProductQualityExceed(uint productId)
    external view returns(uint) {
        return _productManagement.getProductQualityExceed(productId);
    }

    function getQualityBounds(uint qualityId)
    external view returns(Quality.QualityBounds memory) {
        return _productManagement.getQualityBounds(qualityId);
    }

    function getQualityLastId() external view returns(uint) {
        return _productManagement.getQualityLastId();
    }

    //Return stakeholder roles
    function getStakeholderRoles(address stakeholder)
    external view returns(string[] memory) {
        return _account.getAccountRoles(stakeholder);
    }

    function getStakeholderReputation(address stakeholder)
    public view returns(uint) {
        return _rating.getLastReputation(stakeholder);
    }

    function getStakeholderTrust(address stakeholder)
    public view returns(uint) {
        return _rating.getLastTrust(stakeholder);
    }

    function isTrusted(address account)
```

```
public view returns(bool) {
    uint trust = getStakeholderTrust(account);
    return trust >=_minTrust;
}

function isTrustedProducer(address account)
public view returns(bool) {
    return _account.hasRoleProducer(account) &&
        this.isTrusted(account);
}

function isTrustedTrader(address account)
public view returns(bool)
{
     return _account.hasRoleTrader(account) &&
        this.isTrusted(account);
}

function isTrustedTransfer(address from, address to)
public view returns(bool)
{
    return (this.isTrustedProducer(from) ||
            this.isTrustedTrader(from)) &&
           (this.isTrustedProducer(to) ||
            this.isTrustedTrader(to));
}

function removeSensorProduct(uint productId)
external onlyOwner
{
   _productManagement.removeSensorProduct(productId);
}

function removeRole(string calldata role, address stakeholder)
external onlyOwner validRole(role) {
    _account.removeRolefromAccount(role, stakeholder);
}

function reputationCalc(
    uint[] memory ratings,
    uint[] memory history,
    uint[] memory subValues)
external pure returns(uint)
```

```
{   uint[] memory weights = new uint[](3);
    weights[0] = 3;
    weights[1] = 3;
    weights[2] = 3;
    require(weights.length == ratings.length,
        string.concat("Ratings array length must be equal to ",
        Strings.toString(weights.length)));
    uint result = 0;
    for(uint i = 0; i< ratings.length; i++) {
        result += ratings[i] * weights[i];
    }
    for(uint i = 0; i< subValues.length; i++) {
        result -= subValues[i];
    }
    if (history.length > 0) {
        uint average = 0;
        uint i = history.length;
        uint count = 0;
        while(count < 10 && i > 0) {
            count +=1;
            i -=1;
            average += history[i];
        }
        average = (result + average)/(count+1);
        result = uint(average);
    }
    return result;
}

//sensor send value and emit warning
function sensorSendData(uint[] calldata values)
external validSensor(msg.sender)
{

    uint[] memory ids = _productManagement.getSensorProduct(msg.sender);

    for (uint i; i < ids.length; i++) {

        uint productId = ids[i];
        if (_productManagement.checkProductBounds(productId, values)) {

            _productManagement.increaseProductQualityExceed(productId);
            emit QualityWarning(
```

```
                _productManagement.getProductOwner(productId),
                msg.sender,
                productId,
                _productManagement.getProductQuality(productId),
                block.timestamp,
                values
            );
        }
    }
}

function setTransfer(address to, uint productId)
external onlyOwner
{
    address from = _productManagement.getProductOwner(productId);
    require(from != to,
        "Source address and Destination address must be different!");
    require(this.isTrustedTransfer(from,to),
        "Only trusted producer or trader could trade resources!");
    _productManagement.setProductTransferAddress(to, productId);
}

function transferResource(
    uint productId,
    uint[] memory rating,
    uint[] memory ratingOtherValues,
    uint[] memory trustOtherValues
)
external {
    address to = _productManagement.getProductTransferAddress(productId);
    require(msg.sender == to,
        "Only destination trader could complete transfer!!!");
    address from = _productManagement.getProductOwner(productId);
    if (this.isTrustedTransfer(from,to)) {
        _productManagement.setProductTransfer(from, to, productId);
        _rating.setReputation(from, rating, ratingOtherValues);
        _rating.setTrust(from, trustOtherValues);
        _productManagement.resetProductQualityExceed(productId);
        if (_account.setPenalties(to, 2, _rating.getRatings(from))) {
           emit BannedAccount(to);
        }
    } else {
        emit UncompletedTransferWarning(
```

```
                from, to, productId, block.timestamp
            );
        }
        _productManagement.setProductTransferAddress(address(0), productId);
    }

    function trustCalc(
        uint[] memory reputations,
        uint[] memory history,
        uint[] memory subValues
    )
    external pure returns(uint)
    {
        uint result = reputations[reputations.length-1];
        for(uint i = 0; i< subValues.length; i++) {
            result -= subValues[i];
        }
        if (history.length > 0) {
            uint average = 0;
            uint i = history.length;
            uint count = 0;
            while(count < 10 && i > 0) {
                count +=1;
                i -=1;
                average += history[i];
            }
            average = (result + average)/(count+1);
            result = uint(average);
        }
        return result;
    }

    modifier validRole(string calldata role) {
        require(_account.isValidRole(role),"Not valid role!");
        _;
    }

    modifier validSensor(address sensor) {
        require(_account.hasRoleSensor(sensor), "Not valid sensor account!");
        _;
    }

}
```

Product.sol

```
// SPDX-License-Identifier: MIT

pragma solidity 0.8.12;

import "@openzeppelin/contracts@4.8.3/access/Ownable.sol";
import "@openzeppelin/contracts@4.8.3/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts@4.8.3/token/ERC721
        /extensions/ERC721URIStorage.sol";
import "@openzeppelin/contracts@4.8.3/utils/Counters.sol";

contract Product is ERC721, ERC721URIStorage, Ownable
{

    using Counters for Counters.Counter;
    Counters.Counter private _tokenIds;

    struct Resource {
        struct Resource {
        string name;
        string[] activityList;
        uint[] activityGHGList;
        uint[] otherResourceList;
        uint GHG;
    }

    mapping(uint=>Resource) private resources; // resources mapping

    constructor()
        ERC721("Product", "PDC"){
    }

    function _beforeTokenTransfer(
        address from, address to, uint firstTokenId, uint batchSize
    ) internal virtual override onlyOwner
    {
        super._beforeTokenTransfer(from, to, firstTokenId, batchSize);
    }

    function _burn(uint tokenId) internal override(ERC721, ERC721URIStorage) {
        require(tokenId == 0, "Token not burnable!");
        super._burn(tokenId);
```

```
    }

    function create(Resource calldata resource, address owner)
    external onlyOwner returns(uint)  {

        _tokenIds.increment();
        uint tokenId = _tokenIds.current();

        _safeMint(owner, tokenId);
        resources[tokenId] = resource;

        return tokenId;
    }

    function currentToken() external view returns(uint) {
        return _tokenIds.current();
    }

    function getResource(uint tokenId)
    external view validTokenId(tokenId) returns(Resource memory) {
        return resources[tokenId];
    }

    function isValidTokenId(uint tokenId) public view returns(bool) {
        return tokenId != 0 && tokenId <= _tokenIds.current();
    }

    function tokenURI(uint tokenId)
    public view override(ERC721, ERC721URIStorage) returns (string memory)
    {
        return super.tokenURI(tokenId);
    }

    function transferFrom(address from, address to, uint tokenId)
    public virtual override onlyOwner{
        _safeTransfer(from, to, tokenId, "");
    }

    modifier validTokenId(uint tokenId) {
        require(isValidTokenId(tokenId),"Not valid product Token ID!");
        _;
    }
}
```

ProductManagement.sol

```
/ SPDX-License-Identifier: MIT

pragma solidity 0.8.12;
//pragma experimental SMTChecker;

import "./Product.sol";
import "./Quality.sol";
import "./Utils.sol";
import "@openzeppelin/contracts@4.8.3/access/Ownable.sol";

contract ProductManagement is Ownable {

    mapping(uint=>uint) private _productQuality;
    mapping(uint=>uint) private _productQualityExceed;
    mapping(uint=>address) private _productSensor;
    mapping(address=>uint[]) private _sensorProducts;
    mapping(uint=>address) private _productTransfer;

    uint private _maxActivities;
    uint private _maxAggregateProducts;

    Product private _product = new Product();
    Quality private _quality = new Quality(this.checkBounds);

    constructor(
        uint maxActivities, uint maxAggregateProducts
    ) {
        require(maxActivities > 0,
            "Not valid max activites value!");
        require(maxAggregateProducts > 0,
            "Not valid max aggregate products value!");
        _maxActivities = maxActivities;
        _maxAggregateProducts = maxAggregateProducts;
    }

    function checkBounds(uint[] memory values, uint[] memory bounds)
    external pure returns(bool)
    {
        for (uint i=0; i < values.length; i++) {
           uint j = i*2;
           uint z = j+1;
```

```
            if (bounds[j] > values[i] || bounds[z] < values[i]) {
                return true;
            }
        }
        return false;
    }

    function checkProductBounds(
        uint productId,
        uint[] calldata values
    )
    external view validProductId(productId) returns(bool)
    {
        uint qualityId = _productQuality[productId];
        return _quality.executeCheckBounds(qualityId, values);
    }

    function createQuality(
      string calldata name, uint[] calldata bounds
    ) external onlyOwner returns(uint)
    {
        Quality.QualityBounds memory qualityBounds = Quality.QualityBounds(
            name, bounds
        );

        return _quality.create(qualityBounds);
    }

    function createProduct(
        address producer, string calldata name, string[] memory activities,
        uint[] memory activitiesGHG, uint[] memory products,
        uint qualityId, address sensor
    ) external validQualityId(qualityId) onlyOwner returns(uint) {

        require(activities.length > 0 &&
        activities.length <= _maxActivities &&
                activities.length == activitiesGHG.length,
                "Not valid activities number!");
        require(products.length <= _maxAggregateProducts,
            "Not valid aggregate products number!");

        uint totGHG = 0;
```

```
    Product.Resource memory resource;

    for (uint i=0; i < products.length; i++) {
        _product.isValidTokenId(products[i]);
        require(_product.ownerOf(products[i]) == producer,
            string.concat("Producer is not owner of: ",
            Strings.toString(products[i]))
        );
        resource = _product.getResource(products[i]);
        totGHG += resource.GHG;
    }

    for (uint i=0; i < activitiesGHG.length; i++) {
        totGHG += activitiesGHG[i];
    }

    resource = Product.Resource(
        name,
        activities,
        activitiesGHG,
        products,
        totGHG
    );

    uint productId = _product.create(
        resource, producer
    );

    _productQuality[productId] = qualityId;
    _productQualityExceed[productId] = 0;
    _sensorProducts[sensor].push(productId);
    _productSensor[productId] = sensor;

    return productId;
}

function getProductLastId() external view returns(uint) {
    return _product.currentToken();
}

function getProductOwner(uint productId)
external view validProductId(productId) returns(address)
{
```

```
        return _product.ownerOf(productId);
    }

    function getProductQuality(uint productId)
    external view returns(uint)
    {
        return _productQuality[productId];
    }

    function getProductQualityExceed(uint productId)
    external view validProductId(productId) returns(uint)
    {
        return _productQualityExceed[productId];
    }

    function getProductResource(uint productId)
    external view validProductId(productId) returns(Product.Resource memory)
    {
        return _product.getResource(productId);
    }

    function getProductTransferAddress(uint productId)
    external view onlyOwner validProductId(productId) returns(address)
    {
        return _productTransfer[productId];
    }

    function getQualityBounds(uint qualityId)
    external view validQualityId(qualityId) returns(Quality.QualityBounds memory)
    {
        return _quality.getQualityBounds(qualityId);
    }

    function getQualityLastId()
    external view returns(uint)
    {
        return _quality.currentId();
    }

    function getSensorProduct(address sensor)
    external view returns(uint[] memory)
    {
       return _sensorProducts[sensor];
```

```
}

function increaseProductQualityExceed(uint productId)
external onlyOwner validProductId(productId) returns(uint)
{
    _productQualityExceed[productId] += 1;
    return _productQualityExceed[productId];
}

function removeSensorProduct(uint productId)
external onlyOwner validProductId(productId) {
    require(_productSensor[productId]!=address(0),
        "Not valid sensor's product ID!"
    );
    address sensor = _productSensor[productId];
    Utils.orderedUintListRemove(productId, _sensorProducts[sensor]);
    _productSensor[productId]=address(0);
}

function resetProductQualityExceed(uint productId)
external onlyOwner
{
    _productQualityExceed[productId] = 0;
}

function setProductSensor(address sensor, uint productId)
external onlyOwner {
    _sensorProducts[sensor].push(productId);
    _productSensor[productId] = sensor;
}

function setProductTransfer(address from, address to, uint productId)
external onlyOwner validProductId(productId)
{
    _product.transferFrom(from, to, productId);
}

function setProductTransferAddress(address to, uint productId)
external onlyOwner validProductId(productId)
{
    _productTransfer[productId] = to;
}
```

```
    modifier validProductId(uint productId) {
        require(_product.isValidTokenId(productId),"Not valid product ID!");
        _;
    }

    modifier validQualityId(uint qualityId) {
        require(_quality.isValidId(qualityId),"Not valid quality ID!");
        _;
    }

}
```

Quality.sol

```
// SPDX-License-Identifier: MIT

pragma solidity 0.8.12;

import "@openzeppelin/contracts@4.8.3/access/Ownable.sol";
import "@openzeppelin/contracts@4.8.3/utils/Counters.sol";
import "@openzeppelin/contracts@4.8.3/utils/Strings.sol";

contract Quality is Ownable {

    // id's counter
    using Counters for Counters.Counter;
    Counters.Counter private _ids;

    function(uint[] memory, uint[] memory)
        external pure returns(bool) _checkBounds;

    struct QualityBounds {
      string name;
      uint[] bounds;
    }

    mapping(uint=>QualityBounds) private _qualitiesBounds;

    constructor(function(uint[] memory, uint[] memory)
    external pure returns(bool) checkBounds)
    {
        _checkBounds = checkBounds;
    }
```

```
    function create(QualityBounds calldata qualityBounds)
    external onlyOwner returns(uint)
    {
        uint[] calldata bounds = qualityBounds.bounds;
        require(bounds.length > 0, "Inser at least one bound!");
        _ids.increment();
        uint id = _ids.current();
        _qualitiesBounds[id] = qualityBounds;
        return id;
    }

    function currentId() external view returns(uint) {
        return _ids.current();
    }

    function executeCheckBounds(uint id, uint[] memory values)
    external validId(id) view returns(bool)
    {
return _checkBounds(values, _qualitiesBounds[id].bounds);
    }

    function getQualityBounds(uint id)
    external view validId(id) returns(QualityBounds memory) {
        return _qualitiesBounds[id];
    }

    function isValidId(uint id) public view returns(bool) {
        return id!=0 && id <= _ids.current();
    }

    modifier validId(uint id) {
        require(isValidId(id), "Not valid quality  ID!");
        _;
    }
}
```

Rating.sol

```
// SPDX-License-Identifier: MIT

pragma solidity 0.8.12;
```

```
import "@openzeppelin/contracts@4.8.3/access/Ownable.sol";
import "@openzeppelin/contracts@4.8.3/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts@4.8.3/token/ERC721/
       extensions/ERC721URIStorage.sol";
import "@openzeppelin/contracts@4.8.3/utils/Counters.sol";
import "@openzeppelin/contracts@4.8.3/utils/Strings.sol";

contract Rating is ERC721, ERC721URIStorage, Ownable {

    using Counters for Counters.Counter;
    Counters.Counter private _tokenIds;

    function(uint[] memory, uint[] memory, uint[] memory)
        external pure returns(uint) _reputationCalc;
    function(uint[] memory, uint[] memory, uint[] memory)
        external pure returns(uint) _trustCalc;

    mapping(address => uint) private _accountSBT;
    mapping(address => uint[][]) private _ratings;
    mapping(address => uint[]) private _reputations;
    mapping(address => uint[]) private _trusts;

    constructor(
        function(uint[] memory, uint[] memory, uint[] memory)
            external pure returns(uint) reputationCalc,
        function(uint[] memory, uint[] memory, uint[] memory)
            external pure returns(uint) trustCalc
    ) ERC721("Trust", "TRS")
    {
        _reputationCalc = reputationCalc;
        _trustCalc = trustCalc;
    }

    function _beforeTokenTransfer(
        address from, address to, uint tokenId, uint batchSize
    ) internal override(ERC721)
    {
        require(from == address(0), "Token not transferable!");
        super._beforeTokenTransfer(from, to, tokenId, batchSize);
    }

    function _burn(uint tokenId) internal override(ERC721, ERC721URIStorage) {
        require(tokenId == 0, "Token not burnable!");
```

```
    super._burn(tokenId);
}

function createSBT(address account)
external onlyOwner
{
    uint tokenId = _accountSBT[account];
    if (tokenId <= 0) {
        _tokenIds.increment();
        tokenId = _tokenIds.current();
        _safeMint(account, tokenId);
        _accountSBT[account] = tokenId;
    }
}

function getLastRatings(address account) external view returns(uint[] memory)
{
    uint[][] memory accountRatings = _ratings[account];
    return accountRatings[accountRatings.length-1];
}

function getLastReputation(address account) external view returns(uint)
{
    uint[] memory accountReputation = _reputations[account];
    return accountReputation[accountReputation.length-1];
}

function getLastTrust(address account) external view returns(uint)
{
    uint[] memory accountTrust = _trusts[account];
    return accountTrust[accountTrust.length-1];
}

function getRatings(address account)
external view returns(uint[][] memory)
{
    return _ratings[account];
}

function getReputation(address account) external view returns(uint[] memory)
{
    return _reputations[account];
}
```

```
function getTrust(address account) external view returns(uint[] memory)
{
    return _trusts[account];
}

function isValidTokenId(uint tokenId)
public view returns(bool) {
    return tokenId!=0 && tokenId <= _tokenIds.current();
}

function resetTrust(address account, uint newTrust)
external onlyOwner
{
    _trusts[account].push(newTrust);
}

function setReputation(
    address account,
    uint[] memory ratings,
    uint[] memory otherValues
) external onlyOwner returns(uint)
{
    uint reputation = _reputationCalc(
        ratings,
        _reputations[account],
        otherValues
    );
    _ratings[account].push(ratings);
    _reputations[account].push(reputation);
    return reputation;
}

function setTrust(
    address account,
    uint[] memory otherValues
) external onlyOwner returns(uint)
{
    uint trust = _trustCalc(
        _reputations[account],
        _trusts[account],
        otherValues
    );
```

```
        _trusts[account].push(trust);
        return trust;
    }

    function tokenURI(uint tokenId)
    public view override(ERC721, ERC721URIStorage) returns (string memory)
    {
        return super.tokenURI(tokenId);
    }

    modifier validTokenId(uint tokenId)
    {
        require(isValidTokenId(tokenId), "Not valid trust token ID!");
        _;
    }
}
```

Utils.sol

```
// SPDX-License-Identifier: MIT

pragma solidity 0.8.12;
//pragma experimental SMTChecker;/

import "@openzeppelin/contracts@4.8.3/utils/math/Math.sol";
import "@openzeppelin/contracts@4.8.3/utils/Strings.sol";

library Utils {

    function abs(int x)
    internal pure returns (uint) {
        return x >= 0 ? uint(x) : uint(-x);
    }

    function calcStocastic(uint[] memory data, uint historylength)
    internal pure returns(bool, uint, uint)
    {

        if (data.length > 0) {
            uint average = 0;
            uint i = data.length;
            uint count = 0;
            while(count < historylength && i > 0) {
```

```
            count +=1;
            i -=1;
            average += data[i];
        }
        average = average/count;
        uint stdDev = 0;
        i = data.length;
        count = 0;
        while(count < historylength && i > 0) {
            count +=1;
            i -=1;
            unchecked {
                uint diff = abs(int(data[i] - average));
                stdDev += diff * diff;
            }
        }
        stdDev = Math.sqrt(stdDev/count, Math.Rounding.Up);
        return (true, uint(average), uint(stdDev));
    } else {
        return (false, 0, 0);
    }
}

function orderedUintListRemove(uint needle, uint[] storage haystack)
internal
{
    (bool found, uint index) = uintListFindIndex(needle, haystack);
    haystack.pop();
    if (found) {
        for (uint i = index; i < haystack.length; i++) {
            haystack[i] = haystack[i+1];
        }
        haystack.pop();
    }
}

function sqrt(uint y)
internal pure returns (uint)
{
    uint z = 0;
    if (y > 3) {
        z = y;
        uint x = y / 2 + 1;
```

```
        while (x < z) {
            z = x;
            x = (y / x + x) / 2;
        }
    } else if (y != 0) {
        z = 1;
    }
    return z;
}

function stringsEqual(string memory a, string memory b)
internal pure returns (bool)
{
    return keccak256(bytes(a)) == keccak256(bytes(b));
}

function stringListExists(string memory needle, string[] memory haystack)
internal pure returns (bool)
{
   (bool found,) = stringListFindIndex(needle, haystack);
   return found;
}

function stringListFindIndex(string memory needle, string[] memory haystack)
internal pure returns (bool, uint)
{
   for (uint i = 0; i < haystack.length; i++) {
       if (stringsEqual(haystack[i], needle)) {
            return (true, i);
       }
   }
   return (false, 0);
}

function stringListRemove(string memory needle, string[] storage haystack)
internal
{
    (bool found, uint index) = stringListFindIndex(needle, haystack);
    if (found) {
        haystack[index] = haystack[haystack.length - 1];
        haystack.pop();
    }
}
```

```
    function uintListFindIndex(uint needle, uint[] memory haystack)
    internal pure returns (bool, uint)
    {
       for (uint i = 0; i < haystack.length; i++) {
            if (haystack[i] == needle) {
                return (true, i);
            }
       }
       return (false, 0);
    }

}
```

Appendice B

Codice PHP

Di seguito il codice sorgente, in linguaggio PHP, dell'esempio applicativo. Tale codice viene eseguito lato server e utilizza gli smart contract trattati in Appendice A, rispettivamente: *account_trust.php*, *add_role.php*, *complete_transfer.php*, *index.php*, *insert_product.php*, *insert_quality.php*, *library.php*, *logout.php*, *main.php*, *operations.php*, *product_owner.php*, *product_search.php*, *sensor_send.php*, *session.php*, *sign.php*, *transfer_product.php*.

account_trust.php

```
<?php
if (isset($_POST['selectAccount'])) {
    $resultString = 'Account trust value is ' . $contract
    ->call('getStakeholderTrust', [$_POST['selectAccount']])
    ->result . '.';
    $smarty->assign('result', $resultString);
}
```

add_role.php

```
<?php
if (isset($_POST['selectAccount']) &&
    isset($_POST['selectRole'])) {
    $send_data = new stdClass();
    $send_data->role = $_POST['selectRole'];
    $send_data->account = $_POST['selectAccount'];
    $extra_data = ['nonce' => $web3->personal->getNonce()];
    try {
        $result = $contract
        ->send('addRole', get_object_vars($send_data), $extra_data);
```

```
        $resultString = getTransactionResultString($result, $web3);
        $smarty->assign('result', $resultString);
        $accounts = getAccountsRoles($accountsInitial, $contract);
        $smarty->assign('accounts', $accounts);
    } catch (Exception $e) {
        $smarty->assign('result', 'ERROR!!!&#13;&#10' . $e->getMessage());
    }
}
```

complete_transfer.php

```
<?php
if (isset($_POST['productID']) &&
    isset($_POST['traderRating'])
    ) {
    $mockRegulatorRating = 10;
    $productQualityExceed = intval($contract
        ->call('getProductQualityExceed', [$_POST['productID']])
        ->result
    );
    $sensorRating = 10;
    if ($productQualityExceed < $sensorRating) {
        $sensorRating = $sensorRating - $productQualityExceed;
    } else {
        $sensorRating = 0;
    }
    $productGHG = intval($contract
        ->call('getProductResource', [$_POST['productID']])
        ->tuple_1->GHG
    );
    $send_data = new stdClass();
    $send_data->productId = $_POST['productID'];
    $send_data->rating = [
        $mockRegulatorRating, $sensorRating, $_POST['traderRating']
    ];
    $send_data->ratingOtherValues = [];
    $send_data->trustOtherValues = [intval(ceil($productGHG/100))];
    $extra_data = ['nonce' => $web3->personal->getNonce()];
try {
        $result = $contract->send('transferResource',
            get_object_vars($send_data), $extra_data
        );
        $resultString = getTransactionResultString($result, $web3);
```

```
        $smarty->assign('result', $resultString);
    } catch (Exception $e) {
        $smarty->assign('result', 'ERROR!!!&#13;&#10' . $e->getMessage());
    }
}
```

index.php

```
<?php

ini_set('display_errors', 1);
ini_set('display_startup_errors', 1);
error_reporting(E_ALL);

require_once(__DIR__ . '/vendor/autoload.php');
require_once('library.php');
require_once('session.php');

use stdClass;
use SWeb3\SWeb3;
use SWeb3\Utils;
use SWeb3\SWeb3_Contract;
use phpseclib\Math\BigInteger as BigNumber;

define('ADMIN_ACCOUNT', '0xFE3B557E8Fb62b89F4916B721be55cEb828dBd73');
define('ETH_PROVIDER', 'http://localhost:8545');
define('MAIN_CONTRACT_PATH', __DIR__ . '/contracts/Main');
define('MAIN_CONTRACT_PATH_JSON', MAIN_CONTRACT_PATH . '.json');
define('MAX_PRODUCT_ACTIVITIES', 5);
define('MAX_PRODUCT_AGGREGATE', 5);
define('MIN_TRUST', 30);
define('ROLE_ADMIN', 'admin');
define('ROLE_PRODUCER', 'producer');
define('ROLE_SENSOR', 'sensor');
define('ROLE_TRADER', 'trader');

require_once('operations.php');

$smarty = new Smarty();

$smarty->setTemplateDir(__DIR__ . '/smarty/templates');
$smarty->setCompileDir(__DIR__ . '/smarty/templates_c');
$smarty->setCacheDir(__DIR__ . '/smarty/cache');
```

```
$smarty->setConfigDir(__DIR__ . '/smarty/configs');

$boolContractDeploy = false;
$web3 = new SWeb3(ETH_PROVIDER);
$web3->chainId = hexdec($web3->call('eth_chainId', [])->result);
$contractInfo = json_decode(file_get_contents(MAIN_CONTRACT_PATH_JSON));

$accountsInitial = new ArrayObject(array(
    ADMIN_ACCOUNT => [ROLE_ADMIN],
    '0x627306090abaB3A6e1400e9345bC60c78a8BEf57' => [],
    '0xf17f52151EbEF6C7334FAD080c5704D77216b732' => [],
    '0x03Dd24EE515e6c74cE6EA24Ea94aaC55547e5044' => [],
    '0x6d832866fB72F5f78AE7a4B7cddf1721E8343094' => [],
    '0xE6aBF89FCEB42Cfb5F581550A958e2D364375eF2' => [],
    '0x1180b3f981ceb70e2E39e0EEdB0c1C0A1a1e0eB4' => []
));

$abi = file_get_contents(MAIN_CONTRACT_PATH . '.abi');
$res = $web3->call('eth_getCode',
            [$contractInfo->contractAddress, 'latest']);
if ($res->result !== '0x') {
    $contract = new SWeb3_contract($web3,
        $contractInfo->contractAddress, $abi
    );
    $accounts = getAccountsRoles($accountsInitial, $contract);
    $smarty->assign('accounts', $accounts);
    require_once('main.php');
} else {
    $accounts = $accountsInitial->getArrayCopy();
    $boolContractDeploy = true;
    $contract = new SWeb3_contract($web3, '', $abi);
    require_once('sign.php');
    $accountAdmin = [];
    $accountAdmin[ADMIN_ACCOUNT] = $accounts[ADMIN_ACCOUNT];
    $smarty->assign('accounts', $accountAdmin);
}
$smarty->display('index.tpl');
```

insert_product.php

```
<?php
$sensorAccounts = [];
foreach($accounts as $account => $permissions) {
```

```
    if (in_array(ROLE_SENSOR, $permissions, true)) {
        $sensorAccounts[] = $account;
    }
}
$smarty->assign('sensorAccounts', $sensorAccounts);
if (isset($_POST['productName']) &&
    isset($_POST['ghgValue']) &&
    isset($_POST['qualityID']) &&
    isset($_POST['sensorAccount'])
    ) {
    $send_data = new stdClass();
    $send_data->name = $_POST['productName'];
    $send_data->activities = ['Raw Material'];
    $send_data->activitiesGHG = [$_POST['ghgValue']];
    $send_data->products = [];
    $send_data->qualityId = $_POST['qualityID'];
    $send_data->sensor = $_POST['sensorAccount'];
    $extra_data = ['nonce' => $web3->personal->getNonce()];
    try {
        $result = $contract->send('createProduct',
        get_object_vars($send_data), $extra_data
        );
        $resultString = getTransactionResultString($result, $web3);
        $resultString .= 'Product ID: ' .
            $contract->call('getProductLastId', [])
            ->result;
        $smarty->assign('result', $resultString);
    } catch (Exception $e) {
        $smarty->assign('result', 'ERROR!!!&#13;&#10' . $e->getMessage());
    }
}
```

insert_quality.php

```
<?php
if (isset($_POST['qualityName']) &&
    isset($_POST['lowTemperature']) &&
    isset($_POST['highTemperature'])
    ) {
    $send_data = new stdClass();
    $send_data->name = $_POST['qualityName'];
    $send_data->bounds = [$_POST['lowTemperature'], $_POST['highTemperature']];
    $extra_data = ['nonce' => $web3->personal->getNonce()];
```

```
    try {
        $result = $contract->send(
            'createQuality',
            get_object_vars($send_data),
            $extra_data
        );
        $resultString = getTransactionResultString($result, $web3);
        $resultString .= 'Quality Contract ID: ' .
        $contract->call('getQualityLastId', [])->result;
        $smarty->assign('result', $resultString);
    } catch (Exception $e) {
        $smarty->assign('result', 'ERROR!!!&#13;&#10' . $e->getMessage());
    }
}
```

library.php

```
<?php
function getAccountsRoles($accountsInitial, $contract) {
    $resultAccounts = $accountsInitial->getArrayCopy();
    foreach($resultAccounts as $account=>$permission) {
        $roles = (array) $contract->call('getStakeholderRoles',
            [$account]
        );
        foreach($roles['array_1'] as $role) {
            array_push($resultAccounts[$account], $role);
        }
        asort($resultAccounts[$account]);
    }
    return $resultAccounts;
}

function getTransactionResultString($result, $web3) {
    $resultString = 'Transaction succesfully sent: ' .
        $result->result . '&#13;&#10;&#13;&#10;';
    $txResult = null;
    do {
        sleep(2);
        $checkTx = $web3->call('eth_getTransactionReceipt',
            [$result->result]
        );
        if(isset($checkTx->result)) $txResult = $checkTx->result;
    } while (is_null($txResult));
```

```
    $resultString .= json_encode($txResult) . '&#13;&#10;&#13;&#10;';
    return $resultString;
}
```

logout.php

```
<?php
session_start();
unset($_SESSION['account']);
header('Location: index.php');
```

main.php

```
<?php
if (isset($_GET['login'])) {
    require_once('sign.php');
} else {
    $smarty->assign('body', 'main.tpl');
    $newOperation = [];
    if (isset($_SESSION['account'])) {
        foreach($operations as $operation) {
            if ($operation->auth) {
                $result = false;
                foreach($operation->permission as $permission) {
                    if (in_array($permission,
                        $_SESSION['account']->permission,
                        true)) {
                        $result = true;
                        break;
                    }
                }
                if ($result) {
                $newOperation[] = $operation;
                }
            }
        }
        $smarty->assign('account', $_SESSION['account']->address);
        $smarty->assign('permissions',
            implode(', ', $_SESSION['account']->permission)
        );
        $smarty->assign('logged', true);
        $web3->setPersonalData(
            $_SESSION['account']->address,
```

```
            $_SESSION['account']->privateKey
        );
    } else {
        $smarty->assign('logged', false);
    }
    foreach($operations as $operation) {
        if (!$operation->auth)
            $newOperation[] = $operation;
    }
    $smarty->assign('operations', $newOperation);
    if (isset($_GET['page'])) {
        $operation = $newOperation[$_GET['page']];
        $smarty->assign('result', 'Result will be visible here!');
        require_once($operation->page . '.php');
        $smarty->assign('page', $operation->page . '.tpl');
        $smarty->assign('pageFa', $operation->fa);
        $smarty->assign('pageTitle', $operation->title);
    }
}
```

operations.php

```
<?php
$operations = [];

$operation = new stdClass();
$operation->auth = false;
$operation->description = 'Gets the trust value of an account.';
$operation->fa = 'fa-user';
$operation->page = 'account_trust';
$operation->permission = [];
$operation->title = 'Account trust';
array_push($operations, $operation);

$operation = new stdClass();
$operation->auth = false;
$operation->description = 'Gets the information of a product.';
$operation->fa = 'fa-search';
$operation->page = 'product_search';
$operation->permission = [];
$operation->title = 'Product search';
array_push($operations, $operation);
```

```
$operation = new stdClass();
$operation->auth = false;
$operation->description = 'Gets the owner account of the product.';
$operation->fa = 'fa-user-lock';
$operation->page = 'product_owner';
$operation->permission = [];
$operation->title = 'Product owner';
array_push($operations, $operation);

$operation = new stdClass();
$operation->auth = true;
$operation->description = 'Create a quality contract involving temperature.';
$operation->fa = 'fa-thermometer';
$operation->page = 'insert_quality';
$operation->permission = [ROLE_ADMIN];
$operation->title = 'Insert quality contract';
array_push($operations, $operation);

$operation = new stdClass();
$operation->auth = true;
$operation->description = 'Permit to add roles to accounts.';
$operation->fa = 'fa-users';
$operation->page = 'add_role';
$operation->permission = [ROLE_ADMIN];
$operation->title = 'Add role';
array_push($operations, $operation);

$operation = new stdClass();
$operation->auth = true;
$operation->description = 'A sensor that sends the current product temperature.';
$operation->fa = 'fa-tachometer-alt';
$operation->page = 'sensor_send';
$operation->permission = [ROLE_SENSOR];
$operation->title = 'Sensor: send data';
array_push($operations, $operation);

$operation = new stdClass();
$operation->auth = true;
$operation->description = 'Permit to create a new product.';
$operation->fa = 'fa-utensils';
$operation->page = 'insert_product';
$operation->permission = [ROLE_PRODUCER];
$operation->title = 'Insert product';
```

```
array_push($operations, $operation);

$operation = new stdClass();
$operation->auth = true;
$operation->description = ' Initializes the transfer of a product.';
$operation->fa = 'fa-truck-loading';
$operation->page = 'transfer_product';
$operation->permission = [ROLE_ADMIN];
$operation->title = 'Transfer product';
array_push($operations, $operation);

$operation = new stdClass();
$operation->auth = true;
$operation->description = 'Completes the transfer with a rating vote.';
$operation->fa = 'fa-truck-moving';
$operation->page = 'complete_transfer';
$operation->permission = [ROLE_PRODUCER, ROLE_TRADER];
$operation->title = 'Complete transfer';
array_push($operations, $operation);
```

product_owner.php

```
<?php
if (isset($_POST['productID'])) {
    $resultString = 'The owner of the product is ' .
    $contract->call('getProductOwner',
        [$_POST['productID']])->result . '.';
    $smarty->assign('result', $resultString);
}
```

product_search.php

```
<?php
if (isset($_POST['productID'])) {
    $resource = $contract->call('getProductResource',
        [$_POST['productID']])->tuple_1;
    $resultString = 'Product components: &#13;&#10;&#13;&#10;' .
    'Name: ' . $resource->name . '&#13;&#10;' .
    'GHG: ' . $resource->GHG;
    $smarty->assign('result', $resultString);
}
```

sensor_send.php

```
<?php
if (isset($_POST['temperature'])) {
    $extra_data = ['nonce' => $web3->personal->getNonce()];
    try {
        $result = $contract->send(
            'sensorSendData',
            [$_POST['temperature']],
            $extra_data
        );
        $resultString = getTransactionResultString($result, $web3);
        $res = $contract->getLogs('latest');
        if (is_array($res) &&
            count($res)>0) {
            $resultString .= 'In the logs there is an event called ' .
            $res[0]->event_name;
        } else {
            $resultString .= 'There are no events in the logs';
        }
        $smarty->assign('result', $resultString);
    } catch (Exception $e) {
        $smarty->assign('result', 'ERROR!!!&#13;&#10' . $e->getMessage());
    }
}
```

session.php

```
<?php

session_start();

if (!isset($_SESSION['destroyed'])) {
    $_SESSION['destroyed'] = time();
}

if ($_SESSION['destroyed'] < time() - 3600) {
    session_regenerate_id(true);
} else {
    session_regenerate_id();
}
```

sign.php

```
<?php
$smarty->assign('body', 'sign.tpl');
$smarty->assign('boolContractDeploy', $boolContractDeploy);

if (isset($_POST['accountSelect']) &&
    !isset($_SESSION['account'])) {

    $account = new \stdClass();
    $account->address = $_POST['accountSelect'];
    $account->privateKey = $_POST['privateKey'];
    $web3->setPersonalData($account->address, $account->privateKey);
    $account->permission = $accounts[$account->address];
    $_SESSION['account'] = $account;

    if ($boolContractDeploy) {
        $bytecode = file_get_contents(MAIN_CONTRACT_PATH . '.bin');
        $contract->setBytecode($bytecode);
        $nonce =  $web3->personal->getNonce();
        $extra_params = [
            'from' => $account->address,
            'nonce' => $nonce
        ];
        $result = $contract->deployContract(
            [MAX_PRODUCT_ACTIVITIES, MAX_PRODUCT_AGGREGATE, MIN_TRUST],
            $extra_params
        );
        if(isset($result->result))
        {
            $newAddress = '';

            do {
                sleep(5);
                $checkContract = $web3->call('eth_getTransactionReceipt',
                    [$result->result]
                );
                if(isset($checkContract->result->contractAddress))
                    $newAddress = $checkContract->result->contractAddress;
            } while ($newAddress == '');

            file_put_contents(
                MAIN_CONTRACT_PATH_JSON,
```

```
                json_encode(array(
                    'abi' => $abi,
                    'bytecode' => $bytecode,
                    'contractAddress' => $newAddress
                )));
        }
        else
        {
            echo 'Transaction error <br/>';
            echo json_encode($result);
        }
    }
    header("Status: 301 Moved Permanently");
    header("Location: index.php");
    exit();
}
```

transfer_product.php

```
<?php
$transferAccounts = [];
foreach($accounts as $account => $permissions) {
    if (in_array(ROLE_PRODUCER, $permissions, true) ||
        in_array(ROLE_TRADER, $permissions, true)
    ) {
        $transferAccounts[$account] = $permissions;
    }
}
$smarty->assign('accounts', $transferAccounts);
if (isset($_POST['productID']) &&
    isset($_POST['selectAccount'])
    ) {
    $send_data = new stdClass();
    $send_data->to = $_POST['selectAccount'];
    $send_data->productId = $_POST['productID'];
    $extra_data = ['nonce' => $web3->personal->getNonce()];
    try {
        $result = $contract->send('setTransfer',
            get_object_vars($send_data), $extra_data
        );
        $resultString = getTransactionResultString($result, $web3);
        $smarty->assign('result', $resultString);
    } catch (Exception $e) {
```

```
        $smarty->assign('result', 'ERROR!!!&#13;&#10' . $e->getMessage());
    }
}
```

Codice HTML, CSS, Javascript

Come già accennato in precedenza, l'applicativo è fruibile tramite browser web, e quindi comprensivo di parti scritte in codice *HTML5*, *CSS3* e *Javascript*. Il progetto completo con tutto il codice sorgente, quindi anche file *HTML*, *CSS* e *Javascript* è liberamente scaricabile dal repository https://github.com/aleca3112/Enoughchain.

Bibliografia

[1] Massimiliano Pirani, Alessio Cacopardo, Alessandro Cucchiarelli e Luca Spalazzi. «A Soulbound Token-based Reputation System in Sustainable Supply Chains». In: *The 2nd International Workshop on Hybrid Internet of Everything Models for Industry 5.0 (HIEMI 2023)*. ACM. 2023, pp. 1–6.

[2] ENOUGH, g. a. No 101036588. *European Food Chain Supply to Reduce GHG Emissions by 2050*. Available at https://enough-emissions.eu/ (2023/05/31).

[3] DG RTD, EU Commission. *Industry 5.0 - Towards a sustainable, human-centric and resilient European industry*. Available at https://op.europa.eu/en/publication-detail/-/publication/468a892a-5097-11eb-b59f-01aa75ed71a1/ (2023/05/31), 2021.

[4] Jiewu Leng, Weinan Sha, Baicun Wang, Pai Zheng, Cunbo Zhuang, Qiang Liu, Thorsten Wuest, Dimitris Mourtzis e Lihui Wang. «Industry 5.0: Prospect and retrospect». In: *Journal of Manufacturing Systems* 65 (2022), pp. 279–295.

[5] Francesco Longo, Antonio Padovano e Steven Umbrello. «Value-oriented and ethical technology engineering in Industry 5.0: A human-centric perspective for the design of the factory of the future». In: *Applied Sciences* 10.12 (2020), p. 4182.

[6] M. Kjaer. «Analyzing the Use of Blockchains for Challenges in Inter-organizational Business Processes». In: Institute of Electrical e Electronics Engineers Inc., 2023, pp. 137–140.

[7] P. Vilchez, S. Jacques, F. Freitag e R. Meseguer. «LoRaTRUST: Blockchain-Enabled Trust and Accountability Service for IoT Data». In: *Electronics (Switzerland)* 12.9 (2023).

[8] Meimin Wang, Zhili Zhou e Chun Ding. «Blockchain-based decentralized reputation management system for internet of everything in 6G-enabled cybertwin architecture». In: *Journal of New Media* 3.4 (2021), p. 137.

[9] An Binh Tran, Qinghua Lu e Ingo Weber. «Lorikeet: A Model-Driven Engineering Tool for Blockchain-Based Business Process Execution and Asset Management.» In: *BPM (Dissertation/Demos/Industry)*. 2018, pp. 56–60.

[10] Jan Mendling, Ingo Weber, Wil Van Der Aalst, Jan Vom Brocke, Cristina Cabanillas, Florian Daniel, Søren Debois, Claudio Di Ciccio, Marlon Dumas, Schahram Dustdar et al. «Blockchains for business process management-challenges and opportunities». In: *ACM Transactions on Management Information Systems (TMIS)* 9.1 (2018), pp. 1–16.

[11] Peter Bodorik, Christian Gang Liu e Dawn Jutla. «TABS: Transforming automatically BPMN models into blockchain smart contracts». In: *Blockchain: Research and Applications* 4.1 (2023), p. 100115.

[12] Luca Spalazzi, Francesco Spegni, Alessandra Corneli e Berardo Naticchia. «Blockchain based choreographies: The construction industry case study». In: *Concurrency and Computation: Practice and Experience* (2021), e6740.

[13] Francesco Spegni, Lorenzo Fratini, Massimiliano Pirani e Luca Spalazzi. «ChoEn : A Smart Contract Based Choreography Enforcer». In: *BRAIN 2023: Fourth Workshop on Blockchain theoRy and ApplicatIoNs, IEEE PERCOM 2023 The 21th International Conference on Pervasive Computing and. Communications, in press.* IEEE. 2023, pp. 1–8.

[14] Emanuele Bellini, Youssef Iraqi e Ernesto Damiani. «Blockchain-based distributed trust and reputation management systems: A survey». In: *IEEE Access* 8 (2020), pp. 21127–21151.

[15] Matt Bishop. *Computer Security: Art and Science. 2nd Edition.* Pearson, 2019.

[16] M. Aaqib, A. Ali, L. Chen e O. Nibouche. «IoT trust and reputation: a survey and taxonomy». In: *Journal of Cloud Computing* 12.1 (2023).

[17] Y. Liu, J. Wang, Z. Yan, Z. Wan e R. Jantti. «A Survey on Blockchain-Based Trust Management for Internet of Things». In: *IEEE Internet of Things Journal* 10.7 (2023), pp. 5898–5922.

[18] P. Burgess, F. Sunmola e S. Wertheim-Heck. «Blockchain Enabled Quality Management in Short Food Supply Chains». In: a cura di Padovano A. Longo F. Affenzeller M. Vol. 200. Elsevier B.V., 2022, pp. 904–913.

[19] Sidra Malik, Volkan Dedeoglu, Salil S Kanhere e Raja Jurdak. «Trustchain: Trust management in blockchain and iot supported supply chains». In: *2019 IEEE International Conference on Blockchain (Blockchain)*. IEEE. 2019, pp. 184–193.

[20] Y. Bai, K. Fan, K. Zhang, X. Cheng, H. Li e Y. Yang. «Blockchain-based trust management for agricultural green supply: A game theoretic approach». In: *Journal of Cleaner Production* 310 (2021).

[21] Yuan Liu, Chuang Zhang, Yu Yan, Xin Zhou, Zhihong Tian e Jie Zhang. «A semi-centralized trust management model based on blockchain for data exchange in iot system». In: *IEEE Transactions on Services Computing* (2022).

[22] Guntur Dharma Putra, Changhoon Kang, Salil S Kanhere e James Won-Ki Hong. «DeTRM: Decentralised trust and reputation management for blockchain-based supply chains». In: *2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE. 2022, pp. 1–5.

[23] Yue Wu e Yingfeng Zhang. «An integrated framework for blockchain-enabled supply chain trust management towards smart manufacturing». In: *Advanced Engineering Informatics* 51 (2022).

[24] Sana Alam, Shehnila Zardari e Jawwad Ahmed Shamsi. «Blockchain-Based Trust and Reputation Management in SIoT». In: *Electronics* 11.23 (2022), p. 3871.

[25] Andrea Bonci, Sauro Longhi e Massimiliano Pirani. «Prospective ISO 22400 for the challenges of human-centered manufacturing». In: *IFAC-PapersOnLine* 52.13 (2019), pp. 2537–2543.

[26] Giulio Caldarelli. «Understanding the blockchain oracle problem: A call for action». In: *Information* 11.11 (2020), p. 509.

[27] Doriane Perard, Jérôme Lacan, Yann Bachy e Jonathan Detchart. «Erasure Code-Based Low Storage Blockchain Node». In: *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. 2018, pp. 1622–1627. DOI: 10.1109/Cybermatics_2018.2018.00271.

Ringraziamenti

In primis, un ringraziamento particolare al mio relatore, il professor Spalazzi, che mi ha seguito, con la sua infinita disponibilità, in ogni step della realizzazione di questo progetto, fin dalla scelta dell'argomento.

Grazie anche al mio correlatore, l'ingegner Pirani, per la sua gentilezza, pazienza e per i suoi preziosi consigli e suggerimenti.

Ringrazio i miei genitori, per aver creduto in me e per aver sempre appoggiato le mie scelte.

Un grazie infinito a tutti coloro che mi hanno supportato durante questo lungo percorso.

Infine, dedico questa tesi a me stesso, ai miei sacrifici e alla mia perseveranza che mi hanno permesso di raggiungere questo traguardo.