UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI INGEGNERIA
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea Magistrale in Ingegneria Informatica e dell'Automazione

TESI DI LAUREA

Progettazione e implementazione di un sistema basato su blockchain per la certificazione dell'autenticità di prodotti di lusso

Design and implementation of a blockchain-based system for authenticity certification of luxury products

Relatore

Prof. Domenico Ursino

Candidato

Federico Miscia

Correlatori

Prof. Luca Spalazzi

Dott. Gianluca Bonifazi

ANNO ACCADEMICO 2022-2023

Muore lentamente,
chi passa i giorni a lamentarsi
della propria sfortuna o della pioggia incessante.
Lentamente muore,
chi abbandona un progetto
prima di iniziarlo,
chi non fa domande
sugli argomenti che non conosce,
chi non risponde
quando gli chiedono
qualcosa che conosce.

Martha Medeiros, "Lentamente muore"

Sommario

Con la recente introduzione delle blockchain, ben presto si è diffusa la consapevolezza che tale tecnologia potesse avere riscontri interessanti in molteplici applicazioni pratiche. Una di queste è il tracciamento del ciclo di vita dei prodotti, ambito nel quale, spesso, si manifestano problemi di trasparenza e di accentramento del controllo dei dati. Tali limitazioni rendono ulteriormente complesso uno scenario già profondamente eterogeneo, ostacolando la lotta al fenomeno della contraffazione. In questa tesi viene proposto un sistema per la certificazione dell'autenticità di prodotti di lusso. Nello specifico, a partire dai punti deboli delle attuali soluzioni per il tracciamento, sono stati definiti i requisiti su cui soffermarsi. In seguito, la progettazione del *software* ha chiarito i vantaggi dell'utilizzo delle blockchain ed ha posto le basi per l'implementazione di un'applicazione decentralizzata tramite cui gestire, in modo sicuro, il passaggio di proprietà di beni di lusso verificati.

Keyword: Blockchain, NFT, Smart Contract, Tracciamento, Autenticità, Contraffazione, Supply-chain, Prodotti di lusso

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

Certificare l'autenticità dei prodotti è diventato un imperativo per la società moderna, da perseguire tanto nell'interesse delle aziende, quanto in quello dei consumatori.

I fenomeni della globalizzazione e della diffusione del *retailing* online, infatti, hanno reso il processo di tracciamento dei prodotti particolarmente articolato e, talvolta, opaco o afflitto da *gap* informativi.

Allo stesso tempo, gli utenti del web sono continuamente inondati da un'enorme mole di informazioni, notizie e pubblicità, di cui, spesso, è difficile verificarne la genuinità e la provenienza; tale fattore, combinato con l'eccessiva dipendenza dagli algoritmi di raccomandazione dei motori di ricerca, può minare la fiducia nei confronti dei prodotti in cui è possibile imbattersi in rete.

Un contesto del genere, caratterizzato da scarsa trasparenza e da un crescente sentimento di diffidenza, offre ampio raggio d'azione a pratiche illlecite; su tutte citiamo la contraffazione. Tale minaccia, nonostante sia ben nota da diversi decenni, continua ad impattare pesantemente l'intera economia mondiale perchè, al pari del progresso tecnologico, diviene sempre più sofisticata e difficilmente rilevabile; con il passare degli anni, difatti, il fenomeno è profondamente mutato, a causa della diversificazione dei canali di vendita, dell'evoluzione delle pubblicità ingannevoli e della diffusione di repliche ad alta fedeltà.

La contraffazione, alla luce di ciò, costituisce una delle maggiori sfide da affrontare nell'attuale scenario commerciale in quanto, pur con modalità e portate diverse, non risparmia alcun comparto economico.

In riferimento al panorama nazionale, a maggior ragione, la forte tradizione manifatturiera di elevata qualità, che porta al riconoscimento del "Made in Italy" nel mondo, offre, giocoforza, grandi opportunità ai contraffattori per trarre illeciti vantaggi economici. Difatti, l'immissione di prodotti falsi, oltre ad influenzare negativamente le vendite delle aziende legittime, rischia di arrecare un danno d'immagine alle etichette di qualità distintive, poichè, spesso, non sono garantiti neanche i più elementari standard di sicurezza.

Oltre a ciò, in un momento storico in cui la sostenibilità è al centro dell'attenzione, è fondamentale sottolineare come il mercato del falso contribuisca al degrado ambientale; i contraffattori, per limitare il più possibile i costi, sono soliti utilizzare materiali e processi non conformi alle normative ambientali, generando rifiuti ed emissioni oltre le soglie tollerate. Non da meno, la vita media di un prodotto che elude controlli di qualità stringenti risulta più breve, il che comporta maggiori sprechi a discapito di un'economia circolare.

I dati descrivono un quadro attuale in cui l'acquito di prodotti contraffatti è considerabile, a tutti gli effetti, un'esperienza di massa, oltrettutto con una tendenza sempre più marcata a ricorrervi in maniera consapevole; ciò è particolarmente preoccupante perché, anche se i

consumatori, spesso, sono disposti ad ignorare tutti gli aspetti legati all'etica ambientale, la contraffazione può esporli, in prima persona, a rischi per la salute.

Le aziende, dunque, hanno tutto l'interesse nel certificare i propri prodotti attraverso soluzioni all'avanguardia, in modo da limitare la diffusione delle copie, rafforzare il proprio marchio ed attirare l'attenzione della clientela, guadagnandone la fiducia.

Al contempo, poter avere garanzie sulle caratteristiche dichiarate di un prodotto è una delle massime aspirazioni a cui spera di ambire un consumatore, specialmente al giorno d'oggi, in cui si sta registrando la tendenza a preferire la qualità sulla quantità.

In virtù di quanto detto, in questo elaborato è stato progettato e sviluppato un sistema in grado di certificare l'autenticità di prodotti di alta gamma; il settore del lusso, infatti, costituisce un universo variegato, in grado di orientare stili di vita, comportamenti sociali ed immaginario collettivo, oltre che uno dei comparti con il valore generato più elevato; per tali ragioni, deve essere tutelato.

L'obiettivo conclusivo del sistema è quello di tracciare in maniera trasparente tutti i passaggi di proprietà che coinvolgono un bene di lusso, verificando, ad ogni trasferimento, le caratteristiche di qualità e di origine a cui esso risponde. Per riuscire nell'intento, l'implementazione viene svolta sfruttando le potenzialità della tecnologia blockchain, la quale consente di adottare un approccio decentralizzato che evita la necessità di fiducia nei confronti di terze parti.

La presente tesi è composta da sei capitoli organizzati come di seguito specificato:

- Nel Capitolo 1 sarà fornita una panoramica sul mondo delle blockchain, chiarendo i fattori che ne hanno favorito la diffusione, i principi di funzionamento, i risvolti in termini di sicurezza e i possibili impieghi.

- Nel Capitolo 2 verrà analizzato, in dettaglio, il ruolo che le blockchain si stanno ritagliando nel contesto del tracciamento dei prodotti, evidenziandone i benefici rispetto a tecniche più tradizionali.

- Nel Capitolo 3 verrà introdotto il sistema per la certificazione dell'autenticità; in particolare, adottando un opportuno linguaggio di modellazione, sarà svolta una fase di ingegneria dei requisiti, individuando gli attori coinvolti, le loro relazioni e le competenze di ognuno di essi.

- Nel Capitolo 4 sarà discussa la fase di progettazione del sistema in esame, concentrando l'attenzione sugli aspetti architetturali; parte integrante risulterà l'analisi del rischio, attraverso cui verrà data una priorità alle misure di sicurezza da adottare; inoltre, mediante un framework metodologico, verrà chiarita la fattibilità dell'utilizzo delle blockchain.

- Nel Capitolo 5 verrà descritto il processo di implementazione dell'applicazione decentralizzata che realizza il sistema di certificazione proposto; nello specifico, saranno esaminate le tecnologie utilizzate, le strategie di codifica messe in pratica, le problematiche emerse e le risoluzioni adottate, sia per quel che riguarda il *back-end*, sia per la parte di *front-end*.

- Nel Capitolo 6 verranno tratte le conclusioni e saranno illustrati alcuni possibili sviluppi futuri.

CAPITOLO 1

Le blockchain

Il capitolo in questione provvederà a fornire un quadro esaustivo riguardo il mondo delle blockchain, dalle motivazioni che hanno favorito la nascita e la diffusione di tale tecnologia fino agli aspetti più ingegneristici che la caratterizzano, nell'intento di chiarire l'impatto decisamente rivoluzionario che essa sta apportando nel campo delle transazioni digitali e della gestione dei dati. A tal proposito, verranno esplorati l'approccio e i principi che regolano una blockchain, descrivendone, al contempo, la struttura e i meccanismi su cui essa si fonda.

Si proseguirà, poi, a delineare le diverse varianti di blockchain, ciascuna con i propri tratti distintivi, focalizzando, dunque, l'attenzione su un esempio pratico di blockchain, Ethereum, la quale ha introdotto rilevanti elementi di novità strettamente correlati all'ambito dello sviluppo software. Avendo chiare tali nozioni, infatti, verranno affrontati gli argomenti Smart Contracts e Decentralized Apps, che risulteranno essere centrali per l'intera trattazione.

Seguirà, successivamente, un'analisi dal punto di vista delle tematiche della sicurezza e della difesa, esponendo vantaggi e svantaggi delle blockchain anche in relazione a possibili tecnologie alternative.

Il capitolo, infine, sarà concluso con alcune considerazioni riguardanti le numerose possibilità che la tecnologia blockchain offre e con la descrizione dei trend applicativi su cui si sta sempre più indirizzando il mercato.

1.1 Panoramica generale

Le blockchain costituiscono una tecnologia decisamente dirompente teorizzata già nel 1998 e divenuta realtà nel 2009, anno in cui Satoshi Nakamoto ha introdotto il sistema *Bitcoin* e l'omonima criptovaluta. Le blockchain, dunque, sono nate originariamente come un insieme di strutture dati e protocolli a supporto delle criptovalute, ossia monete digitali basate su algoritmi crittografici.

I fattori scatenanti che hanno spinto all'introduzione delle blockchain sono da rintracciare nella natura dei sistemi tradizionali tramite i quali le transazioni avvengono. Le valute classiche, così come i consueti circuiti di pagamento elettronici, rispondono, infatti, ad un paradigma centralizzato in cui è sempre presente una figura predominante, ovvero un singolo ente (nella realtà, una gerarchia di enti legati da rapporti di delegazione) che certifica il valore della valuta piuttosto che un fornitore di servizio che si occupa di verificare la validità dei pagamenti effettuati dagli utenti.

Un sistema centralizzato è senz'altro funzionale e concettualmente semplice; tuttavia, potrebbero sorgere diverse problematiche legate ad un ipotetico abuso di autorità o alla compromissione dell'ente centrale. Da qui l'esigenza di introdurre sistemi decentralizzati per l'emissione di valuta e l'autorizzazione di transazioni, sfociando, così, nelle blockchain.

Le blockchain, al fine di adempiere agli ideali per cui sono nate, sono basate su reti *peer-to-peer* nelle quali non esiste più alcun ente governatore centrale, bensì i nodi sono posti tutti allo stesso livello. A tal proposito, è facilmente intuibile come il più importante vantaggio di questa tecnologia, ovvero l'assenza di centralità, renda necessario un grande sforzo in termini dei criteri di consenso che governano i protocolli sottostanti, come verrà illustrato più in dettagio nella sezione 1.2.

1.1.1 Distributed Ledger Technology

Svolgendo un passo di astrazione, le blockchain costituiscono un caso particolare di DLT[1], ossia tecnologie di registro distribuito.

Una blockchain, di fatto, svolge il ruolo di un registro pubblico, o libro contabile, in cui vengono trascritte tutte le transazioni in maniera sequenziale. Ogni spostamento di moneta, una volta annotato su tale registro, non può essere né cancellato né alterato; infatti, una blockchain è una struttura dati immutabile.

La vera innovazione, ad ogni modo, sta nel fatto che non esiste un'unica copia del registro sulla quale va a scrivere un solo ente. Il registro non ha un'unica sede, bensì risiede in milioni di copie mantenute e aggiornate dalla comunità di nodi che partecipano alla rete, ovvero i nodi validatori.

La forza di una blockchain, dunque, consiste proprio nel creare un protocollo che permetta alle singole copie di risultare sempre sincronizzate tra loro e consenta, a tutti gli effetti, di avere un unico registro nonostante esso sia fisicamente disperso in un gran numero di repliche.

É lecito chiedersi, alla luce di ciò, come sia possibile garantire che tutte queste copie siano tra loro consistenti e come ci si possa assicurare della veridicità di quanto trascritto. L'idea di fondo è quella di procedere per votazione al fine di raggiungere un consenso. La richiesta di notarizzare una certa transazione, infatti, viene inoltrata a tutti i nodi validatori dopopdiché ciascuno di essi controlla autonomamente la validità della richiesta e comunica l'esito della propria verifica a tutti gli altri. Al raggiungimento di un *quorum*, infine, la transazione viene ufficialmente ritenuta legittima (o illegittima) e tutti coloro che mantengono una copia del registro sono tenuti a trascrivere la medesima informazione.

1.1.2 Struttura delle blockchain: blocchi, transazioni e crittografia

Sotto il profilo tecnico, un primo punto di vista dal quale esaminare le blockchain prende in considerazione l'aspetto strutturale.

Un registro tipicamente è costituito da un insieme di pagine ordinate, ciascuna contenente una serie di righe associate alle singole transazioni le quali, a loro volta, presentano una propria struttura interna con i diversi campi di interesse.

Una blockchain, che concettualmente richiama un registro contabile, presenta una struttura analoga. Come suggerisce il nome, si tratta di una catena i cui anelli sono chiamati blocchi ed ospitano al loro interno un certo numero di registrazioni sequenziali.

Sostanzialmente, dunque, una blockchain non è altro che una lista concatenata di blocchi; ciascun nuovo blocco aggiunto presenta un puntatore al blocco precedente al fine di poter stabilire con certezza l'ordine all'interno della catena. Tale puntatore, nello specifico, è l'hash del blocco precedente, il che costituisce un primo esempio pratico di utilizzo di tecniche crittografiche all'interno delle blockchain.

[1] Distributed Ledger Technologies

Il blocco iniziale, non avendo predecessori, ha come puntatore al blocco precedente un hash composto da tutti zeri; essendo il primo della lista, infatti, costituisce il fondamento da cui la blockchain si estenderà nel corso del tempo ed è, per questo, chiamato blocco genesi.

Proseguendo il parallelismo con un registro contabile, ogni blocco della blockchain è assimilabile ad una pagina. Al suo interno è possibile distinguere due sezioni principali, un header e un corpo. Facendo riferimento alla struttura interna di Bitcoin (struttura in gran parte ereditata anche da tutte le blockchain successive, a meno di circostanziali variazioni), un blocco è caratterizzato dai seguenti elementi:

- numero di blocco;
- nonce, ossia un numero pseudo-casuale utilizzato una sola volta, interpretabile come "indicatore di freschezza";
- hash che identifica il blocco;
- riferimento al blocco precedente;
- timestamp della creazione del blocco;
- Merkle root, ossia la radice di una struttura dati ad albero che sintetizza le transazioni contenute nel blocco;
- corpo del blocco, costituito dalle diverse transazioni registrate.

L'hash digest che contraddistingue un blocco deriva dall'applicazione di un'opportuna funzione hash (SHA-256) che riceve in input l'insieme di tutte le informazioni relative al blocco stesso.

La diretta conseguenza dell'utilizzo di un hash come riferimento per un blocco è che un'eventuale modifica all'interno del blocco stesso verrebbe immediatamente riscontratata e creerebbe una serie di effetti collaterali in cascata che porterebbero ad invalidare l'intera catena, come verrà messo in luce nel paragrafo relativo agli aspetti di sicurezza di una blockchain.

Relativamente alla caratterizzazione dei blocchi, inoltre, è importante sottolineare che ciascun blocco ha capacità di memoria limitata (variabile in base alla tipologia di blockchain adottata); di conseguenza, risulteranno limitati anche i dati che è possibile annotare per descrivere una transazione.

A tal punto, dunque, è necessario soffermarsi sulle transazioni, anticipando innanzitutto che il concetto di transazione può assumere anche un significato più ampio non esclusivamente connesso alle criptovalute, così come verrà approfondito nella sezione 1.4.

Ad ogni modo, nel caso più tradizionale, una transazione è uno spostamento di denaro da A a B che, nel mondo delle blockchain, sono due *wallet*. Per interagire con una blockchain, infatti, un utente deve aprire un proprio conto generando innanzitutto una coppia di chiavi: a partire da una chiave privata, e sfruttando algoritmi crittografici ECDSA[2], viene creata una chiave pubblica che, a sua volta, viene utilizzata per calcolare l'indirizzo associato al conto tramite il quale l'utente può originare e ricevere transazioni.

Chiave pubblica e privata di un account, in aggiunta, hanno un ruolo decisamente importante dal punto di vista crittografico. La chiave privata, infatti, viene utilizzata come una sorta di password per firmare digitalmente i dati di una transazione. Invertendo lo schema di firma digitale, lato ricezione, sarà possibile verificare la validità della transazione effettuata da un account sfruttando la chiave pubblica di quest'ultimo. Il workflow è il

[2]Algoritmo a curve ellittiche utilizzato nell'ambito di un cifrario asimmetrico.

seguente: un account firma la transazione da emettere tramite la propria chiave privata (che solo esso conosce); dopodiché chiunque riceva i dati e la firma può applicare la chiave pubblica del mittente; se il risultato di tale applicazione è consistente allora la verifica va a buon fine e ciò significa che quei dati provengono effettivamente dal mittente dichiarato.

I dati relativi ad una transazione, sottoposti, dunque, ad un processo di firma digitale, sono i seguenti:

- *input*, ossia la sorgente della valuta che, a differenza di un tradizionale bonifico, non corrisponde semplicemente al mittente bensì contiene una prova della storia di tutte le transazioni che hanno fornito al wallet in questione quella determinata capacità di spesa;
- *amount*, cioè l'ammontare che l'account intende inviare;
- *output*, ossia l'indirizzo pubblico di destinazione.

L'esistenza di un registro contabile da un lato e delle transazioni dall'altro giustifica, a tutti gli effetti, la presenza di criptovaluta; avendo l'intera successione di transazioni annotate nella blockchain a partire dal tempo zero, infatti, è possibile conoscere l'esatta disponibilità di un conto e rilevare eventuali anomalie nell'importo che si intende trasferire tramite una transazione, evitando, dunque problemi, noti come *double spending*.

L'insieme delle transazioni, infine, è organizzato in una struttura dati chiamata *Merkle tree* la quale, basandosi su funzioni hash applicate iterativamente a coppie di elementi, rappresenta in modo univoco e sicuro tutte le transazioni all'interno del blocco in questione. Difatti, qualsiasi tentativo di modifica a una singola transazione va ad influire sull'hash del suo ramo e, di conseguenza, sulla radice dell'albero, il Merkle root. Ne consegue che se un'entità malintenzionata tentasse di alterare una transazione all'interno del blocco, il Merkle root risulterebbe completamente diverso, rendendo, quindi, possibile segnalare l'inconsistenza e invalidare il blocco.

L'utilizzo del Merkle tree all'interno dei blocchi di una blockchain offre dunque diversi vantaggi. Innanzitutto, consente di verificare in modo efficiente l'integrità delle transazioni senza la necessità di controllare ogni singola transazione nel blocco. Inoltre, semplifica la sincronizzazione dei validatori che mantengono una copia della blockchain, in quanto il Merkle root può essere utilizzato per verificare rapidamente se due nodi hanno la stessa versione di un blocco.

Un sunto della struttura tipica di una blockchain e dell'organizzazione interna dei singoli blocchi è riportato in Figura 1.1

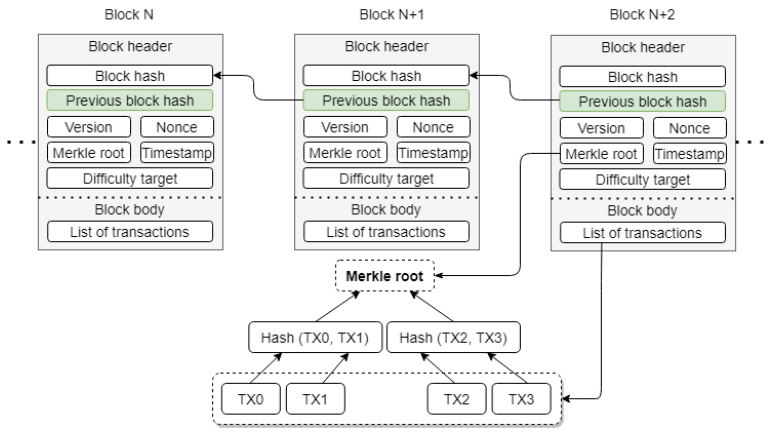

Figura 1.1: Blockchain vista dall'interno

1.2 Consenso distribuito

Il concetto di consenso costituisce uno dei principi fondanti delle blockchain ed è definito come il meccanismo attraverso il quale raggiungere un accordo circa uno stato o un singolo valore nella blockchain.

Le diverse operazioni menzionate nella sezione 1.1.2 relativamente alla verifica della validità delle transazioni, infatti, presuppongono l'esistenza di robusti protocolli tramite i quali la rete possa esprimerere il proprio parere e fornire o meno un'approvazione. Una delle maggiori difficoltà in tal senso è che la rete non ha un coordinatore bensì deve autogestirsi al fine di riuscire ad esprimere un verdetto. Tutto ciò si basa sull'uso intensivo di tecniche e protocolli crittografici; infatti, l'idea cardine è che sulla crittografia si possa reggere un governo decentralizzato.

Come chiarito, ogni transazione va verificata prima di essere aggiunta al registro e, idealmente, tale verifica può essere effettuata da ciascun utente della rete. Tutte le informazioni relative a ciascuna transazione, infatti, sono di dominio pubblico, così come tutta la storia di una valuta, dal momento in cui è stata coniata fino al tempo attuale.

Ogni transazione, come descritto in precedenza, ha un input e un output dove l'input, sostanzialmente, non è altro che l'output di una transazione precedente, ottenendo, perciò, una precisa sequenzialità. Esplorando i legami tra ogni transazione e la precedente, dunque, è possibile ripercorrere all'indietro tutta la storia della valuta fino alla cosiddetta "transazione zero", l'unica che non ha input in quanto generata dal processo di *mining* della moneta.

Nel momento in cui un utente intende effettuare una transazione, deve, innanzitutto, accedere al proprio wallet e specificare il portafoglio elettronico di destinazione; dopodiché, dietro le quinte, viene stabilita una connessione ad un nodo validatore della rete. Il nodo in questione comunica la richiesta agli altri validatori così da prendere una decisione comune a seguito di controlli svolti autonomamente.

Quando un nodo si trova a validare una transazione, dunque, legge l'input e ripercorre a ritroso tutta la catena di transazioni in modo da verificare che effettivamente ci sia la disponibilità di quella valuta. Se qualcuno provasse ad alterare una transazione all'interno di questa catena, come facilmente intuibile, tale operazione invaliderebbe tutte le transazioni successive in quanto l'input risulterebbe inconsistente.

Le transazioni, ad ogni modo, non vengono verificate una per una ma in gruppi: si attende, infatti, di raccogliere un certo numero di transazioni "pendenti" e solo quando viene raggiunta la capacità massima di un blocco quest'ultimo viene "forgiato". A tal punto, previo consenso sul blocco appena creato, ogni validatore aggiunge il blocco in questione alla propria copia della blockchain.

Un blocco, inoltre, non viene ritenuto confermato subito dopo la validazione bensì viene accettato solo quando assume una certa "anzianità" che, solitamente, consiste nella verifica di ulteriori X blocchi (6, nel caso di Bitcoin).

Ipoteticamente, come anticipato, la validazione può essere svolta da chiunque prenda parte alla rete. Sta di fatto, però, che a livello pratico la questione non è così semplice e possono subentrare delle barriere di ingaggio.

Svolgere il compito di un nodo validatore, infatti, presuppone di ospitare sulla propria macchina una copia della blockchain che nel corso del tempo è destinata a crescere. Oltre alla capacità di storage, per giunta, c'è da considerare il dispendio computazionale per effettuare controlli ogniqualvolta arriva una nuova richiesta per registrare una transazione.

Alla luce di ciò, sono previsti dei meccanismi di ricompensa per il lavoro svolto dai nodi validatori, redistribuendo tra essi una commissione pagata dal mittente della transazione. Oltre a tale quota fissa, per un nodo validatore c'è la possibilità di guadagnare un "premio bonus" come ulteriore stimolo a svolgere i propri compiti (le modalità per ottenere questa

ricompensa aggiuntiva dipendono dalla specifica implementazione del protocollo di consenso e verranno illustrate nel prosieguo). Per questo motivo, spesso, ci si riferisce ai nodi validatori anche con il nome di "minatori".

Gli algoritmi di consenso utilizzati variano a seconda della tipologia di blockchain (con permessi o senza) ma anche a seconda del vendor specifico (Bitcoin, Ethereum, Ripple, etc.). Di seguito una descrizione dei due algoritmi più noti.

1.2.1 Proof of Work

Proof of Work (PoW) è l'algoritmo proposto da Bitcoin ed è basato sulla difficoltà computazionale nello svolgimento di un determinato compito crittografico da parte dei nodi validatori.

Ciascun nodo che decide di contribuire alla verifica delle transazioni, innanzitutto, raccoglie in modo casuale alcune transazioni pendenti. Per ciascuna transazione, dunque, ripercorre tutta la storia di quella valuta così da attestarne o meno la legittimità. Ammesso che tutte le transazioni raccolte abbiano esito positivo (eventuali transazioni anomale vengono scartate), il nodo avrà pronto un blocco di transazioni legittime che vorrebbe attaccare alla blockchain.

Forgiare un blocco ed inserirlo nella blockchain, però, richiede la risoluzione di una sfida (puzzle crittografico) che costituisce una prova di lavoro ed ha un notevole costo computazionale.

Il problema da risolvere, infatti, consiste nel trovare un nonce tale per cui l'hash digest generato fornendo in input il nonce stesso, il numero del blocco, i dati contenuti nel blocco e il riferimento al blocco precedente abbia determinate caratteristiche. Nello specifico, si richiede un hash che abbia un certo numero di bit iniziali posti a zero, dove il numero di zeri iniziali è il parametro che modula la difficoltà della prova di lavoro (periodicamente questo parametro viene variato al fine di mantenere costante la quantità di blocchi verificati nell'unità di tempo).

Il numero di zeri iniziali che l'hash deve presentare è un parametro molto importante perché, da un lato, determina la velocità nella verifica dei blocchi, dall'altro è indice della robustezza della rete in quanto comporta che solo la più grande potenza di calcolo della rete sia in grado di mandare avanti i blocchi verificati.

Le risorse computazionali a disposizione dei nodi validatori, dunque, sono indispensabili nell'ottica dell'algoritmo Proof of Work in quanto la risoluzione della sfida crittografica avviene tramite un approccio a forza bruta: variando il nonce, chi esegue la prova di lavoro può calcolare di volta in volta un nuovo hash digest e procedere ad un nuovo tentativo. Da qui la necessità per i nodi validatori di disporre di risorse computazionali elevate.

Ad ogni modo, una volta trovato un hash valido, basta pubblicare il valore del nonce che ha permesso di ottenere quel risultato così da dimostrare di aver risolto la prova di lavoro. A tal punto, il primo nodo validatore che risolve il puzzle crittografico ha diritto a "distribuire" il proprio blocco e va a benificiare di un premio extra (solitamente della moneta fresca di conio) che è decisamente più sostanzioso rispetto la retribuzione fissa per la verifica delle transazioni.

1.2.2 Proof of Stake

Oltre all'approccio Proof of Work, che è quello più classico, nel tempo sono stati concepiti nuovi algoritmi con lo scopo di ridurre l'onere computazionale e il conseguente consumo in termini energetici. Tra questi, uno dei più noti è Proof of Stake (PoS).

Anziché fondarsi sulla potenza di calcolo dei nodi validatori, infatti, il PoS si basa sul possesso di una certa quantità di criptovaluta (chiamata "pegno") come base per determinare chi avrà il diritto di validare le transazioni e creare nuovi blocchi all'interno della blockchain.

Il funzionamento di PoS richiama quello di un'asta: ciascun minatore mette sul piatto una certa quantità di moneta che è disposto a sborsare affinché venga selezionato per creare un nuovo blocco e ricevere dunque un premio extra. La probabilità di essere selezionato dipende dalla quantità di criptovaluta che il partecipante ha bloccato come pegno. In generale, maggiore è la quantità di criptovaluta posseduta, maggiore è la probabilità di essere scelto come validatore.

Il validatore selezionato come vincitore dell'asta raccoglie, quindi, le transazioni all'interno di un blocco e ne verifica la validità e la conformità, secondo i medesimi criteri descritti nelle sezioni precedenti.

Una volta che il nodo in questione termina la validazione delle transazioni, il blocco da esso creato viene trasmesso alla rete per essere esaminato e convalidato anche dagli altri partecipanti. Se la maggioranza dei partecipanti raggiunge un accordo sul blocco proposto, questo viene accettato e aggiunto alla blockchain; inoltre, il creatore del blocco riceve una ricompensa sotto forma di criptovaluta fresca di conio. Il premio ha la finalità di incentivare i nodi ad agire in modo onesto, infatti, in caso di comportamento malevolo, si è soggetti a penalità quali la confisca della ricompensa oppure l'esclusione dall'attività di validazione.

In base al funzionamento descritto, ad ogni modo, si potrebbe essere indotti a pensare che solamente chi detiene grandi somme di criptovaluta possa continuare ad ottenere la posta, escludendo di fatto gli altri grazie alle maggiori capacità di stake. In realtà, però, l'algoritmo prevede dei meccanismi di rotazione per evitare eventuali polarizzazioni nella scelta del vincitore dell'asta: chi prende la posta, infatti, viene escluso dai round successivi per un certo lasso di tempo durante il quale deve, comunque, continuare a svolgere il proprio lavoro per poter tornare a partecipare.

1.2.3 Confronto tra Proof of Work e Proof of Stake

Gli algoritmi di consenso PoW e PoS presentano sia aspetti in comune che sostanziali differenze.

Innanzitutto, entrambi i protocolli prevedono una ricompensa che costituisce una parte integrante del meccanismo.

Anche la soglia da raggiungere per ottenere il quorum è la medesima: in ambedue i casi, infatti, il blocco in questione è ritenuto valido quando ottiene un consenso del 50%+1. Ad ogni modo, comunque, è importante sottolineare che nel PoS tale percentuale non si riferisce al numero totale dei peer partecipanti alla comunità dei validatori, come invece accade in PoW, bensì fa riferimento alla *computational power stake*, dunque dipende dalla somma impegnata nel "piatto"; ciò si traduce a tutti gli effetti in un sistema di voto ponderato in cui la quantità di criptovaluta posseduta da un partecipante, e dunque il rischio assunto bloccando una certa cifra, determina la sua influenza nel processo decisionale. Potendo contare su tali soglie di consenso ed un un elevato numero di minatori, PoW e PoS sono caratterizzate da una bassa probabilità di *deception*, ossia è più difficile che nodi malintenzionati riescano a far passare per buona una transazione illegittima.

Quanto all'impiego di risorse dei due algoritmi, emergono differenze più marcate. Dal punto di vista computazionale, versare sul piatto una certa somma di moneta, così come previsto da PoS, è relativamente semplice e i costi trascurabili. Questione ben più complessa, invece, è risolvere una sfida crittografica, come nel caso di PoW, dove sono richiesti diversi tentativi, dunque tempo ed elevate prestazioni.

Svolgere il ruolo di minatore per una blockchain basata su PoW, dunque, presuppone la disponibilità di hardware sofisticato tant'è che la scena è per lo più dominata dalle Big

Tech o, in generale, da organismi che possono contare su cluster di server dalle elevate capacità computazionali. A ciò si collega direttamente la questione tanto discussa dell'impatto energetico dovuto ad un elevato consumo di energia elettrica per la risoluzione della prova di lavoro (secondo i dati del Bitcoin Electricity Consumption Index dell'Università di Cambridge, il consumo si attesta a circa 134 TWh), fattore che sta portando alcune blockchain a migrare verso algoritmi di consenso più sostenibili in ottica ambientale.

La complessità dei protocolli di consenso, in ultima istanza, ha un impatto anche sull'efficienza in termini del tempo medio utile a trascrivere una transazione in un blocco ed aggiungere il blocco in maniera permanente alla blockchain. In linea generale, sia in PoW che in PoS bisogna raccogliere un certo numero di transazioni, forgiare il blocco, scambiare dei messaggi per raggiungere il quorum ed, infine, stabilire a chi spetta la ricompensa. Ne consegue che l'attesa dell'utente per vedere la propria transazione, nei due algoritmi in esame, è abbastanza elevata risultando inaccettabile in alcuni contesti.

La Tabella 1.1 confronta Proof of Work e Proof Stake con ulteriori algoritmi di consenso nati nel corso degli anni.

	PoW	**PoS**	**PBFT**	**Ripple**
Impatto Energetico	Alto	Moderato	Basso	Basso
Quorum	$> 50\%$	$> 50\%$	$> 33.3\%$	$> 80\%$
Efficienza (tempo)	Bassa ($\sim$ 10 min.)	Bassa ($\sim$ 10 min.)	Alta (pochi ms)	Bassa ($\sim$ 10 min.)
Pr. Deception	Bassa	Bassa	Alta	Bassa
Esempio	Bitcoin	Ethereum 2.0	Fabric	Ripple

Tabella 1.1: Prospetto riassuntivo su analogie e differenze di alcuni algoritmi di consenso. Le proprietà prese in considerazione per il confronto sono: i costi in termini di energia elettrica consumata, la soglia necessaria per raggiungere una decisione, l'efficienza temporale per la verifica di una transazione e la probabilità di deception. Sono, infine, indicati esempi di blockchain che adottano l'algoritmo in questione

1.3 Tipologie di blockchain

Una volta aver messo in luce le caratteristiche strutturali e la logica sottostante le blockchain, è possibile fornire una classificazione delle tipologie di blockchain più comuni in modo da avere un quadro quanto più chiaro possibile delle soluzioni ad oggi disponibili per tale tecnologia ed evidenziare le diverse sfaccettature che esse comportano.

Ad ogni modo, è bene sottolineare che non si tratta di una classificazione rigida in quanto gli elementi che caratterizzano le diverse tipologie di blockchain possono essere combinati in un'ampia varietà di modalità ibride, al fine di creare registri personalizzati per applicazioni specifiche.

Tradizionalmente, ad ogni modo, le blockchain sono distinte in due grandi categorie, in relazione a quelle che sono le modalità di partecipazione alla rete, in particolare per ciò che concerne il processo di validazione del consenso. In quest'ottica, si pongono come alternative le blockchain permissionless e le blockchain permissioned.

1.3.1 Blockchain permissionless VS Blockchain permissioned

Le blockchain permissionless (senza permessi) non prevedono alcun requisito di autorizzazione: chiunque può decidere autonomamente di unirsi alla rete per svolgere il ruolo di "minatore" e convalidare blocchi in quanto l'accesso è aperto e privo di restrizioni. Sostanzialmente, dunque, tale tipologia rispecchia a pieno l'idea originaria di blockchain realmente democratizzata in cui non sono presenti entità centrali con autorità sul sistema.

Dato che l'accesso alle attività di rete non ha limitazioni e, di conseguenza, si ritroverranno ad interagire tra loro parti sconosciute tre le quali non si può instaurare reciproca fiducia, blockchain di questo tipo richiedono provvedimenti più stringenti in termini di sicurezza che si concretizzano, ad esempio, in soglie di consenso più elevate (algoritmi Proof of Work o Proof of Stake) e nella leva dell'incentivo economico per spingere ad un comportamento onesto da parte dei nodi. Tutto ciò, di contro, viene pagato in maggiore lentezza e difficoltà di scalabilità.

D'altro canto, nelle blockchain permissioned (con permessi) la partecipazione alle attività di convalida delle transazioni è limitato a nodi autorizzati o selezionati sulla base di determinati requisiti o criteri. La concessione dei permessi, in tal caso, può avvenire ad opera di un singolo ente oppure di un consorzio di enti.

Trattandosi di reti parzialmente chiuse ed in qualche misura controllate, dunque, le blockchain con permessi si presuppongono costituite da nodi fidati, per tale motivo solitamente sono regolate da protocolli di consenso in cui il quorum da raggiungere è più basso rispetto al caso precedente, a vantaggio di una maggiore velocità.

1.3.2 Ulteriori soluzioni

Le recenti declinazioni del paradigma blockchain, in combinazione con l'avvento degli *smart contract* di cui si parlerà a breve, hanno portato ad un ulteriore livello di raffinamento nella tassonomia delle reti.

Difatti, è possibile trovare sia infrastrutture pubbliche permissioned (ad esempio Ripple) gestite da una comunità con interessi comuni nelle quali l'accesso al ruolo di minatore è limitato ad utenti autenticati e l'uso della blockchain è regolamentato dal consorzio, ma anche blockchain private permissioned, tipiche di istituzioni e agenzie governative, in cui l'accesso a tutti i ruoli è riservato ad utenti autenticati.

Inoltre, in alcune particolari architetture, chiunque può leggere ed emettere transazioni sulla blockchain mentre altre integrano al loro interno moduli di controllo degli accessi basato su ruoli, in cui i nodi vengono gerarchicamente posizionati su livelli distinti ed è possibile effettuare transazioni riservate solo ad alcuni utenti che dispongono di autorizzazioni di lettura.

Un'altra soluzione ibrida, particolarmente utilizzata nell'ambito degli smart contract, prevede reti cosiddette private permissionless. Questo tipo di blockchain è gestito e mantenuto da un'organizzazione o un gruppo ristretto di partecipanti ma l'accesso e la partecipazione alla rete non richiedono autorizzazioni specifiche o permessi; ciò che è soggetto ad autorizzazione, invece, è la facoltà di leggere uno smart contract e i dati ad esso relativi.

1.3.3 Esempio pratico di blockchain: Ethereum

Soffermandosi su un'istanza specifica di piattaforma blockchain, *Ethereum* costituisce sicuramente un esempio interessante da trattare in quanto, grazie agli elementi di novità introdotti, ha dato un forte impulso nell'ambito dello sviluppo software di applicazioni basate su una logica decentralizzata.

In realtà, già con Bitcoin si iniziarono ben presto a diffondere degli usi alternativi della tecnologia blockchain che non riguardavano unicamente il trasferimento di moneta digitale, tuttavia, è stato proprio con l'avvento di Ethereum che il concetto di blockchain ha subito una sostanziale evoluzione ed è stato generalizzato in un nuovo modello che non definisce necessariamente una sola criptovaluta nativa bensì permette di creare una varietà di asset digitali.

Ethereum è una blockchain pubblica permissionless nata nel 2014 con la finalità di dar vita ad una piattaforma digitale destinata a scopo generale in cui chiunque può implementare o fruire di nuovi servizi senza far capo ad un ente privilegiato.

Nello specifico, l'intento degli sviluppatori di Ethereum era quello di fare in modo che la blockchain non si riducesse semplicemente ad un repository di dati memorizzati in maniera immutabile bensì che i nodi minatori potessero anche essere i garanti dell'esecuzione di programmi scritti permanentemente nella blockchain. Tali programmi prendono il nome di *smart contracts* e verranno approfonditi nella sezione 1.4. Ethereum, dunque, può essere definita come una piattaforma di distributed computing pubblica, open source e basata su blockchain.

Per ciò che riguarda il protocollo di consenso, la versione attuale di Ethereum sta sostituendo Proof of Work (utilizzato in Ethereum 1.0) a favore di un meccanismo noto con il nome di *Gasper* il quale consiste in una combinazione di algoritmi di stampo proof-of-stake, volti a ridurre l'impatto energetico, a facilitare la sincronizzazione dei nuovi minatori con la catena valida e a rendere più sicure le operazioni di forking (divisione).

La criptovaluta nativa utilizzata in Ethereum è chiamata Ether. Tale moneta digitale è utilizzata per ricompensare i minatori i quali, oltre a forgiare i blocchi, devono fornire potenza di calcolo per l'esecuzione di codice.

A tal proposito, in Ethereum è stato introdotto il concetto di gas come unità di misura dell'utilizzo delle risorse di calcolo della rete, al fine di garantire l'efficienza, la sicurezza e l'economia del sistema. Maggiore è il lavoro che un minatore deve svolgere, maggiore sarà il gas consumato e, di conseguenza, la commissione da pagare dall'utente. Il meccanismo del gas, infatti, è alla base del calcolo dei costi e delle ricompense dei minatori, processo che è un po' più complicato rispetto, ad esempio, a Bitcoin e chiama in causa due parametri principali:

- *gas price*, ossia il prezzo (in Ether) che l'utente è disposto a pagare per unità di gas;
- *gas cost*, il quale indica il costo totale (in Ether) del gas necessario ad eseguire una transazione o uno smart contract.

Se il prezzo del gas specificato è troppo basso allora nessun minatore sarà interessato a prendere in carico la transazione o l'esecuzione del codice; dunque, tale parametro serve anche come strumento di prioritizzazione. Oltre a ciò, se il costo totale del gas dovesse superare l'ammontare che l'utente è disposto a sostenere allora si va "out of gas", condizione nella quale il minatore viene comunque ricompensato (in quanto ha effettivamente occupato irreversibilmente delle risorse computazionali) ma l'elaborazione viene abortita. Tramite soluzioni di questo tipo, dunque, da una parte si evita il rischio di esecuzioni infinite dovute a possibili errori di programmazione, dall'altra si spinge chi scrive gli smart contract ad utilizzare meno risorse possibili e privilegiare soluzioni cosiddette "off-chain". Il meccanismo del gas, pertanto, ha il fine ultimo di evitare che le risorse vengano utilizzate in maniera impropria (volontariamente o meno) e che la rete sia sottoposta ad un carico di lavoro esagerato, considerando che pezzi di codice verranno eseguiti in parallelo su tutti i nodi che devono validare la transazione associata.

Le risorse computazionali che comportano consumo di gas, e dunque vengono fatte pagare sotto forma di *fee*, sono il tempo e lo storage, entrambe proporzionali alla quantità di dati, rispettivamente, da processare e memorizzare. Se da un lato, però, il tempo di

elaborazione non può essere "rimborsato" una volta che un nodo l'ha impiegato, altro discorso vale per lo spazio; in tal caso, infatti, se viene richiesta l'occupazione di una certa quantità di memoria e in un secondo momento lo spazio viene liberato, allora quella stessa quantità di memoria tornerà a disposizione del minatore e il corrispettivo pagato inizialmente dall'utente verrà restituito.

Per ciò che riguarda la struttura della blockchain, infine, Ethereum è più complessa rispetto, ad esempio, a Bitcoin; Ethereum, difatti, utilizza delle particolari strutture dati chiave-valore chiamate *storage trie* per memorizzare lo stato attuale di ciascun account. Ciò è dovuto al fatto che, oltre alle transazioni, sono presenti particolari programmi (smart contract) con le rispettive variabili. Ethereum, dunque, distingue tra due tipologie di account:

- *Externally Owned Accounts*, cioè account normali controllati dal possessore della chiave privata.
- *Contract Accounts*, ossia particolari account controllati dal codice e creati dalla rete all'atto della registrazione di uno smart contract.

Nella Figura 1.2 viene riportato il logo relativo ad Ethereum.

Figura 1.2: Logo della piattaforma Ethereum

1.4 Smart Contract e Decentralized App

Il concetto di *Smart Contract*, o contratto intelligente, costituisce un vero e proprio punto di svolta nell'utilizzo e nella diffusione della tecnologia blockchain.

Partendo dagli albori, uno smart contract è nato come programma informatico che consente di facilitare, verificare o imporre la negoziazione e l'esecuzione di un contratto senza la necessità di terze parti quali avvocati o giudici. Difatti, due sono i principali problemi riscontrabili nei contratti tradizionali: innanzitutto, la mancanza di uno strumento effettivo per obbligare i contraenti a rispettare le diverse clausole; in secondo luogo, le possibili ambiguità che scaturiscono dall'utilizzo del linguaggio naturale. Da qui gli smart contract, cioè contratti espressi per mezzo di un linguaggio formale che consiste in un linguaggio di programamzione vero e proprio, il che li rende direttamente eseguibili.

Alla luce di ciò, è apparsa immediatamente chiara la necessità che tali programmi fossero scritti, memorizzati ed eseguiti su una piattaforma tracciabile, irreversibile ed incontrovertibile, motivo per cui è facile intuire come l'uso delle blockchain si presti particolarmente alla causa.

Allo stato attuale dell'arte, in realtà, per smart contract si intende più ampiamente l'esecuzione di programmi general purpose, perdendo, dunque, il concetto di contratto a favore di una più generica accezione di programma o algoritmo immutabile e non ripudiabile. In ogni caso, comunque, gli smart contract possono essere considerati come veri e propri "pezzi di software" che, mandati in esecuzione in un ambiente decentralizzato, costituiscono un'alternativa alle infrastrutture software tradizionali, realizzando, a tutti gli effetti, un nuovo pattern architetturale. Integrando blockchain e smart contract, difatti, è possibile implementare il

back-end di quelle che sono ormai note come *Decentralized Applications* (DAPPs), chiamate così perché non vengono eseguite da un server bensì dai nodi della rete.

Le blockchain, congiuntamente ai concetti di smart contract e DAPP, sono i pilastri su cui si fonda la visione del cosiddetto Web 3.0.

1.4.1 Automazione ed esecuzione programmabile

Gli smart contract sono supportati da Ethereum e da altre blockchain preposte alla loro integrazione per mezzo di appositi linguaggi di scripting, i più noti dei quali sono Solidity e Vyper. Trattandosi di porzioni di programmi, dunque, l'esecuzione degli smart contract non è ambigua ed è automatica.

Il workflow tipico che coinvolge uno smart contract prevede a monte un utente che richiede una transazione il cui contenuto consiste nel registrare permanentemente lo smart contract stesso, dunque le condizioni e le azioni codificate al suo interno, su uno dei blocchi della blockchain: è questa la fase del *deploy*. Da tale momento in poi, il pezzo di programma che costituisce lo smart contract è in esecuzione e ad esso viene associato un wallet in modo tale che chiunque voglia fornire dei dati in input allo smart contract possa farlo eseguendo una transazione a suo favore. Le transazioni, nello specifico, svolgono il ruolo di eventi che "triggerano" l'esecuzione automatica del contratto.

Ogniqualvolta qualcuno invia dei dati in input allo smart contract affinché vengano memorizzati o elaborati, tutti i minatori forniscono tale input alla propria copia dello smart contract la quale produrrà un certo risultato. I minatori, a tal punto, esprimono il consenso sul risultato e, successivamente, trascrivono l'esito del voto sulla blockchain.

Le funzioni principali che è possibile automatizzare e regolare tramite uno smart contract, sostanzialmente, sono le seguenti:

- memorizzare dei dati che possono tornare utili nel futuro in maniera permanente sulla blockchain, in modo che vengano "notarizzati" e non possano essere alterati;
- gestire un contratto o un accordo tra utenti tra i quali non ci può essere piena fiducia, proponendosi come automazione del rispetto delle clausole contrattuali;
- fornire servizi ad altri smart contract: ricalcando il principio della modularità delle componenti software di un programma, due smart contract possono interagire reciprocamente attraverso l'invocazione di metodi o funzionalità dell'altro, similmente a quanto accade con una qualsiasi libreria;
- gestire meccanismi di autenticazione più o meno complessi.

Essendo immutabili e registrati su una blockchain, dunque, gli smart contract offrono un alto livello di sicurezza e trasparenza; tuttavia, non sono privi di sfide e rischi in quanto si potrebbe incorrere in bug di programmazione o vulnerabilità appetibili per utenti malevoli. Pertanto, dato che una volta registrati non possono essere alterati, è essenziale progettare, sviluppare e validare gli smart contract accuratamente in apposite blockchain di test (*testnet*) per garantire la loro affidabilità ed integrità.

1.4.2 Token fungibili ed NFT

Una funzionalità interessante, resa possibile grazie agli smart contract, consiste nel creare nuove criptomonete, o più in generale nuovi asset digitali, al di fuori della valuta nativa utilizzata dalla specifica blockchain.

Tali criptomonete create ad hoc prendono il nome di token fungibili e possono essere generate nella quantità e nella qualità desiderate, motivo per cui si prestano per diventare merce di scambio di beni e servizi.

Token di questo tipo, dunque, consistono in unità digitali intercambiabili ed indistinguibili tra loro; ciascun token fungibile, difatti, rappresenta un'unità di valore e può essere scambiato con un altro token dello stesso tipo in modo equivalente, senza alcuna differenza di valore o di caratteristiche. Oltre a ciò, i token fungibili sono contraddistinti dalla proprietà di divisibilità, nel senso che possono essere divisi in parti più piccole, consentendo, di fatto, transazioni di frazioni di un token.

Anziché scrivere smart contract da zero per creare questi nuovi token, però, nel tempo si sono consolidati alcuni modelli che, pur seguendo un'impostazione standard, possono essere adattati alle diverse esigenze.

Il più celebre di tali smart contract predefiniti è identificato con la sigla ERC20 e al suo interno sono codificate le regole per creare, scambiare e conteggiare un token standard Ethereum. ERC, infatti, sta per *Ethereum Request for Comment*, ossia un protocollo ufficiale utilizzato dalla comunità Ethereum per proporre miglioramenti o nuove funzionalità alla rete Ethereum. Il numero 20, invece, è l'identificatore della proposta.

Rispettare l'interfaccia dello smart contract ERC20, dunque, dà agli sviluppatori la possibilità di programmare token tra loro compatibili in quanto aderenti ad una sintassi unica e, di conseguenza, creare beni basati su blockchain con funzionalità simili a Bitcoin, Ether ed altre criptovalute. Un token ERC20, pertanto, può assumere un controvalore in relazione al fatto che venga fissata una quantità limitata all'atto della creazione.

Successivamente, sono nate altre famiglie di token che hanno apportato rilevanti elementi di novità. L'esempio più lampante è lo standard ERC721, il quale consente la creazione dei cosiddetti token non fungibili, o NFT.

La differenza chiave con gli altri token è che ERC721 consente la creazione di asset non interscambiabili. Gli NFT, difatti, sono associati ad esemplari unici: si tratta di beni digitali con caratteristiche e proprietà specifiche, definite all'atto della creazione, che li rendono diversi da tutti gli altri; inoltre, a differenza dei token fungibili, risultano indivisibili.

Tramite gli NFT, dunque, è possibile assegnare ad un utente un oggetto crittografico non riproducibile di cui, per giunta, non è garante un fornitore di servizi o una multinazionale bensì il garante è la blockchain stessa. L'idea di avere una garanzia crittografica di unicità di un asset, come facilmente intuibile, ha aperto nuovi scenari in aree applicative quali il diritto d'autore, l'emissione di titoli o la tracciabilità, così come verrà approfondito nella sezione 1.6.

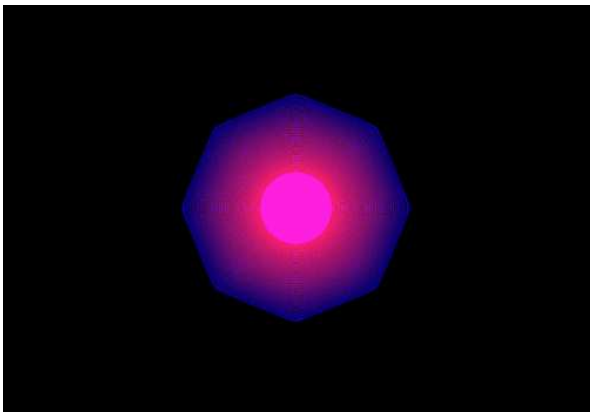

Figura 1.3: Immagine di Quantum, il primo NFT creato sulla piattaforma Ethereum a partire da una forma d'arte generativa e venduto nel 2021 per la cifra di 1,47 milioni di dollari

1.4.3 Problema dell'oracolo e Hybrid Smart Contract

Se gli smart contract devono idealmente sostituire i tradizionali contratti utilizzati nella quotidianità allora vuol dire che, in linea di principio, essi avranno bisogno di dati dal mondo reale o di potenza di calcolo esterna. Le blockchain, però, non possono interagire né leggere dati dall'esterno ed è proprio da tale contraddizione che nasce il cosiddetto "problema dell'oracolo".

Le blockchain costituiscono sistemi deterministici, e tutto ciò che è connesso ad esse avviene in un ecosistema "chiuso". Gli smart contract, infatti, sono eseguiti in modo isolato e non possono accedere direttamente a informazioni del mondo reale (magari anche tempo-varianti), come il prezzo di mercato di un'attività finanziaria o dati meteo. Ciò è dovuto al fatto che le blockchain cercano di garantire la sicurezza e l'immutabilità; di conseguenza permettere agli smart contract di accedere direttamente a fonti esterne potrebbe aprire la porta a minacce di sicurezza e vulnerabilità.

Una possibile soluzione a tale sfida consiste nell'utilizzo degli "oracoli" il cui intento è proprio quello di fornire un "ponte" tra il mondo reale e la blockchain. Un oracolo, dunque, è un'entità o un servizio che trasporta dati esterni verificabili agli smart contract sulla blockchain. Gli oracoli, infatti, possono recuperare, elaborare e convalidare i dati esterni per poi inviarli alla blockchain e consentire agli smart contract di utilizzare tali informazioni nelle loro operazioni.

Affinché un'applicazione costruita su blockchain e smart contract sia realmente decentralizzata, però, sarebbe controintuitivo lavorare con un singolo oracolo, piuttosto che un singolo provider dei dati o una singola sorgente che effettua calcoli esternamente, in quanto tornerebbero alla ribalta problemi di fiducia e di *single point of failure*. È necessaria, invece, una rete di oracoli decentralizzati e modulari.

Alla luce di ciò, un'applicazione può contare sia su una logica totalmente *on chain* decentralizzata ma anche su una parte *off chain* che risulti analogamente decentralizzata: combinando questi due modelli si parla di *hybrid smart contract*, ossia smart contract che possono beneficiare delle caratteristiche sicure delle blockchain utilizzando, al contempo, anche dati esterni accertati.

Al giorno d'oggi, infatti, la maggior parte dei protocolli con cui si interfacciano applicazioni e servizi decentralizzati del Web3 chiamano in causa smart contract ibridi. È questo il caso del protocollo Chainlink, una rete di oracoli affidabile e a prova di manomissione utile per connettere fonti dati o API agli smart contract.

1.4.4 Aspetti legali

Le diverse possibilità che offrono gli smart contract, come già accennato, si scontrano con inevitabili rischi.

Uno smart contract che dovesse presentare un bug di programmazione, infatti, non può essere sostituito una volta registrato sulla blockchain, a differenza di un'infrastruttura tradizionale in cui sarebbe possibile sovrascriverlo. Alla luce di ciò, e considerando le eventuali perdite economiche a cui si sarebbe esposti, un forte dibattito riguarda la questione della responsabilità nel caso in cui dovesse insorgere un malfunzionamento nelle funzionalità dello smart contract. Di conseguenza, se da un lato demandare il potere agli algoritmi è interessante per avere massima trasparenza e inoppugnabilità, dall'altro bisogna tener conto che gli algoritmi sono scritti dalle persone e, dunque, non sono esenti da errori.

Altra discussione aperta nasce dalla seguente considerazione: se è pur vero che uno smart contract può sostituire teoricamente un contratto, è altrettanto vero che, dal punto di vista legale, non è detto che questo venga riconosciuto universalmente. Mentre un contratto tradizionale è scritto in linguaggio naturale, rispecchiando la volontà della parti, uno smart

contract deve essere tradotto in forma algoritmica, il che costituisce una prima problematica. In secondo luogo, i contratti tradizionali possono essere annullati o sovvertiti in tribunale mentre uno smart contract, una volta trascritto nella blockchain, non è più modificabile.

Allo stato attuale dei fatti, dunque, il dibattito più spinoso vede fronteggiare la tesi secondo cui uno smart contract dovrebbe contenere gli elementi di contratto veri e propri, tra cui le diverse clausole, e quella che considera uno smart contract come mero strumento di esecuzione.

1.5 Le blockchain come tecnica di difesa

Indubbiamente, le blockchain costituiscono, a tutti gli effetti, uno dei possibili strumenti per introdurre aspetti di sicurezza all'interno di un software.

Tale tecnologia, infatti, sfrutta al suo interno diversi approcci volti a garantire la *dependability* di un software, ossia il grado di fiducia che è possibile riporre nei confronti di quest'ultimo. Le tecniche di sicurezza insite nelle blockchain, dunque, mirano al soddisfacimento del seguente insieme di policy:

- integrità;
- disponibilità;
- affidabilità;
- autenticità;
- responsabilità
- non ripudiabilità;
- resilienza.

A tali obiettivi di sicurezza si è fatto riferimento, nuovamente, nella fase di progettazione del sistema oggetto di sviluppo, come discusso nel Capitolo 4.

1.5.1 Caratteristiche di sicurezza

Una prima tecnica di difesa adottata nel contesto delle blockchain è l'isolamento. I diversi nodi della rete in grado di verificare la correttezza di una transazione, ad eccezione della comunicazione reciproca che instaurano per il consenso, sono fisicamente isolati gli uni dagli altri: ciascuno di essi effettua i controlli sulle transazioni sottoposte a vaglio in maniera indipendente, dopodiché trascrive sulla blockchain ciò che emerge dalla maggioranza.

Il monitoraggio è un altro approccio di sicurezza a cui aderiscono le blockchain; una transazione viene accettata perché la maggioranza dei nodi l'ha monitorata per mezzo di opportuni controlli crittografici ed ha stabilito la sua validità. Conseguenza diretta di ciò è che quanto scritto all'interno di una blockchain viene considerato autentico; dunque, se ne risponde.

Come già descritto nel paragrafo relativo al funzionamento delle blockchain, inoltre, l'uso intensivo di tecniche crittografiche, su tutte l'hashing e le firme digitali, risulta fondamentale per garantire l'immutabilità dei dati trascritti sul registro. Difatti, una modifica al campo dati di un blocco della blockchain comporterebbe il cambiamento nell'hash di quello stesso blocco e, al contempo, tutti i blocchi ad esso successivi risulterebbero invalidati: il blocco seguente nella catena, infatti, si ritroverebbe con un riferimento al blocco precedente che non coincide più con l'hash attuale. Qualsiasi informazione già memorizzata si provi a cambiare, dunque,

andrà a compromettere il resto della blockchain. Conseguenza diretta di ciò è che più è lunga la blockchain, più essa risulta sicura e resistente ad alterazioni indebite.

L'unica possibilità per compromettere l'integrità consiste nel minare nuovamente i blocchi da quello modificato in poi ma, in generale, in tal caso entra in gioco la natura decentralizzata della tecnologia blockchain (oltre allo spreco di risorse computazionali) dovuta al fatto che ci sono diversi nodi indipendenti che eseguono il software della blockchain e andranno a comparare le singole copie del registro per vedere se ci sono versioni inconsistenti.

Altra caratteristica delle blockchain, motivo di distinzione rispetto ai registri tradizionali, è l'anonimato ottenuto attraverso tecniche di offuscamento. Tutti coloro che utilizzano una blockchain, infatti, non vengono identificati con la loro verà identità bensì per mezzo degli indirizzi dei corrispettivi portafogli elettronici o di certificati crittografici che, a tutti gli effetti, costituiscono degli pseudonimi. Chiaramente questi pseudonimi possono essere tracciati, dato che tutta la storia delle transazioni è presente nella blockchain, ma ciò non significa conoscere l'identità del proprietario di un certo indirizzo. Ad ogni modo è importante sottolineare che, ad eccezione dell'anonimato, non vi sono altre misure di riservatezza in quanto le scritture sulla blockchain, in linea generale, sono consultabili da chiunque entri a farne parte; se si ha a che fare con requisiti stringenti in termini di confidenzialità, pertanto, bisogna valutare attentamente l'utilizzo delle blockchain.

Le blockchain, inoltre, sfruttano al loro interno una serie di tecniche volte a garantire in particolar modo le policy di affidabilità e disponibilità.

Innanzitutto, ogni nodo della rete possiede una copia locale della blockchain che mantiene aggiornata coerentemente con gli altri peer; ciò significa che la blockchain, in quanto a struttura dati, è altamente ridondante, a differenza, ad esempio, di alcuni servizi cloud che mettono a disposizione un numero più contenuto di copie. Conseguenza naturale della ridondanza è che, se anche alcuni nodi in un certo momento dovessero risultare irrangiugibili, esiste un numero talmente elevato di copie della blockchain per cui sarà comunque possibile accedervi.

In combinazione a quanto appena descritto, le blockchain seguono anche l'approccio della distribuzione, tale per cui ogni nodo può trovarsi in un luogo geograficamente distante da tutti gli altri, aumentando, di conseguenza, la capacità che l'intero sistema rimanga affidabile a disponibile anche in situazioni di isolamento di una certa area.

Sempre all'interno di questa famiglia di tecniche, infine, è da citare la diversità, nonostante a primo impatto non sia così evidente, in quanto ogni nodo appare come un'esatta replica degli altri. In realtà, la diversità all'interno delle blockchain si può ravvisare sotto diverse ottiche. Innanzitutto, sono diversi ed indipendenti tra loro gli utenti a cui fanno riferimento i diversi nodi ed è proprio su questo presupposto che si fonda il consenso; avendo utenti diversi, non tutti ragionano allo stesso modo; quindi, se pur ci dovesse essere qualche nodo malevolo, ce ne saranno statisticamente tanti altri onesti. In linea teorica, poi, purché vengano rispettate le stesse regole per la validazione delle transazioni e per la partecipazione ai meccanismi di voto, ogni nodo potrebbe implementare le azioni di verifica e di formazione di un blocco in modo autonomo ed "originale".

1.5.2 Blockchain VS DBMS

Alla luce delle funzionalità messe a disposizione dalle blockchain, si potrebbe essere indotti a pensare a tale tecnologia come strumento di storage alternativo ai database tradizionali, come i DBMS, o ai servizi cloud.

In realtà, però, le blockchain presentano delle caratteristiche peculiari che, in alcuni casi, le rendono vantaggiose, ed in altri meno.

Dal punto di vista architetturale, innanzitutto, le blockchain sono strutture decentralizzate mentre i database sono tipicamente gestiti da una singola entità o un'organizzazione che

controlla l'accesso ai dati. Se nel primo caso, dunque, vige un modello di consenso, nel secondo caso le decisioni spettano ad un amministratore di sistema.

All'interno dei DBMS, inoltre, si adotta il modello delle azioni CRUD (Create, Read, Update, Delete) mentre in una blockchain è possibile soltanto inserire nuovi dati e leggere i dati inseriti. Se bisogna rettificare qualcosa nel contesto delle blockchain, dunque, si rende necessaria una nuova scrittura, il che potrebbe essere inefficiente in scenari che prevedono la gestione di dati da modificare con regolarità.

Quanto alla sicurezza, da un lato la blockchain può contare sulla sua natura crittografica e distribuita, il che riduce il rischio di frodi e manipolazione dei dati; dall'altro, in un database la sicurezza dipende dall'implementazione e dalle politiche di sicurezza adottate dall'organizzazione. In ogni caso, bisogna anche tenere conto del fatto che le blockchain non garantiscono tutte le proprietà di sicurezza perché, come già discusso, soffrono di problemi di confidenzialità.

Come ultimo termine di paragone, vi sono le prestazioni. In tal caso le blockchain, ed in particolar modo quelle basate su algoritmi di consenso più pesanti come Proof of Work, sono penalizzate dalle attività di verifica delle transazioni e dalla generazione della prova crittografica da parte dei minatori, con la possibilità di tempi non trascurabili. I tempi per registrare una transazione, in aggiunta, sono dipendenti dalla congestione della rete e possono risultare piuttosto variabili. Alcune blockchain, inoltre, possono sperimentare problemi di scalabilità a causa della necessità di replicare i dati su tutti i nodi della rete mentre, tramite soluzioni di *Big Data*, i database possono essere progettati al fine di scalare con relativa facilità.

Le blockchain, in conclusione, non costituiscono una soluzione universalmente valida bensì, nel momento in cui si decide di adottare tale sistema come strumento di sicurezza all'interno del proprio software, è necessario analizzare in maniera oculata i requisiti a cui bisogna rispondere modellando opportunamente i diversi casi d'uso.

1.5.3 Possibili attacchi alle blockchain

Nonostante le molteplici tecniche e soluzioni di sicurezza coinvolte nei protocolli che regolano le blockchain, non si possono escludere a priori eventuali rischi.

In primo luogo, ci può essere sempre la possibilità di sovversione del meccanismo di consenso, questione nota come probabilità di *deception*.

Relativamente all'algoritmo Proof of Work, si parla di attacco del 51% nello scenario in cui un utente malintenzionato, o un'organizzazione criminale, riuscisse ad acquisire il controllo di più del 50% della potenza di calcolo (*hash rate*) della rete, potendo contare, così, su una posizione di vantaggio nella risoluzione della prova di lavoro. In tal caso, dunque, l'attaccante sarebbe in grado di falsificare transazioni, imporre la propria versione della blockchain tramite un forking o, in generale, controllare quali transazioni vengono incluse nella blockchain.

In Proof of Stake, come già discusso, invece, non entra in gioco la potenza di calcolo; di conseguenza, affinché venga attuato un attacco analogo al 51%, un utente disonesto dovrebbe controllare più della metà dei token in circolazione, cioè dei diritti di voto.

Un altro attacco al meccanismo di consenso è noto come attacco di Sybil, in cui un attore crea più identità falsificate, o più nodi della rete apparentemenete indipendenti, ma controllati da un unico ente, nell'intento, ancora una volta, di influenzare il processo di voto.

Il meccanismo di consenso, inoltre, in una blockchain ben progettata dovrebbe prevenire attacchi di double spending, ossia l'invio di una transazione in cui gli stessi fondi vengono spesi e riutilizzati più di una volta.

Dopodiché, inevitabilmente, possono essere soggette ad attacchi le primitive crittografiche sottostanti tali protocolli, ad esempio tramite tecniche di *quantum computing*.

Non meno significativi sono gli attacchi che è possibile muovere sfruttando eventuali vulnerabilità riscontrate negli smart contract, i quali possono portare a danni per gli utenti o al furto di fondi. È questo il caso di cronaca che ha coinvolto l'organizzazione "The DAO" (una DAO è una *decentralized autonomous organization*), a seguito del quale è avvenuto il fork della blockchain Ethereum in due versioni.

Dal punto di vista meno ingegneristico, inoltre, l'anonimato e l'assenza di enti di controllo rendono le blockchain appetibili per attacchi di *phishing* e *social engineering*, così come per traffici illeciti; spesso, infatti, i riscatti a seguito di attacchi *ransomware* vengono richiesti in Bitcoin.

Un altro aspetto caratteristico delle criptovalute che espone a rischi, seppur non strettamente considerabile come attacco, è la loro volatilità: le forti oscillazioni nel tempo del valore di tali monete digitali, infatti, rendono le blockchain allettanti per la speculazione finanziaria.

1.6 Applicazioni delle blockchain oltre le criptovalute

Se è vero che le blockchain sono nate con l'intento di creare un sistema di pagamento elettronico esente da autorità centrali, è altrettanto vero che ben presto si sono diffusi molteplici sviluppi di tale tecnologia in contesti variegati e non legati alle sole criptovalute. In particolar modo, tale impulso si deve all'introduzione dei concetti di smart contract ed NFT.

In linea di principio, è possibile raggruppare le applicazioni che possono essere implementate su tecnologia blockchain in due grandi famiglie:

- *notarizzazione*, cioè sfruttare le blockchain per "cristallizzare" dati particolarmente rilevanti;
- *tokenizzazione*, ossia associare un token digitale emesso sulla blockchain ad un bene reale, con la possibilità di effettuare operazioni di compravendita di quel bene tramite il token

Un esempio di notarizzazione è la marcatura temporale digitale la quale sfrutta la cifratura asimmetrica, il calcolo dell'hash e il timestamp contenuto in un blocco con l'obiettivo, ad esempio, di dimostrare la proprietà di un documento ad una certa data.

Relativamente alla tokenizzazione, invece, è possibile classificare i token in diverse tipologie sulla base del loro impiego. A tal proposito, oltre ai *payment token* (analoghi a tutti gli effetti a criptovaluta), sono da citare gli *asset token*, utilizzati per rappresentare la partecipazione al profitto in un'azienda, ed, infine, gli *utility token*, il cui scopo è quello di rappresentare il diritto di accesso ad una piattaforma oppure a benefici o servizi premium (ad esempio, spazio cloud o capacità di calcolo).

Un esempio di utility token è Steem, sperimentato per dare più o meno reputazione all'interno dei cosiddetti decentralized social networks, ossia piattaforme social che hanno lo scopo di evitare censure, ricompensare contenuti di valore e autenticare contenuti. A tal proposito, infatti, è anche in studio un protocollo blockchain chiamato Proof of Brain, con il fine ultimo di ridurre l'impatto delle fake news.

1.6.1 Certificazione dell'autenticità

Una particolare menzione va riservata all'ambito della certificazione dell'autenticità e la tracciabilità di beni, aspetto che sarà il filo conduttore dell'intera trattazione.

Le caratteristiche intrinseche delle blockchain, in comunione con la possibilità di generare token non fungibili unici che rappresentano digitalmente asset di varia natura, costituiscono

le condizioni ideali per l'implementazione di sistemi in grado di fornire una prova dell'origine di dati, prodotti, opere d'arte, attestandone al contempo la loro fedeltà.

Già sperimentati sul campo sono i primi sistemi per la certificazione di dati, dai dati sanitari fino al tracciamento delle manipolazioni e delle condizioni dei prodotti lungo la filiera che ha come estremi i fornitori di materie prime e gli scaffali per la vendita al pubblico.

In quest'ottica, dunque, gli scenari applicativi vanno di pari passo all'immaginazione e sono in continua evoluzione.

Un esempio interessante è la certificazione di competenze come diplomi di laurea; altri utilizzi, anche già affermati e spesso frutto di importanti guadagni, riguardano l'ambito della proprietà intellettuale specialmente in campo artistico. In questo scenario, infatti, un possibile scopo è quello di tutelare i legittimi autori attraverso licenze e royalty gestite ed automatizzate tramite smart contracts, favorendo, al contempo, una maggiore fruizione ed un accesso globale alle opere grazie alla possibilità di raggiungere platee più vaste, aprendo dunque nuove opportunità per il commercio e l'espozione artistica.

Un focus sulle applicazioni attualmente esistenti riguardo gli utilizzi delle blockchain nel settore della tracciabilità dei prodotti verrà affrontato nel Capitolo 2.

CAPITOLO 2

Blockchain per il tracciamento dei prodotti

In questo capitolo l'attenzione verrà posta nei confronti di uno specifico ambito applicativo che si presta particolarmente all'utilizzo delle blockchain, ossia il tracciamento di prodotti.

Innanzitutto, si farà un excursus sulle modalità e le problematiche che caratterizzano gli approcci tradizionali al monitoraggio dei prodotti durante le diverse fasi della filiera produttiva.

A seguire, verranno illustrati il contributo e l'apporto delle blockchain all'interno dello scenario in esame. Nello specifico, dunque, si evidenzieranno i possibili impieghi volti a garantire l'origine e l'autenticità dei prodotti, delineando, di fatto, il quadro della situazione attuale a livello tecnologico.

Trattandosi di soluzioni che, per quanto ancora in corso di assestamento, sono sempre più concrete e diffuse, si proseguirà, successivamente, ad esaminare un esempio pratico di sistema per il tracciamento basato su blockchain.

Infine, il capitolo sarà concluso con alcune considerazioni riguardanti gli aspetti ancora aperti e le sfide future nell'ambito del tracciamento dei prodotti, nonostante l'avvento delle blockchain.

2.1 Tracciamento dei prodotti nelle catene di approvvigionamento

La maggior parte dei prodotti a disposizione dei consumatori sugli scaffali dei negozi, per quanto semplici possano sembrare a primo impatto, nascondono spesso un ciclo di vita più o meno complesso, che prevede, a monte, diverse manipolazioni.

Una catena di approvvigionamento, nota anche con il termine inglese di *supply-chain*, è un processo interconnesso che comprende proprio tutte le attività coinvolte nella trasformazione di materie prime in prodotti finiti e nella consegna di tali prodotti all'utenza.

Affinché l'intera catena proceda nella maniera più regolare, e i prodotti risultanti soddisfino determinati criteri di qualità imposti, si rende necessaria una parallela azione di monitoraggio e tracciamento dei singoli step attraversati.

Tale processo di tracciamento presenta un duplice vantaggio che interessa sia le realtà di business coinvolte a monte che l'utenza finale. Le aziende, da un lato, possono verificare più facilmente la conformità alle normative, attestare la qualità e la sicurezza dei prodotti, identificare e rispondere repentinamente a possibili irregolarità o malfunzionamenti, far fronte ad eventuali colli di bottiglia nella produzione, fidelizzare la clientela ed, in generale, migliorare l'efficienza delle operazioni aziendali; i consumatori, d'altro canto, hanno l'opportunità di conoscere l'origine e la storia di un prodotto, assicurandosi della bontà dei processi a cui esso è stato sottoposto.

2.1.1 Descrizione della problematica

Considerato che una catena di approvvigionamento abbraccia l'insieme di processi che vanno dalla fase di reperimento delle materie prime, passando per la lavorazione, fino alla distribuzione e alla vendita del prodotto, inevitabilmente si tratta di un ecosistema in cui operano numerosi attori, o *stakeholder*, i quali interagiscono con il prodotto stesso secondo modalità diverse. Fornitori, produttori, distributori, trasportatori, rivenditori, ed eventuali intermediari, difatti, lavorano in sinergia per garantire che i prodotti raggiungano i consumatori finali in modo efficiente e conforme alle esigenze del mercato.

Le supply-chain, inoltre, possono variare notevolmente a seconda del tipo di prodotto, dell'industria, della geografia e delle modalità di distribuzione. Alcune catene di approvvigionamento, infatti, sono relativamente semplici, mentre altre possono coinvolgere diverse e più elaborate fasi di produzione, assemblaggio, trasporto internazionale e stoccaggio.

Affinché i prodotti siano disponibili nei tempi giusti, nei luoghi opportuni e nella corretta quantità, in ogni caso, ciascuna fase prevista all'interno di una supply-chain richiede pianificazione, coordinamento, monitoraggio e comunicazione tra le parti coinvolte. In questo contesto, dunque, si è sviluppata una vera e propria disciplina che prende il nome di *Supply-Chain Management* (SCM), di cui parte integrante è l'attività di tracciamento.

Appositi sistemi SCM hanno il compito di supervisionare i tre flussi principali di una catena di approvvigionamento, ossia il flusso dei prodotti, il flusso delle informazioni e il flusso delle finanze, coordinandoli sia internamente alle singole aziende coinvolte che tra di esse. A tale scopo, e per adeguarsi a ritmi di produzione sempre più frenetici e scambi di dati sempre più frequenti e numerosi, il tracciamento delle informazioni e dei movimenti connessi ai prodotti risulta indispensabile.

Più nello specifico, il processo di tracciamento di un prodotto interessa, solitamente, i seguenti elementi chiave:

- *l'origine*, ossia vengono raccolte informazioni riguardo i fornitori, le materie prime utilizzate, il luogo di produzione, i test di qualità soddisfatti, etc.;
- *la catena di lavorazione* seguita dal prodotto, con le diverse condizioni operative e gli spostamenti annessi;
- *la distribuzione e la vendita*, che coinvolge, anche, le spedizioni ed eventuali intermediari;
- *l'utilizzo e il consumo del prodotto* (non sempre), permettendo di raccogliere informazioni anche in seguito alla consegna presso il cliente finale.

Al fine di attuare le operazioni di tracciamento in maniera accurata, affidabile ed efficiente, possono essere presi in considerazione una serie di requisiti, alcuni dei quali ormai standardizzati, altri idealmente da soddisfare.

Innanzitutto, è bene che ogni prodotto sia dotato di un identificatore univoco che ne agevoli il riconoscimento nel corso delle diverse fasi e tra i suoi simili.

In secondo luogo, è doveroso registrare dettagliatamente le informazioni relative a ciascun prodotto e i suoi movimenti tramite modalità sicure e a prova di manipolazioni indebite, onde evitare falsificazioni nei dati e, di conseguenza, la compromissione del prodotto stesso.

Memorizzare tutti i movimenti di un prodotto, nello specifico, risulta fondamentale affinché la tracciabilità possa procedere in una duplice direzione; in molti casi, infatti, è necessario poter seguire il percorso del prodotto non solo in avanti (dall'origine fino alla destinazione finale) ma anche in maniera retroattiva (dalla destinazione finale a ritroso fino all'origine), così da ottenere una visibilità del flusso del prodotto a 360 gradi. Ciò, ad esempio, risulta particolarmente utile in tutti quei contesti in cui si potrebbe celare il rischio di

contraffazione dei prodotti, minaccia che rappresenta una delle maggiori sfide da fronteggiare nell'ottica del tracciamento.

Oltre a ciò, è da tenere in considerazione anche l'aspetto temporale; i dati relativi al monitoraggio di un prodotto all'interno della supply-chain, infatti, dovrebbero essere acquisiti contestualmente all'operazione fisica svolta ed immagazzinati tempestivamente, in modo tale che i diversi attori possano accedere ad informazioni sempre aggiornate e coerenti.

Un'ulteriore questione da trattare riguarda, proprio, l'accesso ai dati. Le modalità e i confini entro cui ciò avviene contribuiscono alla cosiddetta visibilità della supply-chain (SVC), fattore che esprime il grado di tracciabilità e la disponibilità delle informazioni relative all'intera catena di approvvigionamento. L'SVC monitora le diverse operazioni di logistica, consentendo di controllare lo stato attuale degli eventi e di osservare il raggiungimento di particolari *milestone* che riguardano un prodotto o un materiale. A tal fine, è evidente la necessità di facilitare l'accesso alle informazioni da parte dei singoli stakeholder coinvolti, così da raggiungere una collaborazione efficace e guadagnare vantaggi competitivi.

Quando si parla di condivisione di dati, però, bisogna anche prestare massima attenzione alle policy di confidenzialità che essi richiedono; poiché il tracciamento dei prodotti, in linea di principio, può coinvolgere informazioni sensibili, è essenziale garantire sicurezza e protezione da accessi non autorizzati.

Infine, un aspetto spesso sottovalutato, che riguarda tutti i contesti di *data gathering*, consiste nel rispetto di normative, leggi e standard vigenti, sia a livello locale che internazionale; basti pensare, ad esempio, alle regolamentazioni che interessano il settore della filiera agro-alimentare.

2.1.2 Sistemi tradizionali per la gestione delle supply-chain e il tracciamento di prodotti

Prima di analizzare il contributo delle blockchain alla causa, e al fine di comprenderne a pieno le potenzialità, è necessario fornire un quadro generale delle caratteristiche e degli approcci tipici dei sistemi adottati di consueto per il tracciamento di prodotti lungo la filiera.

Da un lato, è evidente come il contesto in cui si ritrovano ad operare le aziende sia in continua evoluzione a causa della diffusione di tecnologie ed infrastrutture informatiche che permettono di automatizzare i processi e gestire efficientemente cospicue moli di dati; l'avvento dell'industria 4.0, ad esempio, ha sicuramente contribuito ad una maggiore sensibilità delle imprese, specialmente quelle manifatturiere, nei confronti del tema dell'innovazione digitale, di cui parte rilevante è la tracciabilità delle informazioni.

D'altra parte, nonostante siano sempre più forti la consapevolezza e la propensione verso l'integrazione di nuove soluzioni in ambito tecnologico, sono ancora numerose le realtà che conservano strumentazioni e metodologie meno evolute per la gestione delle supply chain e, conseguentemente, per il tracciamento di beni all'interno della filiera produttiva, in particolar modo a causa delle limitate possibilità di investimento, delle ridotte dimensioni d'impresa o di strategie aziendali che prediligono know how ormai assodati da tempo. Precisato, dunque, che il termine "tradizionale" non traccia confini netti, e prescindendo dalle ovvie differenze riscontrabili in specifici casi d'uso o in diversi mercati, è possibile, comunque, individuare delle caratteristiche che accomunano i sistemi ad oggi adottati in larga scala per il tracciamento dei prodotti.

Le supply-chain "tradizionali" fanno affidamento, per lo più, su sistemi *standalone*, che lavorano in maniera isolata e prevedono una limitata condivisione dei dati.

Nello specifico, il processo di tracciamento all'interno delle catene di approvvigionamento è coadiuvato da sistemi gestionali tipicamente di natura centralizzata.

Tra i tool che maggiormente possono essere sfruttati per il tracciamento dei beni in una catena di approvvigionamento, oltre ai già menzionati sistemi SCM, è possibile citare sistemi

di *Product Lifecycle Management* (dedicati alla gestione dell'intero ciclo di vita di un prodotto), *Warehouse Management System* (per la gestione di inventari e la tracciabilità dei movimenti dei prodotti nei magazzini), *Manufacturing Execution System* (per monitorare le attività di produzione in tempo reale) e *Transportation Management System* (per gestire aspetti legati al trasporto e alle spedizioni).

I sistemi appena elencati presupongono, a monte, la capacità di identificare con esattezza un prodotto all'interno della filiera, concetto cardine dell'intero processo di tracciamento. A tal fine, nel corso degli anni, sono state proposte molteplici soluzioni e tecnologie, ma, ad oggi, quelle più note e da più tempo presenti sul mercato sono riconducibili a barcode, RFID e RTLS.

I barcode, o codici a barre, rappresentano, sicuramente, la tecnologia più matura e più diffusa nello scenario in questione. L'utilizzo di un barcode consente di acquisire dati automaticamente per mezzo di appositi dispositivi elettronici; tramite lettori di codici a barre collegati ad un PC, infatti, è possibile estrarre in maniera agevole i dati relativi ad un prodotto e recapitarli all'interno dei sistemi informatici.

Tale tecnologia, come facilmente intuibile, garantisce un'acquisizione del dato rapida e senza errori rispetto all'inserimento manuale da parte di un operatore; di contro, richiede la visione diretta dell'etichetta su cui è posto il barcode (con i vincoli pratici del caso); inoltre, se l'etichetta è danneggiata, c'è il rischio di non poter leggere le informazioni contenute. Per ovviare a ciò, sono state introdotte tecnologie più sofisticate, come i *Quick Response Code*, che consistono in codici a barre bidimensionali, adatti per essere utilizzati in supporti non necessariamente cartacei, bensì direttamente stampati sul prodotto e contenenti al loro interno un maggior numero di informazioni.

RFID (Radio Frequency IDentification) è una tecnologia di identificazione alternativa al codice a barre che, per l'acquisizione, sfrutta le onde radio. Un sistema RFID, infatti, è costituito da due elementi principali, un transponder, o tag, applicato al prodotto da identificare e contenente le informazioni interessate, ed un reader, o gate, ossia un dispositivo che, captando onde elettromagnetiche a determinate frequenze, consente di leggere e scrivere quanto contenuto nel transponder.

Rispetto ad un codice a barre, l'RFID permette di processare una maggiore quantità di informazioni con, in più, la possibilità di aggiornarle. Dal punto di vista operativo, inoltre, tale tecnologia vanta una maggiore flessibilità dovuta alla possibilità di leggere in parallelo più tag contemporaneamente ed in automatico, senza la necessità di un contatto visivo diretto o di una scansione manuale; oltre a ciò, le dimensioni ridotte dei tag permettono una facile integrazione anche all'interno del prodotto stesso. Infine, quanto a sicurezza, le informazioni memorizzate possono essere criptate.

D'altra parte, lo stesso principio fisico su cui si basa la tecnologia RFID, l'utilizzo di onde radio, potrebbe costituire un limite dovuto alle potenziali interferenze, in fase di lettura, con gli oggetti su cui i tag sono applicati e, più in generale, con l'ambiente circostante.

Gli RTLS (Real Time Locating System), infine, sono sistemi di localizzazione in tempo reale, utilizzati per identificare e monitorare automaticamente la posizione di oggetti all'interno di un'area circoscritta, come un magazzino; i dati di geolocalizzazione, dopodiché, vengono inviati all'interno delle già citate piattaforme di Warehouse Management System e/o Manufacturing Execution System, così da gestire e controllare in maniera puntuale i processi aziendali interessati.

Quanto alla documentazione ausiliaria al processo di tracciamento, proseguendo la panoramica delle soluzioni tipiche, le aziende mantengono appositi registri, solitamente basati su database, fogli elettronici, o addirittura cartacei nelle realtà più piccole, al fine di annotare le informazioni chiave relative ai prodotti, tra cui la provenienza, la data di produzione, i fornitori coinvolti, la data di spedizione, etc. Tali registri, spesso, vengono

aggiornati manualmente man mano che i prodotti attraversano i singoli step della catena di approvvigionamento.

Oltre a ciò, non è raro che, altrettanto manualmente, vengano svolte le verifiche di uniformità nei confronti di normative e standard di qualità durante il percorso del prodotto nella catena di approvvigionamento.

Alla luce dei molteplici attori coinvolti in una supply-chain, inoltre, appare evidente la necessità di scambi di informazioni e comunicazioni frequenti; le aziende, ad esempio, hanno l'esigenza di condividere con i propri partner dati riguardanti la manipolazione o il transito dei prodotti; a tal proposito, i mezzi di consueto utilizzati sono email, piattaforme Intranet, portali web ad accesso protetto e documenti cartacei, quali fatture, bolle di consegna e ordini d'acquisto. Scambi di messaggi di questo tipo, come è facilmente intuibile, possono essere soggetti ad errori umani, alterazioni e ritardi, come chiarito nella Sezione 2.1.3.

Ad ogni modo, anche per quanto riguarda l'aspetto della comunicazione e della condivisione delle informazioni, va sottolineato che sono sempre più frequenti i casi di utilizzo di strumenti collaborativi, database condivisi o piattaforme cloud, così come è da menzionare l'adozione di metodologie più strutturate per lo scambio elettronico interaziendale di documenti commerciali, sfruttando, ad esempio, il formato standard EDI (*Electronic Data Interchange*).

2.1.3 Limiti delle soluzioni tradizionali

Alla luce degli incalzanti ritmi di produzione odierni, che portano, quotidianamente, alla creazione di milioni di prodotti per mezzo di supply-chain estese a livello globale, e considerate le molteplici modalità di acquisto (in-store, online, etc.) a disposizione dei consumatori, è facile intuire le difficoltà insite nel coordinamento dei processi sottostanti e le nuove sfide che ogni giorno sorgono, specialmente per ciò che riguarda il tracciamento dei beni.

A tal proposito, diversi sono gli articoli scientifici che hanno analizzato l'ecosistema delle attuali catene di approvvigionamento, evidenziandone i punti deboli e motivando, al contempo, l'impiego delle blockchain nello scenario in questione; frequenti, dunque, saranno i riferimenti a tali lavori.

Secondo Agarwal *et al.* [2022], tra le principali problematiche ravvisabili nella gestione delle supply-chain e, di riflesso, nel processo di tracciamento dei prodotti, è da considerare, sicuramente, la natura centralizzata degli approcci adottati, sia per quanto riguarda le metodologie operative che le infrastrutture software.

Spesso, l'organigramma aziendale prevede una serie di manager responsabili di dipartimenti diversi, quali l'approvvigionamento, la logistica e la distribuzione; tali figure si occupano di supervisionare la propria area di competenza lungo l'intera supply-chain, ma operano in maniera pressochè indipendente e rispondono, in ogni caso, unicamente alla singola sede operativa o al singolo magazzino di afferenza. In questo contesto, dunque, le informazioni raccolte vengono memorizzate in database centralizzati. Tutti gli scenari caratterizzati da entità centralizzate, come già anticipato nel Capitolo 1, sono soggetti a possibili problemi in termini di single point of failure, in cui un malfunzionamento o un attacco all'elemento centrale potrebbe causare la totale cessazione del servizio o la compromissione dell'intero sistema, traducendosi, di fatto, in ingenti perdite economiche.

Oltre a ciò, una supply-chain aderente ad un paradigma centralizzato è, spesso, costosa da gestire, sia a livello burocratico che di capacità di storage per la memorizzazione delle informazioni, priva di funzionalità avanzate per l'analisi approfondita del mercato, nonché poco efficiente in termini temporali a causa di una scarsa attitudine alla dinamicità e alla reattività.

Un approccio centralizzato, inoltre, può richiedere ingenti oneri computazionali per processare i dati raccolti e necessita di servizi di terze parti per la loro autenticazione, con l'ulteriore difficoltà della mancanza di armonizzazione tra una piattaforma e l'altra; in un ecosistema centralizzato in cui diversi attori concorrono alla realizzazione di un prodotto finito, infatti, l'amministratore dei dati cambia di volta in volta, dal produttore al rivenditore, e, in ogni step, esso è l'unico responsabile del tracciamento dei prodotti, memorizzando informazioni che, però, risultano mutevoli e possono essere alterate in qualsiasi momento senza notificare gli altri attori.

In assenza di una visione distribuita sul sistema, per di più, può essere difficoltoso rilevare prontamente dove e come ha avuto luogo un'anomalia; in molti casi, inoltre, anche quando viene individuata una difformità, predire gli effetti più probabili e trovare una risoluzione efficace può comportare lo spreco di risorse, considerando, per giunta, la presenza di intermediari cui rendere conto. Come diretta conseguenza, dunque, possono generarsi ritardi di produzione o difetti nei prodotti che, in generale, allungano il tempo necessario per raggiungere il mercato, a discapito della soddisfazione della clientela e dei profitti aziendali.

Dati alla mano, come riportato in Tian [2016], in Cina, il tasso di smarrimenti durante i processi logistici nel settore agro-alimentare supera il 30% annuo, principalmente a causa di sistemi centralizzati.

Non è raro, inoltre, che gran parte delle informazioni relative ai prodotti e al loro status nei vari step della filiera siano memorizzate in maniera offuscata e risultino difficilmente accessibili a causa di policy stringenti di privacy o segretezza aziendale; ciò ha progressivamente portato ad una mancanza di fiducia tra i diversi stakeholder coinvolti, il che, a sua volta, si traduce in spese più alte per le comunicazioni. Tali costi, inevitabilmente, sono assorbiti dal consumatore.

L'opacità costituisce, senz'altro, un significativo punto debole delle usuali soluzioni per la gestione e il tracciamento delle catene di fornitura. Tale fattore, congiuntamente alla mancanza di dati in tempo reale, rende più complicate le operazioni aziendali di *auditing*, con conseguente peggioramento delle attività di *decision-making*, e riduce l'affidabilità complessiva del sistema, esponendo ad eventuali pratiche fraudolente.

Non può essere trascurato, in aggiunta, l'alto rischio di manipolazione o cancellazione (più o meno volontarie) dei dati mantenuti dalle entità che prendono parte alla supply-chain, così come le problematiche legate all'utilizzo di formati incompatibili, entrambe cause di difficoltà e ritardi nelle operazioni di *tracing*.

Nel momento in cui tali dati devono essere scambiati e condivisi, inoltre, appare evidente la necessità di metodi e protocolli in grado di offrire migliori garanzie, dal punto di vista della resistenza ad alterazioni premeditate o ad errori umani, rispetto ai mezzi convenzionali (email, documenti cartacei, etc.).

Da tale quadro, dunque, emergono livelli piuttosto insoddisfacenti per i requisiti di trasparenza, visibilità ed integrità delle supply-chain, fattori che danno adito ad uno dei maggiori problemi, nell'ottica del tracciamento, che le aziende cercano di fronteggiare ormai da decenni, ossia la contraffazione.

Secondo uno studio congiunto dell'Ufficio dell'Unione Europea per la Proprietà Intellettuale (EUIPO) e dell'Organizzazione per la Cooperazione e lo Sviluppo Economico (OCDE) relativo al 2022, il volume del commercio internazionale di prodotti contraffatti ammonta a 464 miliardi di dollari, una fetta pari al 2.5% delle cifre relative all'intero commercio mondiale. Nell'UE, inoltre, si stima che il 5.8% di tutte le importazioni da Paesi terzi sia costituito da prodotti contraffatti, per un valore di 119 miliardi di euro.

Una delle ragioni principali che hanno portato alla proliferazione del mercato del falso è che il processo di manifattura e distribuzione dei prodotti pecca di trasparenza specialmente

nei confronti dei consumatori finali, i quali si possono imbattere in informazioni facilmente falsificabili o replicabili da altri che, una volta immesse, sono difficili da verificare o validare.

Gli attaccanti, in questo scenario, riescono ad inserire sul mercato i propri prodotti contraffatti accanto a quelli legittimi, spesso rendendoli indistinguibili agli occhi dei consumatori, che vedono unicamente il risultato finale di diversi processi di lavorazione, senza, però, poterne entrare nel dettaglio ed avere garanzie di legittimità *step-by-step*.

Alcuni tipici esempi di pratiche che contribuiscono ai reati di contraffazione e furto della proprietà intellettuale prevedono l'apposizione di etichette false sui prodotti, l'utilizzo di materiali e componenti falsi o inappropriati durante i processi manifatturieri, l'abuso e la replica non autorizzata di un marchio. Oltre a ciò, il compito di validare la provenienza e la qualità dei prodotti è reso ulteriormente difficile dalla delocalizzazione delle produzioni anche in aree geografiche in cui le regolamentazioni e le normative da seguire sono meno stringenti o, di per sè, poco trasparenti.

Con l'aumento del numero di prodotti contraffatti presenti sul mercato, dunque, è sempre più difficile scoprirne la reale provenienza, e ciò ha un pesante impatto sia per le aziende che per i consumatori. I proprietari di brand, da un lato, registrano voluminose perdite in termini di fatturato e di reputazione presso la clientela; indirettamente, inoltre, la contraffazione disincentiva l'innovazione e la crescita economica ad essa collegata. I consumatori, d'altro canto, è possibile che abbiano solamente una conoscenza limitata dei prodotti, ragione per cui possono acquistare non intenzionalmente merce contraffatta; diretta conseguenza di ciò è che risultano maggiormente esposti a truffe (specialmente acquistando online), sperimentano più frequentemente difetti nei prodotti e, soprattutto nel settore alimentare, possono incorrere in rischi per la salute.

Tecnologie e dispositivi di tracciamento sempre più "smart", come QR Code e tag RFID, citati nella Sezione 2.1.2, costituiscono sicuramente degli ottimi spunti e mostrano notevoli potenzialità per la lotta alla contraffazione ma, di per sè, non rappresentano soluzioni definitive. Possibili vulnerabilità dei tag RFID, ad esempio, sono legate ad attacchi di clonazione, disturbo del segnale, rilevamento passivo di informazioni sensibili, riutilizzo di dati catturati, spoofing del segnale tramite dispositivi falsi e danneggiamento fisico. Appare evidente, dunque, che tali strumenti, per raggiungere il massimo della loro funzionalità, necessitano dell'integrazione con infrastrutture solide, dal punto di vista dell'affidabilità.

Oltre alla contraffazione, infine, la scarsa trasparenza e le carenze riguardo le capacità di tracciamento in una supply chain possono portare anche alla nascita di controversie e dispute legali tra gli stakeholder; tali contenziosi, in assenza di strumenti per l'enforcement, magari anche in maniera automatizzata, di clausole oggettivamente verificabili, risultano complicate da risolvere tramite terze parti.

Tenendo conto delle esigenze emerse, dei requisiti da rafforzare e delle questioni aperte appena sottolineate, la tecnologia blockchain può trovare, senz'altro, ampio spazio d'applicazione, così come verrà discusso nella prossima sezione.

2.2 Impatto delle blockchain nel tracciamento: stato dell'arte

Le blockchain, grazie alle loro caratteristiche intrinseche di decentralizzazione, immutabilità e trasparenza, si pongono naturalmente come valida tecnologia in grado di fornire, in maniera olistica, un solido supporto al processo di tracciamento di una supply-chain, contesto in cui la gestione dei prodotti è affidata a più parti.

Difatti, sfruttando solide basi crittografiche, le blockchain costituiscono un sistema per la registrazione permanente ed inalienabile di dati ed eventi, rispondendo, perciò, in modo molto efficace all'esigenza di depositare e legittimare le informazioni acquisite da diversi attori, spesso privi di reciproca fiducia, durante i diversi step di una catena di approvvigionamento.

Nelle sezioni seguenti, dunque, verrà analizzato come possono essere sfruttate concretamente le caratteristiche peculiari delle blockchain per soddisfare i requisiti richiesti nell'ottica del tracciamento dei prodotti, evidenziando i benefici che è possibile trarne.

2.2.1 Utilizzo delle blockchain per garantire la provenienza

Uno dei primi compiti che possono assolvere le blockchain nel contesto in esame consiste nel fornire trasparenza circa l'origine dei prodotti.

I consumatori, spesso, hanno accesso solamente ad una porzione limitata delle informazioni rilasciate dai grandi brand e non hanno modo di verificare, in prima persona, l'eticità delle pratiche produttive a cui è sottoposto un bene; conseguentemente, dunque, non possono fare altro che affidarsi al logo di certificazione stampato sul prodotto stesso.

Una volta notarizzate sulla blockchain, invece, le informazioni sull'origine, quali, ad esempio, le materie prime utilizzate, il luogo e le condizioni di produzione, i test superati, vengono indelebilmente correlate ad un marcatore temporale e, da quel momento in poi, risulteranno disponibili alla verifica per tutti gli utenti che prendono parte alla rete, senza che nessuno possa assumerne il totale controllo o abbia la possibilità di alterarle.

I meccanismi che governano le blockchain, difatti, permettono di distribuire in poco tempo i dati relativi all'origine di un prodotto attraverso l'intera rete, agevolandone, da un lato, la condivisione e, dall'altro, l'integrità; maggiore è il numero di nodi a disporre della medesima versione circa la provenienza di un prodotto, più tali informazioni potranno essere ritenute affidabili e saranno difficilmente compromissibili.

Nell'intento di confermare l'origine di un prodotto, inoltre, è fondamentale poter contare su un tracciamento retroattivo; data la cronologicità delle registrazioni e la consequenzialità che si viene a creare nella struttura della blockchain, è possibile seguire il percorso di un prodotto all'indietro, dall'istante corrente fino all'origine, fornendo una chiara visione di quando, dove e come è stato creato e quali processi o movimenti ha subito.

Dopodiché, per poter effettuare controlli mirati sulla base di regole codificate e, parallelamente, ridurre al minimo o eliminare la necessità di processi di auditing che, svolti internamente o tramite terze parti, presupporrebbero un certo livello di fiducia, molteplici possibilità sono offerte dagli smart contract. L'azienda produttrice, infatti, può pensare di predisporre un'apposito smart contract di verifica che implementa le regole e le procedure utili a validare l'origine di un prodotto. Un contratto del genere, sostanzialmente, potrebbe includere del codice che confronta le informazioni registrate sulla blockchain relativamente al prodotto con le informazioni fornite dall'utente che intende effettuare la verifica.

Gli utenti che desiderano ottenere garanzie sulla provenienza di un prodotto, dunque, possono interagire con lo smart contract (tramite un'applicazione web o mobile), richiamando le funzioni di quest'ultimo e fornendo in input i parametri richiesti.

Lo smart contract, quindi, esegue automaticamente i controlli in esso implementati, verificando, ad esempio, se l'identificatore univoco di un prodotto inserito dall'utente corrisponde alle informazioni di origine o se i dati del prodotto sono coerenti e conformi a normative. Dopodiché, sulla base dell'esito della verifica, esso restituirà una risposta appropriata all'utente; alternativamente, nel momento in cui un certo obiettivo è raggiunto o un vincolo è soddisfatto, lo smart contract può essere configurato per svolgere automaticamente un task, come l'esecuzione di un pagamento o il trasferimento di un asset digitale, il tutto senza la partecipazione di terze parti.

Le caratteristiche degli smart contract assicurano che, una volta aver avviato l'esecuzione, il sistema possa proseguire automaticamente fino alla fine del contratto senza interferenze esterne, a favore di una maggiore affidabilità e resilienza.

Grazie ai principi di sicurezza su cui le blockchain si fondano, inoltre, ciascuna operazione effettuata tramite uno smart contract diviene tracciabile e verificabile in qualsiasi momento.

Il monitoraggio delle transazioni ad opera dei nodi validatori della rete, per quanto detto, contribuisce ad ottenere la cosiddetta "proof of provenance" che le blockchain sono in grado di fornire.

Poter contare su una prova documentata e crittograficamente accertata relativa alla storia di un prodotto risulta particolarmente utile in tutti quei settori in cui l'origine, la qualità, l'autenticità e la conformità costituiscono requisiti critici come, ad esempio, nel campo alimentare, dell'arte, dei beni di lusso e dell'industria manifatturiera.

A testimonianza dell'importanza e dell'impatto potenziale di tali applicazioni, secondo uno studio condotto da Gartner [2019], è previsto che l'utilizzo delle blockchain per comprovare l'origine dei beni generi la cifra di 962 milioni di dollari in termini di PIL mondiale entro il 2030.

2.2.2 Tracciabilità della filiera produttiva tramite blockchain: registrazione delle ownership e certificazione della qualità

Nell'intento di supportare il processo di tracciamento nella maniera più esaustiva possibile, oltre alle informazioni relative alla provenienza, è utile registrare sulla blockchain tutti i passaggi di consegne a cui è sottoposto un prodotto.

Fare chiarezza sulla successione degli attori che interagiscono con un prodotto, infatti, è utile non solo per ristabilire un sufficiente grado di fiducia presso i consumatori ma anche per aiutare le aziende produttrici ad avere una migliore prospettiva di eventuali falle nella supply-chain e capire come determinate scelte o fattori ambientali possano influenzare un prodotto.

Come ormai noto, man mano che un prodotto avanza all'interno della catena di approvvigionamento, diversi attori (fornitori, produttori, trasformatori, distributori, rivenditori, consumatori) ne entrano in possesso e contribuiscono, in qualche modo, alla sua storia.

Ciascuna entità coinvolta in una supply-chain, dunque, andrà a rivestire un ruolo fondamentale per l'intero sistema in quanto, anche se geograficamente distante dagli altri partner, è chiamata ad ampliare l'informazione associata ad un prodotto inserendo in un registro comune, la blockchain, dati sullo stato del prodotto che derivano dalle operazioni svolte in prima persona su di esso. Questo, insieme all'annotazione che un bene, in un determinato istante di tempo, è tra le mani di una certa figura, fornisce un quadro decisamente chiaro del ciclo di vita e dei singoli step attraversati dal prodotto stesso. Diretta conseguenza di ciò è che un consumatore, o in generale qualsiasi stakeholder sia in procinto di acquisire il prodotto in questione, può fare affidamento su informazioni che non sono più parziali o frammentate bensì attestano, in maniera incontrovertibile e non ripudiabile, la catena di proprietà e di processi svolti sul prodotto, a monte.

Alla luce di ciò, le blockchain permettono di aumentare la disponibilità e di facilitare la trasmissione dei dati riguardanti i processi di manifattura, assemblaggio, distribuzione e manutenzione del prodotto tra tutti i partecipanti della supply chain. Al tempo stesso, tenendo traccia di tutti i documenti aziendali, quali ordini di acquisto, modifiche agli ordini e ricevute di pagamento, è possibile vigilare il flusso dei dati della catena di fornitura in modo più mirato. Grazie ai principi di decentralizzazione e sicurezza crittografica, inoltre, il trasporto e la proprietà di questi dati possono essere salvaguardati, in modo più efficace, da manipolazioni o hacking.

La tecnologia blockchain si presta in maniera diretta e naturale a tenere traccia di tutti i movimenti e i passaggi di proprietà che coinvolgono un bene, concedendo possibilità di verifica e di *auditing* affidabili al momento del bisogno. Svolgendo un parallelismo, infatti, un blocco della blockchain può rappresentare ciascun collegamento nella catena di approvvigionamento, dalla manifattura alla vendita, e, come nello scenario reale, tali blocchi sono interconnessi per completare il processo.

Tracciare la catena degli attori che si susseguono alla proprietà di un prodotto, scendendo in aspetti più tecnici, prevede una serie di passaggi. Innanzitutto, è necessario che ogni prodotto venga dotato di un identificatore univoco, sfruttando, ad esempio, un codice a barre, un numero di serie o un token digitale. Dopodiché, quando un prodotto cambia proprietà, deve essere generata una transazione sulla blockchain per registrare il passaggio; a tal proposito, vengono inclusi i dettagli del vecchio e del nuovo proprietario, l'identificatore del prodotto ed, eventualmente, ulteriori dati rilevanti. La transazione, dunque, viene firmata digitalmente dal vecchio proprietario utilizzando la propria chiave privata, la quale ha la duplice funzione di confermare l'autenticità della transazione e fornire l'autorizzazione al trasferimento. A questo punto, la transazione firmata viene trasmessa alla rete blockchain e i validatori, secondo i criteri di consenso specifici approfonditi nel Capitolo 1, verificano che essa sia legittima e conforme alle regole della blockchain, esprimendo un'approvazione o meno. Una volta che il blocco contenente la transazione è stato accettato dalla rete, la transazione risulta, a tutti gli effetti, definitiva e immutabile. Tutti i successivi trasferimenti di proprietà, dunque, vengono registrati in modo simile; così facendo, per ogni nodo attraversato dal flusso del prodotto e per ogni operazione di compravendita, gli estremi della transazione saranno impacchettati in un blocco, provvedendo a creare una cronologia facilmente verificabile di tutti i movimenti lungo la catena di fornitura e garantendo, al contempo, l'autenticità e l'integrità delle informazioni.

Oltre a ciò, l'utilizzo delle blockchain permette un monitoraggio puntuale della qualità di un prodotto mentre esso è in transito all'interno della catena di fornitura. Registrando, di volta in volta, chi tiene in custodia un prodotto in un certo lasso di tempo e sotto quali condizioni, uno stakeholder può rilevare, tra le altre cose, se un articolo è stato collocato in un posto non idoneo o è rimasto in una certa sede per troppo tempo, grazie all'analisi dei dati sul percorso di trasmissione e la durata di quest'ultimo. Tale approccio, ad esempio, può risultare di estrema importanza nel settore agro-alimentare, specialmente per prodotti congelati che non possono essere mantenuti a temperatura ambiente.

Nell'ottica del tracciamento finalizzato al controllo qualità, inoltre, diverse soluzioni proposte integrano le blockchain con sistemi IoT. Le indicazioni relative ai processi eseguiti su un prodotto, prelevate tramite QR Code o tag RFID, piuttosto che i cambiamenti delle condizioni ambientali rilevati tramite sensori applicati sui prodotti, costituiscono, infatti, preziose informazioni che possono essere "cristallizzate" sulla blockchain. Dopodiché, dal punto di vista dello sviluppo software, la gestione e la governance automatizzata di tali dati è legata principalmente alla programmazione degli smart contract.

A tal proposito, Azzi *et al.* [2019] riporta, tra gli altri, l'esempio della start-up *Ambrous Network* che definisce due smart contract, uno per i requisiti e uno per le misure. Lo smart contract dei requisiti delinea gli standard di qualità che uno specifico prodotto deve mantenere fino alla consegna. Lo smart contract delle misure, invece, memorizza gli attributi raccolti per un determinato lotto in un punto specifico della catena di fornitura. Dopodiché, la lista di *statement* definiti nello smart contract dei requisiti viene confrontata con quanto memorizzato nello smart contract delle misure e, in caso di sovrapposizione, si è in grado di assicurare la sicurezza e la bontà del prodotto.

Ovviamente, in risposta, i consumatori risulteranno più propensi all'acquisto di marche che, sfruttando soluzioni *blockchain-based*, immettono sul mercato merce tale per cui è possibile provare l'alta qualità di tutti i processi produttivi sottostanti.

Tenere traccia in maniera trasparente di tutti i passaggi di mano di un prodotto, infine, può tornare utile per semplificare il richiamo di un lotto in caso di emergenza, per risolvere, in maniera imparziale, eventuali conflitti legati alla capacità di dimostrare la proprietà di un determinato bene (ad esempio, in caso di furto o smarrimento), così come per risalire ai responsabili di danni o deterioramento sul prodotto stesso.

In Figura 2.1 è riportato uno scenario tipico che schematizza le interazioni tra i diversi attori di una supply-chain per mezzo della blockchain.

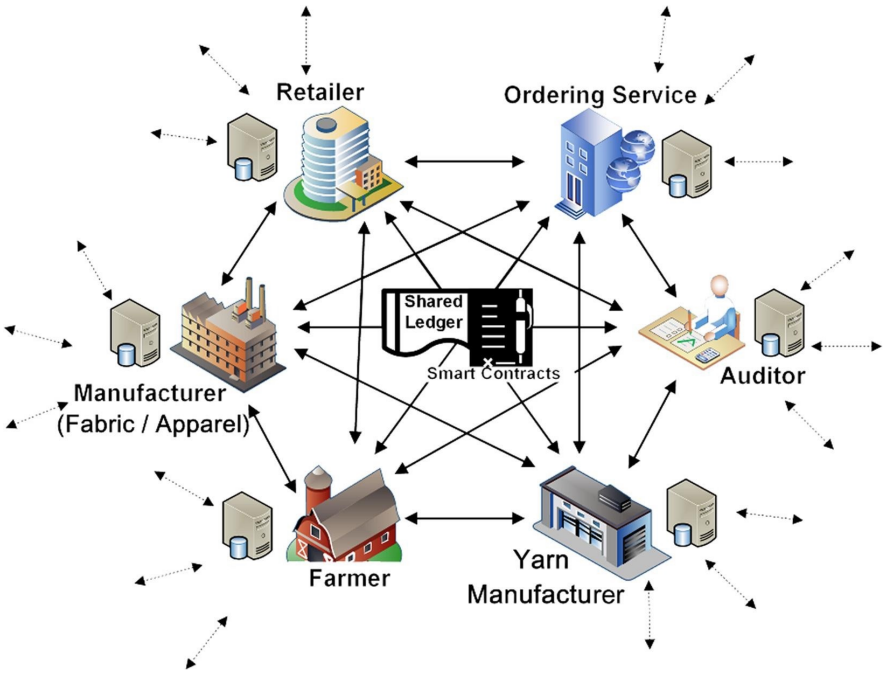

Figura 2.1: Struttura di una catena di approvvigionamento che si interfaccia con la blockchain

2.2.3 Blockchain ed anti-contraffazione: certificazioni di autenticità ed NFT

Una delle maggiori sfide che ci si propone di intraprendere tramite l'utilizzo delle blockchain, nell'ambito del tracciamento, è la lotta alla contraffazione.

Le motivazioni che hanno portato all'ampia circolazione di merce contraffatta sul mercato sono da collegare ad una governance inefficace e all'uso improprio delle pratiche commerciali in contesti poco trasparenti. Oltre a ciò, un notevole impulso alla problematica è dato dalla diffusione dei siti di e-commerce che, nonostante la comprovata praticità, costituiscono, spesso, terreno fertile per frodi e per il transito di repliche non autorizzate o imitazioni, a causa dell'impossibilità di visionare fisicamente un prodotto prima dell'acquisto.

Non da meno, il mercato del falso si macchia, frequentemente, di attività illecite, quali la violazione dei diritti umani sul lavoro, lo sfruttamento minorile, etc.

Le blockchain, dunque, essendo fondate sul principio della trasparenza, costituiscono un'infrastruttura molto promettente per arginare la contraffazione; tramite tale tecnologia, come analizzato nelle sezioni precedenti, è possibile divulgare in maniera affidabile informazioni sull'origine, la proprietà e la qualità di un prodotto, requisiti indispensabili per far luce sull'autenticità.

Mediante le blockchain, infatti, i consumatori possono accedere in prima persona anche ad informazioni che, nelle soluzioni attuali più comuni per il tracciamento, restano spesso nell'ombra o sono filtrate da soggetti intermediari che si occupano dell'ultima parte della supply-chain fino alla vendita al pubblico.

Poter verificare il ciclo di vita di un prodotto nella sua totalità facendo affidamento sulle blockchain, dunque, permette di ridurre la probabilità che gli utenti, in maniera inconsapevole, acquistino prodotti falsi. I consumatori, infatti, possono riporre fiducia nei confronti della blockchain in quanto le informazioni registrate su di essa sono uniche e non possono essere alterate nè duplicate da singoli individui. Seguendo un approccio decentralizzato, per-

tanto, una piattaforma per gli acquisti può fornire, a tutti gli effetti, un certificato depositato sulla blockchain che, rendendo disponibile l'intera storia di un prodotto, è volto a provarne l'autenticità.

La quasi totalità degli strumenti odierni per il tracciamento dei prodotti, in ogni caso, non è in grado di combinare tra loro la resistenza alla contraffazione e la decentralizzazione nella gestione della catena di fornitura. Ciò, secondo Wang *et al.* [2022], è dovuto alla disconnessione e alll'inconsistenza che può venirsi a creare tra i record dei dati e i prodotti fisici. A tal proposito, diverse soluzioni sperimentate, tra cui la stessa proposta da Wang *et al.* [2022], prevedono sistemi di gestione delle informazioni basati su blockchain che combinano, in maniera univoca, l'autenticazione dei record dei dati e l'autenticazione a livello fisico tramite tag RFID.

Sempre relativamente al filone dell'integrazione IoT, recentemente è stato lanciato un progetto pilota da parte dell'azienda Parmigiano Reggiano che prevede l'applicazione di *p-Chip* (piccoli microchip delle dimensioni di un granello di sale) nell'etichetta di caseina di 100.000 forme di formaggio, al fine di garantirne l'autenticità. Tali p-Chip, che per ragioni di sicurezza sono edibili, una volta scansionati, consentono ai consumatori di scoprire se il prodotto in questione è originale o si tratta di un'imitazione. I micro-transponder p-Chip utilizzati, infatti, costituiscono una cosiddetta "ancora crittografica", cioè un identificatore univoco (al pari di un'impronta digitale) inserito nel prodotto, senza la possibilità di rimozione o modifica, e collegato ad una blockchain che assicura l'immmutabilità del dato. Le forme di Parmigiano Reggiano coinvolte nel progetto, alla luce di ciò, possono raccontare, dall'inizio alla fine, la propria unica storia, dall'alimentazione delle mucche, al latte, alla produzione, al processo di stagionatura e a tutti gli altri passaggi necessari, sino all'ispezione qualitativa finale, nell'interesse dei produttori, di tutta la catena distributiva e del consumatore finale.

Uno degli obiettivi raggiungibili con l'ausilio delle blockchain, dunque, consiste proprio nel creare una "carta d'identità digitale" di un prodotto che ne garantisca l'autenticità. A tal fine, assumono un ruolo fondamentale i token non fungibili, o NFT.

Gli NFT, difatti, grazie alla loro carattersitiche intrinseche di unicità, non riproducibilità e programmabilità, consentono la creazione, all'interno di una piattaforma basata su blockchain, dei cosiddetti "digital twin" (gemelli digitali), ossia asset virtuali che diventano la controparte di beni fisici. Per raggiungere lo scopo, un ID univoco viene assegnato a ciascun NFT all'atto della creazione, in modo tale che nessuno di essi possa essere sostituito direttamente da un altro della stessa natura.

Sfruttando le proprietà di immutabilità e trasparenza delle blockchain, pertanto, gli NFT possono essere conservati e fatti circolare in modo sicuro tra i partecipanti di una supply-chain, fornendo una possibile risposta al problema della contraffazione. Nei metadati associati a token di questo tipo, difatti, possono essere immagazzinate le indicazioni sulla provenienza di un prodotto, gli standard di qualità soddisfatti, i materiali utilizzati e la sostenibilità ambientale, tutte informazioni utili per attestarne l'originalità. Tali dati, di conseguenza, verranno notarizzati sulla blockchain a seguito del *minting* dell'NFT.

La soluzione proposta da Philips e Wicaksana [2022], ad esempio, è basata sulla rete blockchain *Hyperledger Fabric* e prevede l'utilizzo di QR Code stampati o integrati nei prodotti fisici come strumenti a basso costo per l'identificazione e la verifica di autenticità. Dietro le quinte, tali QR code rimandano agli NFT creati dal produttore a lavorazione ultimata, ricalcando i corrispettivi prodotti fisici; il sistema, dopodiché, è in grado di trasferire i token non fungibili tra i diversi attori che intendono trasmettere un certificato di autenticità.

Gli smart contract, in questo contesto, offrono la possibilità di codificare regole per la gestione degli NFT adatte alle proprie esigenze. Ad esempio, se un produttore e un fornitore raggiungono un accordo sulla fornitura delle materie prime tramite uno smart contract, oppure un produttore e un rivenditore stabiliscono una collaborazione espressa tramite uno

smart contract, i consumatori saranno in grado di rintracciare la storia della circolazione dell'NFT associato al prodotto in qualsiasi momento, beneficiando, così, della tracciabilità e di una garanzia crittografica di unicità del prodotto.

Tali strategie, dunque, hanno il fine ultimo di ripristinare livelli di trasparenza adeguati in un ecosistema, come quello delle supply-chain, in cui diverse parti, spesso non fidate, hanno l'esigenza di interagire reciprocamente e manipolare, in momenti diversi, uno stesso asset.

2.2.4 Tipologie di blockchain adottate per il tracciamento

L'utilizzo di NFT e smart contract ai fini del tracciamento lascia già intuire che non tutte le reti blockchain possono essere adottate in maniera efficace nello scenario in questione per il semplice fatto che non tutte supportano tali feature.

Tralasciando le specifiche istanze, ad ogni modo, si intende qui chiarire come le più diffuse architetture blockchain rispondono a diverse esigenze e a diversi scenari reali.

Le blockchain pubbliche, essendo totalmente aperte e prive di vincoli di ingaggio, offrono massima trasparenza perché chiunque può esaminare cosa succede nella rete, motivo per cui sono adatte per applicazioni di tracciamento in cui la visibilità collettiva dei dati è cruciale.

L'elevato numero di nodi fa si che risulti decisamente difficile ottenere la maggioranza per compromettere un dato memorizzato. Di contro, però, le prestazioni possono non essere ottimali, soprattutto in termini di scalabilità, e le transazioni prevedono dei costi che devono essere attentamente valutati e confrontati con quelli delle supply-chain tradizionali. In aggiunta, come verrà chiarito nella Sezione 2.4.1, il vantaggio di non avere alcuna restrizione sulla partecipazione, nel caso specifico del tracciamento, potrebbe rilevarsi un ostacolo per autenticare il soggetto che deposita le informazioni circa un prodotto.

Le blockchain private, d'altro canto, sono controllate e mantenute da un gruppo più ristretto di entità e, per questo, si prestano a tutti i contesti consorziali in cui sono presenti partner che si conoscono reciprocamente e possono stabilire un certo grado di fiducia, ad esempio in una supply-chain chiusa o tra singoli dipartimenti di uno stesso gruppo.

Le blockchain permissioned, in cui è necessaria l'autorizzazione per partecipare alla rete e condividere dati, si collocano meglio in scenari in cui è necessario garantire la conformità normativa e avere un maggiore controllo dell'accesso, come nel settore finanziario.

Dato che le transazioni non prevedono costi e le prestazioni sono solitamente migliori a causa di un numero non elevato di nodi, le infrastrutture private permissioned sono tra quelle maggiormente prese in considerazione nelle soluzioni ad oggi proposte. In contesti di questo tipo, la chiave privata di ciascun utente determina il tipo di accesso e il raggio d'azione, permettendo, dunque, di stabilire le regole della rete e le policy per l'interazione tra i diversi stakeholder.

Alla luce di ciò, i grandi brand o i colossi dell'*e-commerce* potrebbero mettere in piedi dei sistemi per il tracciamento e la verifica dell'autenticità dei prodotti trattati fondati su blockchain private permissioned, dove l'accesso viene esteso a fornitori, produttori, distributori, trasportatori, rivenditori, organizzazioni di consulenza ed ulteriori collaboratori ufficiali, nonché, dietro apposita verifica, ai consumatori, così da instituire un'interazione più diretta ed efficace tra le parti. Ad esempio, Amazon ha collaborato con Nestlé per sviluppare una blockchain volta a diffondere le informazioni sui luoghi in cui i semi di cacao vengono piantati e tostati, oppure su come e quando sono cresciuti.

Le blockchain permissionless, infine, non richiedono autorizzazioni di partecipazione, per cui sono adatte in applicazioni dove la decentralizzazione e la resistenza alla censura sono prioritarie, come progetti di tracciamento di prodotti in aree con governi autoritari.

Le scelte riguardo l'utilizzo di un'architettura piuttosto che un'altra, come intuito, devono tenere in considerazione la natura e il grado di interazione delle parti coinvolte nella catena

di fornitura, l'infrastruttura tecnologica già esistente e i requisiti di privacy, efficienza e scalabilità da soddisfare.

2.3 Caso di studio: sistema per il tracciamento di farmaci tramite blockchain

Dato l'oggettivo impatto dovuto al loro volume d'affari, si distinguono, in particolar modo, tre settori nei quali il tracciamento della supply-chain è indispensabile per contrastare il fenomeno della contraffazione, ossia il mercato della moda e dei beni di lusso, la filiera alimentare e l'industria farmaceutica.

Al fine di approfondire le modalità attraverso cui la tecnologia blockchain possa essere integrata concretamente in una supply-chain, verrà preso in considerazione, come caso di studio, il sistema *NFT-IoT Pharma chain* ideato da Turki *et al.* [2022] ed incentrato sul tracciamento di farmaci.

2.3.1 Scenari di utilizzo ed approccio proposto

L'affidabilità della catena di fornitura dei medicinali, data la criticità e delicatezza dei prodotti coinvolti, risulta fondamentale per la salute pubblica.

Negli ultimi anni, infatti, il numero di farmaci contraffatti è cresciuto drasticamente, il che ha portato a migliaia di vittime da avvelenamento o cure errate. L'*Health Research Funding Organization*, a tal proposito, ha stimato che il 30% dei farmaci venduti nei paesi in via di sviluppo è soggetto a contraffazione.

A complicare la situazione, le supply-chain tipiche dell'industria farmaceutica coinvolgono diverse parti che hanno interessi eterogenei e che, spesso, sono avverse a condividere reciprocamente le informazioni sulla tracciabilità, mantenute in banche dati non interconnesse.

La rete degli stakeholder presa in considerazione, nello specifico, è costituita da diverse entità indipendenti, quali produttori, distributori all'ingrosso, farmacie, ospedali e PBM[1].

Il flusso delle operazioni seguito da tale supply-chain è sequenziale. Un produttore, previa autorizzazione di un'agenzia di regolamentazione, riceve gli ordini ed avvia la lavorazione dei farmaci; dopodiché, i farmaci vengono confezionati in lotti e spediti dai grossisti alle farmacie presenti sul territorio o agli ospedali.

La complessità della catena, dunque, può portare all'introduzione, nel sistema, di farmaci che esulano dagli standard di lavorazione o di imballaggio, per cause volontarie o meno. Oltre a ciò, possibili problemi durante il trasporto, dovuti a condizioni ambientali straordinarie in termini di temperatura, vibrazioni, pressione ed umidità, possono rendere i medicinali inefficaci, o addirittura pericolosi per i pazienti.

A tal proposito, la soluzione proposta prevede l'integrazione delle tecnologie IoT e blockchain. Tramite la blockchain, dunque, tutte le transazioni che coinvolgono i farmaci sono marcate temporalmente e memorizzate in maniera immutabile, in modo tale che ciascuna parte coinvolta possa verificare lo status di un ordine e la sua ubicazione in qualsiasi momento. In aggiunta a ciò, l'IoT permette a dispositivi connessi ad Internet di trasferire dati alla rete blockchain, potenziando ed automatizzando il tracciamento delle condizioni di produzione e trasporto dei farmaci.

Il progetto *NFT-IoT Pharma chain* prevede un'architettura modulare organizzata in diversi layer, ciascuno dei quali offre i servizi di tracciabilità di cui ogni singolo stakeholder usufruirà. Gli attori modellati e le responsabilità ad essi assegnate sono riassunte di seguito:

[1] Pharmacy Benefit Managers: si tratta di enti presenti nel sistema sanitario statunitense che hanno il ruolo di intermediario tra fornitori di assicurazioni e produttori di farmaci.

- *HADPI (Higher Authority of Drugs and Pharmaceutical Industry)*: è l'autorità che gestisce il sistema proposto, incaricata di monitorare l'intero ciclo di vita di un farmaco, registrare gli attori e verificare la conformità delle licenze assegnate ad essi.
- *Produttore*: il suo ruolo principale è quello di lavorare alla creazione dei medicinali rispettando le regole di sicurezza sanitaria.
- *Distributore*: una volta ricevuta una richiesta di distribuzione dal produttore, sceglie il veicolo destinato alla spedizione, rendendone pubblico il certificato di compliance. Al termine della spedizione, in base allo stato del lotto consegnato, la qualità del servizio offerto viene valutata dall'HADPI che provvede ad assegnare premi o penalità.
- *IoT del Veicolo*: il veicolo selezionato per trasportare il lotto di farmaci a destinazione interagisce con la blockchain per richiedere i parametri di riferimento di un medicinale ed aggiornare lo status del lotto quando viene superata una soglia di temperatura.
- *Ospedale/farmacia*: prima di accettare un lotto e somministrare i farmaci ai pazienti, tale attore è incaricato di verificarne le condizioni.

  L'infrastruttura alla base del sistema in questione è schematizzata in Figura 2.2.

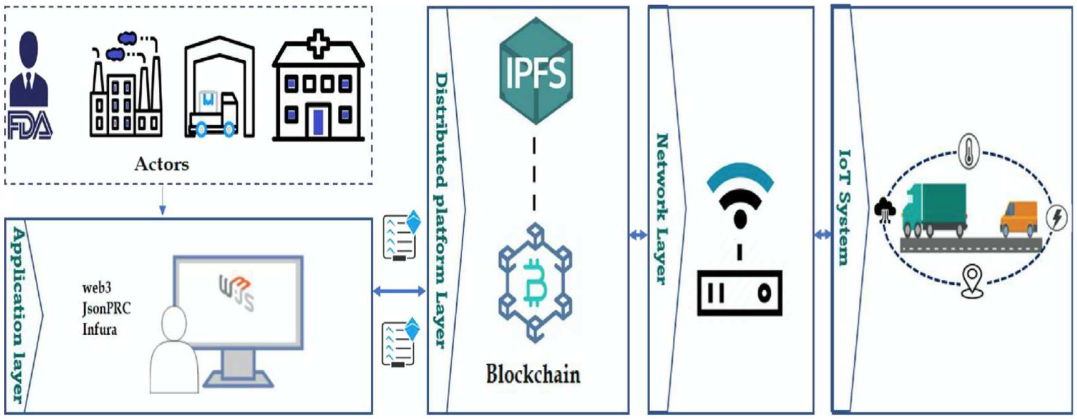

Figura 2.2: Componenti principali del sistema NFT-IoT Pharma chain. Nello specifico, è possibile individuare un application layer, un layer per la piattaforma distribuita, un network layer e un sistema IoT

2.3.2 Uso degli NFT

Affinché la tracciabilità risulti quanto più possibile priva di rischi e trasparente, il sistema proposto mette in pratica il concetto di *tokenizzazione* ed utilizza tre diversi NFT creati attraverso altrettanti smart contract.

Per tenere traccia di tutti i passaggi di proprietà che coinvolgono un lotto, una prima tipologia di NFT è associata a tale asset. NFT di questo tipo, dunque, permettono di avere una visione d'insieme di tutti i dettagli di un lotto, quali il proprietario attuale, la posizione, le condizioni di spedizione, le certificazioni ottenute, etc. Le informazioni su un lotto, inevitabilmente, vengono aggiornate nel corso dei processi eseguiti dalla catena di approvvigionamento ma i dettagli degli step precedenti rimangono, comunque, memorizzati sulla blockchain. Ciò permette ad un paziente o ad un ospedale di ottenere una storia del medicinale *end-to-end*. Oltre a ciò, se un farmaco di un lotto specifico dovesse provocare effetti collaterali in qualche paziente, è possibile localizzare e richiamare tutti gli altri componenti dello stesso lotto.

Altri NFT, diversi dai precedenti, vengono creati per rappresentare gli ordini; tale scelta deriva dalla possibilità di spedire insieme, in uno stesso ordine, più lotti. Il token relativo ad un ordine contiene parametri specifici per registrare il processo di spedizione, come, ad esempio, il veicolo utilizzato, la destinazione, la validità, etc.

Infine, la terza tipologia di NFT è stata creata per identificare i veicoli; ogni distributore, infatti, ha più veicoli a disposizione ed ognuno di essi deve essere certificato ed autorizzato alla spedizione dalla HADPI. Al fine di effettuare transazioni sulla blockchain, dunque, ad ogni veicolo è associato un address.

2.3.3 Implementazione ed integrazione IoT

Il software sviluppato per realizzare il sistema in esame consiste in un'applicazione decentralizzata basata su una rete Ethereum privata con permessi, al fine di enfatizzare la privacy e l'efficienza.

Per poter gestire i permessi, il layer dedicato alla piattaforma distribuita prevede un modulo di controllo degli accessi di tipo RBAC (Role Based Access Control), realizzato tramite la libreria OpenZeppelin; appositi smart contract forniti da tale libreria, infatti, consentono la connessione o l'accesso al sistema esclusivamente ad utenti o dispositivi autorizzati a seguito della verifica delle chiavi crittografiche.

La parte di Back-end dell'applicazione utilizza il runtime system Node.js e gestisce le chiamate alle funzionalità degli smart contract tramite la libreria Web3. In aggiunta, per quanto riguarda la memorizzazione di dati *off-chain*, viene sfruttata la piattaforma IPFS[2].

Per ciò che concerne il Front-end, invece, l'interfaccia utente è stata creata per mezzo del framework Vue.js, basato su JavaScript.

Componente rilevante del progetto, come intuito, è il sistema IoT integrato nei veicoli dei distributori ed utilizzato per monitorare determinate condizioni ambientali a cui un lotto è esposto, quali la temperatura e l'umidità. Il sistema IoT è basato su un Raspberry Pi 3B e comprende, inoltre, il sensore di temperatura e umidità DHT11 e il GPS NEO-6, per la localizzazione del veicolo.

Il sistema prevede che parte dei calcoli richiesti in termini di monitoraggio venga svolta off-chain, sulla piattaforma raspberry, al fine di ridurre il numero di transazioni da effettuare sulla blockchain; a tal proposito, su quest'ultima vengono inviati solamente i valori di temperatura che oltrepassano le soglie imposte.

Una volta acquisiti, infine, i dati del sistema IoT sono trasmessi dal layer di rete alla blockchain utilizzando la comunicazione 4G e la libreria Web3.

2.4 Sviluppi futuri

Le blockchain applicate al tracciamento dei prodotti costituiscono un settore in forte espansione e decisamente innovativo; ciò, unitamente ad un diffuso interesse solo di recente, fa si che siano ancora diversi i quesiti e i dubbi circa le potenzialità inespresse e le possibili strade da percorrere per migliorare ulteriormente i processi di gestione delle informazioni nelle supply-chain tramite tale tecnologia.

2.4.1 Sfide rimanenti e possibili miglioramenti

Se, da un lato, è vero che le blockchain possono offrire garanzie molto più tangibili in termini di trasparenza della filiera produttiva e permettono, in linea di principio, di contrastare la contraffazione, d'altra parte non bisogna dimenticare che tale tecnologia non è

[2]InterPlanetary File System

nata specificatamente per gli obiettivi citati; dunque, nel processo di adattamento, possono restare scoperte alcune questioni.

Una prima sfida fondamentale consiste nel trovare delle modalità efficaci per rendere attendibile l'origine del dato. Difatti, a meno che la blockchain utilizzata ai fini del tracciamento non sia istituita e gestita, attraverso permessi di accesso, da un brand autorevole o da un consorzio di enti universalmente riconosciuti come affidabili, non si può essere certi della bontà delle informazioni riguardanti la prima parte del ciclo di vita di un prodotto, nè dell'identità di chi trascrive tali informazioni. Di conseguenza, se i dati non sono affidabili già all'origine, l'esito dell'intero processo non potrà risultare soddisfacente, in quanto la catena risulta spezzata nel suo primo anello.

Per poter pensare di utilizzare reti blockchain senza particolari vincoli di accesso, giovando a pieno dei benefici di una reale decentralizzazione, dunque, bisogna trovare delle soluzioni al problema che coinvolge tutti quei dati che nascono all'esterno, nel mondo reale, e che devono essere importati sulla blockchain al fine di apporvi un certificato di originalità.

Altra considerazione da fare, inoltre, riguarda l'aspetto economico. Dato che il processo di tracciamento lascia ampio spazio all'utilizzo di funzionalità e accordi codificati tramite gli smart contract, deve essere svolta anche una valutazione circa i consumi di gas, e dunque i corrispettivi costi da sostenere in termini di commissioni per l'esecuzione di programmi sulla blockchain. A tal proposito, va sicuramente effettuato un maggiore sforzo di design volto a minimizzare la quantità di informazioni da notarizzare e, di conseguenza, l'uso di risorse on-chain. Ciò, inoltre, risulta particolarmente importante considerando che un contesto reale di tracciamento prevede un elevato carico di transazioni giornaliere che devono essere gestite e processate in maniera efficiente con tempi accettabili, al fine di ottenere un monitoraggio quanto più possibile in tempo reale.

Non meno importante, l'enorme flusso di dati che deriva dalla tracciabilità dei prodotti potrebbe provocare un problema di ridondanza nel registro, che acquisirebbe in poco tempo dimensioni molto elevate.

Un possibile sviluppo per l'impiego delle blockchain nel contesto della tracciabilità, secondo Trofa [2021], chiama in causa la sostenibilità ambientale. Nello specifico, bisognerebbe essere in grado di discriminare il prodotto in sé, che può essere ecologicamente sostenibile, a filiera corta e supportato da diversi standard qualitativi, dall'intero impianto necessario per renderlo disponibile sul mercato, che, invece, può presentare impatti più pesanti.

Altro miglioramento auspicabile deriva dalla constatazione che le soluzioni ad oggi teorizzate, e talvolta sperimentate, si concentrano prevalentemete sul percorso del prodotto dal reperimento delle materie prime fino alla vendita al pubblico e, solo di rado, interessano la cosiddetta *post supply-chain*, una volta che il prodotto passa in mano al consumatore. Utilizzare le blockchain anche in questo tratto pressoché inesplorato, infatti, permetterebbe non solo alle aziende di sviluppare servizi post-vendita di assistenza e manutenzione molto più oculati, ma contribuirebbe ulteriormente a limitare la diffusione di prodotti contraffatti, che molto spesso vengono inseriti direttamente nei mercati di seconda mano.

In conclusione, le problematiche delineate, più che veri e propori "limiti", possono essere viste come reali opportunità perché indicano in maniera piuttosto mirata la direzione da intraprendere nel processo di avanzamento tecnologico. Difatti, la realizzazione del sistema oggetto della tesi, di cui si inzierà a discutere nel Capitolo 3, si basa proprio su una possibile applicazione delle blockchain per tracciare, in particolare, la rivendita di seconda mano.

CAPITOLO 3

Specifica e analisi dei requisiti

Il capitolo in questione provvederà a presentare il sistema oggetto di sviluppo, destinato alla certificazione dell'autenticità di prodotti di lusso.

A tal proposito, inizialmente, verrà fornita una panoramica relativa alle caratteristiche e alle dinamiche che contraddistinguono il mercato del lusso, analizzando, al tempo stesso, le maggiori problematiche da affrontare e i rischi a cui esso è esposto.

Avendo a disposizione un quadro più chiaro del contesto d'applicazione, verranno illustrate le specifiche a cui il software in questione dovrà attenersi e gli aspetti su cui dovrà focalizzarsi.

Infine, ampio spazio sarà riservato alle considerazioni emerse dall'analisi dei requisiti del sistema, discutendo, di pari passo, la documentazione prodotta durante i diversi step di modellazione svolti nella fase embrionale del lavoro.

3.1 Il settore dei beni di lusso

Prima di addentrarsi nella descrizione del sistema in esame, è necessario illustrare il contesto in cui esso andrà ad operare, ossia il mercato del lusso, così da mettere in luce i fattori scatenanti e le esigenze da soddisfare.

3.1.1 Panoramica generale

Nonostante gli strascichi della pandemia da Covid-19, l'aumento dei costi energetici, l'inflazione crescente e le tensioni geopolitiche scaturite dal conflitto bellico nell'Europa orientale, il report *Luxury Goods Worlwide Market Monitor* (Altagamma *et al.* [2022]), mostra che, nel 2022, l'industria globale dei beni di lusso ha raggiunto un valore di mercato di circa 1400 miliardi di euro, il 21% in più rispetto all'anno precedente, stabilendo un nuovo record storico. In aggiunta, come ulteriore testimonianza delle ottime performance del settore, è emerso che il 95% dei brand ha osservato una crescita positiva nel 2022.

Ciò che si evince, dunque, è che il mercato del lusso, a differenza di altri settori, risente meno dell'incertezza delle condizioni economiche e di consumo e, per giunta, è in grado di superare brillantemente sporadiche flessioni, confermando grande solidità e resilienza.

Facendo un passo indietro, però, è doverso chiarire cosa si intende per beni di lusso.

Da un punto di vista prettamente economico, in accordo con la *legge di Engel*, un bene di lusso è un prodotto la cui domanda aumenta più che proporzionalmente rispetto all'incremento del reddito.

Dopodiché, per tracciare un confine tra beni di lusso e beni normali, è possibile far riferimento anche a nozioni di marketing ed aspetti sociologici.

In quest'ottica, i prodotti di lusso si contraddistinguono, tipicamente, per l'eccellente qualità, il prezzo elevato, il carattere fortemente esclusivo e la componente emozionale che accompagna l'atto di acquisto o l'esposizione al pubblico. Tali caratteristiche, in sostanza, sono proprie di tutti quei prodotti in grado di attribuire uno status di prestigio al soggetto che ne entra in possesso o ne usufruisce.

Difatti, il valore dei *luxury good* non dipende tanto dall'aspetto funzionale; nella gran parte dei casi, beni di questo tipo hanno un carattere superfluo perché, per principio, non rispondono a bisogni primari. Il valore di un prodotto di alta gamma, piuttosto, è influenzato dalla sfera emotiva, dall'aspirazione e dal senso di gratificazione connessi al suo possesso.

Ad ogni modo, l'estrema qualità di un bene di lusso, dovuta all'impiego di materie prime ricercate e alla cura dei processi di lavorazione eseguiti, contribuisce a garantirne l'affidabilittà e la durevolezza nel tempo, trasmettendo, di conseguenza, fiducia al consumatore.

La qualità percepita e l'idea di longevità connessa ad un bene di lusso, inoltre, sono alcuni dei principali elementi che ne legittimano il prezzo elevato rispetto a prodotti che svolgono funzioni analoghe.

In aggiunta, una grande influenza sul prezzo è imputabile alle caratteristiche di unicità e rarità, cruciali per l'industria del lusso; maggiore è l'inaccessibilità di un bene, più alti risultano il desiderio da esso suscitato e la cifra che un consumatore è disposto a sborsare.

Le cause della limitata reperibilità dei beni di lusso sono da rintracciare, per lo più, nella rarità delle materie prime utilizzate e nella specificità delle competenze richieste dai processi di lavorazione, entrambi fattori che escludono le tipiche dinamiche di produzione e consumo di massa. Tale natura non ordinaria, inoltre, si riflette anche dal punto di vista della distribuzione; a differenza dei rivenditori comuni, che offrono ampia scelta di prodotti per diversi livelli qualitativi, i *luxury store*, solitamente, offrono una gamma di prodotti più ristretta ma altamente selezionata ed, inoltre, puntano sempre di più su modalità esperienziali e personalizzate di relazione con i clienti. L'elevata qualità, dunque, è da intendersi anche in relazione al servizio offerto nella fase finale della supply chain; dettagli come il packaging e le confezioni regalo, ad esempio, fanno, spesso, la differenza tra il mondo del *fast fashion* e quello dei beni di lusso. Il cliente che acquista un prodotto di marca, spesso ritenuto un investimento, deve essere deliziato dall'intero percorso di acquisto.

Relativamente alla caratterizzazione di un bene di lusso, in ultima analisi, un ruolo rilevante è svolto dalla componente estetica, nel senso che il "consumo" del prodotto costituisce, a tutti gli effetti, un'esperienza multi-sensoriale, analogamente a quanto avviene con un'opera artistica; proprio per tale motivo, spesso, il giudizio dei consumatori nei confronti dei beni in questione risulta significativamente influenzato dal gusto dei creatori e dai canoni da loro proposti.

Scendendo nel dettaglio del report relativo ai consumi dei beni di lusso nell'anno 2022, innanzitutto, è possibile constatare che la base dei consumatori ultra-abbienti si conferma solida; in aggiunta, si riscontra un desiderio di elevazione che sta spingendo anche clienti più ordinari a cercare la qualità anziché la quantità.

In secondo luogo, va sottolineato che tutte le maggiori categorie del lusso hanno recuperato, e talvolta superato, i livelli pre-Covid. Nello specifico, la fetta maggiore della crescita, nel 2022, è occupata dai beni di lusso personali (moda, gioielleria, accessori, cosmetica, etc.) i quali hanno fatto registrare un +22% rispetto all'anno precedente. Proseguendo, si sono distinte anche le categorie delle auto di lusso, nonostante le difficoltà nell'approvvigionamento di microchip, dei jet e yacht privati, della *fine art* e dell'arredamento di design. Buone performance, infine, sono state ottenute da ristoranti stellati, vini e liquori pregiati ed hotel di lusso. Il grafico in Figura 3.1 mostra il valore delle vendite associato alle principali categorie del lusso nell'anno 2022.

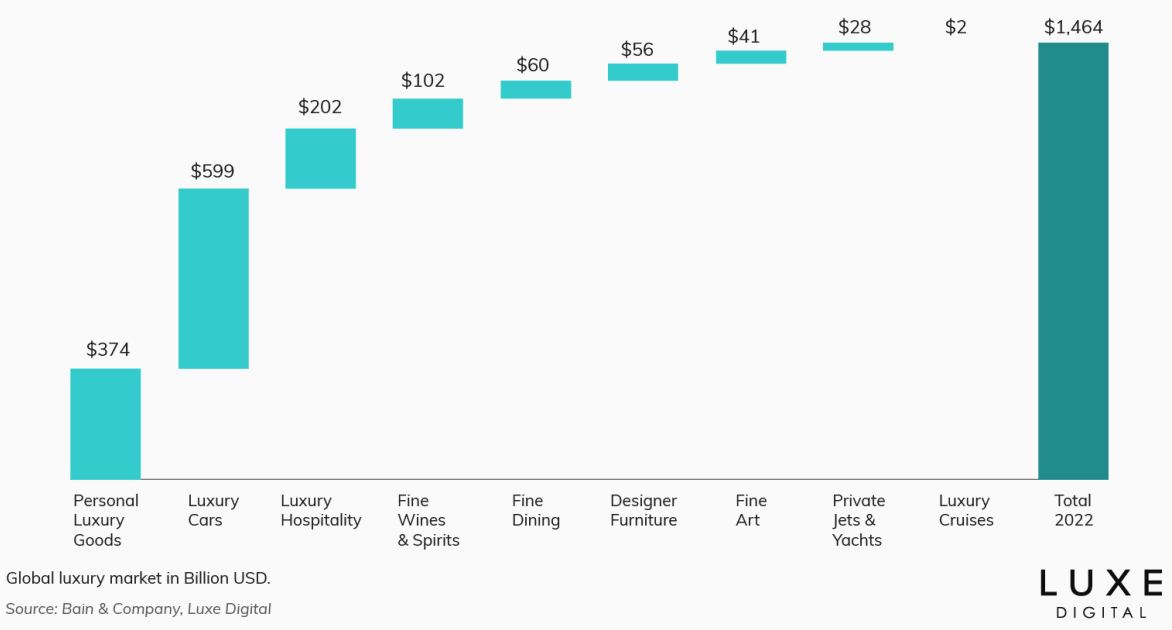

Figura 3.1: Fatturato relativo alle singole categorie di lusso nell'anno 2022. I valori sono espressi in miliardi di dollari.

A fianco ai prodotti di lusso tradizionali, per giunta, una grande prospettiva di crescita è prevista per gli asset digitali e il mondo virtuale legato al metaverso; secondo le previsioni, infatti, entro la fine del 2030 tale categoria rappresenterà il 5-10% del mercato del lusso.

3.1.2 Minacce nel mercato del lusso: impatto della contraffazione

Dato l'enorme giro d'affari connesso al settore del lusso, numerosi sono i tentativi di frode che coinvolgono tale comparto economico e, dati alla mano, ancora molto si può fare dal punto di vista della prevenzione.

Nello specifico, la questione probabilmente più critica che logora il mercato del lusso, sia a livello nazionale che internazionale, consiste nella contraffazione, fenomeno che si concretizza attraverso la creazione e la vendita di imitazioni di prodotti di marca.

Più nel dettaglio, è possibile distinguere tra prodotti contraffatti non ingannevoli (*non-deceptive*) ed ingannevoli (*deceptive*). I primi fanno riferimento a repliche che i consumatori riescono a distinguere dagli originali piuttosto facilmente, a causa di più palesi differenze in termini di qualità, prezzo e punti vendita presso i quali è possibile reperirli.

I prodotti contraffatti ingannevoli, invece, sono in grado di persuadere i consumatori della loro "autenticità". A tal fine, essi sono confezionati, etichettati e venduti fedelmente proprio come i corrispettivi prodotti originali, motivo per cui devono riuscire ad infiltrarsi nelle supply-chain legittime.

In molti casi, infatti, i consumatori del mercato del lusso non sono in grado di distinguere l'autenticità dei prodotti perché non possono ottenere, in maniera diretta e trasparente, dettagli circa le materie prime utilizzate o i processi di produzione e distribuzione coinvolti nel ciclo di vita di un prodotto. Oltre a ciò, le decisioni strategiche di esternalizzare le produzioni, nonostante portino vantaggi immediati in termini di ROI, rendono più difficoltosa la supervisione delle supply-chain da parte dei brand e più semplice l'immissione di repliche apparentemente legittime, a causa della presenza di un numero più elevato di attori dislocati geograficamente.

Secondo le stime effettuate dalle organizzazioni di monitoraggio dei consumi e anti-contraffazione, dei circa 4.5 miliardi di dollari generati, a livello globale, dal commercio di prodotti contraffatti, il 60-70% riguarda esclusivamente il settore del lusso. Ciò è attribuibile, in gran parte, al boom dei siti di *e-commerce*; difatti, i dati rivelano che più del 40% delle vendite di prodotti di lusso falsi avvenga online, dove la facilità di accesso per i consumatori e la possibilità di anonimato offrono ampio spazio a pratiche poco oneste o del tutto illecite.

A complicare lo scenario, accanto ai mercati illegali dove vengono scambiati prodotti contraffatti o rubati, denominati "black market", una sfida per l'industria del lusso è costituita anche dai controversi "gray market", ossia canali di distribuzione non autorizzati, ma non attaccabili dal punto di vista legale, dove vengono venduti prodotti brandizzati.

Ad ogni modo, il rischio di imbattersi in prodotti contraffatti genera, nei consumatori, un problema di fiducia nei confronti dei marketplace online, spingendo diversi brand di lusso, di conseguenza, a non vendere i propri prodotti tramite siti di e-commerce generici, quali, ad esempio, Amazon.

Sempre relativamente ai rischi provenienti dal web, recenti indagini hanno individuato due trend pericolosi, nonché sempre più diffusi, per il mercato del lusso.

In primo luogo, non è raro che alcuni influencer, tramite le piattaforme social, sponsorizzino apertamente imitazioni più economiche di prodotti firmati, condividendo con migliaia di follower gli URL di shop online dedicati, per mezzo di messaggi diretti visibili una sola volta o storie della durata di 24 ore, così da non lasciare traccia.

La seconda minaccia deriva dall'evoluzione della pubblicità fraudolenta. Anziché condurre campagne pubblicitarie globali, relativamente facili da individuare e rimuovere, le organizzazioni criminali dei mercati contraffatti cercano di raggiungere i consumatori sui social attraverso il *micro-targeting*, una forma di pubblicità online basata sulla profilazione degli utenti a partire dall'analisi dei dati personali, sicuramente più difficile da identificare e filtrare per i team che si occupano della protezione dei marchi.

Sulla base dei report rilasciati dagli enti impegnati nella lotta alla contraffazione, la categoria di lusso più colpita risulta quella dei beni personali; vestiti, gioielli, orologi, borse ed accessori firmati, infatti, sono i prodotti che più facilmente possono essere falsificati e spacciati a consumatori non abbastanza informati o, d'altra parte, intenzionati a spendere meno.

Alla luce di ciò, gran parte dei brand che, nel corso degli ultimi decenni, ha accusato ingenti perdite dovute alla contraffazione dei propri prodotti opera nel comparto moda ed include, per citarne alcuni tra i più celebri, Chanel, Givenchy, Louis Vitton, Prada e Gucci.

In aggiunta al danno economico, i marchi del lusso bersagliati dal mercato del falso incorrono in danni d'immagine a causa dei potenziali problemi di salute che repliche di bassa qualità possono arrecare ai consumatori. Oltre a ciò, si è stimato che la riproduzione non autorizzata di prodotti di alta moda, e la relativa commercializzazione, sia responsabile anche della perdita di 2.5 milioni di posti di lavoro in tutto il mondo.

Infine, non meno importante, la contraffazione nell'industria del lusso è spesso sinonimo di un impatto ambientale che infrange le normative a tutela della sostenibilità.

3.2 Introduzione al sistema LuxuryChain

Nonostante i brand di lusso siano costantemente impegnati ad investire nella ricerca e nello sviluppo di tecnologie in grado di limitare il fenomeno della contraffazione, le attuali soluzioni più diffuse per il tracciamento, tra cui QR-Code e tag RFID, non sono in grado di raggiungere sufficienti livelli di sicurezza e, di conseguenza, i risultati sperati.

Alla luce di ciò, il sistema discusso nella presente tesi, al quale viene assegnato il nome *LuxuryChain*, è volto ad offrire garanzia di autenticità di beni di alta gamma.

3.2.1 Scenari e casi d'uso

Il sistema *LuxuryChain* si ripropone di operare nel contesto della compravendita di prodotti di lusso, in particolar modo quelli destinati ad uso personale.

Ad ogni modo, pur avendo individuato un segmento di mercato piuttosto definito, non ci si vuole soffermare su una singola linea di prodotto, né sono stati posti vincoli sulla natura specifica dei beni da trattare; idealmente, dunque, il sistema potrà essere adattato, in base alle esigenze pratiche, a capi di abbigliamento, così come a calzature, gioielli, accessori, cosmetici etc.

Ciò su cui si intende focalizzare l'attenzione, invece, è la possibilità di generare, a livello digitale, una prova inalterabile dell'originalità dei prodotti, ponendosi, di conseguenza, nell'ottica della sicurezza.

Il sistema, dunque, ha l'obiettivo di creare, sotto forma di applicazione web o mobile, un ambiente affidabile, a prova di falsificazione, tramite il quale regolamentare lo scambio di prodotti di lusso tra parti solitamente non fidate, verificando, ad ogni passaggio di mano, le caratteristiche di autenticità e provenienza.

La Figura 3.2 mostra il tipico scenario all'interno del quale avvengono le principali transazioni che coinvolgono beni di lusso. Nello specifico, è messo in evidenza il flusso seguito dal prodotto, utilizzato come base di partenza per l'analisi del sistema sviluppato.

Una volta ultimato e confezionato, il bene passa dal brand di lusso ad uno store, scelto tra i possibili rivenditori autorizzati, in modo che possa essere venduto al pubblico; ad un certo punto della catena, infatti, un cliente effettua l'acquisto del prodotto presso il luxury store e, a meno di eventuale reso, ne diventa ufficialmente il nuovo proprietario.

In un secondo momento, però, è lecito attendersi che il legittimo proprietario decida deliberatamente di cedere il proprio prodotto, perché magari non più utilizzato, e, a tal proposito, può procedere ad una rivendita privata diretta o, in alternativa, può rivolgersi ad un mercato di seconda mano; in ogni caso, interviene un nuovo consumatore che acquista il prodotto usato ed, ancora di più, necessità di garanzie di autenticità. Si immagini, infatti, che le certezze riguardo la bontà del prodotto diminuiscano, progressivamente, spostandosi verso destra nello schema riportato in Figura 3.2.

Infine, la parte conclusiva della catena, che rappresenta la rivendita di seconda mano del prodotto, può ripetersi arbitrariamente in maniera ciclica. Il bene di lusso, dunque, vedrà succedersi tra loro un certo numero di proprietari, ciascuno responsabile delle sue condizioni.

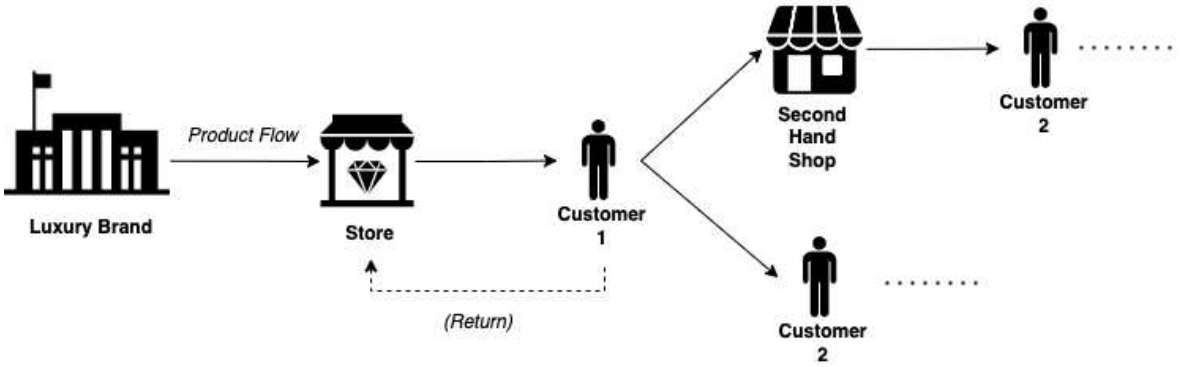

Figura 3.2: Flusso di un prodotto di lusso attraverso le figure che si susseguono alla sua gestione

Lo schema descritto, pur cercando di replicare fedelmete il contesto reale all'interno del quale può essere inserito il sistema *LuxuryChain*, prevede alcune semplificazioni che tracciano i confini entro cui operare.

Innanzitutto, è stata tralasciata volontariamente la sezione della supply-chain dedicata alle fasi iniziali del ciclo di vita di un prodotto; non vengono considerati, infatti, fornitori di materie prime, aziende per la trasformazione dei materiali, lavorazioni intermedie e ditte per il trasporto. Tale scelta, in gran parte, è dovuta alla presenza di altri studi e lavori già focalizzati espressamente sulla tracciabilità di un prodotto durante gli step citati.

L'assunto, in questa sede, è di inaugurare la catena avendo a disposizione un bene già pronto il quale esce dalla casa madre, il brand di lusso, dotato di un qualche dispositivo fisico per il tracciamento (ad esempio, un tag RFID) e di un certificato di autenticità; dopodiché, nel corso del tempo, la proprietà del prodotto passa, di volta in volta, attraverso diverse entità che, se in buona fede, non hanno l'esigenza di apportarvi modifiche.

Un'altra semplificazione insita nello schema di Figura 3.2 prevede la coesistenza di store fisici e online, sia per quanto riguarda la vendita di un prodotto di lusso nuovo, che per il mercato di seconda mano.

Da tale quadro, dunque, emerge la volontà di convogliare le funzionalità del sistema in maniera mirata verso la cosiddetta *post supply-chain*, affinché vengano gestite nel modo più sicuro e trasparente possibile tutte quelle transazioni che mettono di fronte utenti non fidati o non universalmente riconosciuti, tra i quali potrebbero nascondersi potenziali contraffattori.

Una breve analisi delle caratteristiche e dei rischi relativi alle *post supply-chain* è effettuata nella Sezione 3.2.2

3.2.2 Post supply-chain e mercato di seconda mano

Se, da un lato, nel corso degli ultimi decenni, il mondo dell'IoT, ed in particolare la tecnologia RFID, hanno riscosso molta attenzione per il possibile impiego nel rilevamento di beni contraffatti inseriti all'interno delle supply-chain, dall'altro è pur vero che dispositivi di questo tipo perdono di efficacia una volta che i prodotti raggiungono la fine della catena di approvvigionamento e vengono esposti nei negozi. Da questo momento in poi, infatti, l'originalità dei beni di lusso non è più garantita perché i tag RFID, ad esempio, possono essere clonati ed, in generale, il prodotto esce dalla supervisione di enti ufficiali.

Alla luce di ciò, risulta fondamentale fornire ai brand di lusso dei sistemi a prova di contraffazione, in grado di funzionare bene anche quando un prodotto giunge al termine della supply-chain tradizionale e qualora l'integrità delle informazioni dei tag RFID dovesse risultare compromessa. A tale esigenza, infatti, cerca di rispondere il sistema sviluppato.

Tutti i movimenti di un prodotto che si verificano a partire dall'istante in cui esso viene immagazzinato presso un rivenditore, di conseguenza, entrano più specificatamente nella sfera della *post supply-chain*, spesso trascurata dal punto di vista dei rischi.

Innanzitutto, come parte integrante di un'esperienza di acquisto altamente personalizzata in un luxury store, un cliente in procinto di investire somme importanti su un prodotto di alto calibro potrebbe avere l'esigenza di verificare, in prima persona, la storia ad esso collegata, con i dettagli sull'origine, sulla lavorazione eseguita e sull'ecosostenibilità, senza doversi affidare ciecamente all'autorità del mediatore, rappresentato, in questo caso, dallo store stesso. Oltre a ciò, specialmente interfacciandosi con gli shop online, la possibilità di procedere con l'acquisto dei soli prodotti autenticati mediante un apposito sistema di certificazione può aiutare nell'individuare i rivenditori autorizzati ed affiliati con il brand.

Dopo esserne entrato in possesso, inoltre, un consumatore può decidere, a sua volta, di rivendere il prodotto di lusso su Internet o tramite un mercato di seconda mano. Sta di fatto, però, che tali transazioni, spesso, sono tracciate solo parzialmente o del tutto ignorate ed, in ogni caso, non sono sottoposte a solidi processi di verifica di legittimità.

Difatti, un consumatore che intende acquistare un bene di lusso usato, soprattutto se non esperto in materia e non coadiuvato da uno *shop assistant* esperto, ha bisogno di certezze ancora più solide circa la sua provenienza e qualità, dato che, verosimilmente, si affaccerà

in un contesto non monitorato, dove può non bastare effettuare esclusivamente un'analisi visiva dell'oggetto in questione, leggere un certificato cartaceo o basarsi sulle informazioni riportate dal venditore su un annuncio.

È facile intuire, dunque, come anche lo scenario delle *post supply-chain*, altrettanto complesso, possa costituire terreno fertile per l'immissione di prodotti contraffatti, illegali o rubati, che devono essere tenuti alla larga dai consumatori.

Tra i rischi legati alla contraffazione nel mercato di seconda mano, infatti, è da menzionare, in primo luogo, la perdita di valore degli investimenti da parte dei consumatori. I beni di lusso autentici, spesso, mantengono o aumentano di valore nel tempo; tuttavia, se un acquirente investe in un prodotto in realtà contraffatto, credendo che sia originale, potrebbe subire una perdita significativa quando cerca di rivenderlo successivamente.

In seconda istanza, nel caso in cui un acquirente scopra, solo dopo l'acquisto, che un prodotto è contraffatto, può essere difficile risalire al venditore, dimostrare la frode subita ed ottenere un rimborso o un reso.

Fare luce sulle *post supply-chain*, in sostanza, significa adottare un'ottica incentrata sul consumatore, cercando di soddisfare le sue aspettative in modo tempestivo ed efficiente, nonché, dal punto di vista dei brand, poter giovare di una migliore profilazione della propria clientela e delle relative abitudini di consumo. In aggiunta, poter contare su un sistema di attestazione dell'autenticità anche nel mercato di seconda mano non può che favorire lo scambio di prodotti che, nonostante usati, risultano certificati, contribuendo, quindi, a migliorare l'economia circolare e ridurre gli sprechi.

3.2.3 Specifiche desiderate

Una volta delineate le diverse sfaccettature che riguardano il contesto applicativo, di seguito sono illustrate le funzionalità ideate, in una prima fase di studio di fattibilità, per il sistema in questione.

L'applicativo software, innanzitutto, deve essere in grado di tracciare, senza possibilità di alterazione ed in modo trasparente, il passaggio di beni di lusso sia tra venditori pubblici, quali brand e negozi, che tra utenti privati.

A tal proposito, ciascuna transazione deve indicare le parti coinvolte, da identificare opportunamente, deve riportare un *timestamp* ed, infine, deve essere registrata in maniera sincrona con il movimento fisico del bene, in modo tale da risultare disponibile alla consultazione in qualsiasi momento.

Il sistema deve prevedere diversi livelli di utenza da associare, sulla base di tecniche di controllo degli accessi, agli attori modellati nella Sezione 3.3. Il riconoscimento del ruolo degli utenti, dunque, deve sbloccare le funzionalità specifiche di immissione, vendita, acquisto ed eventuale restituzione di prodotti certificati.

È, infatti, richiesta la possibilità di generare e memorizzare, tramite modalità sicure, un certificato digitale volto ad attestare l'autenticità di un bene. Tale certificato deve far riferimento, in maniera incofondibile, ad uno e un solo prodotto fisico e deve essere corredato di tutte le informazioni di rilievo utili a rintracciarne la qualità, al pari di un'impronta digitale. Esso, inoltre, circolerà tra i diversi utenti sulla base dei reali movimenti del prodotto e costituirà l'oggetto principale di verifica, sia in caso di acquisto del nuovo che dell'usato, condizione che, per giunta, diverrà molto più semplice da rilevare.

In quest'ottica, eventuali problematiche possono insorgere per certificare, in un secondo momento, prodotti già esistenti o in circolazione; a tal proposito, seguiranno ulteriori considerazioni in fase di progettazione.

Oltre a ciò, all'atto della creazione del certificato univocamente associato ad un prodotto, deve risultare possibile un certo grado di *customizzazione*, con la facoltà di specificarne gli

attributi caratteristici. L'intento, infatti, è quello di rendere il sistema modulare ed in grado di trattare, al suo interno, prodotti di lusso di varia natura.

In ultima analisi, devono essere ottimizzati quanto più possibile le prestazioni e i costi del sistema.

Nello specifico, il tempo necessario a creare un certificato, piuttosto che a registrare la transazione di un bene o effettuare le necessarie verifiche di autenticità, deve risultare accettabile in relazione alle reali dinamiche di compravendita e, dunque, deve rientrare nell'ordine dei secondi o, al più, di un paio di minuti.

Dopodiché, deve essere valutata attentamente l'infrastruttura tramite cui memorizzare, in modo quanto più sicuro possibile, una mole di informazioni destinata a crescere nel tempo. L'obiettivo da raggiungere, a tal proposito, è quello della scalabilità, anche perché, all'aumentare del numero di utenti, corrisponderà una maggiore affidabilità complessiva del sistema proposto, dovuta alla riduzione del rischio di falsificazione. Per assolvere a tale richiesta, pertanto, è necessario cercare di limitare, al meglio delle possibilità, sia i costi associati allo spazio di archiviazione occupato per la registrazione delle transazioni e dei certificati che il lavoro computazionale da svolgere per le verifiche di legittimità.

Se i costi medi del sistema, tramite opportune scelte in fase di progettazione ed implementazione, dovessero risultare inferiori ai costi di gestione, soprattutto a livello burocratico, delle attuali soluzioni per la tracciabilità, si avrà la possibilità di contenere il prezzo finale dei prodotti ed aumentare il *surplus del consumatore*.

Infine, si richiede un'elevata disponibilità del sistema, da valutare in termini di percentuale di *uptime*, al fine di rendere inefficaci attacchi di *Denial of Service* che, di fatto, aprirebbero la strada all'inserimento di prodotti contraffatti.

3.3 Analisi dei requisiti

Avendo chiarito, seppur ancora ad un alto livello di astrazione, le principali *feature* desiderate per il sistema, il passo successivo prevede di raccogliere ed analizzare, più in dettaglio, i requisiti che devono essere soddisfatti e i vincoli da rispettare. Tale attività, infatti, costituisce la prima fase fondamentale dell'ingegneria del software.

3.3.1 Metodologia adottata

Il fine ultimo dell'analisi dei requisiti è quello di modellare il software in tutti i suoi aspetti e, parallelamente, le entità esterne che interagiranno con esso.

A tal proposito, è stato utilizzato il linguaggio di modellazione *i** che, a differenza del più noto UML, si concentra esclusivamente sulla descrizione dei requisiti e fornisce un maggiore potere espressivo per quanto riguarda, in modo specifico, l'analisi delle caratteristiche di sicurezza e la modellazione dei comportamenti di un utente.

Ai fini di una migliore comprensione dei diagrammi riportati nelle sezioni successive, dunque, viene fornita prima una breve descrizione delle carattersitiche e delle possibilità espressive offerte dal *framework* adottato.

I principali elementi presenti nel linguaggio *i** sono i seguenti:

- *Attore*, ossia un'entità che deve eseguire un'attività o dei compiti interagendo con il software ed utilizzando, all'occorrenza, delle risorse. Un attore può fare riferimento sia ad una persona fisica, e al ruolo da essa ricoperto, che ad un altro sistema, come un DBMS o un server, ed in tal caso si parla di agente.

- *Hard goal*, cioè un obiettivo che è possibile definire con precisione e tale per cui esiste un criterio rigoroso per stabilirne il soddisfacimento, utile al sistema per prendere delle

decisioni. La necessità di inserire *hard goal* deriva dalla constatazione che gli attori, nel loro operato, sono mossi da obiettivi.

- *Soft Goal*, ossia un obiettivo che, rispetto al precedente, è più complesso da formalizzare o la cui formulazione è più vaga. Nonostante non siano ben determinati, obiettivi di questo tipo possono risultare molto importanti nell'economia del sistema perché possono esprimere, ad esempio, requisiti di sicurezza più generici.
- *Task*, utile per modellare un'attività che deve essere svolta da un attore affinché esso possa raggiungere un obiettivo.
- *Risorsa*, ossia un bene che rappresenta il risultato di un'attività o ciò di cui ha bisogno un attore per svolgere un determinato compito. Una tipica risorsa è costituita sicuramente dai dati ma, adottando una visione olistica del sistema informatico e dell'ambiente in cui esso dovrà operare, possono essere rappresentate anche risorse fisiche.

La Figura 3.3 mostra, a livello grafico, gli elementi che caratterizzano il linguaggio *i**. Di tali elementi, *hard goal* e risorse possono essere condivisi tra due attori, dando luogo a relazioni di delega di cui va espressamente indicata la direzionalità.

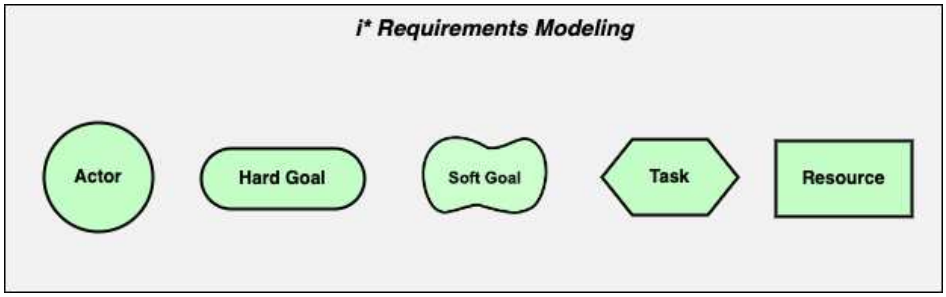

Figura 3.3: Elementi modellabili tramite il linguaggio i*

3.3.2 Early Requirement Analysis: Strategic Dependency Model

Tramite l'analisi preliminare dei requisiti è stato possibile svolgere considerazioni puntuali sulle dinamiche, sui processi e sull'organizzazione relativi al dominio di business che si intende automatizzare per mezzo del software, così come si presenta nello scenario tipico di riferimento descritto nella Sezione 3.2.1.

Difatti, a questo livello di analisi, in accordo con i principi generali dell'ingegneria dei requisiti, non è stata formulata alcuna ipotesi sull'apporto e sugli oneri del software, limitandosi esclusivamente ad indivudare gli attori coinvolti, a rintracciare le loro attitudini mentali e a stabilire le relazioni che li coinvolgono.

Nello specifico, grande attenzione è stata posta nei confronti delle relazioni di delega tra coppie di attori, ottenendo il cosiddetto *Strategic Dependency Model* (diagramma delle dipendenze strategiche) mostrato in Figura 3.4.

I principali attori modellati nell'ecosistema del passaggio di beni di lusso sono:

- *Brand*;
- *Sistema informativo prodotti (SIP)*;
- *Retailer*;
- *Product Owner* il quale, a sua volta, si specializza in *Product Buyer* e *Second Hand Reseller*.

L'attore *Brand* rappresenta l'azienda produttrice dalla quale esce il bene di lusso "ultimato", pronto per essere immesso sul mercato e per essere tracciato anche nella *post supply-chain*. Il *Brand*, alla luce di ciò, è il responsabile della creazione delle risorse *High-end product* e *Certificato di autenticità* che, una volta correlate tra loro, si sposteranno sempre insieme.

Per legare il prodotto di lusso fisico e il relativo certificato di originalità, il brand ha a disposizione diverse soluzioni, tra cui l'utilizzo di un codice seriale impresso in un barcode o, in alternativa, l'applicazione, sul prodotto stesso, di un tag che sfrutta la tecnologia RFID piuttosto che NFC. Tali possibilità sono sintetizzate, nel diagramma *i** di Figura 3.4, dalla generica risorsa *Barcode/Tag*.

Proseguendo, l'agente *Sistema Informativo Prodotti* (SIP) indica un database o un repository, mantenuto dal *Brand* e condiviso con i rivenditori, all'interno del quale sono contenute le informazioni relative ai beni di lusso gestiti nel corso della *supply chain*. Tale sistema è interrogato nel momento in cui, ai fini dell'identificazione di un prodotto, viene scandito il *Barcode/Tag* posto su di esso; di conseguenza, è necessario, a monte, che il *Brand* abbia predisposto correttamente tale risorsa.

Il SIP è stato modellato per poter integrare, all'interno del sistema *LuxuryChain*, le soluzioni ad oggi maggiormente utilizzate per il tracciamento dei beni nelle fasi della catena di approvvigionamento che conducono all'ingresso negli store; i consumatori, dunque, non si interfacceranno con esso.

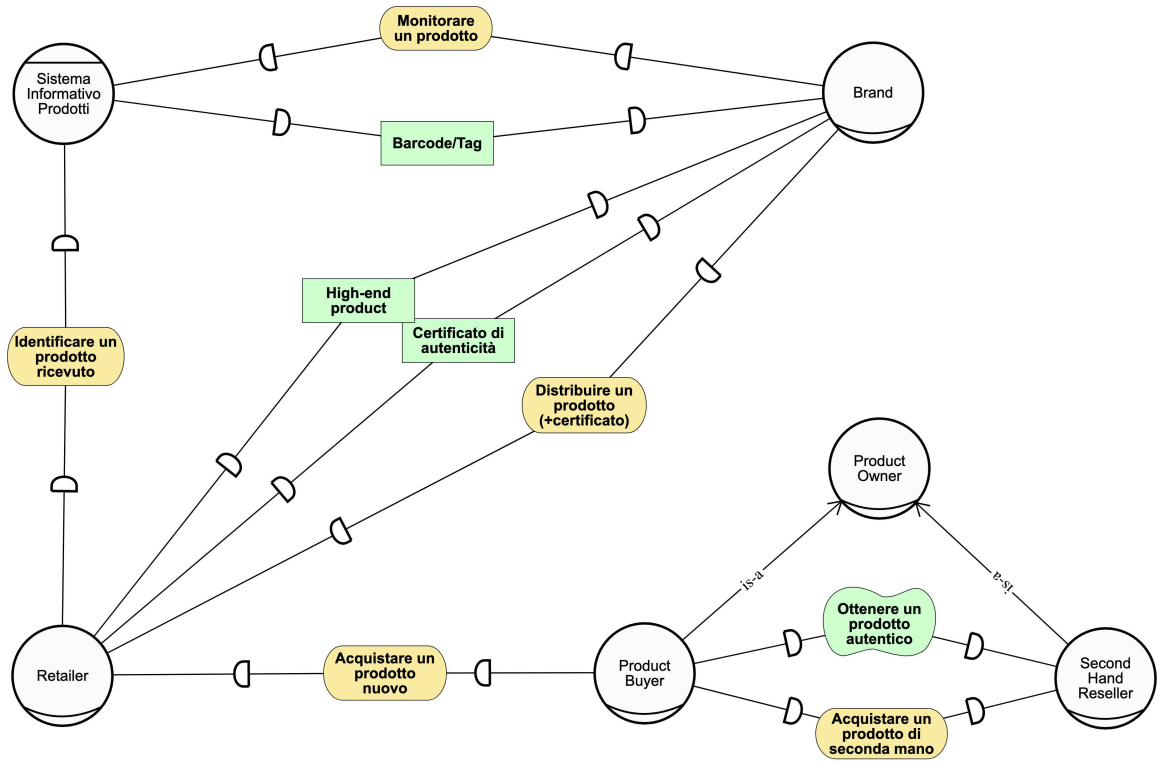

Figura 3.4: Diagramma delle dipendenze strategiche tra gli attori coinvolti nella comprevendita di beni di lusso

Il *Retailer* è l'attore incaricato della vendita al pubblico dei beni di lusso e può rappresentare sia uno store fisico che uno shop online.

Per poter svolgere il proprio compito, il rivenditore necessita delle risorse *High-end product* e *Certificato di autenticità* provenienti dal *Brand*; una volta ricevute, il *Retailer* procede alla scansione del *Barcode/tag*, così da interfacciarsi con il SIP e leggere o aggiornare le informazioni relative al prodotto.

Nel momento in cui il *Retailer* effettua una vendita, logicamente, cede sia il bene fisico che il certificato di autenticità.

L'attore *Product Owner* modella, sostanzialmente, il consumatore, dunque un privato divenuto proprietario di un prodotto di lusso corredato del relativo certificato.

Il *Product Owner* rappresenta un ruolo "astratto"; non ha particolari compiti da assolvere, se non la gestione e il mantenimento del prodotto che detiene. Ad ogni modo, da esso discendono due attori specifici.

L'attore *Product Buyer* è colui che non possiede ancora un prodotto ma intende acquistarlo. A tal fine, ha due possibilità, ossia rivolgersi ad un rivenditore ufficiale (*Retailer*) o, in alternativa, reperire il prodotto usato da un *Second Hand Reseller*. Nello specifico, per questa seconda possibilità, il diagramma di Figura 3.4 mette bene in evidenza l'obiettivo dell'acquirente di ottenere un bene originale.

In ogni caso, una volta aver completato l'acquisto, il *Product Buyer* diventa ufficialemnte *Product Owner*.

Il *Second Hand Reseller*, d'altra parte, possiede già un bene di lusso e, dunque, ha la facoltà di rivenderlo.

La ciclicità dei legami che, nel diagramma delle dipendenze strategiche, coinvolgono gli attori *Product Owner*, *Product Buyer* e *Second Hand Reseller* ha lo scopo di descrivere la possibilità di transazioni reiterate che interessano un prodotto usato.

Infine, come nota a margine, ogniqualvolta è stato chiamato in causa il passaggio di mano di un prodotto (e del relativo certificato di autenticità), si sottintende, nel diagramma, il transito delle rispettive risorse che, per motivi di leggibilità, è stato omesso.

3.3.3 Early Requirement Analysis: Strategic Rationale Model

Dopo aver esplicitato le principali relazioni di delega tra gli attori, l'analisi è proseguita sviluppando, più in dettaglio, le attitudini e le competenze di ciascuno di essi.

Infatti, tramite una decomposizione degli obiettivi primari dei singoli attori, ed inserendo task specifici, è stato possibile tracciare i confini e realizzare una "descrizione interna" di tutte le entità coinvolte.

Il diagramma risultante, descritto nel seguito, è noto come *Strategic Rationale Model* ed è riportato in Figura 3.5.

La gestione di un prodotto di lusso, obiettivo principale del *Brand*, passa attraverso il raggiungimento di tre sotto-obiettivi.

In primo luogo, il *Brand* deve immettere, dopo averli generati, il prodotto fisico e il relativo certificato di autenticità. Queste due risorse, accoppiate irreversibilmente, sono legate tra loro per mezzo di un medesimo codice seriale o, nel caso di utilizzo di un tag RFID, attraverso un *Electronic Product Code* (EPC).

Il *Brand*, creando prodotto e certificato, ne diventa ufficialmente il primo proprietario ed inaugura la catena degli attori che si succederanno, di volta in volta, al controllo di tali risorse.

Affinché un prodotto possa essere identificato in maniera efficace, inoltre, il *Brand* deve etichettarlo correttamente il che, a più basso livello, significa generare il codice a barre piuttosto che programmare opportunamente il tag RFID/NFC.

Infine, per poter monitorare il prodotto nel corso del tempo, è necessario effettuare la connessione al *SIP*.

Passando all'analisi del *boundary* dell'attore *Retailer*, come è possibile notare in Figura 3.5, l'obiettivo primario da soddisfare consiste nella commercializzazione di un prodotto. A tal proposito, il *Retailer* ha una duplice esigenza; da un lato, egli deve riuscire ad avere i prodotti disponibili nel proprio catalogo, rifornendo il magazzino; dall'altro, egli deve procedere, concretamente, alla vendita.

Il rifornimento del magazzino è possibile solo dopo aver completato i task di invio e ritiro di un ordine.

Dopodiché, nel momento in cui il *Retailer* riceve la coppia "prodotto e certificato", c'è una prima fase di identificazione in cui l'attore si rivolge al *SIP* per leggere le informazioni sul bene in questione e, parallelamente, ispeziona il certificato di autenticità per accertare l'origine del prodotto. In assenza di anomalie, dunque, il *Retailer* registra autonomamente la ricezione.

A tal punto, non resta che vendere il prodotto di lusso al pubblico. Decomponendo tale obiettivo, se ne ricava che è necessario, innanzitutto, incassare il pagamento dall'acquirente, per poi trasferire ad esso le due risorse *High-end Product* e *Certificato di autenticità*. Così come fatto lato ricezione, inoltre, è necessario che il *Retailer* registri la vendita, una volta che la transazione risulti effettuata con successo.

Proseguendo, il *Product Buyer* ha interesse nel diventare il proprietario di un bene di lusso. Tale obiettivo porta l'attore in questione ad interfacciarsi o con il *Retailer* o con il *Second Hand Reseller*, in base alla volontà di ottenere un prodotto nuovo o usato.

In ogni caso, prima di procedere all'acquisto, subentrano due step principali di verifica. Il primo riguarda l'autenticità e, sostanzialmente, consiste nel risalire all'origine del prodotto leggendo il corrispettivo certificato, mostrato dal venditore. La seconda fase di verifica, di particolare rilevanza nel caso di acquisto di seconda mano, ha lo scopo di accertare che il venditore risulti, effettivamente, il legittimo proprietario del prodotto. A tal fine, può essere richiesta una ricevuta che attesti il precedente passaggio di consegne.

Come è facile intuire, tali fasi risultano particolarmente critiche ed esposte a rischio; di conseguenza, costituiscono il fulcro attorno al quale si è concentrato l'intero lavoro.

A questo livello di analisi preliminare, che ancora esclude la presenza del software oggetto di sviluppo, però, il *Product Buyer* non può fare altro che fidarsi di ciò che legge su un certificato, solitamente stampato su carta, facilmente falsificabile ed, in ogni caso, compilato da un ente centrale, il *Brand*, senza ulteriori supervisioni, quindi potenzialmente sottoposto a filtraggio dei contenuti.

In aggiunta, il *Product Buyer*, allo stato attuale dei fatti, non dispone di mezzi efficaci per verificare la legittimità della proprietà di un prodotto usato, in un dato istante di tempo. Nello specifico, se il venditore, in precedenza, ha acquistato il prodotto da uno store ufficiale, allora mostrare la ricevuta di acquisto o uno scontrino fiscale ha effettivamente senso; se, invece, il venditore stesso ha acquistato il prodotto di seconda mano, allora la ricevuta potrebbe non essere disponibile o, comunque, potrebbe essere stata emessa non a nome dell'attuale proprietario. Da qui l'esigenza di trovare un modo attendibile per attestare la proprietà di un prodotto di lusso.

Entrambe queste riflessioni, dunque, costituiscono i motivi scatenanti dell'introduzione del sistema *LuxuryChain*, come discusso nella Sezione 3.3.4.

Ammettendo il successo delle verifiche descritte, comunque, il *Product Buyer* procede ad inoltrare il pagamento, completando l'acquisto.

Successivamente, la ricezione di prodotto e certificato deve essere, in qualche modo, ufficializzata; ottenere la prova di acquisto, infatti, garantisce il passaggio di proprietà del prodotto e, in riferimento al diagramma di Figura 3.4, sancisce l'evoluzione da *Product Buyer* a *Product Owner*.

Il *Second Hand Reseller*, continuando l'analisi delle competenze dei singoli attori, presenta, come unico obiettivo, esclusivamente la gestione di un bene di lusso regolarmente acquisito. Ciò si traduce nella facoltà di rivendere il prodotto di seconda mano o, qualora richiesto, nella possibilità di dimostrarne pubblicamente l'autenticità.

Provare l'originalità del proprio prodotto, sostanzialmente, consiste nell'esibire il relativo certificato.

Rivendere il prodotto, d'altra parte, prevede, innanzitutto, di dimostrarne l'effettiva proprietà, ad esempio fornendo la ricevuta di acquisto presso un rivenditore ufficiale; dopodiché, si incassa il pagamento ed, infine, si provvede a cedere fisicamente il prodotto e il certificato di autenticità, risorse che il *Second Hand Reseller* attualmente possiede.

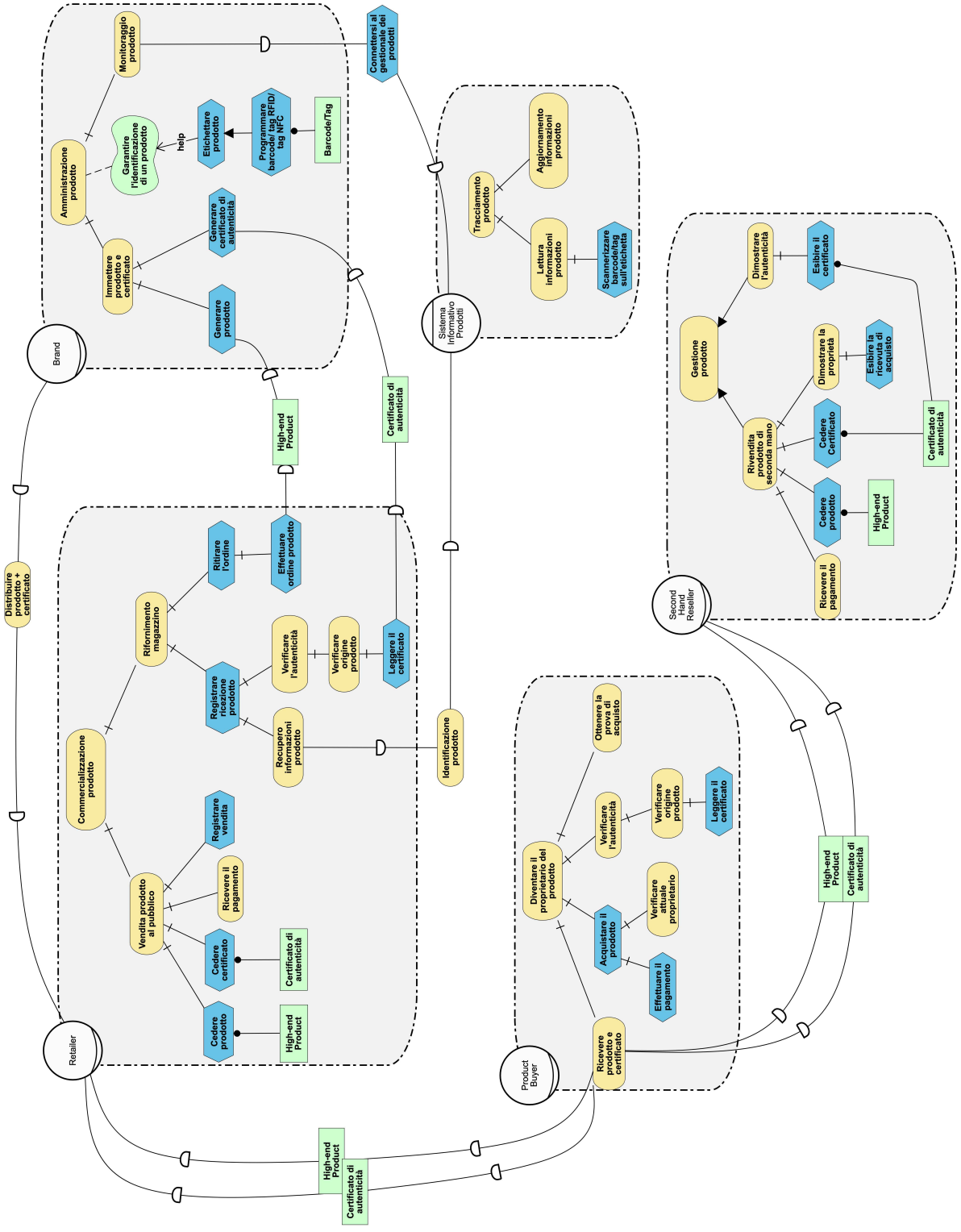

Figura 3.5: Strategic Rationale Model relativo a tutti gli attori individuati durante l'analisi preliminare dei requisiti

In conclusione, il *Sistema Informativo Prodotti* ha il fine ultimo di tracciare un bene di lusso, fornendo delle *query* per la lettura e l'aggiornamento delle informazioni ad esso relative attraverso la connessione ad un sistema RFID.

3.3.4 Late Requirement Analysis

Nell'analisi postera dei requisiti, potendo contare su un quadro ben preciso del dominio applicativo, è stato possibile iniziare a ragionare sul software vero e proprio, cercando di intuire come esso potesse intervenire.

Il passo compiuto in questa fase di lavoro, dunque, è stato quello di modellare un nuovo agente *System*, rappresentante del sistema software per la certificazione dei beni di lusso, come una piattaforma con la quale tutti gli altri attori possono interfacciarsi e che sia in grado di raccogliere tutte le informazioni necessarie da essi scambiate.

Alla luce di ciò, il modello delle dipendenze strategiche di Figura 3.4 è stato aggiornato con l'introduzione del sistema *LuxuryChain*, ottenendo il diagramma riportato in Figura 3.6.

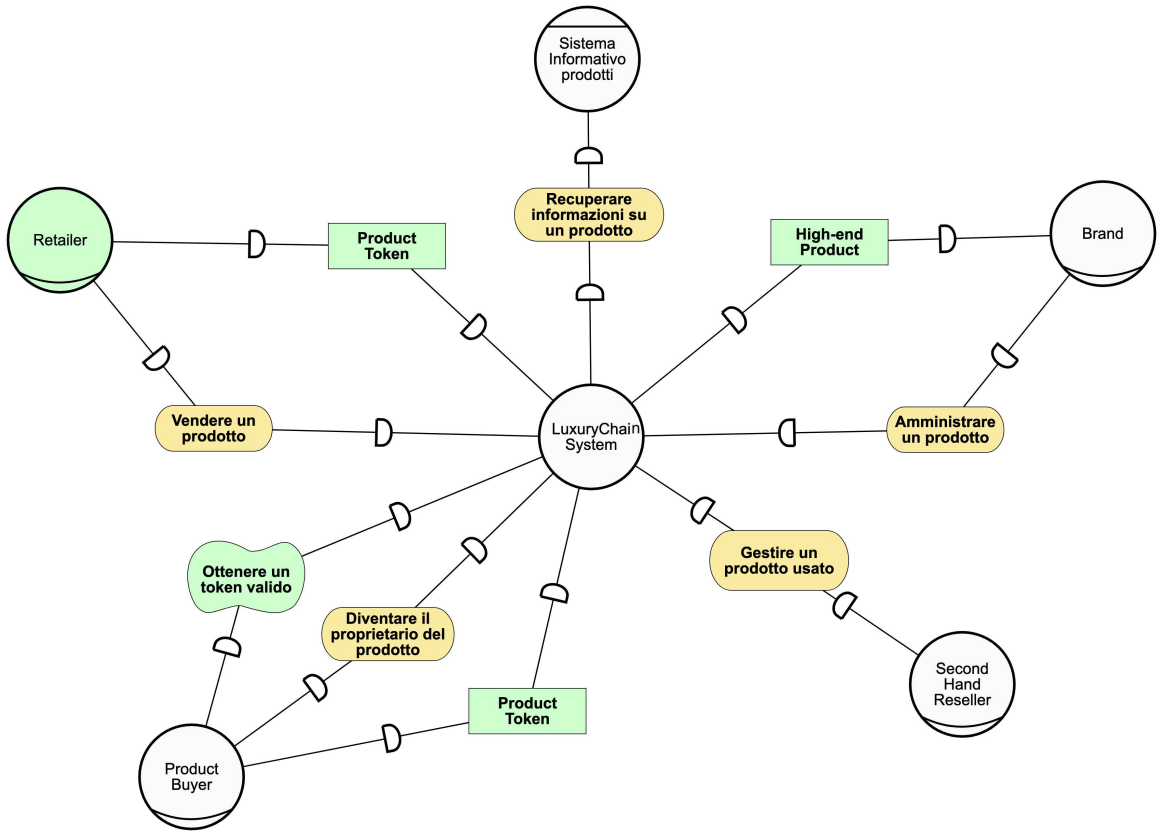

Figura 3.6: *Strategic Dependency Model* realizzato durante la fase di *Late Requirement Analysis*

Obiettivi, task e risorse che, precedentemente, gli attori delegavano a vicenda tra loro, ora sono demandate e prese in carico dal software, il quale dovrà automatizzare la porzione di mondo descritta nella Sezione 3.3.2.

Tramite l'analisi svolta in questa sede, innanzitutto, è possibile evincere che l'attore *Brand* è ancora il responsabile della creazione del prodotto, inteso come bene materiale, ma, apparentemente, è stata rimossa la risorsa *Certificato di autenticità*.

Effettivamente, tramite il software, il *Brand* ha la possibilità di generare la "versione digitale" del prodotto che, al tempo stesso, ingloba anche la funzione del certificato; di fatto, ha luogo un processo di *tokenizzazione*.

Nel mondo reale, dunque, il bene di lusso viene recapitato secondo le consuete modalità, a seguito di una regolare transazione; parallelamente, nel sistema modellato, avviene il passaggio del suo *digital twin* tra i due utenti coinvolti nel processo di compravendita.

L'attore *System*, ad ogni modo, dipende dal *Brand* per la risorsa *High-end Product*, in quanto l'azienda produttrice deve plasmare la controparte virtuale di un bene di lusso rispecchiandone le reali carattersitiche e codificando, al suo interno, le informazioni relative alla tracciabilità e alla qualità.

Ad operazione ultimata, il *Brand* può gestire in maniera efficace il prodotto digitalizzato, comprensivo di un certificato di autenticità *embedded*; interfacciandosi con il sistema, infatti, il *Brand* ha la facoltà di registrare ufficialmente la cessione o di tracciare tutti i cambi di proprietà che il prodotto subirà, monitorando il suo ciclo di vita.

Da questo momento in poi, dunque, ai fini dell'analisi dei requisiti, scompare il prodotto fisico e subentra la risorsa *Product Token* la quale, per mezzo del sistema *LuxuryChain*, può essere scambiata in maniera affidabile e sicura da compromissioni; tutti gli altri attori, alla luce di ciò, agiranno sul token del prodotto.

Il *Product Token*, inoltre, deve contenere un codice identificativo da utilizzare, come riferimento, anche all'interno del *Sistema Informativo Prodotti*, in qualità di *foreign key* (ipotizzando l'adozione di un database relazionale). Nel SIP, infatti, si trovano memorizzate ulteriori informazioni, magari non rilevanti ai fini dell'autenticità, connesse al prodotto stesso; di conseguenza, è necessario un vincolo che permetta di sfruttare, al tempo stesso, i due sistemi.

Per il resto, i *goal* che avevano gli attori *Retailer, Product Buyer e Second Hand Reseller* rimangono invariati rispetto al diagramma di Figura 3.4, se non che, ora, vengono delegati al sistema *LuxuryChain*.

Uno degli obiettivi cruciali da perseguire con l'introduzione del sistema discusso nella presente tesi è quello di rendere affidabile, in particolar modo, l'acquisto di seconda mano, fornendo al *Product Buyer* la capacità di legittimare un rivenditore privato ed attestare l'originalità del prodotto offerto da tale attore. Uno studio, a livello di *design* del sistema, delle modalità per ottenere tale risultato e dei rischi informatici connessi a ciò, sarà illustrato nel Capitolo 4.

CAPITOLO 4

Progettazione ed analisi del rischio

In questo capitolo è riportata una disamina di tutte le fasi di lavoro relative alla progettazione del sistema LuxuryChain.

Sulla base dei requisiti emersi dall'analisi svolta nel Capitolo 3, innanzitutto, verrà impostata l'architettura generale del software da sviluppare.

Successivamente, dopo aver individuato gli elementi cruciali del sistema, avrà luogo l'analisi preliminare dei rischi che, sostanzialmente, costituisce il filo conduttore dell'intero processo di progettazione.

Proprio a partire dalla valutazione dei rischi, infatti, sarà possibile determinare delle tecniche di mitigazione che avranno un impatto, a livello pratico, sulle scelte di design.

Infine, il capitolo sarà concluso con l'applicazione di un framework volto a chiarire la fattibilità dell'utilizzo delle blockchain.

4.1 Design preliminare del sistema

Archiviata la *Late Requirement Analysis*, che ha permesso di definire gli obiettivi di cui si deve far carico il sistema software in esame, l'attenzione si è focalizzata sull'impatto di tali requisiti nei confronti dell'architettura, il tutto a monte dello sviluppo vero e proprio.

Entrando nella fase di *design* del software, infatti, due sono i principali fili conduttori seguiti; oltre ai requisiti funzionali, emersi dai diagrammi *i** del Capitolo 3, si è fatto riferimento ai requisiti non funzionali, in particolare quelli attinenti alla sicurezza e alla *dependability*.

4.1.1 Assetto architetturale

Nell'intento di definire l'architettura del sistema di certificazione dell'autenticità di beni di lusso, è stato creato un modello razionale strategico incentrato sull'attore *LuxuryChain System*, effettuando una decomposizione degli obiettivi e delle attitudini ad esso delegati.

Come è possibile notare dal diagramma *i** di Figura 4.1, risultante dal processo appena descritto, il sistema è stato suddiviso in ulteriori agenti, ciascuno destinato alla sovrintendenza di uno specifico *goal* e, di conseguenza, di tutte le attività e le risorse necessarie al suo soddisfacimento.

Scomporre il sistema, che inizialmente si presentava monolitico, in un certo numero di attori di più basso livello, di fatto, equivale ad identificare dei moduli software. Tali moduli possono essere dotati o di interfacce uomo-macchina o di API (*Application Programming Interface*), nel caso in cui il componente in questione debba comunicare con altri agenti per mezzo di opportuni protocolli.

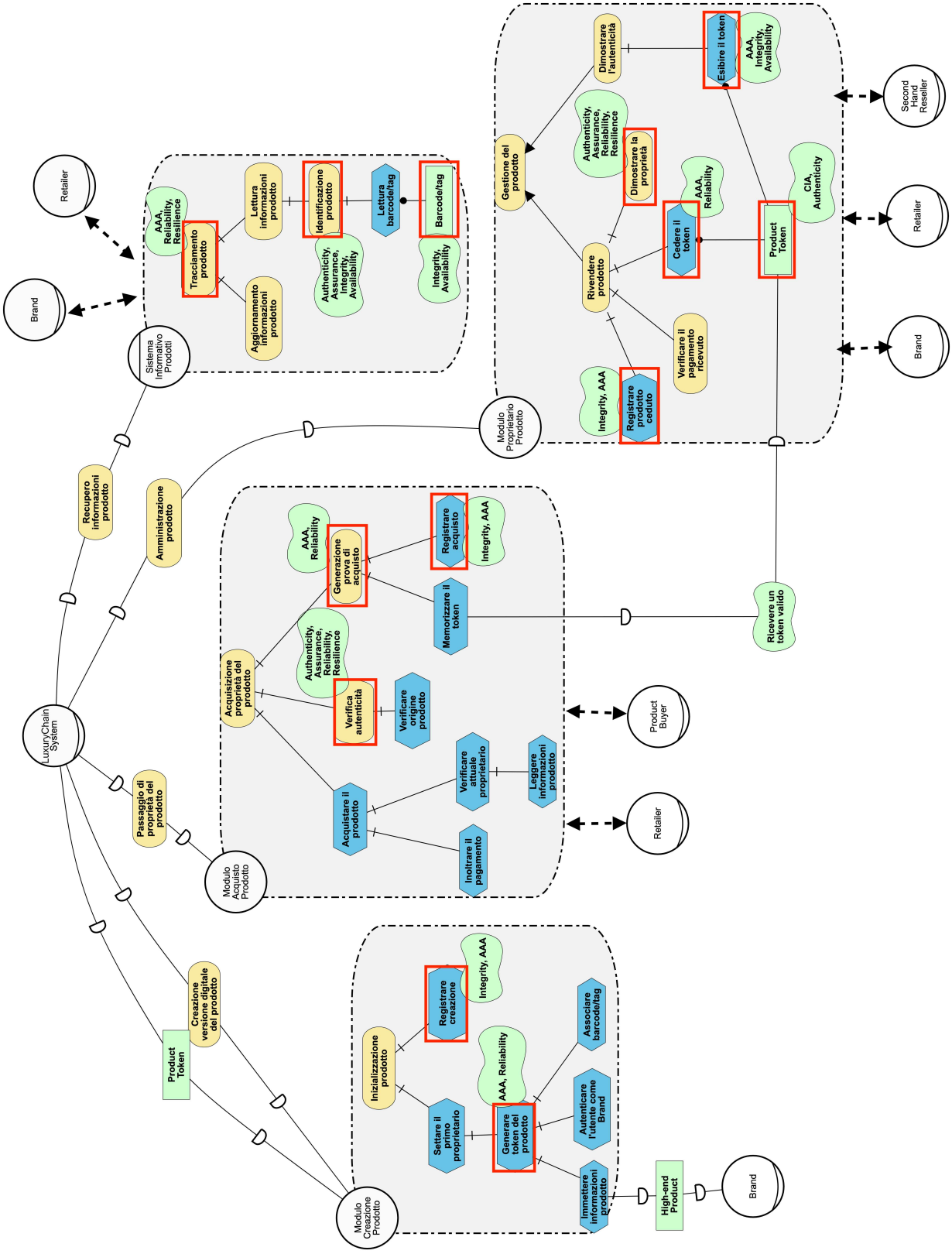

Figura 4.1: Strategic Rationale Model relativo al sistema *LuxuryChain*, dove sono stati evidenziati anche gli asset rilevanti

Sulla base della Figura 4.1, è possibile rintracciare tre moduli che costituiscono il nucleo architetturale del sistema:

- Modulo Creazione Prodotto (MCP);
- Modulo Acquisto Prodotto (MAP);
- Modulo Proprietario Prodotto (MPP).

In aggiunta alle componenti citate, permane la possibilità di interfacciamento, tramite apposita API, con il *Sistema Informativo Prodotti*, già descritto nel Capitolo 3, al fine di recuperare le informazioni sui prodotti relative alla prima parte della *supply-chain*.

Il modulo per la creazione di un prodotto (MCP) viene richiamato, previa autenticazione, dall'attore *Brand*, allo scopo di inizializzare la controparte digitale di un bene di lusso ed assicurarla all'interno del sistema di certificazione.

Avendo a disposizione un prodotto fisico pronto per essere immesso sul mercato, il *Brand* deve raccogliere ed inserire nel sistema le informazioni di rilievo utili a contraddistinguere il bene in questione e a verificarne, in un secondo momento, l'originalità. Il modulo software, alla luce di ciò, riceve in input e trasforma in un opportuno formato i dati relativi agli standard di qualità soddisfatti, alle caratteristiche fisiche peculiari, alla provenienza dei materiali e all'impatto ambientale di un bene di lusso, così da poter generare un *Product Token* ad esso fedele e "congelare" al suo interno le informazioni in merito alla genuinità. Una volta immesso, infatti, il *Product Token* non è più modificabile e può essere soltanto scambiato.

All'atto della creazione, inoltre, il token digitale viene correlato, per mezzo di un identificativo, al barcode, piuttosto che al QR-Code o al tag RFID, posto sul prodotto reale.

Infine, per completare il processo, il sistema deve registrare, in maniera indelebile, l'evento di creazione del bene virtuale, specificandone l'artefice; egli, infatti, diventa ufficialmente il primo proprietario e, dal quel momento in poi, sarà sempre rintracciabile.

Da sottolineare, in merito all'operazione di *tokenizzazione*, la necessità di un certo grado di sincronismo con il completamento del prodotto fisico. Poichè il corrispettivo digitale del bene di lusso costituisce, a tutti gli effetti, anche il certificato che ne attesta l'autenticità, sarebbe rischioso tentare di introdurre nel sistema prodotti già in circolazione, che non si trovano ad un tempo di vita t=0. Da qui il problema di certificare un bene in un secondo momento, possibilità che, in questo studio, non viene presa in considerazione.

Proseguendo l'analisi dell'architettura, il modulo per l'acquisto di un prodotto (MAP) è chiamato a gestire tutte quelle operazioni che conducono al trasferimento di un *Product Token* tra due utenti, parallelamente allo spostamento del corrispondente bene reale. Tale componente, ponendosi nell'ottica dell'ipotetico acquirente, prende in carico la verifica dell'origine e della proprietà attuale di un prodotto, analizzando la storia del token all'interno del sistema e confrontandola con le informazioni codificate al suo interno.

Una volta che il trasferimento viene completato con successo, il modulo software in questione ha il compito di memorizzare, tramite modalità sicure, il token ricevuto dall'acquirente e registrare definitivamente la transazione, generando, dunque, una prova ufficiale d'acquisto che sancisce il passaggio di proprietà. È da evidenziare, inoltre, che tale componente viene richiamata sia nel caso di acquisto di un prodotto nuovo che di seconda mano.

Infine, il modulo che "assiste" il proprietario di un prodotto (MPP) racchiude tutte le risorse e i task necessari per amministrare un *Product Token*, dopo averlo regolarmente acquisito. Tale componente, infatti, espone due principali funzionalità che consentono, rispettivamente, la rivendita del prodotto, nel senso di cessione della proprietà, e l'esibizione del token, così da poter dimostrare, qualora richiesto ed in qualsiai momento, l'autenticità.

Per una maggiore chiarezza, il diagramma di Figura 4.1 segnala quali attori interagiscono con ciascun modulo descritto.

4.1.2 Identificazione degli asset

Al fine di formulare ulteriori ipotesi sull'organizzazione del software, da tradurre successivamente in soluzioni specifiche, è fondamentale individuare gli *asset* interessanti, ossia tutti quegli elementi che, nell'economia del sistema e dello scenario in cui esso si pone, hanno un valore comprovato o un ruolo strategico. Un asset può essere una risorsa fisica, un servizio oppure qualcosa di completamente intangibile; in ogni caso, esso rappresenta un bene che costituisce una forma di guadagno o, dualmente, una perdita, in caso di compromissione.

Avendo realizzato un'accurata modellazione *i** dei potenziali moduli del software, gli asset di rilievo derivano, innanzitutto, dalle risorse previste nei diagrammi discussi; oltre a ciò, sono stati inclusi anche alcuni *task* ed *hard-goal*, in quanto servizi da proteggere e salvaguardare. Gli asset individuati per il sistema *LuxuryChain* sono evidenziati in rosso in Figura 4.1.

In aggiunta, per procedere in modo ottimale nella progettazione di un software in cui l'aspetto della sicurezza risulta cruciale, a ciascun asset è stato associato un *soft-goal* che esplicita le politiche di difesa ritenute fondamentali per il corretto operato. Tali requisiti non funzionali interessano i rami dell'*information security*, della *system security* e, più in generale, della *dependability*.

A livello pratico, si è fatto riferimento alle seguenti definizioni:

- Triade CIA
  - *Confidentiality*: garanzia che i dati non vengano letti da parti non autorizzate; può declinarsi, a sua volta, in privacy e anonimato.
  - *Integrity*: requisito tale per cui i dati non vengano alterati o distrutti in modo improprio, volontariamente o meno.
  - *Availability*: capacità di accedere ai dati al momento del bisogno; tale definizione può essere estesa anche nei confronti di un sistema.
- Triade AAA
  - *Authenticity*: capacità di accertare la veridicità di un'affermazione riguardo l'origine o l'identità .(*Authentication*).
  - *Assurance*: garanzia che un'entità si comporti come atteso; può assumere l'accezione di *Authorization* in relazione ai permessi per compiere una certa azione.
  - *Accountability*: capacità di risalire all'origine di un'azione e tracciare l'attività di un'entità; una specializzazione di tale requisito è la non ripudiabilità, ossia l'impossibilità di negare quanto fatto o detto in precedenza.
- Triade di *dependability*:
  - *Safety*: fiducia sul fatto che un sistema non arrechi danni a cose o persone.
  - *Reliability*: capacità del sistema di erogare il servizio come l'utente si aspetta.
  - *Resilience*: capacità del sistema di mantenere la continuità operativa delle proprie funzioni critiche, anche in presenza di eventi dirompenti, quali la rottura di un componente o un attacco informatico.

Un riepilogo degli asset, degli obiettivi di sicurezza da garantire per ciascuno di essi e delle policy organizzative emerse a questo livello di analisi è riportato in Figura 4.2. In particolare, si noti la predominanza dei requisiti che hanno un nesso diretto e naturale con l'ambito della certificazione dei beni di lusso; su tutti, l'integrità, l'autenticazione e l'affidabilità.

ASSET IDENTIFICATION		
Asset	Security Policy Objectives	Organizational Security Policies
PRODUCT TOKEN	Confidentiality Integrity Availability Authenticity	Il legittimo proprietario di un prodotto detiene, a livello di software, il Product Token e può leggere tutte le informazioni relative al prodotto stesso. E' necessario distinguere, tra le informazioni codificate nel Product Token, quelle che devono essere mantenute confidenziali e quelle che, invece, possono essere condivise nel momento in cui il token viene esibito. Dopo che il token viene generato, nessun utente può accedere in scrittura ad esso.
CEDERE IL TOKEN	Authenticity (Authentication) Assurance (Authorization) Accountability Reliability	La funzionalità in questione è disponibile per chiunque detenga il token relativo ad un prodotto. Il proprietario di un prodotto può accordarsi con un altro utente per la rivendita e, a seguito della verifica della transazione, a livello di software, avviene la cessione del product token attraverso la quale si verifica il passaggio di proprietà.
GENERARE IL TOKEN	Authenticity (Authentication) Assurance (Authorization) Accountability Reliability	Il Brand, dopo essere riconosciuto come tale dal sistema previa autenticazione, è l'unico che può richiedere la generazione di un token che verrà legato indissolubilmente ad un prodotto fisico. Bisogna evitare che utenti disonesti provino ad impersonare il ruolo del Brand.
VERIFICA AUTENTICITA' PRODOTTO	Authenticity Assurance Reliability Resilience	Nel momento in cui un utente (Retailer o Product Buyer) desidera acquistare un prodotto, ha l'interesse nell'assicurarsi che questo sia originale e che risalga effettivamente al Brand dichiarato. Tale operazione chiama in causa il Product Token di cui l'attuale proprietario del prodotto è dotato.
GENERAZIONE PROVA DI ACQUISTO	Authenticity Assurance Accountability Reliability	La generazione della prova di acquisto viene invocata a seguito della verifica della transazione e contestualmente al trasferimento del Product Token dal venditore all'acquirente. La prova di acquisto sarà dotata di un timestamp e potrà essere usata per un'ulteriore verifica di idoneità qualora l'utente decida in futuro di rivendere il prodotto.
DIMOSTRARE LA PROPRIETA'	Authenticity (Authentication) Assurance Reliability Resilience	Dopo essersi autenticato, l'utente vedrà nella propria area riservata la lista di prodotti che attualmente detiene, ad ognuno dei quali è associato un token. Mostrare il token è una prima modalità diretta per sostenere l'affermazione di proprietà di un prodotto. Un token può essere posseduto esclusivamente da uno e un solo utente in ogni istante.
ESIBIRE IL TOKEN	Authenticity Assurance (Authorization) Accountability Integrity Availability	Soltanto chi detiene il token di un prodotto ha a disposizione la funzionalità in questione. L'esibizione del token può avvenire a seguito della richiesta di dimostrazione dell'autenticità del prodotto, ad esempio prima della rivendita di quest'ultimo. Durante l'operazione bisogna assicurarsi che non vengano diffuse informazioni critiche riguardo il prodotto o il suo attuale proprietario. Uno stesso token non deve essere riutilizzato per dimostrare l'autenticità di due prodotti che, pur diversi, rispondono allo stesso modello
REGISTRAZIONE OPERAZIONE PRODOTTO	Integrity Authenticity Assurance Accountability	Qualsiasi operazione e/o flusso coinvolga un prodotto deve essere tracciato in maniera permanente così da poter ricreare in qualsiasi momento la "storia" di un prodotto nel tempo
TRACCIAMENTO PRODOTTO (API)	Authenticity Assurance Accountability Reliability Resilience	L'API per il tracciamento di un prodotto si interfaccia con il Sistema Informativo Prodotti, un DBMS mantenuto dal Brand all'interno del quale sono memorizzate informazioni relative ai singoli prodotti e al loro stato. Gli attori Brand e Retailer possono sfruttare questa API per monitorare un prodotto lungo la filiera che lo porterà alla vendita al pubblico. Il Brand può accedere sia in lettura che in modifica, il Retailer può accedere solo in lettura. Il Brand è l'unico attore che può leggere tutta la successione di proprietari di un proprio prodotto "a valle", anche dopo averlo ceduto; tutti gli altri attori possono soltanto leggere i diversi proprietari che un prodotto ha avuto "a monte", prima di riceverlo.
IDENTIFICAZIONE PRODOTTO	Authenticity Assurance Integrity Availability	Gli attori Brand e Retailer (ma in generale chiunque sia dotato di appositi lettori) possono accedere alle informazioni che permettono di riconoscere un prodotto attraverso l'utilizzo di un barcode o di un tag RFID/NFC posti sull'etichetta del prodotto fisico. La lettura del barcode/tag inoltrerà una richiesta al Sistema Informativo Prodotti, il quale restituirà i dati sulla natura del prodotto in questione.
BARCODE/TAG	Integrity Availability	Il Brand ha l'onere di applicare sull'etichetta del prodotto fisico un barcode o un tag RFID/NFC. Tutti sono in grado di leggere il barcode/tag e risalire alle informazioni del prodotto. Una modifica impropria a livello di barcode/tag deve risultare inconsistente con quanto riportato nel Product Token ad esso associato all'interno del software.

Figura 4.2: Asset individuati per il sistema *LuxuryChain* e loro caratterizzazione in termini di requisiti di sicurezza

Successivamente, per disporre di una visione ancora più dettagliata dei beni e dei servizi critici, è stato compilato un prospetto descrittivo di ciascun asset individuato; il template adottato a tal fine segue le cosiddette *specifiche di Jacobson* e prende in considerazione, per ogni caso d'uso che coinvolge un asset, gli attori coinvolti, i dati scambiati, le precondizioni da soddisfare, il flusso operativo standard e quello anomalo, gli effetti sul sistema e i requisiti non funzionali desiderati.

La documentazione prodotta in questa sede è parzialmente riportata nelle Figure 4.3, 4.4 e 4.5, in cui vengono approfonditi quattro asset fondamentali.

Use case ID:	**LC-01**
Use case Name:	**Product Token**
Actors	System, Brand, Retailer, Product Owner
Description	Il Product Token è l'asset principale che viene gestito per mezzo del software e rappresenta, nel dominio digitale, il prodotto di cui deve essere garantita l'autenticità
Data	Dati utili alla certificazione dell'originalità del prodotto
Stimulus and Preconditions	Deve esistere uno e un solo prodotto fisico a cui il token è legato indissolubilmente
Basic Flow	1. L'utente che possiede un prodotto X effettua il login al sistema
	2. Nell'area utente, tra gli oggetti di proprietà, è presente il token del prodotto X
Exception Flow	1. L'utente che possiede un prodotto X effettua il login al sistema
	2. L'utente non vede il token del prodotto X nella propria area riservata
Response and Postconditions	In ogni istante, il token può essere di proprietà di uno ed un solo utente
Non Functional Requirements	Confidentiality, Integrity, Availability, Authenticity
Comments	A seconda della tipologia di utenza sono disponibili operazioni diverse per manipolare il token di un prodotto.

Figura 4.3: Specifiche relative all'asset *Product Token*

Use case ID:	**LC-02**
Use case Name:	**Generazione token del prodotto**
Actors	Brand, System
Description	Il Brand crea un nuovo prodotto fisico e, sfruttando il software, ha la facoltà di generare il token che lo rappresenta digitalmente.
Data	Product Token (Output), High-end Product (Input)
Stimulus and Preconditions	Il Brand deve autenticarsi come tale e deve essere verificato dal sistema
Basic Flow	1. Il Brand inserisce le informazioni del prodotto all'interno del software
	2. Il Sistema genera un token che fa riferimento al prodotto
	3. Prodotto e relativo token vengono legati indissolubilmente
Exception Flow	1. Il Brand inserisce le informazioni del prodotto nel software
	2. Il Sistema fallisce la generazione del token con un messaggio di errore
Response and Postconditions	Una volta completata la procedura di generazione del token, di fatto il Brand viene riconosciuto come primo proprietario del prodotto e, dunque, possederà il token
Non Functional Requirements	Authenticity (Authentication), Assurance (Authorization), Accountability, Reliability
Comments	L'input del caso d'uso consiste nelle informazioni relative al prodotto fisico. Dopo che prodotto e token vengono associati tra loro, il sistema si concentrerà soltanto sul token per tutte le successive operazioni.

Figura 4.4: Specifiche relative all'asset *Generazione del token*

Use case ID:	LC-03
Use case Name:	**Cedere il token del prodotto**
Actors	Brand, Retailer, Product Owner, System
Description	Gli attori indicati hanno la capacità di scambiare un prodotto di cui sono proprietari e, di conseguenza, cedere il token gestito per mezzo del software
Data	Product Token
Stimulus and Preconditions	L'attore che intende cedere il token deve risultare il legittimo proprietario del prodotto
Basic Flow	1. L'attore che intende vendere un prodotto si autentica come proprietario
	2. Il cedente esibisce il token legato al prodotto interessato
	3. La transazione tra cedente e ricevente viene verificata
	4. Il ricevente ottiene il prodotto fisico e il relativo token (tramite il software)
Exception Flow	1. L'attore che intende vendere il prodotto si autentica come proprietario
	2. Il cedente esibisce il token legato al prodotto interessato
	3. La transazione tra cedente e ricevente non va a buon fine
	4. Il ricevente non ottiene il prodotto e il relativo token
Response and Postconditions	Il ricevente, nel momento in cui ottiene il token, diventa il nuovo proprietario del prodotto, sbloccando, dunque, tutte le operazioni previste per tale ruolo
Non Functional Requirements	Authenticity (Authentication), Assurance (Authorization), Accountability (Non Repudiation), Reliability
Comments	Il trasferimento del token deve essere notarizzato. La consegna del prodotto fisico è inserita per completezza ma non è interessata dal software, il quale, invece, gestisce il token. La transazione prevede, in linea generale, la connessione ad un circuito per il pagamento

Figura 4.5: Specifiche relative all'asset *Cedere il token*

4.2 Risk Analysis

Definiti i requisiti di sicurezza e *dependability* fondamentali per ogni asset, è stato possibile valutarne l'impatto, a livello di architettura del software, attraverso un'analisi del rischio.

4.2.1 Obiettivi generali e pianificazione

Effettuare un'analisi del rischio nelle fasi preliminari della progettazione consente di prevedere eventuali problematiche di sicurezza, per quanto possibile, con un certo anticipo ed ottenere un notevole risparmio in termini di costi; demandare tali valutazioni a momenti successivi, infatti, si dimostra essere più oneroso perchè certe scelte, probabilmente, saranno già assodate ed eventuali cambi di direzione avrebbero un peso maggiore. Analogamente, risolvere errori rilevati in fase di design è sicuramente meno impegnativo rispetto a dover correggere errori ravvisati in fasi di scrittura del codice o, peggio ancora, a seguito del rilascio.

La Figura 4.6 mostra, a livello macroscopico, gli step seguiti per svolgere l'analisi del rischio.

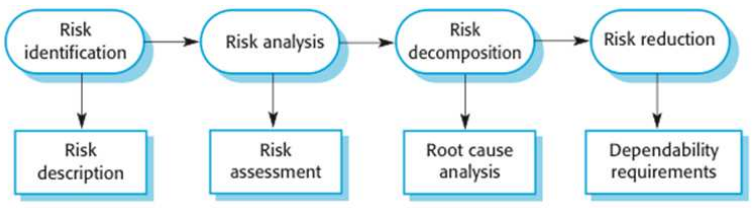

Figura 4.6: Attività svolte ai fini dell'analisi del rischio

Tramite l'identificazione del rischio, sono state individuate le potenziali problematiche di sicurezza a cui il sistema può esporsi, anche in relazione all'ambiente in cui andrà ad operare.

Successivamente, la valutazione del rischio ha portato a definire le minacce da trattare con maggiore priorità.

Per mezzo della decomposizione del rischio è stato possibile rintracciare le cause prime da cui determinate minacce derivano.

Infine, la riduzione del rischio, approfondita nella Sezione 4.3.1, ha permesso di individuare e valutare delle tecniche volte a garantire sufficienti livelli di sicurezza per il sistema.

Seppur presentata come una sequenza ordinata di attività, ad ogni modo, è bene sottolineare che l'analisi del rischio costituisce un processo evolutivo ed iterativo che non si esaurisce in una sola applicazione bensì prevede raffinamenti successivi da riprendere più volte durante lo sviluppo del software.

Prima di addentrarsi nell'analisi, inoltre, sono state stabilite delle regole per poter fornire, nella maniera più uniforme ed omologata possibile, giudizi circa le probabilità, gli impatti e i costi che entrano in gioco. A tal proposito, è stato adottato un approccio qualitativo basato sull'utilizzo di una *scala di Likert* a 3 valori e dell'aritmetica tradizionale come metodologia di aggregazione dei diversi livelli assegnati.

Per quanto riguarda la valutazione del rischio, esprimibile moltiplicando probabilità e impatto, si fa riferimento alla matrice riportata in Figura 4.1.

		Impatto		
		Trascurabile (1)	Moderato (2)	Severo (3)
Probabilità	Molto probabile (3)	Medio (3)	Alto (6)	Alto (9)
	Possibile (2)	Basso (2)	Medio (4)	Alto (6)
	Improbabile (1)	Basso (1)	Basso (2)	Medio (3)

Tabella 4.1: Matrice per la valutazione del rischio; le diverse colorazioni rappresentano il grado di rischio risultante moltiplicando probabilità e impatto: basso (verde), medio (giallo), alto (rosso)

4.2.2 Asset assessment

Un'analisi del rischio realmente efficace richiede, a monte, di determinare il valore dei singoli asset individuati, così da stabilire su quali di essi confluire i maggiori sforzi, in termini di protezione. Ai fini di tali valutazioni, oltre al valore intrinseco di ciascun asset, si è tenuto conto dell'esposizione, ossia del potenziale impatto nel caso in cui si verifichi un incidente che coinvolge il bene o servizio in questione. Valutare l'impatto, infatti, è utile per avere un'idea del costo da sostenere per porre rimedio ad un rischio concretizzato.

La tabella riportata in Figura 4.7 chiarifica, sia in termini descrittivi che numerici, il valore e l'impatto attribuiti a ciascun asset del sistema.

Come è possibile notare dalla tabella in questione, gran parte degli asset assumono una valutazione piuttosto elevata. Ciò è stato ritenuto, comunque, lecito ed è spiegabile alla luce del fatto che, a monte, durante la fase di identificazione, l'attenzione era già stata posta sugli asset che potessero risultare nevralgici, così da non appesantire il lavoro successivo.

Ad ogni modo, si evince che la risorsa *Product Token*, insieme alle funzionalità ad essa connesse, ossia la generazione, il trasferimento, la verifica di autenticità e la pubblicazione, sono da ritenersi le più importanti e meritevoli di protezione in quanto, se compromesse,

manderebbero in tilt l'intero sistema e aprirebbero la strada all'immissione di prodotti contraffatti.

Asset	Value		Exposoure (Impact)	
PRODUCT TOKEN **GENERARE IL TOKEN** **CEDERE IL TOKEN**	Asset nevralgici per l'intero sistema sui quali va riposta la massima attenzione. Hanno tutti influenza sulla verifica dell'autenticità e dell'originalità di un prodotto. Potenzialmente critici per la sicurezza.	3	In caso di problemi che coinvolgono gli asset indicati, il corretto funzionamento dell'intero sistema verrebbe compromesso, con momentanea sospensione del servizio. Sono previsti costi per il ripristino del sistema.	3
VERIFICA AUTENTICITA' **ESIBIRE IL TOKEN**	Fondamentali per ottenere la prova di originalità di un prodotto, che è l'obiettivo principale da garantire. Potenzialmente critici per lo scambio di un prodotto in sicurezza e affidabilità.	3	In caso di compromissione, non si potrà riporre fiducia nel prodotto e nella sua provenienza. Il prodotto perderebbe di valore e deve essere bloccato dal sistema, impedendone lo scambio. Perdite economiche e di reputazione per il Brand e possibili danni per i Product Owner a causa dell'inserimento nel sistema di prodotti contraffatti.	3
GENERAZIONE PROVA DI ACQUISTO **DIMOSTRARE LA PROPRIETA'**	Richiesti per verificare l'identità del legittimo proprietario di un prodotto. Potenzialmente critici nel caso di dispute o controversie sul proprietario dell'oggetto fisico e, dunque, per la successiva rivendita.	2	Non riuscire a dimostrare di aver acquisito legalmente un prodotto provoca una perdita diretta per l'utente interessato. In caso di compromissione potrebbero iniziare a circolare nel sistema prodotti rubati, così come un utente disonesto potrebbe simulare di possedere un prodotto. Il trasferimento del prodotto (token) deve essere interrotto e/o negato.	3
REGISTRARE OPERAZIONE PRODOTTO	Richiesto per il monitoraggio dei movimenti legati ad uno specifico prodotto e per il rilevamento di eventuali anomalie.	2	La "storia" di un prodotto presenterebbe lacune e diminuirebbe il livello di fiducia verso il prodotto stesso. Perdite dirette non elevate ma possibili problemi amministrativi. Previsti costi per recuperare le informazioni.	3
TRACCIAMENTO PRODOTTO (API) **IDENTIFICAZIONE PRODOTTO** **BARCODE/TAG**	Richiesti per l'individuazione di un prodotto e per l'accesso alle sue informazioni, mettendolo in relazione con il token che lo rappresenta all'interno del sistema.	2	In caso di compromissione verrebbe meno l'affidabilità di un prodotto a causa della possibilità di alterazione di informazioni sensibili (ad esempio, controlli di qualità soddisfatti). Perdite economiche e di reputazione per il Brand e il Retailer. Perdita di efficienza nella filiera.	2

Figura 4.7: Valorizzazione degli asset e valutazione dell'impatto

In secondo luogo, per dettagliare ulteriormente la valutazione degli asset, è stato assegnato un opportuno valore all'impatto di ogni singolo obiettivo di sicurezza da soddisfare. La documentazione prodotta è consultabile in Figura 4.8.

Dalla tabella riportata in tale figura, è possibile leggere le seguenti informazioni, nell'ordine in cui si presentano le diverse colonne:

- il nome dell'asset;
- il requisito di sistema derivato dall'implementazione delle policy di sicurezza dell'asset;
- il valore dell'asset;
- le possibili violazioni degli obiettivi di sicurezza previsti per l'asset;
- il valore dell'impatto associato ad ogni violazione;
- un valore sintetico relativo all'esposizione complessiva di ciascun asset, ottenuto sommando tra loro i singoli impatti ed ipotizzando, dunque, la violazione di più obiettivi di sicurezza.

Dall'analisi svolta, come era prevedibile, risulta che le conseguenze più severe si avrebbero in seguito a potenziali incidenti che coinvolgono, in primo luogo, l'esibizione del token

(esposizione 15) ed, in seconda istanza, la generazione, il trasferimento e la verifica di autenticità del token (esposizione 12). Effettivamente, mostrare il token digitale associato ad un bene di lusso è la funzionalità fondamentale per dimostrarne l'originalità ma, coinvolgendo una risorsa chiamata a circolare su un canale condiviso, è altamente soggetto ad attacchi.

In aggiunta, un'elevata esposizione (12) è associata anche all'asset *API per il tracciamento prodotto*, nonostante non sia una componente *core* del sistema oggetto di sviluppo. È da ricordare, infatti, che tale interfaccia prevede la connessione al sistema utilizzato per l'identificazione dei beni di lusso nella prima parte della supply-chain; se tale asset viene compromesso, dunque, non può essere assicurata neanche l'affidabilità del sistema *LuxuryChain*, che dal precedente dovrà prelevare dati.

ASSET	SYSTEM REQUIREMENT (SR)	VALUE	IMPACT	IMPACT VALUE	EXPOSURE
PRODUCT TOKEN	SR1 - Implementare dei meccanismi per assicurare gli obiettivi **CIA + Authenticity** per il token relativo ad un prodotto	3	Violazione di Confidentiality	1	10
			Violazione di Integrity	3	
			Violazione di Availability	3	
			Violazione di Authenticity	3	
GENERARE IL TOKEN	SR2 - Implementare dei meccanismi per assicurare **AAA + Reliability** per l'operazione di generazione del token	3	Violazione di Authenticity	3	12
			Violazione di Assurance	3	
			Violazione di Accountability	3	
			Violazione di Reliability	3	
CEDERE IL TOKEN	SR3 - Implementare dei meccanismi per assicurare **AAA + Reliability** per il trasferimento di un token	3	Violazione di Authenticity	3	12
			Violazione di Assurance	3	
			Violazione di Accountability	3	
			Violazione di Reliability	3	
VERIFICA AUTENTICITA' PRODOTTO	SR4 - Implementare dei meccanismi per assicurare **Authenticity, Assurance, Reliability, Resilience** nel processo di verifica dell'autenticità di un prodotto	3	Violazione di Authenticity	3	12
			Violazione di Assurance	3	
			Violazione di Reliability	3	
			Violazione di Resilience	3	
GENERAZIONE PROVA DI ACQUISTO	SR5 - Implementare dei meccanismi per assicurare **AAA + Reliability** nel momento in cui viene acquistato un prodotto	2	Violazione di Authenticity	3	10
			Violazione di Assurance	2	
			Violazione di Accountability	2	
			Violazione di Reliability	3	
DIMOSTRARE LA PROPRIETA'	SR6 - Implementare dei meccanismi per assicurare **Authenticity, Assurance, Reliability, Resilience** nel processo di verifica del proprietario di un prodotto	2	Violazione di Authenticity	3	10
			Violazione di Assurance	3	
			Violazione di Reliability	2	
			Violazione di Resilience	2	
ESIBIRE IL TOKEN	SR7 - Implementare dei meccanismi per assicurare **AAA + Integrity + Availability**	3	Violazione di Authenticity	3	15
			Violazione di Assurance	3	
			Violazione di Accountability	3	
			Violazione di Integrity	3	
			Violazione di Availability	3	
REGISTRAZIONE E OPERAZIONE PRODOTTO	SR8 - Implementare dei meccanismi per assicurare **Integrity, Authenticity, Assurance, Accountability** nelle operazioni di log dei movimenti di un prodotto	2	Violazione di Integrity	3	10
			Violazione di Authenticity	2	
			Violazione di Assurance	2	
			Violazione di Accountability	3	
TRACCIAMENTO PRODOTTO (API)	SR9 - Implementare dei meccanismi per assicurare **AAA + Reliability + Resilience** nell'API per il monitoraggio di un prodotto	2	Violazione di Authenticity	2	12
			Violazione di Assurance	3	
			Violazione di Accountability	2	
			Violazione di Reliability	3	
			Violazione di Resilience	2	
IDENTIFICAZIONE PRODOTTO	SR10 - Implementare dei meccanismi per assicurare **Authenticity, Assurance, Integrity, Availability** nel processo di identificazione di un prodotto	2	Violazione di Authenticity	3	10
			Violazione di Assurance	2	
			Violazione di Integrity	3	
			Violazione di Availability	2	
BARCODE/TAG	SR11 - Implementare dei meccanismi per assicurare **Integrity e Availability** della risorsa Barcode/Tag	2	Violazione di Integrity	3	5
			Violazione di Availability	2	

Figura 4.8: Documento relativo alla valutazione dell'esposizione dei singoli asset

Infine, in merito all valutazione del *Product Token*, si noti un impatto giudicato trascurabile per la violazione di confidenzialità. Ciò è spiegabile perchè il sistema si pone in un contesto in cui è richiesta elevata trasparenza, ai fini dell'autenticità; di conseguenza, il token associato ad un bene di lusso deve memorizzare, per lo più, informazioni di pubblico dominio, limitando allo stretto indispensabile eventuali dati personali.

4.2.3 Identificazione delle minacce

Nonostante sia stata utilizzata per fornire una stima dell'esposizione di ogni singolo asset, la valutazione dell'impatto di ogni violazione è, in realtà, fortemente legata al tipo di incidente.

Il passo successivo, dunque, è stato identificare le minacce da cui possono scaturire attacchi concreti ai danni degli asset individuati, il che, a livello pratico, ha portato a compilare la tabella illustrata in Figura 4.9.

THREAT IDENTIFICATION (STRIDE Model)									
Property Violated		Authentication	Integrity	Non Repudiation	Confidentiality	Availability	Authorization	Reliability	Resilience
Asset	Value (1-3)	Spoofing	Tampering	Repudiation	Information Disclosoure	Denial of Service	Elevation of privilege	Unreliability	Absence of Resilience
PRODUCT TOKEN	3	X	X		X	X			
GENERARE IL TOKEN	3	X		X			X	X	
CEDERE IL TOKEN	3	X		X			X	X	
VERIFICA AUTENTICITA' PRODOTTO	3	X					X	X	
GENERAZIONE PROVA DI ACQUISTO	2	X		X			X	X	
DIMOSTRARE LA PROPRIETA'	2	X					X	X	X
ESIBIRE IL TOKEN	3	X	X	X		X	X		
REGISTRAZIONE OPERAZIONE PRODOTTO	2	X	X	X			X		
TRACCIAMENTO PRODOTTO (API)	2	X		X			X	X	X
IDENTIFICAZIONE PRODOTTO	2	X	X			X	X		
BARCODE/TAG	2		X			X			

Figura 4.9: Identificazione delle minacce attraverso il modello STRIDE

L'attività di identificazione delle minacce si è basata sull'utilizzo del cosiddetto modello STRIDE, il quale prende in considerazione il seguente insieme di minacce:

- *Spoofing*, ossia la falsificazione dell'identità in fase di autenticazione;
- *Tampering*, cioè la manomissione o l'alterazione di una risorsa;
- *Repudiation*, ovvero la negazione di un'azione effettivamente sostenuta;
- *Information Disclosure*, ossia la divulgazione non autorizzata di informazioni confidenziali;

- *Denial of Service (DoS)*, cioè l'interruzione di un servizio o di un sistema come, ad esempio, un server;
- *Elevation of Privilege*, nel senso di acquisizione indebita di privilegi di accesso di livello superiore;
- *Unreliability*, dunque l'inaffidabilità del sistema;
- *Absence of Resilience*, cioè l'incapacità di un sistema di continuare a funzionare in condizioni critiche.

Per una maggiore chiarezza, in riferimento al contesto della certificazione dei beni di lusso, lo *spoofing* è da intendersi nell'accezione più ampia possibile e, oltre al furto di identità in senso stretto, comprende anche la contraffazione dei contenuti digitali (testi, immagini, video, etc.) finalizzata alla disinformazione o, ancora peggio, a trarre in inganno gli utenti e i sistemi di verifica (*content spoofing*). Questo spiega la presenza di tale minaccia per la stragrande maggioranza degli asset.

4.2.4 Modellazione degli attaccanti

Una volta individuate, le minacce sono state analizzate più in dettaglio e, a tal proposito, la logica adottata si è spostata, inevitabilmente, dalla difesa all'attacco.

Un contributo alla causa è dato dai cosiddetti *Abuse Case* e *Misuse Case*, ossia diagrammi che estendono la nozione di caso d'uso anche alla descrizione di quegli scenari in cui c'è qualcosa che non va, dove, di conseguenza, può nascere un incidente informatico.

Il termine "abuse case", infatti, può essere tradotto come "caso di abuso" ed il relativo diagramma *i** è stato utilizzato per modellare l'interazione, dalle conseguenze dannose, tra il sistema ed un attore malevolo, ossia un attaccante esterno.

Ad ogni modo, non è sufficiente considerare esclusivamente utenti disonesti o attaccanti "di professione"; incidenti altrettanto pericolosi, infatti, possono scaturire dall'azione inconsapevole di utenti che, pur senza fini illeciti, si interfacciano in modo "maldestro" o irresponsabile con il sistema. Alla luce di ciò, un "misuse case", interpretabile come "caso degli usi improri", è stato utilizzato per descrivere funzioni, apparentemente legittime, che il sistema dovrebbe impedire perché potrebbero comportare una perdita per qualche *stakeholder*.

Le Figure 4.10 e 4.11 mostrano, rispettivamente, porzioni salienti dell'*Abuse Case* e del *Misuse Case* relativi al sistema *LuxuryChain*, realizzati aggiungendo opportuni attori "pericolosi" nel diagramma *i** di Figura 4.1, secondo quanto detto poc'anzi.

Come è possibile notare dai diagrammi, gli *hard-goal* rintracciati, rispettivamente, per l'attaccante esterno e per gli utenti "maldestri" consistono, sostanzialmente, nella violazione delle proprietà di sicurezza previste per gli asset del sistema.

A partire dagli *abuse case* e *misuse case*, tuttavia, da un lato è stato possibile avere contezza delle superfici di impatto, ossia delle parti del sistema che possono essere colpite, dall'altro si sono poste le basi per descrivere i passi attraverso cui le minacce individuate precedentemente tramite il modello STRIDE possano concretizzarsi.

I diagrammi risultanti da questa attività di decomposizione del rischio sono definiti "alberi di attacco"; infatti, seguendo il percorso dalla radice ad una delle foglie, è possibile evidenziare una tra le diverse alternative di azione a disposizione dell'attaccante.

In Figura 4.12 è mostrato l'albero di attacco più articolato tra quelli prodotti, relativo all'attore che rappresenta l'attaccante esterno. Come è possibile notare, man mano che si scende verso le foglie dell'albero le azioni individuate sono sempre più specifiche; inoltre, è da sottolineare che singoli attacchi possono confluire nella violazione di più obiettivi di sicurezza contemporaneamente.

Prendendo in considerazione il requisito focale per il sistema *LuxuryChain*, l'autenticità, tre risultano le principali strategie di attacco a suo discapito emerse dall'albero, ovvero:

- Contraffazione del token, che, a sua volta, può concretizzarsi tramite attacchi *Man-in-the-middle* (MITM), attacchi di *replay*, falsificazione dei dati di origine e spoofing dei contenuti.
- Accesso non autorizzato, attuabile, ad esempio, tramite attacchi di *phishing*, e particolarmente critico nel caso in cui si riesca ad impersonare il *Brand*;
- Creazione di transazioni fittizie, ad esempio, a seguito dell'alterazione dei protocolli di comunicazione, della modifica dei dati di transazioni esistenti o della falsificazione delle richieste inviate al server.

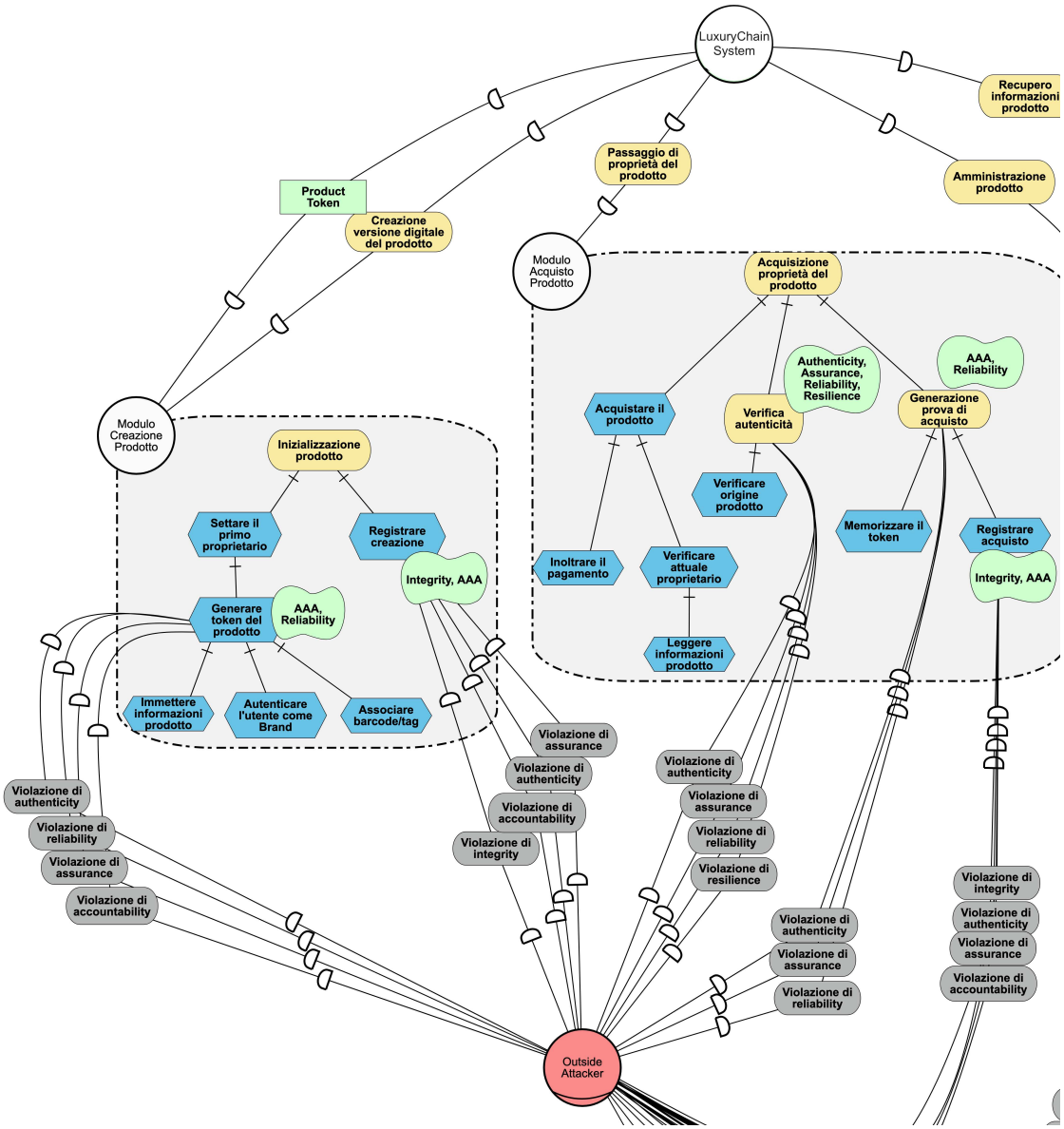

Figura 4.10: Estratto del diagramma *i** realizzato per descrivere le attitudini di un attaccante esterno (*abuse case*)

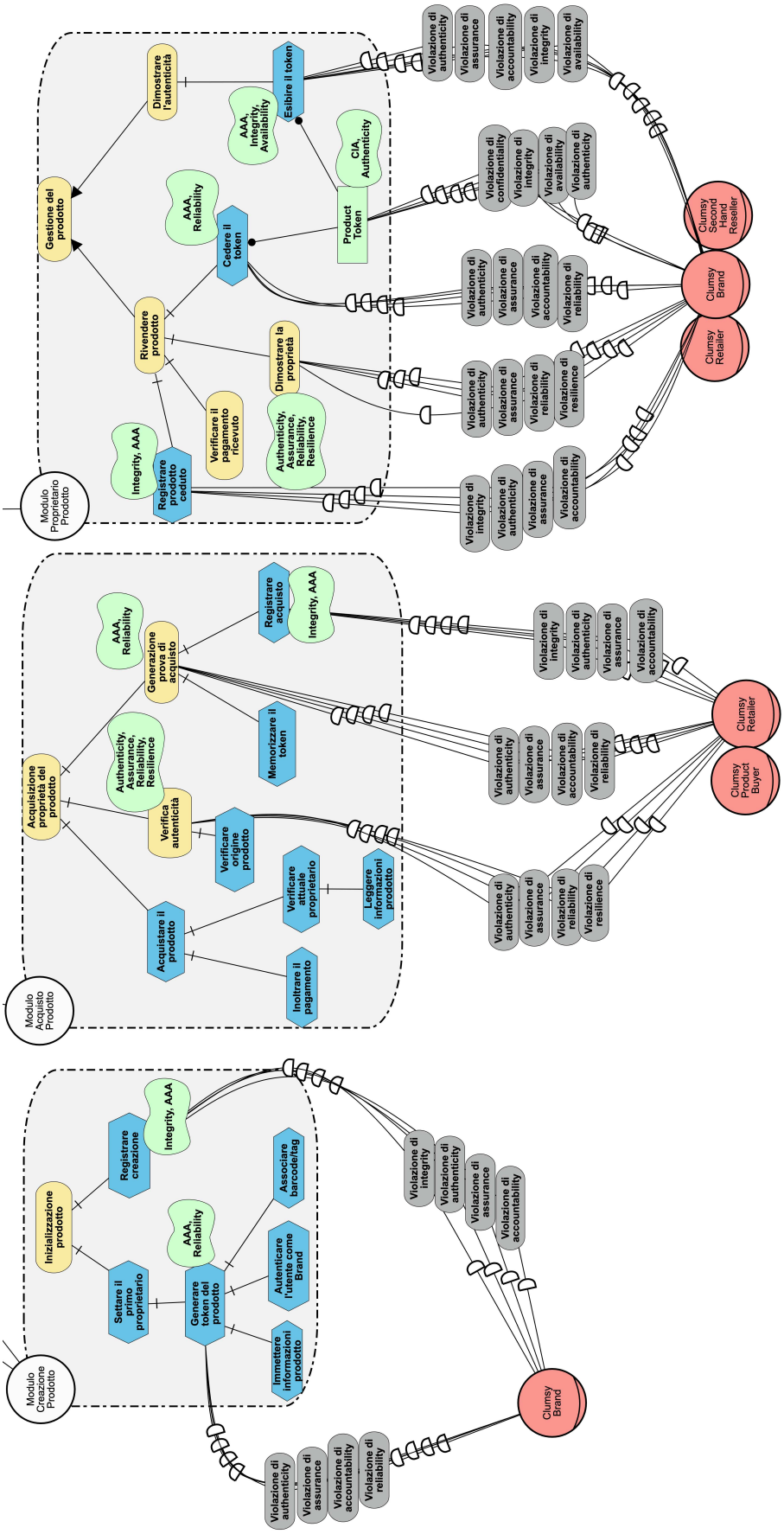

Figura 4.11: Estratto del diagramma *i** realizzato per descrivere le attitudini di utenti disattenti (*misuse case*); per ciascun attore legittimo è stato inserito un "alter-ego" che può arrecare danni al sistema

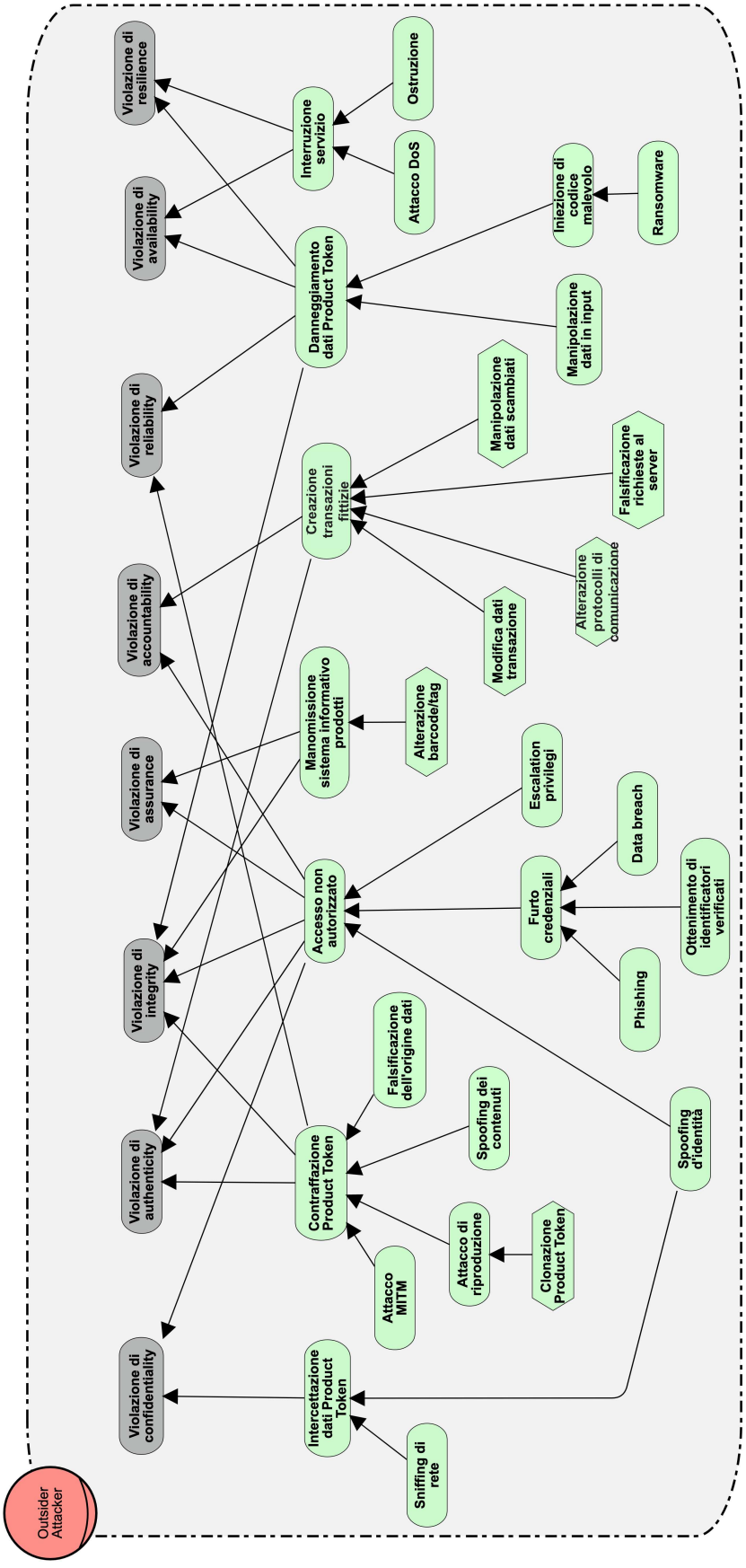

Figura 4.12: Attack tree per l'attore *Outsider Attacker*

4.2.5 Attack Assessment

Gli *attack tree* discussi nella sezione precedente hanno consentito di individuare i vettori di attacco, ciascuno dei quali rappresenta una possibile realizzazione della violazione di un obiettivo di sicurezza ben preciso di cui, in precedenza, è stato valutato anche l'impatto.

Di conseguenza, combinando i valori di impatto e le probabilità di occorrenza dei vettori di attacco, secondo le scale qualitative e l'opportuna aritmetica adottate, è stato possibile calcolare il rischio inerente attribuibile ad ogni attacco, ossia il rischio in assenza di particolari contromisure di arginamento.

Il tutto è stato riassunto nel documento mostrato in Figura 4.13 che, per motivi di concisione, fa riferimento soltanto ad un sottoinsieme degli asset totali.

Il documento citato prende in considerazione, per ogni asset, le violazioni degli obiettivi di sicurezza che lo minacciano, riportandone il possibile impatto in caso di concretizzazione; dopodiché, sono elencate specifiche alternative di attacco in grado di portare a compimento ogni singola violazione considerata, seguite dalla relativa probabilità di accadimento. Moltiplicando probabilità e impatto si ottiene il rischio inerente per ogni attacco, la cui severità è evidenziata su base colore. Infine, nell'ultima colonna, i singoli rischi sono sommati tra loro al fine di valorizzare la voce del rischio totale inerente, relativo allo scenario peggiore in cui si dovessero verificare, contemporaneamente, tutti gli attacchi plausibili per il requisito di sicurezza in questione.

Gli attacchi riportati in Figura 4.13 sono stati individuati, nella gran parte dei casi, attraverso un approccio basato sull'uso di cataloghi compilati, nel corso degli anni, da istituzioni ed organizzazioni preposte alla valutazione degli attacchi; in sostanza, quindi, si è fatto riferimento a problemi noti emersi dall'esperienza comune passata.

Il catalogo adottato come riferimento in questa sede va sotto il nome di CAPEC (*Common Attack Pattern Enumeration and Classification*); tale repository, non legandosi a tecnologie specifiche, risulta particolarmente utile in fase di progettazione.

All'interno di CAPEC sono classificati i possibili pattern che, solitamente, gli attaccanti seguono per provocare un incidente informatico. Ogni tipologia di attacco è corredata di una scheda tecnica che ne descrive le finalità, le possibili cause, gli scenari di applicabilità, il *workflow* di esecuzione, i requisiti colpiti, la severità ed eventuali altri attacchi correlati. Ogni *attack pattern*, inoltre, può essere seguito da un *security pattern* che fornisce indicazioni su come impostare il software affinché l'attacco non possa essere perpetrato.

Esempi di tipologie di attacco particolarmente rilevanti ai fini della tesi, utilizzati in questa fase di valutazione, sono *CAPEC-194: Fake the Source of Data*, relativo all'inserimento di dati sotto falsa identità, *CAPEC-148: Content Spoofing*, ossia apportare modifiche improprie mantenendo inalterata l'origine del dato, *CAPEC-544: Counterfeit Organizations*, attinente al ramo della *social engineering*, e *CAPEC-399: Cloning RFID Cards or Chips*, collegato alla manipolazione dei dispotivi IoT per il tracciamento. Gli attacchi citati, insieme agli altri riportati in Figura 4.13, sostanzialmente, possono essere considerati come ulteriori foglie dell'*attack tree* di Figura 4.12.

Come è possibile leggere dalla tabella mostrata in Figura 4.13, infine, i rischi inerenti complessivi più elevati si trovano in corrispondenza delle violazioni dell'obiettivo di autenticità, a causa dell'impatto massimo ad esso attribuito e dei molteplici vettori di attacco ad esso collegati su cui si è posta l'attenzione.

ATTACK ASSESSMENT						
Asset	**Requirement**	**Impact Requirement**	**Attack**	**Inherent Probability**	**Inherent Risk**	**Total Inherent Risk**
PRODUCT TOKEN	Violazione di Confidentiality per il requisito SR1	1	CAPEC-138: Sniffing Network Attack	3	3	8
			CAPEC-93: Man-in-the-Middle (MITM)	3	3	
			Pubblicazione accidentale dati token	2	2	
	Violazione di Integrity per il requisito SR1	3	CAPEC-138: Content Spoofing	3	9	30
			CAPEC-130: Resource Injection	2	6	
			CAPEC-193: Fake the Source of Data	3	9	
			Modifica accidentale Product Token	2	6	
	Violazione di Availability per il requisito SR1	3	CAPEC-607: Obstruction	2	6	18
			CAPEC-339: Local Execution of Code	2	6	
			Cancellazione accidentale Product Token	2	6	
	Violazione di Authenticity per il requisito SR1	3	CAPEC-93: Man-in-the-Middle (MITM)	3	9	51
			CAPEC-193: Fake the Source of Data	3	9	
			CAPEC-11: Exploitation of Trusted Identifiers	3	9	
			CAPEC-131: Identity Spoofing	3	9	
			CAPEC-333: Counterfeit Organizations	3	9	
			Duplicazione accidentale Product Token	2	6	
GENERARE IL TOKEN	Violazione di Authenticity per il requisito SR1	3	CAPEC-132: Input Data Manipulation	2	6	45
			CAPEC-11: Exploitation of Trusted Identifiers	3	9	
			CAPEC-131: Identity Spoofing	3	9	
			CAPEC-193: Fake the Source of Data	3	9	
			Inserimento informazioni erronee	2	6	
			Errata associazione prodotto-token	2	6	
	Violazione di Assurance per il requisito SR1	3	CAPEC-122: Privilege Escalation	3	9	24
			CAPEC-98: Phishing	3	9	
			Creazione accidentale Product Token	2	6	
	Violazione di Accountability per il requisito SR1	3	CAPEC-131: Identity Spoofing	3	9	18
			CAPEC-193: Fake the Source of Data	3	9	
	Violazione di Reliability per il requisito SR1	3	CAPEC-193: Fake the Source of Data	3	9	36
			CAPEC-132: Input Data Manipulation	3	9	
			Inserimento informazioni erronee	2	6	
			Errata associazione prodotto-token	2	6	
			Creazione accidentale Product Token	2	6	
VERIFICA AUTENTICITA' PRODOTTO	Violazione di Authenticity per il requisito SR3	3	CAPEC-93: Man-in-the-Middle (MITM)	3	9	39
			CAPEC-131: Identity Spoofing	3	9	
			CAPEC-372: Signature Spoof	2	6	
			CAPEC-333: Counterfeit Organizations	3	9	
			Esibizione di un diverso Product Token	2	6	
	Violazione di Assurance per il requisito SR3	3	CAPEC-122: Privilege Escalation	3	9	24
			CAPEC-11: Exploitation of Trusted Identifiers	3	9	
			CAPEC-372: Signature Spoof	2	6	
	Violazione di Reliability per il requisito SR3	3	CAPEC-663: Server Side Request Forgery	2	6	33
			CAPEC-93: Man-in-the-Middle (MITM)	3	9	
			CAPEC-113: Shared Resource Manipulation	2	6	
			CAPEC-283: Transaction or Event Tampering	2	6	
			Esibizione di un diverso Product Token	2	6	
	Violazione di Resilience per il requisito SR3	3	CAPEC-607: Obstruction	2	6	15
			CAPEC-113: Flooding	3	9	

Figura 4.13: Valutazione dei rischi inerenti associati ai singoli attacchi e, complessivamente, a ciascun asset

4.3 Risk-driven Design

I valori di rischio inerente ottenuti tramite l'analisi svolta nella Sezione 4.2.5 rientrano, in gran parte, nella categoria massima di severità, motivo per cui non possono essere trascurati. Tale considerazione, infatti, ha portato ad interrogarsi sulle tecniche da implementare per fronteggiare eventuali attacchi e, di conseguenza, ridurre l'esposizione del sistema *LuxuryChain*.

4.3.1 Strategie di mitigazione

Al fine di ridurre l'esposizione complessiva del sistema ai possibili attacchi rintracciati, è stata valutata, per ogni singolo rischio, l'opportuna strategia di risposta; tra le diverse misure di controllo plausibili, infatti, alcune di esse agiscono in riduzione della probabilità di occorrenza di un evento rischioso; altre, invece, mirano a limitarne l'impatto.

Accanto alle considerazioni circa l'efficacia delle diverse tecniche di mitigazione, ad ogni modo, è stata effettuata anche una valutazione della loro fattibilità, tenendo conto non solo dei costi effettivi per l'implementazione, ma anche di tutta una serie di costi indiretti quali, ad esempio, il dispendio di risorse computazionali, l'usabilità e l'accettabilità degli utenti.

Le tecniche di mitigazione valutate nell'ottica dell'asset *Product Token*, i costi stimati per la loro introduzione e una descrizione testuale della relativa fattibilità sono riportati in Figura 4.14. Le misure di sicurezza che è possibile leggere in tale documento sono state individuate, prelevantemente, a partire dai *security pattern* offerti dal catalogo CAPEC e dalle soluzioni proposte nella sezione "*Mitigations*" del catalogo *ATT&CK* redatto dalla *Mitre Corporation*.

Tra le possibili tecniche a disposizione per rispondere ad un medesimo rischio, inoltre, si noti che alcune di esse risultano, di fatto, complementari, mentre altre sono da ritenersi alternative, in quanto si fondano su approcci incompatibili. In relazione al vettore di attacco *CAPEC-194: Fake the Source of Data*, ad esempio, le strategie "Validazione decentralizzata delle operazioni" e "Affidarsi ad un provider dati centralizzato" seguono linee d'azione diametralmente opposte, dunque pongono di fronte ad un bivio di progettazione; le mitigazioni "Ridondanza" e "Diversità", valutate per il rischio *CAPEC-607: Obstruction*, invece, possono essere integrate insieme anche perché, come si vedrà nella Sezione 4.3.2, la prima punta a ridurre l'impatto mentre la seconda agisce sulla probabilità. Riflessioni di questo tipo, dunque, derivano dalla volontà di considerare, all'occorrenza, anche delle combinazioni multiple di tecniche ed, in generale, conducono ad una visione più eterogenea.

4.3.2 Definizione dei requisiti di sicurezza

L'ultimo passo decisivo della fase di riduzione del rischio consiste nella selezione, tra tutte le mitigazioni plausibili, di quelle misure che entreranno a far parte, ufficialmente, delle specifiche di progettazione del software.

Difatti, accanto ai requisiti funzionali, che sicuramente costituiscono una traccia importante da seguire, gli aspetti di sicurezza hanno un riscontro altrettanto tangibile sull'architettura del sistema, indirizzando su determinate scelte di design piuttosto che su altre.

Il documento prodotto in quest'ultima attività di analisi, consultabile in parte in Figura 4.15, ha l'intento di fornire una prioritizzazione delle diverse misure di controllo a disposizione per fronteggiare gli attacchi. Effettivamente, il rischio a cui si va incontro, in assenza di una valutazione di questo tipo, è duplice; da un lato c'è il pericolo concreto di gravare sulle prestazioni del sistema, introducendo più strumenti di difesa di quelli realmente necessari; d'altro canto, il risvolto di soluzioni di sicurezza adottate senza un criterio ben preciso può risultare particolarmente invasivo in termini di interfaccia utente, impattando negativamente

sulla facilità d'uso o sull'accettabilità, fattori da cui, spesso, dipende il "successo" di un sistema informatico. Da tali considerazioni, infatti, si può dedurre che, non sempre, le strategie tecnologicamente migliori sono, automaticamente, anche quelle più convenienti.

Alla luce di ciò, si è reso necessario trovare un compromesso tra le misure di sicurezza da implementare e gli aspetti appena citati, basandosi su un approccio metodologico ben definito. Infatti, per giungere alle scelte e ai risultati mostrati in Figura 4.15, sono stati calcolati, relativamente a ciascuna mitigazione considerata, i due seguenti indici:

- *Value-to-cost ratio*, ossia il rapporto tra il valore di un requisito di sicurezza e il costo di una specifica tecnica di difesa ad esso riferita;
- *R.o.C (Rate of Change)*, il quale valuta la riduzione del rischio dovuta all'adozione di una determinata misura di controllo, rispetto al costo da sostenere per la sua introduzione; operativamente, esso si può ricavare come $\frac{RischioInerente - RischioResiduo}{Costo} - 1$

In entrambi i casi, seppur con scale diverse, valori più elevati giocano a favore della mitigazione considerata. Nello specifico, è stata adottata la seguente etichettatura:

▶ Value-to-cost ratio

- *alto*, per valori maggiori o uguali di 1.5; l'investimento in una misura di difesa è ben giustificabile dall'importanza del requisito di sicurezza coinvolto;
- *medio*, per valori compresi tra 1 e 1.5 (escluso); l'investimento in una misura di difesa non è così conveniente, ma può essere preso comunque in considerazione;
- *basso*, per valori compresi tra 0 e 1 (escluso); l'investimento non è giustificabile rispetto al valore del requisito di sicurezza.

▶ R.o.C

- *alto*, per valori maggiori o uguali di 1; la tecnica considerata è efficace nel ridurre il rischio, apportando decisamente maggiori benefici rispetto ai costi previsti;
- *medio*, per valori compresi tra 0 e 1, estremi esclusi; la riduzione del rischio apportata non è cosi palese nei confronti dei costi da sostenere;
- *basso*, per valori minori o uguali a 0; *R.o.C.* nullo indica che i vantaggi nella riduzione del rischio sono pareggiati esattamente dai costi; se negativo, il costo supera addirittura il valore ottenuto dalla riduzione del rischio.

In riferimento alla tabella illustrata in Figura 4.15, le misure selezionate con alta priorità sono esclusivamente quelle tali per cui sia il *value-to-cost* che il *R.o.C.* rientrano nella rispettive categorie massime; se uno solo dei due indici, invece, dà luogo ad un valore mediocre, allora la tecnica di controllo viene comunque scelta, ma con minore urgenza; infine, in tutti gli altri casi, la mitigazione è stata scartata.

Ad ogni modo, sono da segnalare alcune eccezioni a quanto detto. Ad esempio, le misure "validazione decentralizzata delle operazioni" e "affidarsi ad un provider dati centralizzato" risultano, in base agli indici calcolati, entrambe molto efficaci per la risposta al medesimo rischio "*CAPEC-194: Fake the Source of Data*"; in situazioni come questa, dunque, è stata data priorità alla tecnica con i valori puntuali migliori o, in alternativa, si è preferita quella che prospetta un maggiore campo di applicazione, considerando anche gli altri rischi. Nel caso specifico discusso, è stata confermata la soluzione "validazione decentralizzata delle operazioni", scartando completamente la seconda; tali misure, infatti, non possono essere combinate tra loro.

In linea generale, dall'analisi del rischio eseguita, si distinguono, per la loro notevole efficacia, diverse strategie di mitigazione legate ad approcci decentralizzati. Ciò non è casuale; i vantaggi di un'architettura basata sulle nozioni di decentralizzazione e distribuzione sono particolarmente rilevanti nei contesti di tracciabilità e verifica dell'autenticità, come quello in esame. Schemi di questo tipo, infatti, basandosi su topologie a maglie, anziché gerarchiche, consentono sia di ottenere maggiore trasparenza, sicurezza ed equità di trattamento sia, in molti casi, di velocizzare l'accesso ai dati rispetto a soluzioni tradizionali monolitiche, incoraggiando, di conseguenza, la condivisione delle risorse. Dal punto di vista delle performance, inoltre, architetture di questo tipo sono in grado di raggiungere una considerevole *fault tolerance*, a favore della resilienza complessiva del sistema. Tali evidenze, insieme alle considerazioni svolte nella Sezione 4.4.1, motiveranno l'adozione delle blockchain come infrastruttura a supporto del sistema in esame.

Più in dettaglio, le misure di controllo che, in base all'analisi effettuata, sono state selezionate come ulteriori specifiche di progettazione e, al tempo stesso, si riveleranno particolarmente attinenti al mondo blockchain, sono quelle elencate di seguito:

- notarizzare i dati in maniera immutabile;
- verifica decentralizzata delle operazioni;
- verifica incrociata delle informazioni da più "oracoli";
- verifica dell'autenticità delle chiavi pubbliche in modo decentralizzato;
- verifica continua del passato di un'entità + implementazione di meccanismi di reputazione;
- utilizzo di tecniche di hashing;
- cifratura e firma digitale dei token di autenticazione in transito;
- utilizzo di strumenti di monitoraggio delle attività;
- meccanismi di enforcement di clausole per la gestione dei privilegi.

4.3.3 Linee guida per la progettazione degli asset

I requisiti funzionali emersi dalla modellazione del sistema e gli aspetti di sicurezza trattati nel corso dell'analisi del rischio hanno permesso di delineare buona parte dell'architettura del software; dopodiché, tale processo si è completato fornendo delle direttive incentrate sulla progettazione degli asset individuati. Per adempiere a ciò, si è fatto riferimento alle linee guida proposte, rispettivamente, da OWASP (*Open Worldwide Application Security Project*) e Sommerville [2015], adattandole al caso di studio specifico.

Al centro dell'attenzione c'è, indubbiamente, il *Product Token*, ovvero la risorsa principale scambiabile tramite il software.

Affinché, all'interno del sistema, possa essere garantita l'attendibilità delle affermazioni circa la qualità e l'autenticità di un prodotto, è necessario evitare condivisioni inappropriate tra gli utenti o i diversi moduli software. Le informazioni da "assicurare", invece, devono essere incapsulate insieme e veicolate all'interno di un unico "contenitore" sicuro, il *Product Token* per l'appunto, così da non disperdere le forze nel tentativo di controllare molteplici flussi dati che, potenzialmente, potrebbero arrecare problemi; si tratta, cioè, dell'applicazione del principio "*Least common mechanism*".

In aggiunta, la controparte digitale di un prodotto non deve contenere più informazioni di quelle strettamente necessarie all'identificazione del bene fisico corrispondente e alla verifica

dell'originalità, rispettando la *best practice* di compartimentare gli asset. Così facendo, infatti, un'ipotetica compromissione non va ad incidere, a cascata, su altre componenti del sistema ed, in maniera analoga, chi ha accesso al token non può risalire deliberatamente ad ulteriori risorse come, ad esempio, i dati del proprietario. Questo "isolamento", tra l'altro, motiva anche la coesistenza del *Sistema Informativo Prodotti*, dal quale possono essere lette ulteriori informazioni secondarie.

Il token di un prodotto, inoltre, costituisce, a tutti gli effetti, una concretizzazione del principio dell'*open design*, dato che la sua solidità, in termini di sicurezza, non deve risiedere nella segretezza del design sottostante bensì si deve fondare sulla robustezza dei protocolli attraverso i quali viene gestito. Nello specifico, il *Product Token* è pensato come una sorta di "ancora digitale" con lo scopo di sopperire, nello scenario attuale del tracciamento dei prodotti, all'assenza di un "ponte" stabile tra un bene fisico e il relativo certificato di autenticità; in virtù di ciò, è proprio sulla base delle modalità attraverso le quali avviene il collegamento, prima, e il trasferimento, poi, di questi due elementi che si gioca la lotta alla contraffazione, piuttosto che sulla riservatezza in sè del certificato. Nell'ottica del sistema *LuxuryChain*, per di più, la circolazione del token è addirittura cercata, in modo che tale risorsa possa essere validata da più entità contemporaneamente.

Un ulteriore principio di progettazione adattabile al contesto in esame è noto come "fail-safe defaults"; più in dettaglio, una possibile declinazione di questa buona norma prevede di impostare come predefinito, già all'atto dell'installazione di un sistema software, il massimo livello di "alert" di sicurezza previsto. Estendendo tale idea alle funzionalità del sistema *LuxuryChain*, ne consegue la scelta di trattare l'acquisto del nuovo e dell'usato alla stessa stregua; in entrambi i casi, infatti, verrà adottato lo stesso grado di severità, in termini di verifiche di legittimità, necessario per la compravendita di seconda mano che, tra le due modalità, è decisamente la più critica.

Oltre a ciò, la necessità di regolamentare, tramite appositi strumenti di *enforcement*, lo scambio di beni di lusso in un contesto di scarsa fiducia, unitamente all'esigenza di definire delle modalità di certificazione dell'autenticità che siano trasparenti e a prova di contraffazione, confluiscono nella buona norma di basare ogni decisione su policy di sicurezza esplicite; dopodiché, tali regole dovranno essere codificate all'interno del software, accertandosi, tramite strumenti automatizzati di *checking*, che non vengano violate.

La progettazione degli asset, inoltre, deve seguire, idealmente, il principio di evitare *single point of failure*, così da non affidare il corretto funzionamento dell'intero sistema ad un unico pilastro o, se proprio ciò non è possibile, quantomeno adottare tecniche di ridondanza di quegli elementi al centro delle funzionalità. Tale pratica è ben supportata dalle tecniche di controllo, emerse dall'analisi del rischio, che prediligono soluzioni decentralizzate.

Infine, risulterà cruciale, per l'economia del sistema *LuxuryChain*, il principio di registrare tutte le attività, così che sia sempre possibile riesaminare quanto successo in passato e rilevare eventuali anomalie. A livello architetturale, dunque, il risvolto di tale pratica consiste nell'adozione di tecniche di monitoraggio, volte a ridurre la probabilità che si verifichi un evento negativo.

Nel contesto in esame del passaggio dei beni di lusso, nello specifico, l'analisi delle transazioni registrate fino all'istante di riferimento è utile per risalire, in maniera inequivocabile, al legittimo proprietario di un *Product Token* e, analogamente, a colui che l'ha immesso nel sistema, in modo da verificare la provenienza.

Asset	Attack	Inherent Probability	Control	Cost	Feasibility
PRODUCT TOKEN	CAPEC-138: Sniffing Network Attack	3	Cifrare le informazioni sensibili che devono viaggiare su canali non sicuri	1	Implementazione relativamente semplice, buoni risultati, necessità di rendere consapevoli gli utenti
	CAPEC-93: Man-in-the-Middle (MITM)	3	Verificare l'autenticità delle chiavi pubbliche in modo decentralizzato	2	Ottiima affidabilità grazie al concetto di consenso, tempi più lunghi
			Adottare protocolli per rendere sicure le comunicazioni (SSL/TLS)	1	Implementazione relativamente semplice, utilizzo di protocolli consolidati
	Pubblicazione accidentale dati token	2	Separazione dei dati	2	Maggiore effort di design, nessun apprendimento richiesto per l'utente
	CAPEC-138: Content Spoofing	3	Notarizzare i dati in maniera immutabile	2	Le modifiche diventano complicate a fronte di maggiore trasparenza e affidabilità, richiesta disponibilità di memoria
			Tamper-resistant protocols	3	La fattibilità dipende dalla complessità dei protocolli e dalla facilità di integrazione con il sistema
	CAPEC-130: Resource Injection	2	Sanificazione degli input	2	Attenzione richiesta in fase di implementazione, necessità di test mirati
	CAPEC-193: Fake the Source of Data	3	Validazione decentralizzata delle operazioni + verifica incrociata delle informazioni da più "oracoli"	2	Possibile allungamento dei tempi per l'elaborazione a fronte di maggiore affidabilità, necessario un meccanismo per gestire il consenso
			Affidarsi ad un provider dati centralizzato	2	Maggiore semplicità nella gestione dei flussi dati, necessità di fiducia verso il provider
	Modifica accidentale Product Token	2	Notarizzare i dati in maniera immutabile	2	Le modifiche diventano complicate a fronte di maggiore trasparenza e affidabilità, richiesta disponibilità di memoria
	CAPEC-607: Obstruction	2	Ridondanza e bilanciamento del carico	3	Necessità di mantenere aggiornate e coerenti le diverse copie
			Diversità	2	E' richiesto uno sforzo di sviluppo affinché le diverse copie non siano sottoposte ad un single point of failure
	CAPEC-339: Local Execution of Code	2	Eseguire tutti i file sospetti in una sandbox	1	Implementazione relativamente semplice, necessità di consapevolizzare gli utenti
	Cancellazione accidentale Product Token	2	Notarizzare i dati in maniera immutabile	2	Le modifiche diventano complicate a fronte di maggiore trasparenza e affidabilità, richiesta disponibilità di memoria
			Ridondanza e bilanciamento del carico	3	Necessità di mantenere aggiornate e coerenti le diverse copie
	CAPEC-11: Exploitation of Trusted Identifiers	3	Cifratura e firma digitale dei token di autenticazione in transito	1	Buoni risultati, nessun impatto per l'utente
			Adozione di una Certification Authority centralizzata	3	Tecnicamente fattibile, necessità di gestire una Public Key Infrastructure, necessità di fare capo ad un'entità centrale
			Verifica continua del passato di un'entità + meccanismo di reputazione	2	Ottimi risultati, necessità di maggiori controlli anche ridondanti; da valutare l'accettabilità da parte dell'utente
	CAPEC-131: Identity Spoofing	3	Autenticazione multi-fattore + assicurarsi che le chiavi private degli utenti siano protette e cifrate	3	Disponibilità dell'utente a fornire un ulteriore fattore di autenticazione, richiesti ulteriori costi in termini di infrastruttura e di processi di gestione delle chiavi
			Kerberos Authentication	3	Tecnicamente fattibile, utilizzo di protocolli noti, necessità di gestione di certificati
	CAPEC-333: Counterfeit Organizations	3	Validazione decentralizzata delle operazioni + verifica incrociata delle informazioni da più "oracoli"	2	Possibile allungamento dei tempi per l'elaborazione a fronte di maggiore affidabilità, necessario un meccanismo per gestire il consenso
			Verifica continua del passato di un'entità + meccanismo di reputazione	2	Ottimi risultati, necessità di maggiori controlli anche ridondanti, da valutare l'accettabilità da parte dell'utente
	Duplicazione accidentale Product Token	2	Utilizzare identificatori univoci per le risorse	1	Implementazione semplice ma soluzione non definitiva
			Notarizzare i dati in maniera immutabile	2	Le modifiche diventano complicate a fronte di maggiore trasparenza e affidabilità, richiesta disponibilità di memoria

Figura 4.14: *Mitigation Table* relativa all'asset *Product Token*

SECURITY REQUIREMENT DEFINITION												
Asset	Requirement	Impact Requirement	Attack	Inherent Risk	Control	Cost	Residual Probability	Residual Impact	Residual Risk	Value-to-cost ratio	R.o.C	Prioritization
PRODUCT TOKEN	Violazione di Confidentiality per il requisito SR1	1	CAPEC-138: Sniffing Network Attack	3	Cifrare le informazioni sensibili che devono viaggiare su canali non sicuri	1	1	1	1	3,00	1,00	
			CAPEC-93: Man-in-the-Middle (MITM)	3	Verificare l'autenticità delle chiavi pubbliche in modo decentralizzato	2	1	1	1	1,50	0,00	
					Adottare protocolli per rendere sicure le comunicazioni (SSL/TLS)	1	1	1	1	3,00	1,00	
			Pubblicazione accidentale dati token	2	Separazione dei dati	2	1	1	1	1,50	-0,50	
	Violazione di Integrity per il requisito SR1	3	CAPEC-138: Content Spoofing	9	Notarizzare i dati in maniera immutabile	2	1	3	3	1,50	2,00	
					Tamper-resistant protocols	3	2	2	4	1,00	0,67	
			CAPEC-130: Resource Injection	6	Sanificazione degli input	1	1	3	3	3,00	2,00	
			CAPEC-193: Fake the Source of Data	9	Validazione decentralizzata delle operazioni + verifica incrociata delle informazioni da più "oracoli"	2	1	3	3	1,50	2,00	
					Affidarsi ad un provider dati centralizzato	2	2	2	4	1,50	1,50	
			Modifica accidentale Product Token	6	Notarizzare i dati in maniera immutabile	2	1	3	3	1,50	0,50	
	Violazione di Availability per il requisito SR1	3	CAPEC-607: Obstruction	6	Ridondanza e bilanciamento del carico	2	1	3	3	1,50	0,50	
					Diversità	2	1	3	3	1,50	0,50	
			CAPEC-339: Local Execution of Code	6	Eseguire tutti i file sospetti in una sandbox	1	2	2	4	3,00	1,00	
			Cancellazione accidentale Product Token	6	Notarizzare i dati in maniera immutabile	2	1	2	2	1,50	1,00	
					Ridondanza	3	2	1	2	1,00	0,33	
	Violazione di Authenticity per il requisito SR1	3	CAPEC-93: Man-in-the-Middle (MITM)	9	Verificare l'autenticità delle chiavi pubbliche in modo decentralizzato	2	1	3	3	1,50	2,00	
					Adottare protocolli per rendere sicure le comunicazioni (SSL/TLS)	1	2	2	4	3,00	4,00	
			CAPEC-193: Fake the Source of Data	9	Validazione decentralizzata delle operazioni + verifica incrociata delle informazioni da più "oracoli"	2	1	3	3	1,50	2,00	
					Affidarsi ad un provider dati centralizzato	2	2	3	6	1,50	0,50	
			CAPEC-11: Exploitation of Trusted Identifiers	9	Cifratura e firma digitale dei token di autenticazione in transito	1	2	3	6	3,00	2,00	
					Adozione di una Certification Authority centralizzata	3	2	2	4	1,00	0,67	
					Verifica continua del passato di un'entità + meccanismo di reputazione	2	2	2	4	1,50	1,50	
			CAPEC-131: Identity Spoofing	9	Autenticazione multi-fattore + assicurarsi che le chiavi private degli utenti siano protette e cifrate	2	1	3	3	1,50	2,00	
					Kerberos Authentication	3	2	2	4	1,00	0,67	
			CAPEC-333: Counterfeit Organizations	9	Validazione decentralizzata delle operazioni + verifica incrociata delle informazioni da più "oracoli"	2	2	2	4	1,50	1,50	
					Verifica continua del passato di un'entità + meccanismo di reputazione	2	2	3	6	1,50	0,50	
			Duplicazione accidentale Product Token	6	Utilizzare identificatori univoci per le risorse	1	2	2	4	3,00	1,00	
					Notarizzare i dati in maniera immutabile	2	1	3	3	1,50	0,50	

Misura adottata con priorità alta | Misura adottata con priorità media | Misura scartata

Figura 4.15: *Security Requirements Definition* per l'asset *Product Token*

4.4 Scelte tecnologiche

L'ultimo passo della progettazione ha portato, definitivamente, all'individuazione delle tecnologie da adottare per lo sviluppo vero e proprio del sistema, quantomeno a livello di infrastruttura di base.

Difatti, in ottemperanza alle buone pratiche dell'ingegneria del software, il punto di partenza non è stato, di certo, il mezzo tramite il quale implementare le funzionalità desiderate. A conferma di ciò, si noti che il termine "blockchain" non è mai stato utilizzato nei capitoli dedicati all'analisi del sistema in esame, se non per qualche sporadico riferimento; piuttosto, l'idea dell'utilizzo di tale tecnologia è frutto di un processo di maturazione iniziato con la modellazione dei requisiti e proseguito con le valutazioni di sicurezza e le conseguenti considerazioni architetturali, trovando la propria consacrazione solo al termine di queste attività.

4.4.1 Framework per la valutazione dell'applicabilità delle blockchain

Giungendo alle ultime scelte di progettazione, dunque, la tecnologia blockchain si presenta come una seria candidata per costituire le fondamenta del sistema di certificazione di beni di lusso discusso con la presente tesi. A questo punto, però, si è resa necessaria una metologia più rigorosa per valutarne l'adeguatezza in relazione agli obiettivi da perseguire. A tal fine, sono risultati utili i framework proposti negli articoli Scriber [2018] e Lo *et al.* [2017].

Sostanzialmente, per determinare se l'architettura del sistema in esame potesse beneficiare o meno dell'adozione delle blockchain, si è cercato di dare una risposta precisa ad una serie di domande guida che prendono in considerazione alcune caratteristiche peculiari, a livello di *design*, di tale tecnologia.

La Figura 4.16 mostra il diagramma di flusso seguito per valutare l'aderenza delle blockchain al problema da affrontare. Ogni domanda prevede esclusivamente risposta affermativa o negativa e, di conseguenza, costituisce un nodo decisionale che permette di creare un percorso ben definito. I possibili punti di arrivo sono due ed indicano, rispettivamente, la scelta più appropriata, tra le blockchain e un database tradizionale, in relazione alle risposte fornite. In figura, inoltre, sono evidenziate in rosso le risposte fornite.

Innanzitutto, è stato necessario interrogarsi sulla presenza o meno di molteplici attori all'interno dello scenario considerato. La risposta è stata affermativa; come già discusso in precedenza, infatti, le *supply-chain*, siano esse riferite a prodotti di lusso o di altra natura, richiedono accordi complessi e dinamici tra più parti, con vincoli normativi e logistici non uniformi, in quanto sottoposti a diversi confini giurisdizionali.

Le blockchain, in virtù di ciò, sono in grado di fornire un'infrastruttura condivisa con un governo neutrale in cui nessuno dei partecipanti può imporre la propria autorità, con il fine ultimo di eliminare la necessità di intermediari che gestiscono le operazioni o le transazioni tra le parti.

La seconda domanda è riferita alla necessità di una *trusted authority*. In tal caso, la riflessione che ha portato a dare una risposta è stata più complessa. Difatti, se è vero che l'intento del sistema *LuxuryChain* è quello di considerare alla pari tutti gli attori coinvolti, rivolgendo ad ognuno di essi le medesime attenzioni in termini di monitoraggio delle azioni, è altrettanto vero che l'attore *Brand* è l'unico che ha la facoltà di inserire dei prodotti all'interno del sistema, generandone il relativo certificato. A tal proposito, l'ipotesi di inaugurare la catena del tracciamento dei passaggi di proprietà a partire da un bene finito che esce dal *Brand* presuppone un minimo livello di fiducia nei confronti di quanto trascritto, dall'attore in questione, all'interno di un *Product Token*.

Ad ogni modo, la forma di *trusted authority* individuata non è così forte e può essere decentralizzata senza troppe difficoltà. Il *Brand*, effettivamente, non ha mansioni di controllo

vero e proprio sul sistema nè dispone di privilegi di modifica delle policy in gioco o delle modalità di esecuzione di determinate operazioni; esso, più che altro, assume il ruolo di un "oracolo" che registra le informazioni del mondo esterno sulla blockchain; dopodiché, è il sistema stesso ad assicurarsi di preservare l'integrità di tali informazioni. In conseguenza di ciò, gli utenti possono spostare la loro fiducia da un singolo ente specifico, o da eventuali intermediari, al software blockchain e alla comunità dei nodi di elaborazione che vi è dietro, incentivata alla buona condotta tramite meccanismi di consenso distribuito.

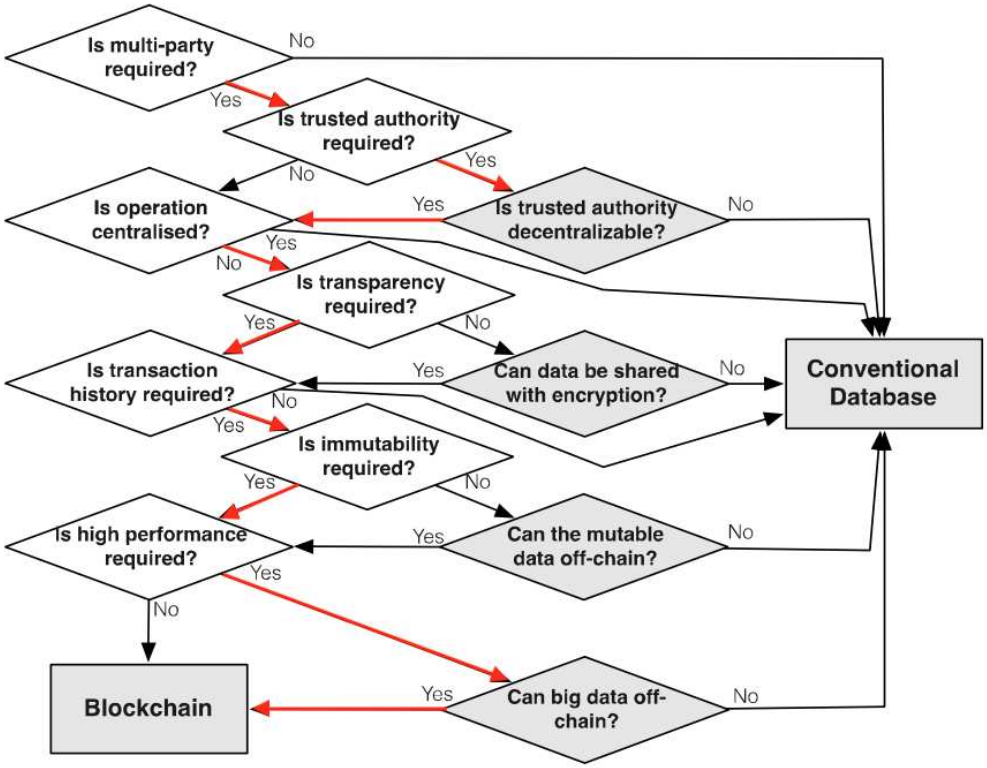

Figura 4.16: Framework adottato per valutare l'applicabilità delle blockchain

Un ulteriore passo in direzione dell'adozione delle blockchain deriva dalla constatazione che le operazioni da svolgere, all'interno del sistema *LuxuryChain*, non sono centralizzate; ogni singolo utente, infatti, è in controllo dei propri dati e dei propri asset senza una supervisione centrale; affinché ciò sia possibile, però, le interazioni tra le singole parti mutuamente non fidate, come nel caso di acquisto di seconda mano, devono essere regolamentate ed automatizzate a livello di codice. A tale esigenza risponde, in maniera naturale, il concetto di smart contract, il cui operato può essere ritenuto affidabile in quanto il codice al suo interno è difficilmente modificabile e direttamente verificabile.

Proseguendo, la necessità di trasparenza all'interno del sistema non è in discussione, trattandosi di uno dei requisiti maggiormente richiesti per la certificazione dell'autenticità. Effettivamente, come già anticipato nel corso dell'analisi dei rischi, gli obiettivi di confidenzialità sono marginali o, comunque, non riguardano l'attestato di originalità di un prodotto di lusso. La trasparenza, inoltre, va a favore delle capacità di auditing del sistema perché, nelle blockchain in grado di eseguire gli smart contract, rende possibile effettuare controlli mirati sulle condizioni e le clausole programmate a livello di codice.

Inoltre, di grande importanza per il sistema è l'integrità della storia delle transazioni relative allo scambio della risorsa *Product Token*. Tale caratteristica, infatti, è la chiave per creare la nozione di "provenienza" che, a sua volta, può essere utilizzata per tracciare i

beni fisici nel corso dei relativi cambiamenti di proprietà e di gestione. In relazione a ciò, è fondamentale riuscire a capire il *workflow* di un bene all'interno del sistema, determinando con esattezza chi è l'ultimo proprietario o chi ne è entrato in possesso in precedenza. Un sistema in cui i partecipanti devono essere mappati specificatamente alle transazioni che effettuano, così come nello scenario in esame, possono beneficiare delle blockchain, le quali richiedono un processo di firma digitale.

Allo stesso modo, come emerso dall'analisi dei requisiti, l'immutabilità dei dati è un requisito profondamente desiderato, specialmente per ciò che riguarda il certificato di autenticità connesso ad un bene fisico, ossia il *Product Token*. Di conseguenza, poichè uno degli obiettivi primari, a livello architetturale, consiste nell'assicurarsi che i diversi attori presenti nell'ecosistema non possano cambiare i dati storici (ad esempio, relativi alla qualità di un prodotto), le blockchain offrono, sicuramente, benefici significativi; esse, infatti, forniscono persistenza e, al tempo stesso, sono basate su solidi supporti crittografici volti a raggiungere gli obiettivi di inalterabilità e non ripudio.

Ad ogni modo, a livello di design, è da tenere in considerazione che le blockchain non hanno capacità di *rollback*; dunque, non si potrà far altro che "convivere" con quanto inserito al loro interno.

Infine, l'ultimo quesito riguarda le performance richieste. Diverse implementazioni blockchain esistenti, infatti, non sono di certo note per l'efficienza dal punto di vista delle prestazioni, a causa del rigoroso framework crittografico sottostante, dell'utilizzo di un'unica lista concatenata come struttura dati e, non meno importante, dei meccanismi di consenso per la verifica delle transazioni.

Il sistema in esame, dovendo rispecchiare le consuete dinamiche di compravendita di prodotti, non può prevedere tempi particolarmente lunghi per la verifica dell'autenticità e la notarizzazione di una transazione che coinvolge il trasferimento di un *Product Token*, bensì deve operare in regime di *near-real time*. In tal senso, si deve tenere conto anche che maggiore è la mole di dati di cui deve essere garantita la persistenza e l'immutabilità e più le prestazioni del sistema ne risentono; le blockchain, infatti, non sono indicate come soluzioni di *storage* per i *Big Data*. Nonostante ciò, è previsto che tutte quelle informazioni secondarie legate ad un prodotto, non interessanti dal punto di vista dell'originalità, siano memorizzate al di fuori del *Product Token*, utilizzando, ad esempio, quello che è stato modellato come *Sistema Informativo Prodotti* il quale, di fatto, agirebbe *off-chain*.

In aggiunta, per ottenere maggiori garanzie sulle prestazioni, possono essere valutate anche infrastrutture private o consorziali con mirate strategie di *tuning* in fase implementativa.

Alla luce di tali considerazioni, dunque, si giunge alla conclusione che fondare l'architettura del sistema oggetto di sviluppo sulla tecnologia blockchain costituisce un'opportunità interessante. Le modalità attraverso cui ciò avverrà e i dettagli del processo di implementazione saranno descritti nel Capitolo 5.

CAPITOLO 5

Implementazione e manuale utente

In questo capitolo verranno messi in luce gli aspetti salienti, a livello implementativo, del sistema LuxuryChain per la certificazione dell'autenticità di beni di lusso tramite blockchain.

In primo luogo, verranno descritte le scelte di codifica inerenti alla logica operativa dell'applicazione, soffermandosi, dunque, sul lato back-end e sulla programmazione degli smart contract. A tal proposito, grande enfasi verrà posta nei confronti delle strategie adottate per la gestione degli aspetti più delicati, come, ad esempio, la restituzione di un prodotto a seguito dell'acquisto.

Successivamente, saranno illustrati i processi di testing e deploy che hanno completato la prima parte dello sviluppo, fornendo dettagli sulle reti di prova utilizzate, sul consumo di risorse stimato per gli smart contract codificati e su alcuni accorgimenti adottati in fase di messa in opera.

Archiviata la discussione del back-end, l'attenzione si sposterà sull'implementazione del front-end dell'applicazione decentralizzata, con una panoramica sulle strategie di realizzazione delle diverse sezioni dell'applicazione. In quest'ottica, l'attenzione verrà posta, più che altro, sulle modalità di interfacciamento con la blockchain sottostante.

Tra le diverse tecnologie adottate, un tool risultato particolarmente utile in sede di sviluppo UI è TheGraph, le cui caratteristiche saranno esaminate nel dettaglio.

Infine, il capitolo si concluderà riportando alcune istruzioni utili ai fini dell'installazione e dell'avvio dell'applicazione, mostrando anche i risultati di una tipica esperienza di navigazione.

5.1 Back-end dell'applicazione

La prima fase dello sviluppo software si è concentrata sull'implementazione della logica sottostante al sistema *LuxuryChain*, secondo la quale dovranno essere elaborati, successivamente, i dati provenienti dall'interazione con gli utenti. La parte di *back-end*, destinata alla realizzazione delle funzionalità *core*, è risultata una componente nevralgica per l'intero esito del lavoro ed anche quella di maggiore interesse; è qui, infatti, che è entrata in gioco l'infrastruttura blockchain, la quale ha permesso di realizzare, a tutti gli effetti, un'applicazione decentralizzata (dApp).

5.1.1 Solidity ed Ethereum Virtual Machine

Solidity è uno dei più noti linguaggi di programmazione ad alto livello per l'implementazione di smart contract sulla blockchain *Ethereum* e su altre piattaforme *Ethereum-based*; esso, dunque, consente la scrittura di codice per eseguire operazioni automatizzate e verificabili all'interno di una blockchain.

Nato nel 2014, Solidity è un linguaggio *object-oriented* ispirato a C++, Python e JavaScript; tra le *feature* più caratteristiche, prevede la tipizzazione statica delle variabili ed, inoltre,

supporta l'ereditarietà, l'utilizzo di librerie e la definizione di tipi di dato complessi ad opera degli utenti.

Un file scritto utilizzando il linguaggio Solidity ha estensione `.sol` e, come intestazione, presenta due elementi ricorrenti; il primo è un commento che indica la licenza, espressa tramite un identificatore SPDX leggibile a macchina; segue poi la keyword **`pragma`**, la quale permette di specificare la versione di Solidity, utile in fase di compilazione del file.

Un file `.sol` può contenere la definizione di un numero arbitrario di smart contract, processo che avviene per mezzo della keyword **`contract`** seguita dal nome del contratto.

Un contratto può essere assimilato al concetto di classe e, proprio come una classe, presenta una serie di attributi che ne determinano lo stato, i cui valori verranno permanentemente memorizzati nello *storage*, ossia all'interno della blockchain stessa.

Per la definizione delle variabili, siano esse di stato o meno, Solidity distingue tra tre tipi principali:

- *Value Types*, ossia variabili che saranno passate per valore; tale categoria comprende interi, booleani e *address*, questi ultimi utilizzati per identificare i *wallet*.
- *Reference Types*, cioè variabili che possono essere passate per riferimento; rientrano in questa famiglia gli array, le stringhe e i tipi definibili dagli utenti sotto forma di *struct*. Per variabili di tipo riferimento deve essere specificata la locazione, scegliendo tra *memory, storage e calldata*.
- *Mapping Types*, ossia particolari tipi di variabili, analoghe alle *hash table*, costituite da coppie chiave-valore.

Oltre alle variabili di stato, parte integrante di uno smart contract sono le funzioni, cioè quelle unità eseguibili di codice che, a tutti gli effetti, si traducono nelle transazioni da inoltrare, verificare e registrare sulla blockchain. È proprio qui che subentra il meccanismo del gas, il quale determina il costo, in termini di risorse consumate, delle istruzioni eseguite all'attivazione di una funzione.

Ad ogni modo, man mano che, nel corso del tempo, vengono eseguite transazioni nei confronti di uno smart contract, richiamando le funzioni in esso definite, può avere luogo il cambiamento delle variabili di stato, facendo evolvere, di conseguenza, la blockchain stessa.

Proseguendo, altri elementi rilevanti all'interno della struttura tipica di un contratto sono i modificatori, utilizzati per rettificare o estendere la semantica delle funzioni in modo condizionale, gli eventi, che possono essere emessi per fini di *logging*, e gli errori, definibili per descrivere situazioni di esecuzione fallimentare.

Ulteriori peculiarità del linguaggio Solidity verranno illustrate contestualmente al commento degli smart contract realizzati, nelle sezioni successive.

Una volta compilato, in ogni caso, uno smart contract può essere eseguito per mezzo di una macchina virtuale nota come *Ethereum Virtual Machine* (EVM), realizzata per evitare problemi di compatibilità con la piattaforma hardware a disposizione.

A tal proposito, quando uno smart contract viene caricato sulla blockchain, operazione nota come *deploy*, il programma Solidity viene tradotto in un codice di basso livello chiamato *bytecode*, che sarà quello mandato in esecuzione, a tutti gli effetti, dai diversi *peer* della rete adottata.

5.1.2 Ambiente di sviluppo e struttura della cartella di lavoro

Per lo sviluppo software, nel *back-end* così come nel *front-end*, è stato utilizzato l'IDE *VisualStudio Code*.

Relativamente al *back-end*, inoltre, si è fatto ricorso ad *Hardhat*, un framework molto noto per lo sviluppo professionale di smart contract, basato su linguaggio *JavaScript SS (Server Side)*.

Hardhat, facilmente integrabile all'interno di *VisualStudio Code* tramite apposita estensione, offre un ambiente di sviluppo altamente flessibile, costituito da una serie di componenti che consentono di eseguire il *deploy* dei contratti, nonchè il *testing* e il *debugging* del codice Solidity in maniera agevole.

Il componente principale con il quale è possibile interagire è *Hardhat Runner*, un esecutore di task che consente di gestire ed automatizzare compiti ricorrenti relativi allo sviluppo della dApp; lanciando il task `hardhat deploy` da linea di comando, ad esempio, è possibile mandare in esecuzione gli script che effettuano il *deploy* degli smart contract sulla rete. A tal proposito, un altro strumento molto utile, specialmente in fase di testing, è *Hardhat Network*, il quale consente di interfacciarsi con una rete locale *Ethereum* progettata appositamente per lo sviluppo, senza dover ricorrere necessariamente ad ambienti reali.

Sfruttando il framework in questione, dunque, si hanno già tutti gli strumenti necessari per procedere alla realizzazione di una dApp. Una volta effettuata l'inizializzazione di Hardhat per mezzo del gestore di pacchetti *yarn* (o equivalentemente *npm*), infatti, il progetto ha assunto la seguente struttura, in termini di cartelle e file:

- cartella `contracts`, destinata ai file `.sol` in cui sono stati definiti gli smart contract;
- cartella `node_modules`, relativa a tutte le dipendenze necessarie per il corretto funzionamento degli smart contract;
- cartella `scripts`, in cui sono stati inseriti opportuni file *JavaScript* da mandare in esecuzione per esigenze specifiche, come il caricamento dei contratti su una rete o l'interazione con i contratti stessi;
- cartella `tests`, dedicata ai test delle funzionalità degli smart contract;
- file `hardhat.config.json`, in cui sono stati forniti parametri utili ad Hardhat per la connessione alle reti blockchain specifiche, per il processo di compilazione e per la connessione ad eventuali API;
- file `package.json`, il quale elenca tutte le dipendenze del progetto.

Tale organizzazione predefinita proposta da Hardhat, pur rimanendo inalterata nel suo scheletro, ha subìto alcune aggiunte nelle fasi successive dello sviluppo.

5.1.3 Smart Contract "Product NFT" e libreria OpenZeppelin

Il primo contratto implementato è stato quello relativo alla generazione del *Product Token*. I requisiti di tale risorsa, sulla base di quanto illustrato nei capitoli precedenti, trovano una corrispondenza pressoché "uno ad uno" con le proprietà che contraddistinguono il concetto di NFT.

Il token di un prodotto, infatti, è pensato come un asset digitale dotato di caratteristiche di unicità che, rispecchiando le connotazioni di un bene di lusso, riveste il ruolo di certificato di autenticità.

Traducendo il *Product Token* in un NFT, dunque, è possibile garantirne la tracciabilità; tutte le transazioni che coinvolgono un NFT, difatti, vengono registrate sulla blockchain, il che provvede a fornire trasparenza e immunità da eventuali contraffazioni.

Nell'ottica del sistema *LuxuryChain*, inoltre, disporre della controparte digitale di un bene di lusso significa attestarne la proprietà, oltre che l'originalità. Le caratteristiche degli NFT, a

tal proposito, consentono di realizzare la cosiddetta *"proof of possession"*, sulla quale fondare una gestione inequivocabile della catena di *ownership*; gli NFT, infatti, sono incorruttibili e presentano una storia permanente relativa ai diversi proprietari, fino a risalire a chi li ha generati.

Oltre alle considerazioni più tecniche, va sottolineato che gli NFT, per loro natura, si prestano particolarmente alla rappresentazione di beni ad elevata complessità, variabili, ad esempio, in quanto a taglia, colore, modello e forma, proprio come possono esserlo i prodotti che rientrano nel lusso personale. Non da meno, la rarità e la limitatezza che contraddistinguono, in molti casi, beni di questo tipo si riflettono nel senso di scarsità, in termini di quantità, e nella percezione di valore degli NFT correlati.

Il Listato 5.1 mostra la codifica, in linguaggio Solidity, dello smart contract *ProductNft*, a partire dal quale, come suggerisce il nome, è possibile generare gli NFT che rappresentano i certificati da associare ai beni di lusso fisici.

```
1  // SPDX-License-Identifier: MIT
2
3  pragma solidity ^0.8.7;
4
5  // Imports
6  import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
7  import "hardhat/console.sol";
8  import "@openzeppelin/contracts/utils/Strings.sol";
9  import "@openzeppelin/contracts/access/AccessControlEnumerable.sol";
10
11 contract ProductNft is ERC721, AccessControlEnumerable {
12     // Brand Role definition for OpenZeppelin Access Control
13     bytes32 public constant BRAND_ROLE = keccak256("BRAND_ROLE");
14
15     // State Variables
16     uint256 private s_tokenCounter;
17     address private s_owner;
18     string internal s_tokenURI;
19
20     // Errors
21     error ProductNft__NotOwner();
22     error ERC721Metadata__URI_QueryFor_NonExistentToken();
23
24     // Events
25     event MintedNFT(uint256 indexed tokenId, address indexed firstOwner);
26
27     // Constructor
28     constructor(
29         string memory tokenUri
30     ) ERC721("Luxury Chain Certificate", "LCC") {
31         _grantRole(BRAND_ROLE, msg.sender);
32         s_tokenCounter = 0;
33         s_owner = msg.sender;
34         s_tokenURI = tokenUri;
35     }
36
37     // Main Functions
38
39     // NFT Creation
40     function mintNft(address _to) public onlyRole(BRAND_ROLE) {
41         address defaultAddress; //0x0000000000000000000000000000000000000000
42         if (_to != defaultAddress) {
43             _safeMint(_to, s_tokenCounter); // ERC721 inherited function
44             s_tokenCounter += 1; // ID update
45             emit MintedNFT(s_tokenCounter - 1, msg.sender);
46         } else {
```

```
47              _safeMint(msg.sender, s_tokenCounter); // ERC721 inherited function
48              s_tokenCounter += 1; // ID update
49              emit MintedNFT(s_tokenCounter - 1, msg.sender);
50          }
51      }
52
53      // Getters
54      function getTokenUri() public view returns (string memory) {
55          return s_tokenURI;
56      }
57
58      function getContractOwner() public view returns (address) {
59          return s_owner;
60      }
61
62      // ------------------------------------------
63      // Overrides REQUIRED by Solidity.
64
65      function supportsInterface(
66          bytes4 interfaceId
67      ) public view override(ERC721, AccessControlEnumerable) returns (bool) {
68          return super.supportsInterface(interfaceId);
69      }
70  }
```

Listing 5.1: Smart Contract per la creazione del token di un prodotto

La creazione degli NFT, come discusso nel Capitolo 1, avviene tramite standard di tokenizzazione ormai assodati, il più noto dei quali, utilizzato anche in questa sede, è ERC-721.

Per costruire NFT secondo lo standard ERC-721 è possibile sfruttare la libreria *open-source* `OpenZeppelin`, la quale fornisce una vasta gamma di smart contract predefiniti e solidamente testati, utili per diverse esigenze. Tali contratti, infatti, sono implementati secondo le *best practices* di sicurezza assodate nel mondo blockchain ed includono protezioni contro vulnerabilità legate, ad esempio, ad *overflow* aritmetici, alla gestione degli errori, ad autorizzazioni di accesso e molto altro. Oltre a ciò, la comunità OpenZeppelin e i team di sviluppo sottopongono i contratti a revisione e aggiornamento costante, in modo da riflettere gli sviluppi della tecnologia blockchain ed affrontare prontamente nuove sfide.

Lo smart contract "ProductNFT" mostrato nel Listato 5.1, nello specifico, estende due contratti fondamentali importati da `OpenZeppelin`, ossia `ERC721.sol` e `AccessControlEnumerable.sol`. Il primo, come anticipato, raccoglie le funzionalità necessarie per gestire token non fungibili. Il secondo, invece, permette di stabilire un modello di controllo degli accessi alle funzionalità dello smart contract basato su ruoli; nel caso specifico, è stato utilizzato per limitare il privilegio di generazione dei token esclusivamente a coloro i quali rispondono al ruolo del "Brand". Tale ruolo viene assegnato, all'atto della creazione del contratto, all'indirizzo dal quale viene effettuato il *deploy*.

Infatti, proprio come una classe, uno smart contract prevede un metodo costruttore che viene richiamato esclusivamente all'atto di "istanziazione", utile per passare parametri dall'esterno e fornire una "configurazione" di base.

Il costruttore mostrato nel Listato 5.1, in realtà, richiama, a sua volta, il costruttore specifico del contratto *ERC721* ereditato. Solo in questo modo, infatti, è possibile utilizzare lo smart contract *ProductNft* per generare nuovi token non fungibili. A tal proposito, come richiesto, sono stati passati due parametri che indicano il nome e il simbolo da attribuire agli NFT di questa classe. Il nome scelto è *"Luxury Chain Certificate"* ed il simbolo è l'acronimo *"LCC"*. Oltre a ciò, per mezzo del costruttore, viene memorizzato il proprietario del contratto (colui

che invia la transazione di *deploy*), il valore di partenza dell'ID da associare ai token ed, infine, il *tokenURI* prelevato dal parametro di input.

Il *token ID* rappresenta univocamente un asset digitale ed è associato, in ogni istante, ad un unico proprietario. Per la generazione degli identificatori è stato utilizzato un semplice contatore (`s_tokenCounter`) che parte da 0; esso, infatti, ha valenza solo all'interno dell'ecosistema blockchain, al fine di individuare i diversi NFT generati nel corso tempo. Identificatori più articolati provocherebbero soltanto un maggiore dispendio di risorse, considerando che andrebbero notarizzati sul registro.

Il *tokenURI*, invece, è destinato ad ospitare i metadati relativi ad un NFT, ossia gli attributi caratteristici che lo rendono unico, utili anche per poterne visualizzare l'aspetto fisico. A tal proposito, dato che le spese per il consumo di gas possono schizzare in alto in piattaforme come *Ethereum*, c'è da chiedersi se sia opportuno, per descrivere le sembianze dei diversi asset, memorizzare una gran quantità di dati, ed eventualmente immagini, direttamente *on-chain*.

Alla luce di tali riflessioni, è stato introdotto lo standard *tokenURI*. L'URI (*Unique Resource Identifier*) è un indicatore univoco ed universalmente valido che definisce come un certo token appare e quali sono i suoi attributi. Nel contesto in esame, dunque, può essere utilizzato per scrivere informazioni sul produttore di un bene di lusso, sulla data di creazione, sugli standard di qualità soddisfatti ed altri dati utili per accertarne l'autenticità.

Proseguendo, all'interno dello smart contract *ProductNft*, la funzione più importante e degna di nota è `mintNft`, la quale provvede alla generarazione di un nuovo token. Tale funzione è attivabile soltanto da chi risponde al ruolo del "Brand" (si noti, infatti, l'utilizzo del modificatore `onlyRole(BRAND_ROLE)`) e richiama, al suo interno, la funzione `_safeMint` ereditata dallo smart contract *ERC-721*; è questa, a tutti gli effetti, la vera responsabile del *minting* di un nuovo NFT.

All'atto della creazione di un nuovo NFT è prevista la possibilità di specificare l'indirizzo nei confronti del quale dovrà essere effettuato il *minting*, così da risparmiare, laddove possibile, un trasferimento. Se non viene specificato nulla allora il proprietario rimane il creatore stesso.

In aggiunta, quando viene richiamata la funzione `_safeMint`, viene indicato anche l'ID univoco da associare al token, contenuto nella variabile di stato `s_tokenCounter`. Ad operazione completata, l'identificatore viene incrementato di un'unità.

Infine, sono previsti dei metodi *getters* che effettuano operazioni di lettura dello stato dello smart contract.

Oltre alle funzioni espressamente definite, ad ogni modo, lo smart contract *ProductNft* eredita anche tutta la *suite* di funzionalità propria del contratto genitore *ERC721*. Una di queste funzioni, risultata particolarmente utile ai fini del lavoro, è la `approve`; tale funzione, richiamabile solo dal legittimo proprietario di un NFT o da un indirizzo a sua volta approvato, consente di specificare un account al quale si intende concedere la delega per trasferire l'NFT in questione.

5.1.4 Smart Contract "NftMarketplace"

Assodata la generazione dei *ProductToken* per mezzo degli NFT, è stato implementato un contratto che si occuppasse di regolamentarne la gestione e, soprattutto, il trasferimento, secondo le tipiche relazioni di compravendita.

Lo smart contract *NftMarketplace*, infatti, è stato ideato come una piattaforma per lo scambio sicuro e verificato dei *Luxury Chain Certificate* (LCC), da effettuare in contemporanea con le effettive transazioni che coinvolgono i corrispondenti beni di lusso reali. A tal proposito, è necessario precisare che, a discapito dei termini "acquisto" e "vendita" utilizzati in questa

sede, ai fini del lavoro non sono state implementate operazioni di pagamento, né sotto forma di criptovaluta né per mezzo dei circuiti tradizionali.

Il contratto *NftMarketplace*, dunque, è quello a cui si rivolgeranno gli utenti per trasferire un LCC di cui sono attulmente in possesso. Più in dettaglio, chiunque decida di rivendere un prodotto di lusso, certificato tramite apposito NFT, dovrà affidare il token allo smart contract *NftMarketplace*, in modo che tutti coloro che sono interessati all'acquisto possano leggerlo ed avere una prova di autenticità. Dopodiché, quando un utente procede effettivamente all'acquisto, il trasferimento viene automatizzato dal contratto in questione. In virtù di ciò, prima di procedere alla codifica vera e propria, sono state esaminate due alternative implementative:

- trasferire la proprietà di un NFT dal venditore al *Marketplace* e, successivamente, dal *Marketplace* all'acquirente;
- concedere al *Marketplace* esclusivamente i privlegi per spostare un NFT, senza però assegnargli la proprietà.

Considerazioni riguardo il consumo di gas hanno portato ad optare per la seconda strada, nel tentativo di ridurre il più possibile le commissioni che gli utenti sono chiamati a pagare.

I dettgli implementativi dello smart contract *NftMarketplace* sono riportati nel Listato 5.2.

Una prima operazione prevista all'interno del contratto è il *listing* di un NFT, analogo alla pubblicazione di un annuncio di vendita. La funzione `listItem`, predisposta a ciò, riceve in input l'indirizzo e l'ID di un token, dopodiché verifica che il contratto *NftMarketplace* sia stato abilitato al suo trasferimento ed, in caso affermativo, aggiunge una nuova *entry* alla struttura dati che tiene traccia di tutti i token attualmente "sul mercato", espressa dalla variabile `s_listings`. Tale variabile è un *mapping type* su due livelli; al suo interno, infatti, un primo mapping ha, come chiave, l'indirizzo dell'NFT e, come valore, un ulteriore mapping innestato che, a sua volta, realizza l'associazione tra i possibili *token ID* e gli indirizzi dei rispettivi venditori.

Proseguendo la descrizione, all'interno dello smart contract *NftMarketplace* è stata codificata la funzione `buyItem`, utile ad avviare la procedura di acquisizione di un *LCC*. Il processo in questione, infatti, è stato suddiviso in due fasi; a seguito dell'acquisto di un bene di lusso, la proprietà del corrispondente certificato non viene trasferita immediatamente all'acquirente bensì si innesca un meccanismo di *staking*, utile per gestire un eventuale reso.

La funzione `buyItem`, sostanzialmente, si limita a tenere traccia dell'acquisto di un prodotto, inserendo gli estremi della transazione (venditore, acquirente, *tokenID*, indirizzo del token e timestamp) all'interno di un'apposita struttura dati identificata attraverso la variabile `s_stakings`, di tipo mapping. Il trasferimento vero e proprio, invece, avviene per mezzo di una seconda funzione, `transferTokenAfterTime`, illustrata nel seguito.

La funzione prevista per la restituzione di un prodotto, o meglio del relativo certificato, è stata chiamata `takeBackToken`. Prima di procedere, logicamente, tale funzione controlla che l'NFT interessato sia ancora nel tempo utile di reso e, per far ciò, utilizza il *modifier* `inStaking`; esso verifica che il token associato alla coppia "indirizzo - ID" fornita in input sia presente all'interno del mapping `s_stakings`.

Le modalità attraverso le quali è stata implementata la restituzione di un NFT saranno spiegate, in dettaglio, nella Sezione 5.1.5.

Ipotizzando che non avvenga nessuna restituzione entro il tempo massimo consentito (definito tramite il costruttore del contratto), l'NFT deve essere definitivamente trasferito dal venditore all'acquirente. A tal proposito, verrà richiamata la funzione `transferTokenAfterTime`, la quale richiede, in input, unicamente indirizzo ed identificatore dell'NFT; gli indirizzi di mittente e destinatario, infatti, vengono recuperati dal mapping che lega il token alla transazione ad esso riferita.

La funzione per il trasferimento di un token, come è possibile notare dal Listato 5.2, è impostata in modo tale che, prima, venga effettuato l'aggiornamento della struttura dati contenente gli estremi della transazione (`s_stakings`) e solo dopo si proceda al trasferimento. Invertire l'ordine di queste due operazioni avrebbe esposto ad una delle più note vulnerabilità in ambito blockchain, traducibile nei cosiddetti *Re-entrancy attack*, ossia attacchi che puntano a ripetere indebitamente una determinata operazione favorevole, come può essere, in questo caso, il passaggio di proprietà di un token.

Una prima pratica di sicurezza, dunque, consiste nell'effettuare eventuali transazioni sempre come ultimo step della funzione da implementare, aggiornando a monte le informazioni utili alla transazione stessa.

Un'altra possibilità per evitare il rischio di attacchi di *re-entrancy* prevede l'utilizzo di un *mutex lock*, concetto ripreso dalla gestione della sezione critica nei sistemi operativi; un *mutex lock* è, a tutti gli effetti, un "lucchetto", rappresentabile tramite una variabile booleana, che viene "chiuso" all'inizio di una funzione particolarmente delicata e "riaperto" al termine. Sfruttando questa variabile di lock, dunque, si fa in modo che, in ogni istante, le operazioni comprese tra la chiusura e la riapertura possano essere eseguite da un solo pezzo di software.

La tecnica appena descritta è facilmente integrabile grazie all'utilizzo dell'apposita libreria *ReentrancyGuard* fornita da *OpenZeppelin*. Tale libreria, infatti, ha il permesso di inserire il modifier `nonReentrant` che implementa il meccanismo del *lock*.

Oltre a ciò, per avere ulteriori garanzie di sicurezza nei confronti di possibili errori durante il trasferimento di un NFT, la funzione `transferTokenAfterTime` richiama al suo interno la funzione `safeTransferFrom`, definita nell'interfaccia *IERC721* al fine di evitare che l'NFT venga bloccato irreversibilmente.

L'ultima funzione implementata nello smart contract *NftMarketplace*, chiamata `cancelListing`, serve per rimuovere un token dal marketplace, qualora il venditore decida di ritirare il proprio annuncio.

Da evidenziare, infine, che tutte le funzioni implementate sono definite come **`external`**, in quanto verranno richiamate esclusivamente da contratti diversi da quello in questione o da altri account.

```
1  // SPDX-License-Identifier: MIT
2
3  pragma solidity ^0.8.7;
4
5  import "@openzeppelin/contracts/token/ERC721/IERC721.sol"; // Needed for token
       approval
6  import "@openzeppelin/contracts/security/ReentrancyGuard.sol"; // Needed for
       nonReentrant modifier
7
8  // ... Errors definition ...
9
10 contract NftMarketplace is ReentrancyGuard {
11     // ... Events definition ...
12
13     // Data structure for staking management
14     struct Transaction {
15         address seller;
16         address buyer;
17         uint buyTimestamp;
18         bool inStaking;
19     }
20
21     // NFT contract address -> NFT Token Id -> Seller address
22     mapping(address => mapping(uint256 => address)) private s_listings;
23
24     // NFT contract address -> NFT token ID -> Transaction data type
```

```
25      mapping(address => mapping(uint256 => Transaction)) private s_stakings;
26
27      // Deadline for product return
28      uint s_returnTime;
29
30      // ... Modifiers ...
31
32      // Constructor
33      constructor(uint returnTime) {
34          s_returnTime = returnTime;
35      }
36
37      //////////////////////
38      // Main Functions //
39      //////////////////////
40
41      // Publishing an NFT for selling
42      function listItem(
43          address nftAddress,
44          uint256 tokenId
45      )
46          external
47          checkListed(nftAddress, tokenId)
48          onlyNftOwner(nftAddress, tokenId, msg.sender)
49      {
50          IERC721 nft = IERC721(nftAddress);
51
52          // Nft listed only if Marketplace is approved
53          if (nft.getApproved(tokenId) != address(this)) {
54              revert NftMarketplace__NotApprovedForMarketplace();
55          }
56
57          s_listings[nftAddress][tokenId] = msg.sender; // Update listing
58          emit ItemListed(msg.sender, nftAddress, tokenId);
59      }
60
61      // NFT goes in staking but not transfered yet
62      function buyItem(
63          address nftAddress,
64          uint256 tokenId
65      ) external isListed(nftAddress, tokenId) nonReentrant {
66          address nftSeller = s_listings[nftAddress][tokenId];
67
68          // Updating Transaction data structure
69          s_stakings[nftAddress][tokenId].seller = nftSeller;
70          s_stakings[nftAddress][tokenId].buyer = msg.sender;
71          s_stakings[nftAddress][tokenId].buyTimestamp = block.timestamp;
72          s_stakings[nftAddress][tokenId].inStaking = true;
73
74          // Item bought, not available in marketplace anymore
75          delete (s_listings[nftAddress][tokenId]);
76
77          emit ItemBought(msg.sender, nftAddress, tokenId);
78      }
79
80      // Sending the NFT to the buyer after return time expired
81      function transferTokenAfterTime(
82          address nftAddress,
83          uint256 tokenId
84      )
85          external
86          stakeCheck(nftAddress, tokenId)
```

```
87          inStaking(nftAddress, tokenId)
88          nonReentrant
89      {
90          address from = s_stakings[nftAddress][tokenId].seller;
91          address to = s_stakings[nftAddress][tokenId].buyer;
92
93          // Reset dell'entry legata all'NFT in questione all'interno del mapping
94          delete (s_stakings[nftAddress][tokenId]);
95
96          IERC721(nftAddress).safeTransferFrom(from, to, tokenId);
97          emit Token_Transfered(from, to, nftAddress, tokenId);
98      }
99
100     // Returning an NFT
101     function takeBackToken(
102         address nftAddress,
103         uint256 tokenId
104     ) external inStaking(nftAddress, tokenId) onlyBuyer(nftAddress, tokenId) {
105         address owner = s_stakings[nftAddress][tokenId].seller;
106         delete (s_stakings[nftAddress][tokenId]);
107         emit TokenGetBack(nftAddress, tokenId, owner);
108     }
109
110     // Removing a listing: the seller doesn't want to sell the product anymore
111     function cancelListing(
112         address nftAddress,
113         uint256 tokenId
114     )
115         external
116         onlyNftOwner(nftAddress, tokenId, msg.sender)
117         isListed(nftAddress, tokenId)
118     {
119         delete (s_listings[nftAddress][tokenId]);
120         emit ItemCancelled(msg.sender, nftAddress, tokenId);
121     }
122
123     // ...Getters....
124 }
```

Listing 5.2: Smart Contract *NftMarketplace*

5.1.5 Gestione del reso di un NFT ed automazione del trasferimento

Particolarmente interessante e stimolante, nell'ottica dello smart contract *NftMarketplace*, si è rivelata la gestione dei meccanismi di reso e dell'eventuale trasferimento automatizzato di un token.

Come anticipato, dopo aver richiamato la funzione `buyItem` del Listato 5.2, l'NFT non viene direttamente trasferito ma entra in uno stato concettualmente paragonabile allo *staking*, fino a quando non viene richiesto un reso o scade il tempo utile alla restituzione. Nonostante tale soluzione, a primo impatto, possa sembrare articolata, ad un'attenta analisi si può rivelare economicamente vantaggiosa. Effettuare il passaggio di proprietà contestualmente all'acquisto, infatti, renderebbe necessario ripetere il trasferimento dell'NFT a parti invertite, in caso di reso; un'ulteriore transazione, dunque, si traduce in costi aggiuntivi.

Chiarita la necessità di avere due funzioni separate, una per l'acquisto e una per il trasferimento vero e proprio, da richiamare eventualmente in un secondo momento, non rimane che descrivere come sia stato implementato lo *staking*.

Il reale compito della funzione `buyItem`, alla luce di ciò, consiste nel memorizzare i dati sulla transazione di acquisto in corso. A tal proposito, è stato creato, a monte, un tipo di dato

strutturato *ad hoc*, chiamato `Transaction`, il quale prevede, tra i suoi campi, l'indirizzo del venditore, l'indirizzo dell'acquirente, il *timestamp* dell'acquisto e un booleano che assume valore **`true`** se il token è effettivamente in *staking*. La struttura `Transaction` costituisce una parte integrante della variabile `s_stakings`, definita come un mapping su due livelli; nel momento in cui si accede a tale variabile, a partire dall'indirizzo di un NFT, è possibile risalire ai diversi *token ID* collegati; dopodiché, utilizzando uno specifico ID come chiave, si può leggere la `Transaction` associata a quel determinato token.

La funzione `buyItem`, dunque, verifica che il token fornito in input sia effettivamente in vendita, recupera l'indirizzo del venditore dalla variabile `s_listings` e popola la `Transaction` associata al `tokenId` specificato, memorizzando il venditore, l'acquirente (cioè l'indirizzo che ha chiamato la `buyItem`) e il timestamp, prelevato dal blocco tramite il quale la transazione verrà registrata sulla blockchain; dopodiché, la variabile *booleana* che indica la presenza del token nello *staking* viene settata a **`true`** e, come ultimo step, si provvede a rimuovere il token dalla lista di vendita.

Una volta terminata l'esecuzione della funzione `buyItem`, idealmente, si avvia il periodo utile alla restituzione.

La funzione per effettuare il reso, `takeBackToken`, prende in input indirizzo ed identificatore del token e, dopo essersi assicurata che esso sia effettivamente in *staking*, a seguito di un acquisto, provvede semplicemente a rimuovere la `Transaction` associata all'NFT in questione dalla struttura dati `s_stakings`. Così facendo, dunque, non risulterà esserci più alcun passaggio di proprietà in attesa di essere formalizzato e, come anticipato, si evita una transazione.

A questo punto, si è reso necessario un meccanismo che, rispondendo comunque al principio della decentralizzazione, fosse in grado di attivare la funzione `transferTokenAfterTime` sulla base di un *trigger*, quindi senza dover interagire direttamente con essa.

Nello specifico, il trasferimento del token deve avvenire allo scadere naturale del periodo di reso. La nozione temporale, però, è un concetto estraneo all'ecosistema "chiuso" che caratterizza una blockchain; nel momento in cui si intende far interagire uno smart contract con il mondo reale, prelevando da esso dati variabili o determinati eventi, infatti, c'è il rischio di contravvenire al principio di determinismo che regola le blockchain, nel senso che i diversi nodi della rete potrebbero fornire risultati diversi senza riuscire, di conseguenza, a raggiungere un consenso.

Per ovviare a questa problematica, si è fatto ricorso al tool *OpenZeppelin Defender*, il quale offre un modulo *Actions* che consente di implementare una logica personalizzata per lo svolgimento di operazioni *on-chain* e *off-chain*. Seppur nato in ambito di *Incident Response*, il modulo in questione può essere sfruttato per automatizzare le funzionalità di uno smart contract ed eseguire determinate azioni come conseguenza di un *monitor alert*.

Le azioni vengono definite creando degli *autotask*, ossia pezzi di codice *JavaScript* automatizzati, eseguibili tramite dei trigger emanati da una "sentinella".

I trigger possono essere impostati sulla base di una frequenza predefinita, in concomitanza con l'invio di una richiesta *HTTP POST* oppure, come nel caso in questione, al verificarsi di determinate condizioni in uno smart contract.

Tramite l'interfaccia web di *OpenZeppelin Defender*, dunque, è stata creata una sentinella per il monitoraggio dello smart contract *NftMarketplace*, incaricata di sollevare un *alert* nel momento in cui la transazione legata alla funzione `buyItem` ha successo. Dopodiché, la configurazione della sentinella è stata completata indicando il nome dell'*autotask* da eseguire in risposta ad una segnalazione.

Prima di passare alla codifica dell'*autotask*, però, è stato altresì necessario definire un *Relayer*, componenente che permette di inviare transazioni *on-chain* per mezzo di *OpenZeppelin*

Defender. Il processo di definizione e *deploy* di un *Relay Client* ha previsto la codifica di un apposito script che, sostanzialmente, effettua una chiamata API verso *Defender*.

L'azione da automatizzare, infine, è stata implementata all'interno di un file *JavaScript* inserito nella cartella *autotask*, creata per l'occasione nella *directory* del progetto. Il Listato 5.3 mostra lo script che realizza l'*autotask* per l'attivazione della funzione di trasferimento dell'NFT, scaduto il tempo di reso.

Il nucleo dello script è costituito da una funzione asincrona `handler`, richiamata quando la sentinella entra in azione. Tale funzione riceve, come parametro di input, un file *JSON* che contiene i dati dell'evento scatenante; a partire da esso, dunque, vengono estratte alcune informazioni di interesse, quali l'indirizzo dello smart contract monitorato e la coppia *(indirizzo, ID)* dell'NFT corrispondente al prodotto acquistato.

A questo punto, viene stabilita una connessione con il *Relayer* inizializzato in precedenza, affinché si possa interfacciare con lo smart contract *NftMarketplace*. In primo luogo, infatti, deve essere letto il tempo utile alla restituzione di un prodotto, parametro che, dopo opportuna conversione, viene passato alla funzione ausiliaria `delay`. Tale funzione, di fatto, sospende l'esecuzione dello script per il lasso di tempo specificato; a tal proposito, la sua attivazione è preceduta dalla *keyword* `await` che realizza un meccanismo di esecuzione asincrona. Ciò è in linea con l'obiettivo di posticipare il trasferimento del token allo scadere del periodo di reso.

Una volta che lo script viene "risvegliato", è necessario accertarsi che l'NFT sia ancora in *staking*, accedendo alla struttura dati `Transaction` ad esso legata; difatti, se la variabile *boolena* `inStaking` è posta a **`false`** vuol dire che, nel frattempo, l'acquirente ha esercitato il diritto di recesso, perciò si può uscire dal flusso di esecuzione senza alcuna azione; se, invece, la variabile è posta a **`true`** allora viene richiamata la funzione `transferTokenAfterTime` dello smart contract.

```
1  // Autotask for automating code execution using Open Zeppelin Defender
2
3  // ... Required modules ...
4
5  // Auxiliary function for setting up a delay
6  function delay(ms) {
7    return new Promise((resolve) => setTimeout(resolve, ms));
8  }
9
10 // JS Function to be called when sentinel alerts
11 async function handler(event) {
12   const evt = event.request.body.events[0];
13   const abi = evt.sentinel.abi;
14   const contractAddress = ethers.utils.getAddress(evt.matchReasons[1].address);
15   console.log(contractAddress);
16
17   // Extracting NFT parameters from the body of JSON generated by sentinel
18   const nftAddress = ethers.utils.getAddress(evt.matchReasons[1].args[0]);
19   const tokenId = parseInt(evt.matchReasons[1].args[1]);
20   console.log(nftAddress);
21   console.log(tokenId);
22   console.log("-------------------");
23
24   const matches = [];
25   matches.push({ nftAddr: nftAddress });
26   matches.push({ id: tokenId });
27
28   // The new signer will be the Open Zeppelin Defender Relay
29   const provider = new DefenderRelayProvider(event);
30   const signer = new DefenderRelaySigner(event, provider, { speed: "fast" });
31   const marketplace = new ethers.Contract(contractAddress, abi, signer);
32
```

```
33    // Extracting delay time for executing the token transfer
34    let requiredTimeInSeconds = await marketplace.getReturnTime();
35    console.log(requiredTimeInSeconds);
36    console.log("----------------------");
37
38    // Applying delay ...
39    await delay(requiredTimeInSeconds * 1000);
40
41    // Reading NFT's associated Transaction
42    const transaction = await marketplace.getTransaction(nftAddress, tokenId);
43    const inStaking = transaction[3];
44    if (!inStaking) {
45      console.log("Rilevata restituzione: nessuna azione da intraprendere");
46      return { matches };
47    } else {
48      console.log(
49        "Tempo utile di restituzione terminato: trasferimento automatico token"
50      );
51
52      // Finally calling the transfer function
53      const tx = await marketplace.transferTokenAfterTime(nftAddress, tokenId);
54      await tx.wait(1);
55      return { matches };
56    }
57  }
```

Listing 5.3: Script che realizza l'attivazione automatizzata e ritardata del trasferimento di un token

Una volta configurata, dunque, l'interazione tra sentinella, *relayer* ed *autotask* ha permesso di raggiungere con successo gli obiettivi sperati, permettendo, tra l'altro, di non demandare a nessuno dei due diretti interessati, cioè venditore e acquirente, il compito di richiamare la funzione per trasferire l'NFT associato ad un bene di lusso.

Come nota a margine, va specificato che, per finalità di test, il tempo utile ad effettuare il reso, da attendere prima di effettuare il passaggio automatico del token, è stato definito nell'ordine di qualche ora, anziché dei giorni, come negli scenari reali.

Oltre a ciò, il corretto funzionamento di *OpenZeppelin Defender* ha richiesto, a monte, il *deploy* degli smart contract su una blockchain attiva che non fosse quella locale.

5.2 Testing e deploy

In seguito all'implementazione delle funzionalità principali del *back-end*, una fase altrettanto importante del lavoro ha previsto la realizzazione di test finalizzati a stabilizzare il codice ed assicurarsi che tutto funzionasse come previsto, in modo da procedere in maniera relativamente sicura al successivo *deploy* dei contratti su una *TestNet* reale.

5.2.1 Unit test degli smart contract

Eseguire dei test è fondamentale nel processo di sviluppo degli smart contract; buona parte dell'esito del software, infatti, dipende dalla capacità di scrivere test efficaci.

Assicurarsi che il codice si comporti come preventivato in fase di realizzazione, inoltre, risulta di grande rilievo specialmente nel mondo decentralizzato delle blockchain, dove gli smart contract, una volta registrati, non possono essere modificati, sono visibili a tutti ed, in linea di principio, chiunque vi può interagire, motivo per cui sono potenzialmente soggetti ad *exploit*. Scrivere dei test robusti, dunque, è sicuramente la prima linea di difesa.

L'obiettivo, in questa fase, è stato quello di verificare la correttezza delle operazioni eseguibili per mezzo dei due contratti implementati, svolgendo tale processo in maniera programmatica.

A tal proposito, è stato utilizzato il framework di testing *Mocha* offerto da *Hardhat* e basato su *JavaScript*. Scrivere i test in un moderno linguaggio di programmazione, come *JavaScript* per l'appunto, offre maggiore flessibilità e possibilità di interazione con gli smart contract.

Nello specifico, il primo passo è stato quello di scrivere e lanciare degli *unit test*, ossia test incentrati su unità individuali e porzioni ristrette del codice sorgente.

Il Listato 5.4, spiegato nel seguito, mostra alcuni esempi di *unit test* relativi allo smart contract *NftMarketplace*.

Trattandosi di *unit test*, innanzitutto viene verificato che la rete utilizzata sia una rete di sviluppo locale; in caso contrario, infatti, il test viene ignorato.

Dopodiché, il file si articola in una serie di funzioni `describe`, definite da *Mocha*, utili per organizzare le *suite* di test in base alle diverse tipologie di funzionalità sotto esame.

La prima sezione provvede a predisporre le pre-condizioni valide per tutti i test; più in dettaglio, per mezzo dell'*hook* `beforeEach`, è stato specificato che, prima di ogni test, deve essere effettuato il *deploy* dei due smart contract discussi, il *mining* di un NFT ed, infine, deve essere concessa, al contratto *NftMarketplace*, l'approvazione per spostare il token creato. In questo modo, dunque, ci si assicura di ripartire, ad ogni nuova esecuzione, da un ambiente "pulito".

All'interno del `describe` dedicato ad una specifica funzionalità, i test veri e propri vengono codificati richiamando una serie di funzioni `it`, ognuna delle quali prevede due parametri; ovvero una stringa che descrive il comportamento testato ed una funzione all'interno della quale vengono svolte le operazioni e le chiamate agli smart contract.

Il Listato 5.4, nello specifico, si sofferma sulla verifica del comportamento della funzione `buyItem`, utile per registrare l'acquisto di un prodotto. Un primo test sollecita una condizione di errore, assicurandosi che venga correttamente segnalata; nello specifico, verifica che non sia possibile acquistare un prodotto che non sia stato preliminarmente messo in vendita tramite la funzione `listItem`. In secondo luogo, è stato testato che, una volta richiamata la funzione per l'acquisto, il *mapping* che tiene traccia delle transazioni venga correttamente aggiornato con gli indizzi di venditore ed acquirente. Infine, l'ultimo test mostrato nel listato in questione accerta che, al termine dell'esecuzione della `buyItem`, venga emesso l'evento previsto.

All'interno dei test descritti, così come in quelli non riportati, sono state sfruttate alcune funzioni particolari predisposte da *Mocha*. Su tutte, quelle più utili sono `assert` ed `expect`; la prima è utile per effettuare delle asserzioni, come nel caso in cui bisogna verificare l'uguaglianza di due variabili; la seconda viene utilizzata per esprimere il comportamento atteso, ad esempio, a seguito della chiamata ad una funzione. Tali funzioni, dunque, permettono di sollevare degli errori nei test, nel caso in cui la condizione da esse imposta non risulti verificata.

```
1  const { assert, expect } = require("chai");
2  const { network, deployments, ethers, getNamedAccounts } = require("hardhat");
3  const { developmentChains } = require("../../helper-hardhat.config.js");
4
5  !developmentChains.includes(network.name)
6    ? describe.skip // Tests executed only on a local network
7    : describe("Nft Marketplace Tests", function () {
8        let nftMarketplace, productNft, deployer, user;
9        const TOKEN_ID = 0;
10       beforeEach(async function () {
11         // Extracting signers used for sending transactions
12         const accounts = await ethers.getSigners();
```

```
13          deployer = accounts[0];
14          user = accounts[1];
15
16          await deployments.fixture(["main"]); // Running scripts for deploying
                contracts
17
18          nftMarketplace = await ethers.getContract("NftMarketplace");
19          productNft = await ethers.getContract("ProductNft");
20
21          await productNft.mintNft();
22          await productNft.approve(nftMarketplace.address, TOKEN_ID);
23        });
24
25        describe("buyItem test", function () {
26          it("buy item only if listed", async function () {
27            // Connecting to smart contract with a different account (buyer = user)
28            const userConnectedNftMarketplace = await nftMarketplace.connect(
29              user
30            );
31            // Trying to buy an NFT not listed
32            await expect(
33              userConnectedNftMarketplace.buyItem(productNft.address, TOKEN_ID)
34            ).to.be.revertedWithCustomError(
35              nftMarketplace,
36              "NftMarketplace__NotListed"
37            );
38          });
39
40          it("updating Transaction for bought item", async function () {
41            // Listing item before
42            await nftMarketplace.listItem(productNft.address, TOKEN_ID);
43
44            // Buying an Item with another account (user = buyer)
45            const userConnectedNftMarketplace = await nftMarketplace.connect(
46              user
47            );
48            await userConnectedNftMarketplace.buyItem(
49              productNft.address,
50              TOKEN_ID
51            );
52            // Extracting Transaction Data
53            const [seller, buyer] =
54              await userConnectedNftMarketplace.getTransaction(
55                productNft.address,
56                TOKEN_ID
57              );
58            assert.equal(seller.toString(), deployer.address);
59            assert.equal(buyer.toString(), user.address);
60          });
61
62          it("buying an item emits an event", async function () {
63            // Listing item before
64            await nftMarketplace.listItem(productNft.address, TOKEN_ID);
65            // Connecting with another account to buy the item
66            const userConnectedNftMarketplace = await nftMarketplace.connect(
67              user
68            );
69
70            expect(
71              await userConnectedNftMarketplace.buyItem(
72                productNft.address,
73                TOKEN_ID
```

```
74              )
75            ).to.emit("ItemBought");
76          });
77        });
78        // ... Continue ...
79    });
```

Listing 5.4: Unit test per lo smart contract *NftMarketplace*

Una volta terminata la scrittura dei test, l'ambiente *Hardhat* fornisce un task apposito, richiamabile da terminale tramite il comando `yarn hardhat test`, che permette di avviare il processo di testing in maniera automatica e di monitorarne l'esito.

Un *plugin* utile durante il processo di testing è *solidity-coverage*. Tale strumento, facilmente integrabile all'interno di *Hardhat*, esamina i test scritti, al fine di rilevare quante linee di codice dei contratti sono coinvolte e coperte da essi. Al termine dell'esecuzione viene fornito un prospetto riassuntivo che indica le percentuali di copertura ottenute, distinguendo tra *statements*, *branch*, funzioni e righe di codice; oltre a ciò, sono segnalate anche le linee rimaste al di fuori dei test.

In Figura 5.1 è mostrato l'output ottenuto lanciando il plugin *solidity-coverage* sulla base dei test previsti per gli smart contract implementati. Come è possibile notare, sono state raggiunte percentuali di copertura soddisfacenti, cercando di esplorare tutte le principali casistiche possibili.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	90.24	83.33	86.36	88.41	
NftMarketplace.sol	94.44	87.5	94.12	91.38	... 115,117,164
ProductNft.sol	60	50	60	72.73	30,57,61
All files	90.24	83.33	86.36	88.41	

Figura 5.1: Output di Solidity Coverage relativo alla copertura dei test effettuati

Gli *unit test* sono stati eseguiti sulla rete locale *hardhat*, descritta nella Sezione 5.2.3.

Il livello successivo di verifica, chiamato *stage testing* o *integration testing*, invece, si è basato sull'utilizzo di una *TestNet* reale. Tale tipologia di test, infatti, può essere pensata come l'ultimo step di collaudo prima del *deploy* definitivo su una *MainNet*.

5.2.2 Gas Report

Uno dei vantaggi dei test è che permettono di capire come ottimizzare gli smart contract affinché siano più funzionali e *gas-efficient*.

Per avere informazioni circa il consumo di risorse dovuto alle singole funzioni implementate, è stata abilitata l'estensione *hardhat-gas-reporter*, integrabile durante l'esecuzione dei test.

La Figura 5.2 mostra il riepilogo del consumo medio di gas legato alle funzioni e al *deploy* degli smart contract *ProductNft* e *NftMarketplace*, in cui viene riportata, anche, la stima dei costi in termini monetari, in riferimento alla blockchain *Ethereum*.

Come è possibile notare, i costi maggiori derivano dalle funzionalità di generazione del token di un prodotto (`mintNFT`), di acquisto di un prodotto (`buyItem`) e di trasferimento di un token (`transferTokenAfterTime`). Ad ogni modo, è da evidenziare anche che le strategie che consentono al *marketplace* di gestire i prodotti e che forniscono agli utenti la possibilità di reso,

implementate senza il trasferimento della proprietà del token, risultano effettivamente più vantaggiose.

Solc version: 0.8.18		Optimizer enabled: false		Runs: 200	Block limit: 30000000 gas	
Methods		10 gwei/gas			1536.09 eur/eth	
Contract	Method	Min	Max	Avg	# calls	eur (avg)
NftMarketplace	buyItem	–	–	94660	9	1.45
NftMarketplace	cancelListing	–	–	31252	2	0.48
NftMarketplace	listItem	–	–	57229	17	0.88
NftMarketplace	takeBackToken	–	–	32364	2	0.50
NftMarketplace	transferTokenAfterTime	–	–	71516	3	1.10
ProductNft	approve	–	–	49310	22	0.76
ProductNft	mintNft	–	–	95238	22	1.46
Deployments				% of limit		
NftMarketplace		–	–	1528928	5.1 %	23.49
ProductNft		–	–	2379519	7.9 %	36.55

Figura 5.2: Gas Report

5.2.3 Reti utilizzate

All'interno dell'ambiente *Hardhat* è integrato un tool chiamato *Hardhat Network*, il quale, avviando un nodo locale *Ethereum*, mette a disposizione una rete "*fake*" appositamente pensata per lo sviluppo. Tale rete consente di effettuare il *deploy* dei contratti, di lanciare i test e di eseguire il *debug* del codice in modo agevole e privo di rischi, dato che tutto rimane confinato nella propria macchina e, ad ogni nuova esecuzione di uno script, la rete viene "abbattuta" e resettata.

Tale componente è stato utilizzato sia nella modalità di default appena descritta, sia nella sua forma *stand-alone*, dotata di una UI in cui sono esposte informazioni quali il *websocket* per la connessione, gli indirizzi e le chiavi private dei vari nodi, nonchè i dati relativi alle transazioni effettuate di volta in volta. Oltre a ciò, la rete in questione continua ad essere attiva anche al termine dell'esecuzione di uno *script*, fornendo una maggiore flessibilità. Sfruttando l'URL mostrato, per giunta, è possibile connettersi tramite *client* esterni; infatti, tale rete locale è stata utilizzata anche per sperimentare il *front-end* dell'applicazione decentralizzata nelle prime fasi di sviluppo.

Strumenti come quelli descritti, dunque, si sono rivelati molto potenti per testare in maniera veloce le funzionalità implementate e per rendersi conto di come gli smart contract potessero interagire su una *TestNet* reale, dove i tempi si allungano. In queste reti di test locale, infatti, viene forgiato un blocco per ogni transazione ricevuta, in ordine e senza ritardi, rendendo l'interazione con i contratti decisamente snella.

Una volta aver raggiunto una versione stabile del *back-end*, ad ogni modo, si è deciso di passare su una *TestNet* reale, così da avere un riscontro anche sulle *feature* più complesse, come, ad esempio, l'automatizzazione del trasferimento del token al termine del periodo di reso, per mezzo di *OpenZeppellin Defender*.

A tal proposito, è stata adottata la rete di test *Sepolia*, creata dagli sviluppatori di *Ethereum* ed introdotta abbastanza recentemente.

Le *TestNet* reali, come *Sepolia*, sono progettate per replicare condizioni operative analoghe a quelle di una *MainNet* ma si sviluppano su registri separati. In questo modo, dunque, consentono agli sviluppatori di accertare il corretto funzionameno degli smart contract e

delle applicazioni decentralizzate senza incorrere in rischi, potendo contare su un ambiente sicuro e affidabile.

La rete *Sepolia*, nello specifico, è costruita sul meccanismo di consenso *Tendermint*, di stampo BFT (*Byzantine Fault Tolerant*), ma, a seguito del *merge* di *Ethereum*, ha introdotto anche un meccanismo di consenso *PoS*.

Ad ogni modo, *Sepolia* è stata scelta perché in grado di processare migliaia di transazioni al secondo, con tempi di conferma abbastanza veloci, e di gestire, in maniera efficace, una grande quantità di utenti. A dispetto di altre reti di test, infatti, il numero di *testnet token* (criptovaluta di test) a disposizione degli utenti non è limitato, rendendo più semplice il compito agli sviluppatori.

All'interno della *directory* di progetto, il file `hardhat.config.js` esporta un modulo `networks` dedicato alla configurazione delle reti, sia locali che reali; il modulo in questione, dunque, è richiamabile nei diversi script, al fine di ricavare le informazioni sulla blockchain in uso al momento dell'esecuzione.

Per effettuare la connessione ad una rete specifica, è necessario fornire una serie di parametri, tra cui l'RPC-URL al quale rivolgersi, il *chainId* (identificatore della rete), l'account con cui effettuare la connessione, il numero di conferme da attendere in termini di blocchi, etc. Alcuni parametri caratteristici della rete, su tutti l'RPC-URL, sono stati ricavati tramite il *tool Alchemy*, il quale consente la connessione ad una serie di blockchain supportate, tra cui *Sepolia*.

Dopo aver predisposto la configurazione delle reti, è stato possibile realizzare gli *script* per il *deploy* dei contratti.

5.2.4 Deploy sulla TestNet: utilizzo di IPFS e verifica su Etherscan

La registrazione degli smart contract *ProductNft* e *NftMarketplace* sulla blockchain di test *Sepolia* è avvenuta per mezzo di due script dedicati.

In realtà, gli script per il *deploy* erano già stati impostati in precedenza per consentire lo svolgimento dei test sulla rete locale *hardhat*; in questa fase, però, sono stati revisionati e modularizzati, aggiungendo delle funzionalità specifiche per la *TestNet*.

Il Listato 5.5 mostra il fulcro del processo di *deploy* del contratto *ProductNft*.

Prima di arrivare al *deploy* vero e proprio, però, sono state svolte alcune operazioni preaparatorie. La prima sezione dello script, infatti, è dedicata alla generazione del *token URI*, tramite il quale poter leggere le informazioni che caratterizzano un NFT e richiesto dal costruttore dello smart contract.

In primo luogo, dunque, viene definito il template dei metadati da associare ai token che verranno creati tramite lo smart contract in questione. La struttura dei metadati può essere organizzata arbitrariamente, adattandola alle specifiche esigenze del *brand* che rilascerà gli NFT. Ai fini del lavoro, comunque, è stato fornito un template di esempio con una plausibile valorizzazione degli attributi, ipotizzando di dover certificare dei gioielli; a tal proposito, alcuni dati interessanti possono essere il materiale, i colori, il peso e la purezza, i quali sono stati espressamente codificati. Altri aspetti da includere nei metadati, come il creatore dell'NFT e l'immagine ad esso associata, invece, saranno aggiunti in un secondo momento.

Ad ogni modo, quello appena definito è semplicemente un oggetto *JSON*; a partire da esso, infatti, deve essere ottenuto l'URI da associare all'NFT. Tale trasformazione avviene richiamando la funzione ausiliaria `handleTokenUri`, la cui implementazione, per motivi di concisione, non è mostrata nel Listato 5.5 ma è comunque spiegata, data la sua rilevanza.

La funzione `handleTokenUri` riceve, come parametro di *input*, l'indirizzo dell'account con cui si intende effettuare il deploy dello smart contract; esso, infatti, viene trascritto nei metadati del token in qualità di creatore, fornendo, dunque, un'indicazione dell'origine.

Una volta inserita l'informazione sulla provenienza, la funzione citata gestisce l'immagine del prodotto di lusso da associare al token; per motivi di efficienza e di consumi di risorse, infatti, l'NFT non può includere il file immagine direttamente in formato *png*. In virtù di ciò, una volta prelevata dalla cartella *images* del progetto, l'immagine di esempio è stata trasferita su IPFS *(InterPlanetary File System*).

IPFS è un sistema *peer-to-peer* progettato per affrontare alcune delle sfide associate alla distribuzione e all'archiviazione di contenuti online, come la decentralizzazione, la ridondanza e la resistenza alla censura.

Una delle caratteristiche di IPFS è che i file, una volta caricati, vengono identificati tramite un hash crittografico derivato dal loro contenuto, il che rende più efficace la verifica dell'integrità dei dati e l'accesso, non più basato su un *path*, di directory.

IPFS è stato utilizzato per l'*hosting off-chain* sia del file immagine da associare all'NFT, sia dei metadati stessi del token, come sarà chiarito a breve,

Per procedere all'upload dei contenuti su IPFS sono state inoltrate delle chiamate *API* al servizio *Pinata*, dedicato a tale scopo. Affidare il *pinning* dei dati ad un servizio come *Pinata* reintroduce un minimo di centralizzazione ma consente, comunque, di rendere disponibili i contenuti presso più *peer* IPFS, diversamente dall'utilizzo di un solo nodo locale che, di fatto, avrebbe costituito un *single point of failure*. Ai fini del lavoro, dunque, la soluzione descritta è stata ritenuta accettabile, seppur migliorabile.

Tornando alla funzione `handleTokenUri`, innanzitutto, è stato effettuato il *pinning* dell'immagine del token. L'hash retituito da *Pinata* è stato utilizzato per creare l'*IPFS URL*, tramite il quale la risorsa risulterà raggiungibile; è proprio tale indirizzo, infatti, ad essere inserito nei metadati dell'NFT, andando a "valorizzare" il campo `image`.

Successivamente, una seconda chiamata a *Pinata* è stata avviata per realizzare l'upload dell'oggetto *JSON* contenente i metadati del token, ormai completo di tutte le sue voci. Analogamente al caso precedente, a partire dall'hash restituito, è stato costruito l'*IPFS URL*, il quale, di fatto, costituisce il *token URI*. Il *token URI*, dunque, punta ad una risorsa IPFS (i metadati dell'NFT) che, a sua volta, fa riferimento ad un'ulteriore contenuto su IPFS (l'immagine del token).

Una volta terminata l'esecuzione della funzione `handleTokenUri`, lo script procede al *deploy* del contratto, sfruttando l'apposita funzione fornita dalla libreria `ethers.js`, dedicata all'interfacciamento del codice *JavaScript* con gli smart contract. All'atto del *deploy*, dunque, viene specificato l'account tramite il quale inviare la transazione e il *token URI* da notarizzare, appena ricavato.

L'ultima operazione svolta per mezzo dello script mostrato nel Listato 5.5 è la verifica automatica dello smart contract su *Etherscan*, il *block explorer* che permette di monitorare le transazioni sulla blockchain *Sepolia*, fornendo, in opportune *dashboard*, informazioni quali l'hash della transazione, lo *status*, il *timestamp*, il numero di blocco nel quale la transazione è inserita, il mittente, il destinatario, la *fee* pagata, etc.

Il processo di verifica su *Etherscan*, realizzato configurando e richiamando l'apposito plugin *hardhat-etherscan*, consente a chiunque si affacci sulla rete di esaminare il codice sorgente dello smart contract, a favore di una maggiore trasparenza. Ciò, infatti, contribuisce a creare fiducia tra gli utenti e gli sviluppatori, poiché è possibile accertare che il contratto agisca effettivamente come dichiarato e che non ci siano *bug* o comportamenti indesiderati. Per giunta, in molti contesti, come, ad esempio, la vendita dei token, la verifica di un contratto sul *block explorer* può essere addirittura un requisito ai fini della conformità normativa, nonché utile in fase di *auditing*.

La verifica programmatica è prevista anche durante il *deploy* dello smart contract *Nftmarketplace*, non discusso in questa sede perché decisamente più lineare.

```
1 // ... Imports ...
```

```
2
3  // Defining NFT metadata template
4  let metadataTemplate = {
5    name: "Bulgari jewel",
6    description: "An authentic luxury product",
7    creator: "",
8    image: "",
9    attributes: [
10     {
11       material: "diamond",
12       colour: ["grey", "blue"],
13       weight: "50g",
14       pureness: 100,
15     },
16   ],
17 };
18
19 // Before first deploy, tokenURI isn't available yet and needs to be created
20 let tokenURI = "";
21
22 module.exports = async function ({ deployments }) {
23   const { deploy, log } = deployments;
24
25   // Account used for deploying
26   const { deployer } = await getNamedAccounts();
27
28   if (process.env.UPLOAD_TO_PINATA == "true") {
29     // URI creation: uploading image and metadata to IPFS using Pinata
30     tokenURI = await handleTokenUri(deployer);
31   }
32
33   let arguments = [tokenURI];
34
35   // Deploying contract ProductNft providing tokenURI to constructor
36   console.log("-------------------------");
37   const productNft = await deploy("ProductNft", {
38     from: deployer,
39     log: true,
40     args: arguments,
41     waitConfirmations: network.config.blockConfirmations || 1,
42   });
43   if (process.env.UPLOAD_TO_PINATA == "false") {
44     console.log(`NFT token URI: ${tokenURI}`);
45   }
46
47   // Contract programatic verification on Etherscan
48   if (
49     !developmentChains.includes(network.name) &&
50     process.env.ETHERSCAN_API_KEY
51   ) {
52     log("Verifica del contratto ...");
53     await verify(productNft.address, arguments);
54   }
55 };
```

Listing 5.5: Deploy dello smart contract *ProductNft*

5.2.5 Metamask

Metamask è un'estensione per il *web browser* che permette di creare e gestire *wallet* dedicati alla criptovaluta, consentendo, al tempo stesso, di interagire con applicazioni decentralizzate.

La creazione di un *wallet* avviene tramite la generazione di una frase mnemonica di 12 parole, da mantenere segreta perché responsabile dell'accesso al portafoglio elettronico.

All'interno di un singolo *wallet* possono essere creati diversi account, ciascuno dei quali identificato univocamente da un indirizzo pubblico e dotato di una propria chiave privata, utilizzata per firmare le transazioni e, per questo, da non condividere.

Tramite *Metamask*, inoltre, ogni singolo account può essere connesso ad una rete diversa, scegliendo tra le numerose blockchain supportate. Oltre a ciò, viene fornito un chiaro resoconto dei token, fungibili e non, attualmente posseduti da ogni account, rendendo molto intuitivo e semplice lo scambio di tali asset.

Metamask si è rivelato uno dei principali strumenti utilizzati a seguito del *deploy* degli smart contract sulla rete *Sepolia*, al fine di testare in maniera *user-friendly* le funzionalità del *back-end* ed inviare transazioni sulla blockchain, simulando un'esperienza quanto più vicina alla realtà.

La Figura 5.3 mostra l'icona che appare nella barra delle estensioni del *browser* una volta installato il *wallet manager Metamask*.

Figura 5.3: Icona di Metamask

5.3 Front-end dell'applicazione

Avendo a disposizione degli smart contract stabili, attivi sia sulle reti locali di test che sulla blockchain *Sepolia*, è stato possibile dedicarsi all'implementazione del *front-end*, in modo da ottenere un'applicazione decentralizzata completa di tutte le sue componenti ed utilizzabile anche dagli utenti finali.

Lo sviluppo del *front-end*, infatti, ha condotto alla realizzazione di una semplice interfaccia *web* che consente di richiamare, interattivamente, le principali funzionalità previste dal sistema, interagendo con la blockchain sottostante.

5.3.1 Framework adottato e struttura del progetto

L'implementazione del *front-end* ha visto l'utilizzo del *framework Next.js*, basato su *JavaScript* e *React*.

Next.js è un *framework* orientato allo sviluppo web in grado di offrire diverse *feature* interessanti; lo *streaming* HTML dinamico consente di apportare modifiche ad una pagina in costruzione istantaneamente, senza doverla necessariamente ricaricare da zero; la flessibilità del *rendering* permette di scegliere se generare una pagina lato *server* o lato *client*; un sistema di *routing* dinamico agevola la gestione delle rotte delle pagine; oltre a ciò, *Next.js* supporta le operazioni di *data fetching* sia lato *client* che lato *server*, semplifica l'integrazione dei componenti *React* e garantisce un'ampia compatibilità nei confronti di diversi *tool* e librerie di stile.

Un componente *React* può essere considerato come un blocco di codice indipendente e riutilizzabile, che restituisce del codice HTML.

Per procedere allo sviluppo della *web application*, è stata creata una nuova *directory* di progetto indipendente dal *back-end*. Una volta eseguita la procedura di *set up* indicata dalla documentazione di *Next.js*, il progetto ha assunto la seguente organizzazione:

- la cartella `pages` è la destinazione dei file *JavaScript* che realizzano le diverse pagine del sito; tra questi file ci sono sicuramente `index.js`, ossia la pagina di default che appare avviando l'applicazione, e `_app.js`, che rappresenta l'*entry-point* per tutto il *front-end*;
- la cartella `components`, è quella che contiene i singoli componenti *React* sfruttati per la costruzione delle pagine;
- la cartella `public`, contenente le immagini pubbliche utilizzate;
- la cartella `styles`, al cui interno sono stati inseriti i fogli di stile per la formattazione;
- la cartella `constants`, dove sono state inserite informazioni statiche da prelevare all'occorrenza, come, ad esempio, gli indirizzi degli smart contract a cui fare riferimento nel caso di utilizzo di una rete locale, piuttosto che di una rete reale;
- file vari di configurazione.

5.3.2 Header, connessione al wallet e lettura dalla blockchain

L'header è un elemento ricorrente in tutte le pagine; di conseguenza, per modularizzare la scrittura del codice è stato realizzato predisponendo un componente *React* dedicato.

Alla luce di ciò, è stata definita una funzione `Header`, analoga ad una funzione *JavaScript*, la quale, però, restituisce l'HTML per il *rendering* degli elementi che costituiscono il componente in questione, ossia il nome del sito, un pulsante per la connessione dell'applicazione al wallet dell'utente e la navbar con i link che consentono di spostarsi tra le varie pagine.

Il pulsante per la connessione del wallet è stato inserito in maniera agevole sfruttando un componente predefinito importato dalla libreria *web3uikit*.

Una volta prevista la possibilità di connessione ad un wallet, le componenti messe a disposizione dalla piattaforma *Moralis* consentono di rispondere a gran parte delle esigenze, a livello di codice, legate alle funzionalità *Web3* e blockchain. Ad esempio, per capire se l'applicazione sta utilizzando le funzionalità *Web3* e, di conseguenza, l'utente è connesso per mezzo di un wallet, può essere interrogato l'oggetto `isWeb3Enabled`, importabile da *Moralis*.

La navbar, infatti, è stata progettata in modo tale da visualizzare determinate sezioni del menù soltanto a seguito della connessione di un wallet o, addirittura, in relazione allo specifico account utilizzato. È questo il caso dell'accesso alla pagina per creare un nuovo "LuxuryChain Certificate', consentito soltanto nel momento in cui l'account connesso corrisponde al legittimo proprietario dello smart contract *ProductNft*, cioè al *brand*.

Per poter riuscire nell'intento, si è resa necessaria una lettura dalla blockchain. Il Listato 5.6 mostra parte dell'implementazione della funzione `Header`, concentrandosi sul codice che permette di interfacciarsi con lo smart contract *ProductNft* ed innescare, al contempo, l'aggiornamento dell'interfaccia utente.

Al fine di richiamare le funzioni degli smart contract, nell'esempio mostrato, così come in tutti gli altri casi, è stato utilizzato l'hook `useWeb3Contract`, fornito dal plugin *react-moralis*.

In riferimento al Listato 5.6, `useWeb3Contract` viene attivato inizialmente per dichiarare la funzione `getContractOwner`, attraverso la quale si potrà interagire con la blockchain. I parametri per effettuare tale operazione, però, non sono ancora definiti; infatti, è necessario

ricavare l'indirizzo dello smart contract a cui rivolgersi, variabile in base alla rete alla quale l'utente è connesso.

Per ovviare a ciò, viene utilizzato un altro hook, `useEffect`, il quale consente di gestire gli effetti collaterali legati al montaggio o all'aggiornamento di un componente *React*. Nel momento in cui viene rilevata la connessione ad un wallet e, di conseguenza, ad una rete blockchain, infatti, `useEffect` provvede a definire i parametri necessari per la chiamata allo smart contract, ossia l'*ABI*, l'indirizzo e il nome della funzione. L'*ABI* e l'indirizzo del contratto, in particolare, vengono prelevati da opportuni file *JSON* creati nella cartella `constants`. Infine viene innescata una funzione `updateHeader`.

La funzione `updateHeader` riceve le opzioni per la chiamata alla funzione `getContractOwner` che, a questo punto, può essere effettuata. La lettura dalla blockchain restituisce l'indirizzo del proprietario del contratto, il quale viene memorizzato in una variabile di stato tramite l'hook `useState`.

Dopodiché, avviene l'eventuale aggiornamento dell'interfaccia; la variabile di stato, infatti, viene utilizzata nel codice HTML affinché il *rendering* del link che consente l'accesso alla pagina per la creazione di un nuovo NFT avvenga in maniera condizionale, solo se c'è corrispondenza tra l'account attualmente connesso e il proprietario dello smart contract.

```
1  // .. Imports ..
2
3  export default function Header() {
4    const { account, chainId, isWeb3Enabled } = useMoralis();
5    const stringChainId = parseInt(chainId).toString();
6    const [brand, setBrand] = useState("");
7
8    // Function declaration for SC call
9    const { runContractFunction: getContractOwner } = useWeb3Contract();
10
11   // Hook for collateral effects' management
12   useEffect(() => {
13     if (isWeb3Enabled) {
14       console.log(`Account Detected: ${account}`);
15       const options = {
16         abi: nftAbi,
17         contractAddress: nftAddresses[stringChainId][0],
18         functionName: "getContractOwner",
19         params: {},
20       };
21
22       updateHeader(options);
23     }
24   }, [isWeb3Enabled]);
25
26   // Handler for header updating
27   async function updateHeader(options) {
28     const nftBrand = await getContractOwner({
29       params: options,
30       onError: (error) => {
31         console.log(error);
32       },
33     });
34     console.log(`Brand: ${nftBrand}`);
35     // Updating state
36     if (nftBrand) {
37       setBrand(nftBrand);
38     }
39   }
40
41   // ... return HTML ...
```

```
}
```

Listing 5.6: Codice per richiamare la funzione `getContractOwner` dello smart contract *ProductNft* all'interno del componente Header

Una volta implementato, un componente può essere esportato nelle diverse pagine dell'applicazione, all'interno delle quali è possibile farvi riferimento tramite una notazione simile ad un tag HTML.

5.3.3 Pagina principale

La home page riporta, in breve, la *mission* dell'applicazione ma, soprattutto, mostra un elenco di token associati a prodotti attualmente in vendita. L'idea è stata quella di costruire una sorta di "vetrina virtuale" composta da una serie di box, ognuno dedicato ad un singolo token.

La lista degli NFT è stata generata per mezzo di una query al servizio *TheGraph*, in ascolto sugli eventi emessi dallo smart contract *NftMarketplace* sulla blockchain. Le motivazioni che hanno spinto all'adozione di tale strumento e le modalità attraverso cui è stato possibile effettuare le interrogazioni saranno approfondite nella Sezione 5.4.

Dando momentaneamente per assodato il recupero dei token disponibili all'acquisto, il Listato 5.7 mostra l'implementazione della pagina principale del sito.

All'interno della funzione `Home`, l'unica istruzione presente memorizza il risultato della *query* a *TheGraph* nella variabile `listedNfts`.

Segue, poi, direttamente il **`return`** della funzione con l'HTML da renderizzare.

Dopo un tag `<div>` con del testo statico, è presente una sezione dinamica che costituisce un tipico esempio di utilizzo della sintassi JSX (o *React Syntax*); tale sezione, infatti, combina codice *JavaScript* con HTML.

Nello specifico, sono stati utilizzati degli operatori condizionali ternari innestati al fine di stabilire il corretto contenuto da visualizzare. Se è ancora in corso l'esecuzione della query allora viene mostrata un'animazione di *loading*; a caricamento ultimato, se la query non ha restituito alcun risultato viene renderizzato del testo statico, altrimenti si procede a processare la lista degli NFT risultanti, accedendo alla variabile `listedNfts`.

A tal proposito, viene utilizzata una funzione `map`, la quale scandisce la lista in input e, per ogni elemento al suo interno, applica una serie di operazioni, codificate per mezzo di una *arrow function*. Di ogni singolo NFT vengono estratte le informazioni restituite dalla query precedente, ossia l'indirizzo, l'ID e il venditore. Infine, tali parametri vengono forniti al componente `<NFTBox/>` che si occupa di renderizzare il riquadro dedicato ad ogni token.

```
// ... Imports ...

export default function Home() {
  // The Graph query -> list of tokens returned
  const { loading, error, data: listedNfts } = useQuery(GET_ACTIVE_ITEMS); //
      Mandiamo in esecuzione la query importata dal file delle subgraph query

  return (
    <div className="px-8 mx-auto">
      <h1 className="py-4 font-bold text-2xl">Product Tokens Available</h1>
      <div className="flex flex-wrap ">
        <p className="py-4 ">
          Hello! Here you can see LCCs (LuxuryChain Certificates) related to
          products that are currently on the market. Each NFT is an authenticity
          proof of the corresponding product. Connect your wallet to start!
        </p>

        {loading ? (
```

```
          <div className="px-4">
            <LoadComponent></LoadComponent>
          </div>
        ) : !listedNfts.activeItems ? (
          <div>No Token available at this moment</div>
        ) : (
          listedNfts.activeItems.map((nft) => {
            console.log(nft);
            const { nftAddress, tokenId, seller } = nft;
            return (
              <div className="py-4 place-content-center">
                <NFTBox
                  nftAddress={nftAddress}
                  tokenId={tokenId}
                  seller={seller}
                  key={`${nftAddress}${tokenId}`}
                />
              </div>
            );
          })
        )}
      </div>
    </div>
  );
}
```

Listing 5.7: Implementazione della home page del sito

Il Listato 5.8 mostra una parte dell'implementazione del componente *NFTBox*, dedicata, nello specifico, alla lettura e alla gestione dei metadati di un token.

All'interno della funzione `NFTBox`, la prima sezione di codice dichiara una serie di variabili di stato tramite le quali verranno memorizzati alcuni dati da visualizzare legati all'NFT. Dopodiché, richiamando l'hook `useWeb3Contract` e fornendo i parametri di collegamento, vengono dichiarate le funzioni da utilizzare per leggere il *tokenURI* e il nome dell'NFT dallo smart contract *ProductNft*.

Segue, poi, la definizione di una funzione `updateUI`, innescata, nel momento in cui un utente connette il proprio wallet all'applicazione, al fine di aggiornare l'interfaccia. Al suo interno, innanzitutto, vengono attivate le funzioni `getTokenUri` e `getNFTName`, che interagiscono con il contratto sulla blockchain.

Il *tokenURI* ricavato consiste in un URL IPFS che necessita di essere processato. La parte iniziale dell'URL, infatti, viene rimpiazzata in modo tale da realizzare una chiamata HTTPS; tramite questa operazione, dunque, la visualizzazione dei metadati del token ha esito positivo anche per chi non dispone di IPFS, in quanto viene utilizzato un *gateway IPFS* pubblico.

Una volta apportata la modifica descritta, l'URL viene passato alla funzione `fetch`, tramite cui è possibile prelevare il file JSON che contiene i metadati del token.

Accedendo al campo `image` dell'oggetto JSON si ricava l'URL IPFS che punta all'immagine del token, anch'esso processato come il precedente.

Al termine di ciò, l'URL dell'immagine e le informazioni ricavabili dal JSON (nome del prodotto, descrizione, creatore, attributi) vengono memorizzate nelle variabili di stato precedentemente definite, così che possano essere utilizzate all'interno del codice HTML.

Nello specifico, l'immagine sarà renderizzata all'interno del box dell'NFT collocato nella home page, mentre i dati del token sono inviati al componente descritto nella Sezione 5.3.4

```
// ... Imports ...

export default function NFTBox({ nftAddress, tokenId, seller }) {
  const { isWeb3Enabled, account } = useMoralis();

```

```
6   // State variables definition. Used for store token metadata after contract
        call
7   const [imageURI, setImageURI] = useState("");
8   const [tokenName, setTokenName] = useState("");
9   const [tokenDescription, setTokenDescription] = useState("");
10  const [tokenCreator, setTokenCreator] = useState("");
11  const [nftName, setNftName] = useState("");
12  const [nftAttributes, setNftAttributes] = useState("");
13
14  // SC call to get tokenURI
15  const { runContractFunction: getTokenUri } = useWeb3Contract({
16    abi: nftAbi,
17    contractAddress: nftAddress,
18    functionName: "getTokenUri",
19    params: {},
20  });
21
22  // SC call to get NFT name
23  const { runContractFunction: getNFTName } = useWeb3Contract({
24    abi: nftAbi,
25    contractAddress: nftAddress,
26    functionName: "name",
27    params: {},
28  });
29
30  // Function called when user is connected to a wallet
31  async function updateUI() {
32    // Execution of SC calls
33    const tokenURI = await getTokenUri();
34    const name = await getNFTName();
35
36    if (tokenURI) {
37      // URL Substitution: an IPFS gateway provider is used in order to make HTTP
            request
38      const requestURL = tokenURI.replace("ipfs://", "https://ipfs.io/ipfs/");
39
40      // Fetching metadata JSON
41      const tokenURIResponse = await (await fetch(requestURL)).json();
42
43      // Extracting and processing image URL IPFS
44      const imageURI = tokenURIResponse.image;
45      const imageURIURL = imageURI.replace("ipfs://", "https://ipfs.io/ipfs/");
46
47      // Setting state variables with data extracted
48      setImageURI(imageURIURL);
49      setTokenName(tokenURIResponse.name);
50      setTokenDescription(tokenURIResponse.description);
51      setTokenCreator(tokenURIResponse.creator);
52      setNftName(name);
53      setNftAttributes(tokenURIResponse.attributes);
54    }
55  }
56  // ... Return HTML ...
57 }
```

Listing 5.8: Implementazione del box di un NFT: estrazione dei metadati

5.3.4 Dettagli di un token e scrittura sulla blockchain

Il box di un NFT prevede un *handler* di tipo `onClick` che attiva la visualizzazione, in sovrimpressione, di una "card" contenente tutti i dettagli del token, così da poter leggere la

provenienza, le caratteristiche e la qualità del prodotto a cui esso è legato.

L'elemento grafico in questione è, sostanzialmente, una finestra modale, realizzata codificando un nuovo componente *React*, chiamato *TokenModal*. Tale componente, all'atto del caricamento, riceve in input le informazioni sull'NFT e restituisce un blocco di codice HTML che le visualizza secondo un opportuno layout.

Oltre ai dati inerenti al prodotto certificato, però, la finestra a comparsa prevede anche alcuni pulsanti, attivati in modo condizionale. Se l'NFT è stato caricato dall'utente attualmente connesso allora viene abilitato esclusivamente il pulsante che permette di rimuovere l'annuncio di vendita. Se, invece, l'NFT appartiene ad un altro utente, il solo pulsante cliccabile è quello che avvia la procedura di acquisto.

Acquistare un prodotto ed ottenere il relativo NFT che ne attesta l'autenticità è un'operazione che richiede l'invio di una transazione in scrittura sulla blockchain, interagendo con lo smart contract *NftMarketplace*.

Il Listato 5.9 mostra il codice degli *handler* attivati, in cascata, in seguito al click sul pulsante per l'acquisto di un prodotto, all'interno del componente *TokenModal*.

La prima funzione chiamata in causa è `handleOkClick`, la quale, innanzitutto, controlla che l'utente sia conesso tramite un wallet e verifica che l'account utilizzato non corrisponda al venditore. In caso affermativo, viene utilizzato l'hook `useWeb3Contract` per effettuare una chiamata alla funzione `buyItem` del contratto *NftMarketplace*, fornendo, come parametri, l'indirizzo dell'NFT e il *tokenID*. All'interno di tale chiamata, inoltre, vengono definiti due ulteriori *event handler*, uno in caso di errore ed uno in caso di successo.

Se la transazione va a buon fine, infatti, viene attivata la funzione `handleBuyItemSuccess` la quale, dopo aver atteso la conferma di un ulteriore blocco forgiato nella blockchain, mostrerà sull'interfaccia web, una notifica che segnala l'avvenuto acquisto.

Dopodiché, il trasferimento effettivo del token avviene, in automatico, decorso il tempo utile alla restituzione, secondo le modalità previste dal *back-end* e già trattate.

```
1  // ... Imports ...
2
3  export default function TokenModal({
4    // Input Parameters
5    nftAddress,
6    tokenId,
7    tokenDescription,
8    nftAttributes,
9    nftName,
10   seller,
11   creator,
12 }) {
13   const { isWeb3Enabled, account, chainId } = useMoralis();
14   const dispatch = useNotification();
15   const { runContractFunction } = useWeb3Contract();
16   const isOwnedByUser = seller === account;
17   const stringChainId = parseInt(chainId).toString();
18
19   // Function called after click
20   async function handleOkClick() {
21     if (isWeb3Enabled && !isOwnedByUser) {
22       // Parameters definition for SC call
23       const options = {
24         abi: marketplaceAbi,
25         contractAddress: marketplaceAddresses[stringChainId][0],
26         functionName: "buyItem",
27         params: { nftAddress: nftAddress, tokenId: tokenId },
28       };
29       // SC call
30       await runContractFunction({
```

```
31          params: options,
32          onError: (error) => console.log(error),
33          onSuccess: (tx) => handleBuyItemSuccess(tx),
34        });
35      }
36    }
37
38    async function handleBuyItemSuccess(tx) {
39      await tx.wait(1);
40      // Notification displayed
41      dispatch({
42        type: "success",
43        message:
44          "Congraturations! Token now is in staking. You have 14 days to get back
                the product (and the token)",
45        title: "Item Bought",
46        position: "topR",
47      });
48    }
49
50    // ... Code continues ...
51  }
```

Listing 5.9: Funzione che effettua l'acquisto di un prodotto richiamando l'apposita funzione `buyItem` dello smart contract *NftMarketplace*

5.3.5 Riepilogo token di un utente

Nel momento in cui l'applicazione risulta connessa al wallet di un utente, viene sbloccata anche la voce del menù che rimanda alla pagina di riepilogo dei *LuxuryChain Certificate* in suo possesso.

Un problema implementativo emerso in questa fase è nato dall'esigenza di mostrare sia i token effettivamnete di proprietà dell'utente, sia quelli relativi a prodotti regolarmente acquistati per i quali, però, è ancora prevista la possibilità di reso. Questi ultimi, infatti, non hanno subìto l'aggiornamento dell'*owner* perché, a tutti gli effetti, non sono stati trasferiti.

In virtù di ciò, la pagina è stata organizzata in due sezioni, individuate da altrettante tabelle, che separano, rispettivamente, i token relativi ai diversi casi descritti.

Per ottenere la lista degli NFT in staking, piuttosto che quella degli NFT ricevuti, è stato necessario, anche in questo caso, rivolgersi al tool *TheGraph*, tramite delle opportune *query*.

Le tabelle sono state realizzate attraverso un componente *React* dedicato, importato dalla libreria `web3uikit`.

Il Listato 5.10 mostra il codice che effettua la generazione dinamica delle righe della tabella relativa ai token in *staking*.

Il blocco di codice riportato applica la funzione predefinita `map` all'elemento `items`, cioè al vettore contenente tutti i token che l'utente ha in staking.

La funzione `map` restituisce un vettore delle stesse dimensioni di `items`, dove, però, ciascun elemento è ottenuto applicando la *arrow function* passata come parametro.

Ad ogni iterazione, dunque, i dati estratti dal token processato vengono disposti all'interno di un array che rappresenta una singola riga della tabella. Nello specifico, ogni riga contiene, nell'ordine, l'indirizzo dell'NFT, il *token ID*, la data entro la quale effettuare il reso ed, infine, un pulsante che, una volta cliccato, avvia la procedura di reso, interrompendo il trasferimento del token.

Indirizzo e ID del token sono ricavabili in maniera diretta.

Il termine ultimo per il reso è stato ottenuto aggiungendo, al timestamp relativo all'acquisto del prodotto, il tempo utile alla restituzione ed effettuando le necessarie conversioni e formattazioni per visualizzare la data in maniera leggibile per l'utente.

Inserire il pulsante per effettuare il reso all'interno dell'array che rappresenta la riga della tabella ha richiesto uno sforzo maggiore. Il pulsante, infatti, è un componente *React* e per passare un componente come elemento di un array è stato necessario utilizzare la funzione `React.createElement`. Tale funzione, a sua volta, riceve come parametro una *arrow function* all'interno della quale viene restituito il componente `<Button>` desiderato.

A questo punto, la riga è completata e si può procedere alla generazione successiva, fin quando non si esauriscono gli elementi dell'array `items`.

Il risultato complessivo dell'elaborazione, memorizzato nella variabile `stakingTableData`, è il parametro da fornire al componente *React* che visualizza l'intera tabella; esso assume, come richiesto, la forma di un array di array.

La tabella relativa ai token effettivamente trasferiti è stata generata in maniera analoga.

```
1  // ... Imports ...
2  //...Handler for table button ...
3
4  // Data sent to Table Component
5  const stakingTableData = items.map((item) => {
6    // The object returned is a row
7    return [
8      // Each element is a cell into the row
9      `${item.nftAddress}`,
10     `${item.tokenId}`,
11     item.timestamp
12       ? `${format(
13           parseInt(item.timestamp) * 1000 + returnTime,
14           "dd/MM/yyyy HH:mm:ss"
15         )}`
16       : "",
17
18     // React component as last element of row
19     React.createElement(() => {
20       return (
21         <Button
22           color="yellow"
23           onClick={() => {
24             handleTakeBack(item.nftAddress, item.tokenId); // L'handler deve
                   essere attivato all'interno di una arrow function altrimenti
                   come se si attivasse di continuo anche senza il click
25           }}
26           text="Give Back"
27           theme="colored"
28         />
29       );
30     }),
31   ];
32 });
33
34 // ... Table Component ...
```

Listing 5.10: Funzione per la generazione dinamica delle righe della tabella relativa ai certificati in *staking*

5.3.6 Creazione del token di un prodotto e pubblicazione di un annuncio di vendita

Le pagine dedicate, rispettivamente, alla generazione di un *LuxuryChain Certificate* e alla pubblicazione, finalizzata alla vendita, di un certificato di cui si è in possesso, rispondono unicamente ad esigenze esemplificative e sono sicuramente perfezionabili in ottica futura.

Come anticipato durante la descrizione dell'header, la possibilità di creare un certificato spetta unicamente al *brand*, ossia al proprietario del contratto *ProductNft*. La sezione dell'applicazione web che consente di effettuare tale operazione, dunque, è raggiungibile solo quando è connesso l'account che ha registrato sulla blockchain lo smart contract citato.

Ai fini del progetto, i dati da assicurare all'interno del certificato sono stati inseriti, tramite opportuno script, al momento del *deploy*. Sostanzialmente, dunque, possono essere generati certificati relativi ad una stessa linea di prodotto, dove i singoli pezzi, di conseguenza, rispondono alle medesime caratteristiche.

Alla luce di ciò, la pagina per la creazione di un nuovo *product token* presenta, semplicemente, un pulsante che, una volta cliccato, lancia la chiamata alla funzione del *back-end* tramite cui avviene il *minting* di un nuovo NFT.

La pagina dalla quale è possibile pubblicare un annuncio di vendita, caricando l'NFT corrispondente, è raggiungibile solo dopo aver connesso un wallet. Al suo interno, è prevista una form con due campi di input, uno testuale ed uno numerico, dove devono essere riportati l'indirizzo e l'ID del token che si intende pubblicare. L'inserimento dell'indirizzo, in particolare, è sottoposto ad alcune regole di validazione che impongono un numero minimo e massimo di caratteri, in relazione alla lunghezza tipica degli indirizzi blockchain.

L'handler definito per l'evento `onSubmit` preleva i dati inseeriti dall'utente nella form ed avvia una chiamata alla funzione `approve` dello smart contract *ProductNft*, tramite la quale si autorizza lo smart contract *NftMarketplace* a spostare il token in questione. Se l'operazione va a buon fine, un secondo handler innesca automaticamente una nuova transazione sulla blockchain, richiamando, questa volta, la funzione `listItem` del contratto *NftMarketplace*.

Al termine del workflow descritto, il token risulta disponibile alla vendita e, di conseguenza, viene visualizzato nella home page del sito, secondo le modalità già discusse.

Il corretto esito o meno delle interazioni con gli smart contract, inoltre, viene sempre notificato all'utente tramite dei pop-up.

5.4 Approfondimento: TheGraph Indexer

The Graph è un tool che offre un *layer* decentralizzato per l'indicizzazione degli eventi; a livello pratico, si tratta di una rete di nodi che va a prelevare e memorizzare i dati degli eventi emessi sulla blockchain, esponendo un'API che ne consente la lettura da *front-end*.

TheGraph, dunque, può essere utilizzato come protocollo di indicizzazione e interrogazione per reti decentralizzate, come Ethereum, IPFS, etc.

5.4.1 Motivazioni

Spostandosi verso il *front-end* del sistema e ragionando nell'ottica dell'usabilità da parte dell'utente, sono nate particolari esigenze, come, ad esempio, quella di mostrare tutti gli NFT disponibili ad essere ceduti, in quanto associati a prodotti in vendita.

In riferimento allo smart contract *NftMarketplace*, discusso nella Sezione 5.1.4, al suo interno è definita una variabile di stato che tiene traccia dei token pubblicati da ciascun account e, di conseguenza, utilizzabile ai fini dell'obiettivo; tale struttura dati, però, è un mapping e, in base alle specifiche di *Solidity*, non risulta iterabile.

Anche l'eventuale utilizzo di un array, al posto del mapping, si sarebbe rivelata una soluzione non percorribile, a causa dell'elevato consumo di gas per iterare in lettura e scrittura su tale struttura dati.

In ogni caso, il desiderio è anche quello di evitare di cambiare l'assetto del *back-end*, una volta assodato, semplicemente per finalità inerenti al sito web del *front-end*.

Tutte le motivazioni citate spingono all'utilizzo degli eventi.

Ogniqualvolta viene richiamata una funzione di uno smart contract che emette un evento, i dati da esso veicolati sono memorizzati *on-chain* e risultano accessibili soltanto dallo smart contract stesso. Oltre ad utilizzare gli eventi per fini di monitoraggio dello stato di un contratto, però, può essere utile fare in modo che servizi *off-chain*, come un sito web, abbiano accesso ad essi, ad esempio per innescare una reazione dell'interfaccia.

A tal proposito, deve essere configurato un nodo che, al di fuori della catena, possa mettersi in ascolto degli eventi emessi tramite le funzioni degli smart contract, aggiungendoli, di volta in volta, all'interno di un *repository* che potrà essere interrogato tramite delle *query*. Per fare ciò, sono stati valutati due tool, *Moralis* e *TheGraph*. Il primo, però, svolge il processo di indicizzazione in maniera centralizzata, tramite un server e un database; di conseguenza, la scelta è ricaduta su *TheGraph*, il quale consente di effettuare tutte le operazioni descritte aderendo al principio della decentralizzazione.

5.4.2 Creazione di un subgraph

La realizzazione di applicazioni al di sopra delle blockchain prevede modalità costruttive decisamente diverse rispetto alle strategie dello sviluppo software tradizionale; in una blockchain, infatti, i dati sono memorizzati in un formato tale per cui non possono essere "consumati" così facilmente dal *front-end*.

Al fine di un recupero efficiente, inoltre, è necessario disporre di dati indicizzati ed organizzati. Le trasformazioni dei dati che avvengono all'interno di uno stack tradizionale, costituito da database, web server ed API non sono applicabili, però, quando si effettuano letture da *Ethereum* o da altre blockchain; prima dell'introduzione di tool come *TheGraph*, infatti, i team di lavoro dovevano predisporre server di indicizzazione proprietari che, di conseguenza, richiedevano significative risorse hardware e di ingegnerizzazione, nonché rischiavano di infrangere i principi di sicurezza richiesti per la decentralizzazione.

Tramite *TheGraph*, dunque, è possibile sopperire al layer di indicizzazione che mancava nello stack *Web3*; tale strumento, infatti, consente di costruire e pubblicare delle *open API*, chiamate *subgraph*, integrabili all'interno delle applicazioni al fine di ottenere maggiori capacità di interrogazione dei dati memorizzati sulle blockchain, in quanto a ricerca, filtraggio, raggruppamento ed ordinamento.

Un *subgraph*, dunque, si colloca nel mezzo tra l'infrastruttura blockchain e l'interfaccia utente, costituendo un layer decentralizzato.

Alla luce di ciò, iniziare ad indicizzare gli eventi emessi dallo smart contract *NftMarketplace* tramite la rete *TheGraph*, ha richiesto, in primo luogo, l'implementazione locale di un *subgraph* e, in seguito, il *deploy* sul sito *Subgraph Studio*.

Per lo sviluppo locale è stata inizializzata una nuova directory di progetto dedicata, utilizzando le istruzioni riportate nella documentazione di *TheGraph*.

Le sezioni principali del progetto risiedono nei file `schema.graphql` e `mapping.ts`.

Il file `schema.graphql` consente di definire le entità che devono essere ricavate a partire dallo smart contract *NftMarketplace*; facendo un'analogia con i database relazionali, dunque, il file in questione predispone la struttura delle tabelle da interrogare.

Un'entità, infatti, è un tipo di dato presente in *TheGraph* sul quale è possibile effettuare delle *query*; le entità ricalcano gli eventi emessi dalle diverse funzioni dello smart contract,

dai quali vogliamo estrarre dati di interesse da mostrare nel front-end. Nel caso specifico, sono state definite le seguenti entità:

- prodotti attivi (*ActiveItem*);
- prodotti messi in vendita (*ListedItem*);
- prodotti acquistati (*ItemBought*);
- token trasferiti (*TokenTransfered*);
- token restituiti (*TokenGetBack*);
- annunci di vendita rimossi (*RemovedItem*).

Ad eccezione di "prodotti attivi", tutte le altre entità derivano in maniera diretta dagli eventi codificati all'interno dello smart contract *NftMarketplace*. L'entità "prodotti attivi" è stata creata *ad hoc* per gestire gli NFT da riportare nella home page del sito; ogni volta che un prodotto viene messo in vendita, infatti, il relativo certificato dovrà risultare attivo, così da essere mostrato nel *front-end*; dopodiché, quando il prodotto viene acquistato o l'annuncio di vendita rimosso, il relativo token non deve risultare più attivo e, di conseguenza, non più visibile.

In aggiunta, sono stati definiti anche i parametri che caratterizzano ciascuna entità, come fossero gli attributi di una tabella in un database relazionale.

Una volta definito lo schema *Graph QL* da utilizzare per le *query*, il passo successivo è far capire a *TheGraph* come mettersi in ascolto degli eventi e come manipolare le entità definite. A tal proposito, l'attenzione si è spostata sul file `mappings.ts`, tramite il quale è stato possibile specificare come mappare le informazioni degli eventi.

Nel file `mappings.ts` sono state definite una serie di funzioni che si comportano come *handler* degli eventi; nel momento in cui un evento viene emesso sulla blockchain, la funzione ad esso associata all'interno del *subgraph* viene attivata.

Il Listato 5.11 mostra, nella prima sezione, la definizione delle entità `ActiveItem` e `ItemListed` e, successivamente, la codifica della funzione `handleItemListed`.

Le entità `ActiveItem` e `ItemListed` sono molto simili tra loro; entrambe presentano, come campi, un ID, l'indirizzo del venditore, l'indirizzo dell'NFT associato al prodotto messo in vendita e l'identificatore dell'NFT. L'entità `ActiveItem`, in più, presenta un parametro `buyer` destinato a memorizzare l'indirizzo dell'acquirente ed, inoltre, è dichiarata come entità che può essere modificata; il campo `buyer`, infatti, sarà sottoposto ad aggiornamento.

La funzione `handleItemListed` viene attivata in seguito all'emissione dell'evento `ItemListedEvent` da parte dello smart contract *NftMarketplace*. Attraverso tale funzione, innanzitutto, vengono istanziati due nuovi oggetti di tipo `ActiveItem` e `ItemListed`, rispettivamente. L'identificatore da associare ad ognuno di essi viene creato "al volo", richiamando una funzione ausiliaria che, sostanzialmente, concatena indirizzo e ID dell'NFT, estratti dai parametri dell'evento intercettato.

Dopodiché, la funzione `handleItemListed` valorizza i campi previsti dalle entità, andando a leggere i dati veicolati con l'evento; vengono memorizzati, dunque, l'indirizzo del rivenditore, l'indirizzo dell'NFT e l'identificatore dell'NFT.

Per quanto riguarda l'entità `ActiveItem`, inoltre, l'indirizzo dell'acquirente viene settato a `null` perché il prodotto e il relativo token sono appena stati messi in vendita. Fin quando il parametro `buyer` rimane nullo, infatti, l'NFT è disponibile e, dunque, deve essere mostrato nell'apposita sezione del *front-end*. Successivamente, al momento dell'acquisto, tramite un nuovo *handler*, si accederà nuovamente all'entità `ActiveItem` per "valorizzare" il parametro `buyer`, rendendo, così, l'NFT non più attivo.

Infine, le entità create ed opportunamente processate, vengono salvate.

```
// ... File schema.graphql ...

type ActiveItem @entity (immutable: false) {
  id: ID!
  buyer: Bytes // buyer null until the product is on the market
  seller: Bytes!
  nftAddress: Bytes!
  tokenId: BigInt!
}

type ItemListed @entity {
    id: ID!
    seller: Bytes! # address
    nftAddress: Bytes! # address
    tokenId: BigInt! # uint256
  }

// ... File mappings.ts ...

  export function handleItemListed(event: ItemListedEvent): void {
    // When an item is listed, two entities are affected

    // Creation of new objects for entities
      itemListed = new ItemListed(
        getIdFromEventParams(event.params.tokenId, event.params.nftAddress)
      );

      activeItem = new ActiveItem(
        getIdFromEventParams(event.params.tokenId, event.params.nftAddress)
      );

    // Updating entity parameters with event data
    itemListed.seller = event.params.seller;
    activeItem.seller = event.params.seller;

    itemListed.nftAddress = event.params.nftAddress;
    activeItem.nftAddress = event.params.nftAddress;

    itemListed.tokenId = event.params.tokenId;
    activeItem.tokenId = event.params.tokenId;

    activeItem.buyer = null;

    itemListed.save();
    activeItem.save();
  }
```

Listing 5.11: Definizione delle entità e di un *event handler* per il *subgraph* con cui si interfaccia il *front-end*

Funzioni analoghe a quella mostrata nel Listato 5.11 sono state definite per rispondere anche agli eventi che segnalano l'acquisto di un prodotto, la restituzione, il trasferimento del token e la rimozione di un annuncio di vendita. Il trasferimento di un token, infatti, apporterà delle modifiche alla sezione privata di ogni utente, aggiornando le tabelle relative ai token in *staking* e ricevuti.

Definito il mapping tra gli eventi emessi dalla blockchain e le entità dello schema *Graph QL*, sono stati aggiornati alcuni file di configurazione, specificando l'indirizzo e l'ABI del contratto sottoposto ad indicizzazione.

Una volta implementato il *subgraph* in locale, ne è stato effettuato il *deploy*, in modo tale che la rete *TheGraph* potesse mettersi in ascolto degli eventi emessi dallo smart contract *Nftmarketplace*, al fine di aggiornare il *front-end* in modo decentralizzato.

5.4.3 Lettura da TheGraph

Spostandosi nell'applicazione dedicata al *front-end*, la possibilità di inoltrare delle *query* a *TheGraph* è stata introdotta importando il tool *Apollo*, il quale offre delle funzioni dedicate.

Le singole *query* sono state dichiarate tutte all'interno di un singolo file *JavaScript* ed, in seguito, esportate nelle diverse pagine del sito, in base alle necessità.

Il Listato 5.12 mostra la definizione della *query* per estrarre i prodotti "attivi" dal *subgraph* e il relativo uso all'interno della home page dell'applicazione.

Per introdurre una porzione di codice interpretabile tramite la sintassi *Graph QL* viene utilizzata la funzione `gql`, seguita dal testo della *query*.

L'obiettivo, nel caso del Listato 5.12, è di estrarre gli NFT messi in vendita attualmente disponibili. La *query*, infatti, specifica che devono essere prelevati gli elementi relativi all'entità `ActiveItem` definita nel *subgraph*, tali per cui l'indirizzo dell'acquirente sia nullo. La condizione sul parametro `buyer` è stata espressa tramite una clausola `where`. Dopodiché, vengono specificati gli attributi di interesse che devono essere restituiti.

Il testo della *query*, infine, viene memorizzato in una costante `GET_ACTIVE_ITEMS`, richiamabile all'occorrenza.

Una volta definita, la *query* viene attivata all'interno della funzione `Home` che implementa la pagina principale dell'applicazione, dove i token che rappresentano i prodotti disponibili verranno mostrati in appositi box.

L'esecuzione della *query* avviene richiamando la funzione `useQuery`, con parametro di input `GET_ACTIVE_ITEMS`. Questa restituisce tre elementi, ossia l'indicazione sullo stato di caricamento, l'eventuale errore sollevato ed i dati veri e propri.

La lista di NFT che soddisfano i criteri di ricerca, e i relativi attributi, vengono organizzati all'interno di un oggetto strutturato di tipo JSON, restituito dalla funzione `useQuery` e memorizzato nella variabile `listedNfts`. Infine, quest'ultima viene utilizzata all'interno del codice HTML, richiamando su di essa una funzione `map` che ne processa il contenuto; accedendo ai campi dell'oggetto JSON, infatti, è possibile ricavare i dati da visualizzare nel box di ciascun NFT, secondo quanto già discusso.

```
1  import { gql, useQuery } from "@apollo/client";
2
3  // ... File with all subgraph queries ...
4
5  export const GET_ACTIVE_ITEMS = gql`
6    {
7      activeItems(where: { buyer: null }) {
8        id
9        seller
10       nftAddress
11       tokenId
12     }
13   }
14 `;
15
16 // ------------------------------------------------------------
17
18 // .... File index.js for home page ....
19
20 export default function Home() {
21   // The Graph query -> list of tokens available on the market
22   const { loading, error, data: listedNfts } = useQuery(GET_ACTIVE_ITEMS);
23
24   return (
25     <div className="px-8 mx-auto">
26       {loading ? (
27         <div className="px-4">
```

```
28            <LoadComponent></LoadComponent>
29          </div>
30        ) : (
31          listedNfts.activeItems.map((nft) => {
32            console.log(nft);
33            const { nftAddress, tokenId, seller } = nft;
34            return (
35              <div className="py-4 place-content-center">
36                <NFTBox
37                  nftAddress={nftAddress}
38                  tokenId={tokenId}
39                  seller={seller}
40                  key={`${nftAddress}${tokenId}`}
41                />
42              </div>
43            );
44          })
45        )}
46      </div>
47    );
48  }
```

Listing 5.12: Definizione ed attivazione della *query* per estrarre i token disponibili

5.5 Manuale utente

Dopo aver implementato il *front-end* e aver stabilito la connessione alla blockchain e al servizio *TheGraph*, è stato possibile testare tutto il sistema, simulando l'esperienza utente completa.

5.5.1 Istruzioni per il set-up e l'avvio dell'applicazione

I *code repository* relativi ai progetti sviluppati sono consultabili sulla piattaforma *GitHub*, tramite i seguenti link:

- Back-end e smart contract: `https://github.com/FedeMiscia/products-authenticity`;
- Subgraph: `https://github.com/FedeMiscia/graph-authenticity`;
- Front-end: `https://github.com/FedeMiscia/nextjs-products-authenticity`

Per poter utilizzare l'applicativo *LuxuryChain*, innanzitutto, è necessario che gli smart contract *ProductNft* e *NftMarketplace* siano correttamente registrati sulla rete di test *Sepolia*.
Di seguito sono indicati gli indirizzi dei contratti all'interno della blockchain adottata:

- *ProductNft*: 0x1Fb05F071da6536bfF1Cd8035EF08994b869F0Da;
- *NftMarketplace*: 0xAc7a9EC00c94421fDfc373eE68867194Ef36862e

Incollando tali indirizzi all'interno del block explorer *Etherscan* (`https://sepolia.etherscan.io/`) è possibile monitorare le transazioni che coinvolgono i due contratti citati.
In secondo luogo, il *subgraph* discusso nella Sezione 5.4.2 deve risultare attivo; per assicurarsi di ciò, è possibile monitorare lo stato dell'API attraverso apposita *dashboard* fornita dal sito *Subgraph Studio*, una volta aver connesso il wallet con il quale è stato effettuato il *deploy*.

Infine, deve essere avviato il *front-end*. A tal proposito, secondo quanto riportato dalla documentazione di *Next.js*, occorre aprire un terminale all'interno della directory di progetto e lanciare il comando `yarn dev`, il quale avvia il server locale tramite cui l'applicazione risulta raggiungibile.

Per iniziare la navigazione all'interno del sito, è necessario collegarsi, tramite *browser*, all'URL `http://localhost:3000/` il quale rimanda alla pagina principale.

Trattandosi di un'applicazione che sfrutta funzionalità *Web3*, inoltre, è richiesto che l'utente disponga di un proprio wallet, che può essere creato, ad esempio, tramite *Metamask*.

5.5.2 Navigazione nell'applicazione

Una volta raggiunta l'applicazione tramite un *web browser*, la home page mostra semplicemente una breve descrizione degli obiettivi e del funzionamento. Per iniziare ad utilizzare le funzionalità previste è necessario cliccare sul pulsante "Connect Wallet".

La Figura 5.4 mostra le fasi iniziali della navigazione.

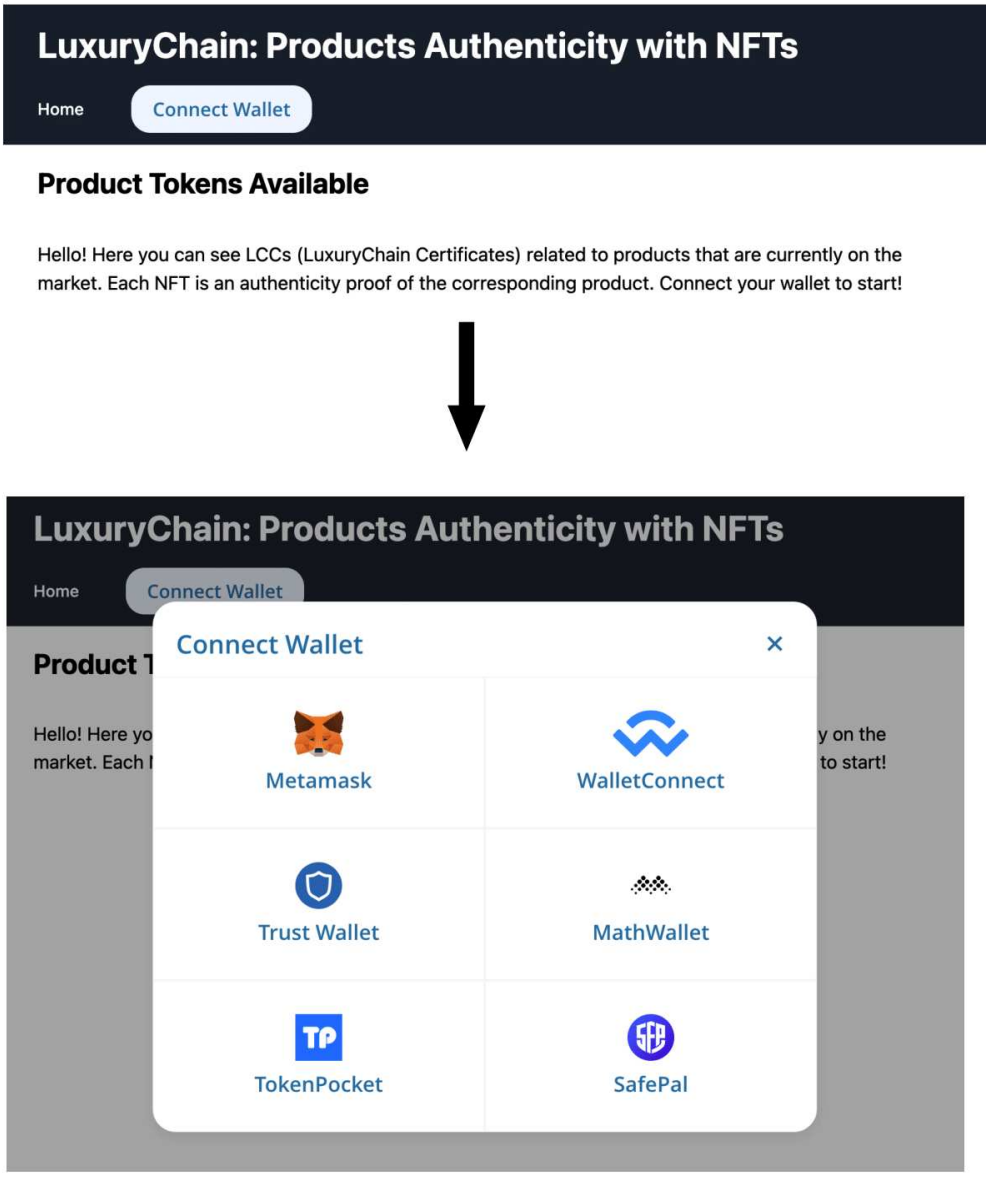

Figura 5.4: Ingresso nell'applicazione e connessione di un wallet

In seguito alla connessione, la home page mostra tutti i *LuxuryChain Certificate* che rappresentano prodotti autenticati attualmente disponibili alla vendita, sia nuovi che di seconda mano.

Ipotizzando che sia stato già pubblicato un token, la Figura 5.5 illustra ciò che vede l'utente nella home page.

Ciascun NFT che può essere acquisito è mostrato in un box, all'interno del quale l'utente può leggere l'ID, l'attuale proprietario (ossia il venditore), l'immagine, il nome del prodotto a cui si riferisce e l'account che l'ha creato, dunque l'indicazione sull'origine.

Nel caso specifico della Figura 5.5, l'attuale proprietario e il creatore dell'NFT coincidono; ciò significa che si tratta di un prodotto nuovo, venduto direttamente dal *brand*.

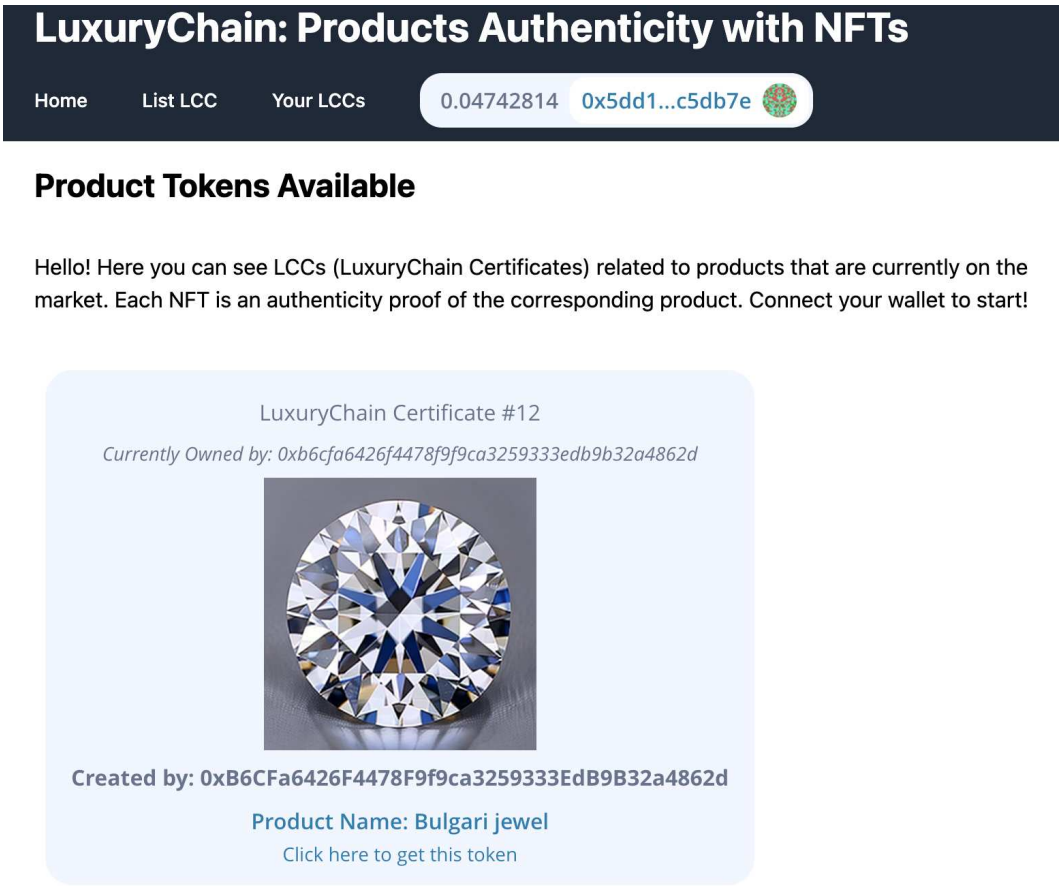

Figura 5.5: Home page dell'applicazione con i token disponibili alla vendita

Cliccando sul box di un *LuxuryChain Certificate*, si apre in sovrimpressione una scheda che riepiloga tutti i dettagli del prodotto che il token certifica. È importante sottolineare che tutte le informazioni qui riportate costituiscono dati notarizzati sulla blockchain, dunque immutabili e verificati.

La Figura 5.6 mostra la "scheda tecnica" di un *LuxuryChain Certificate*. Se il token non è di proprietà dell'account attualmente connesso allora, all'interno della scheda prodotto, è cliccabile unicamente il pulsante "Get it!", il quale rimanda alla procedura di acquisto. Se l'account è quello del venditore, invece, l'utente può cliccare sul tasto "Remove Listing", qualora intenda rimuovere l'annuncio di vendita.

Le operazioni per inoltrare e verificare il pagamento durante l'acquisto non sono state trattate; dunque, il pulsante "Get it!" si limita ad avviare l'interazione con la blockchain,

secondo quanto discusso nella Sezione 5.3.4, ipotizzando che il pagamento sia andato a buon fine.

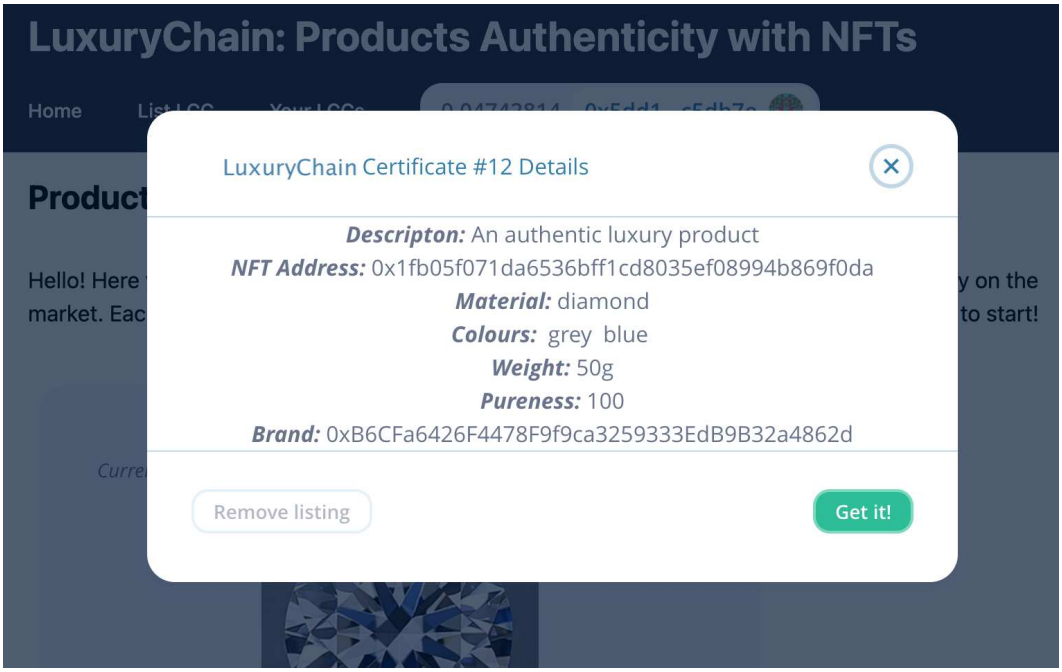

Figura 5.6: Scheda tecnica relativa ad un *LuxuryChain Certificate* che autentica un diamante

La conferma dell'acquisto viene notificata tramite un *pop-up*; dopodiché, l'utente viene reindirizzato alla propria pagina personale, in cui ha modo di consultare un riepilogo dei *LuxuryChain Certificate* attualmente in *staking*, per i quali è disponibile il reso, e quelli già ricevuti, dunque ufficialmente in suo possesso.

La Figura 5.7 mostra la pagina di riepilogo dei token di un utente, dopo aver effettuato l'acquisto.

Il *LuxuryChain Certificate* associato al prodotto appena acquistato è stato inserito nella sezione dello *staking*, dove viene indicata anche la data entro la quale richiedere il reso. In più, dopo i dati del token, è presente anche il pulsante che permette di effettuare la restituzione.

Nella sezione in basso, invece, sono riportati i *LuxuryChain Certificate* già trasferiti all'utente da acquisti precedenti. Per questi token è possibile cliccare sul pulsante "List item", nel caso in cui si decida di metterli in vendita.

Allo scadere del periodo di reso, qualora non sia stata effettuata richiesta di restituzione, avviene automaticamente il trasferimento del *LuxuryChain Certificate* che, da questo momento, diviene ufficialmente di proprietà dell'utente, come mostrato in Figura 5.8.

Per mettere in vendita un prodotto, pubblicando il relativo *LuxuryChain Certificate*, può essere utilizzato l'apposito pulsante disponibile nella sezione di riepilogo dei token ufficialmente trasferiti o, in alternativa, si può accedere alla pagina "List LCC", dove è richiesto l'inserimento dell'indirizzo e dell'ID dell'NFT in questione. Tale pagina è mostrata in Figura 5.9.

Infine, supponendo di simulare le operazioni svolte da un *brand*, è stata effettuata la connessione all'applicazione tramite l'account utilizzato per il *deploy* dello smart contract *NftMarketplace*. Alla luce di ciò, viene abilitato l'accesso alla pagina "Create LCC", mostrata in Figura 5.10, utile per generare un nuovo *LuxuryChain Certificate*.

Come per tutte le azioni che innescano delle scritture sulla blockchain, nel momento in cui viene cliccato il pulsante "Create Product Token" si apre la finestra di *Metamask* che richiede la conferma della transazione.

Dopodiché, per mettere in vendita il prodotto del quale è stato generato il certificato di autenticità, il *brand* può accedere alla pagina "List LCC", mostrata in Figura 5.9.

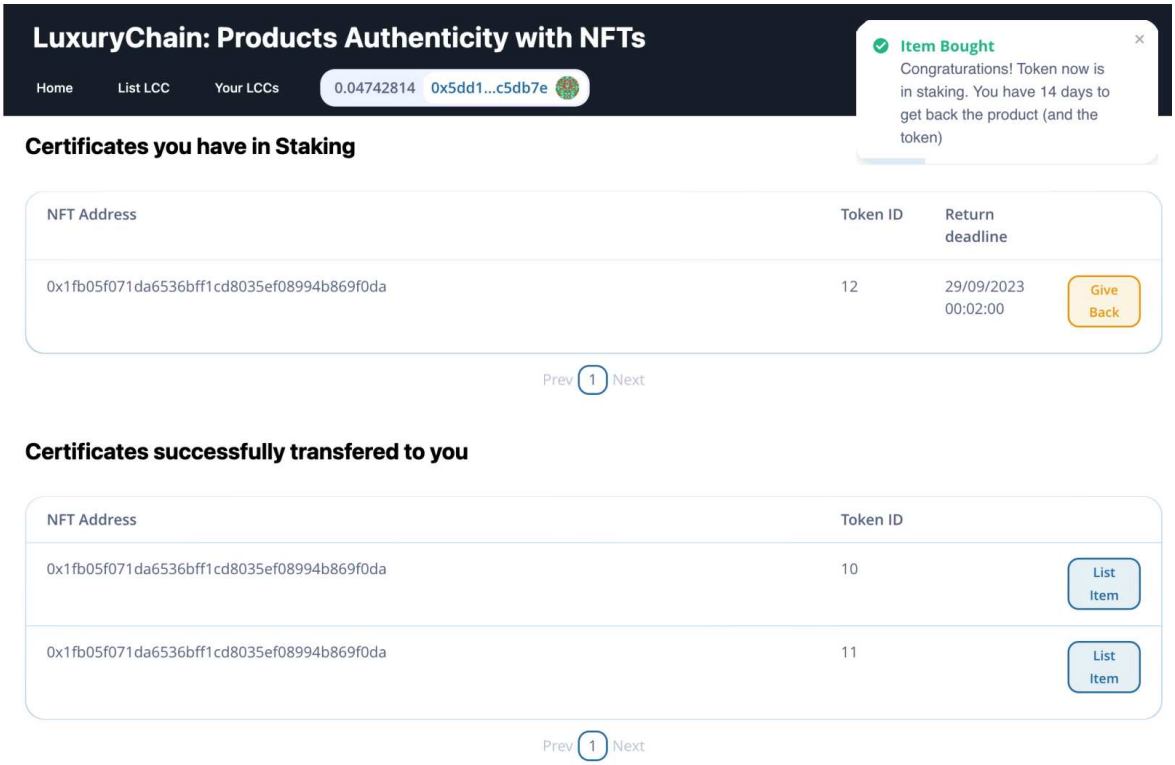

Figura 5.7: Pagina di riepilogo dei certificati di un utente in seguito all'acquisto di un prodotto

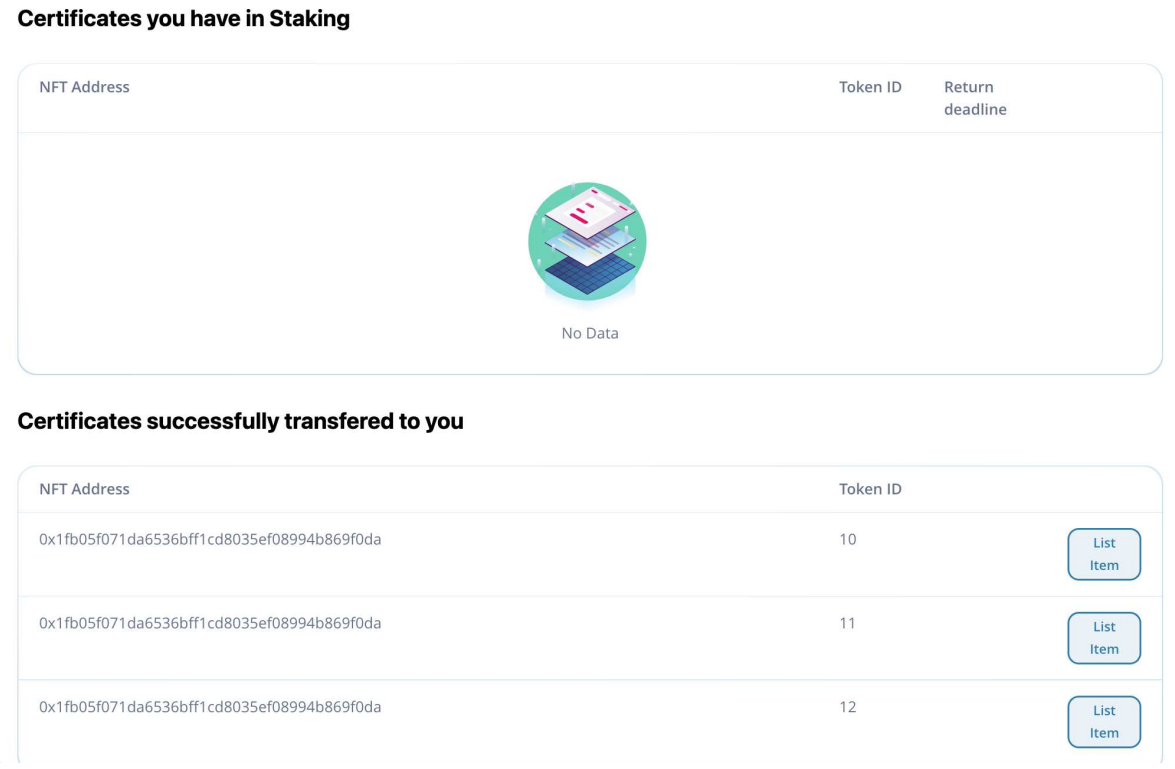

Figura 5.8: Pagina di riepilogo dei certificati di un utente a seguito del trasferimento di un token

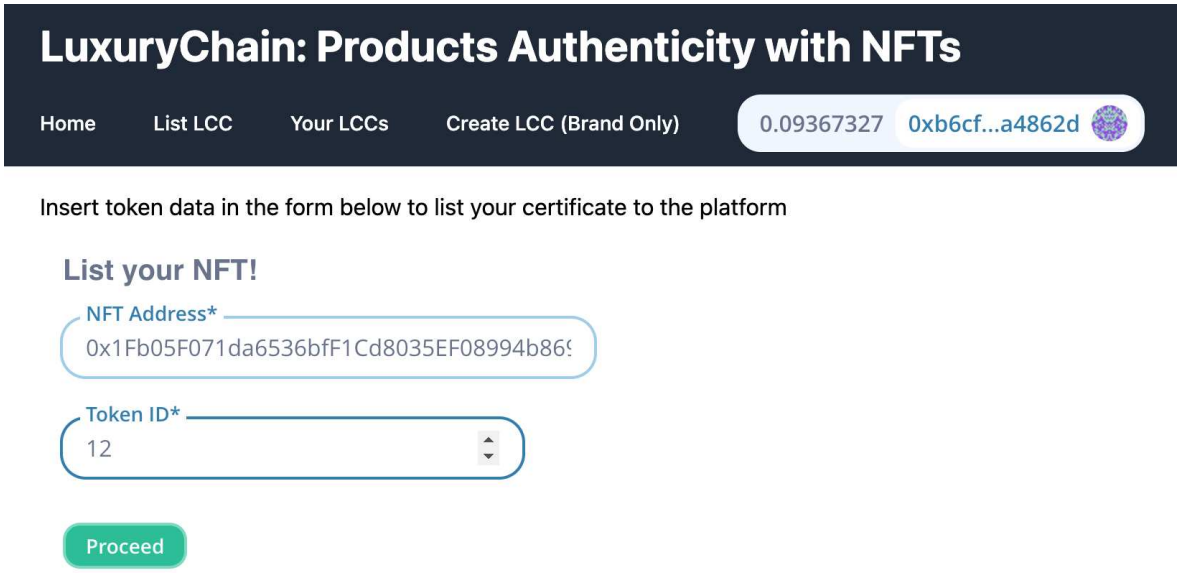

Figura 5.9: Pagina per la pubblicazione di un *LuxuryChain Certificate* ai fini della vendita

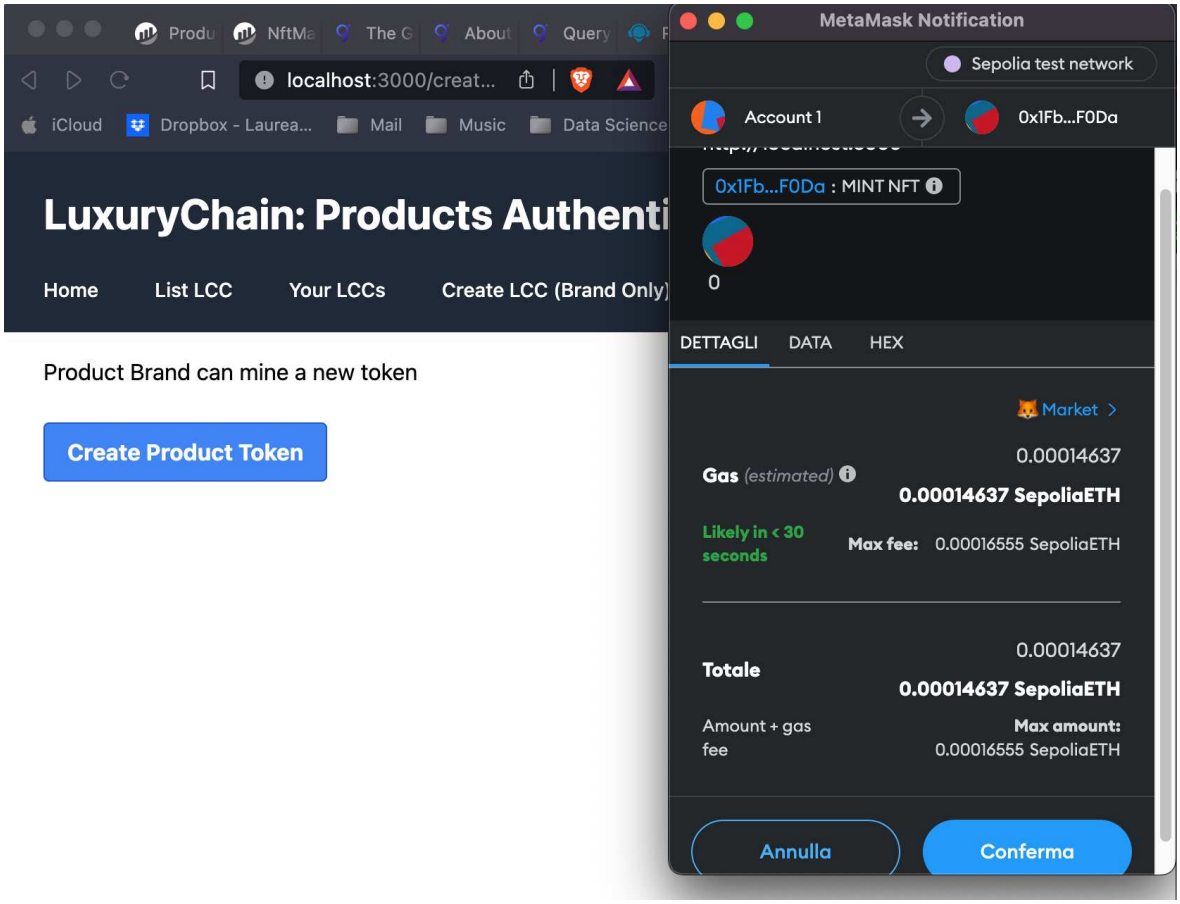

Figura 5.10: Generazione di un nuovo *LuxuryChain Certificate*

CAPITOLO 6

Conclusioni e sviluppi futuri

In questa tesi è stata progettata ed implementata un'applicazione decentralizzata che, sfruttando le nozioni di token non fungibili e blockchain, è in grado di certificare l'autenticità di prodotti di lusso e tracciarne il passaggio di proprietà in maniera indelebile. In particolare, l'attenzione è stata focalizzata su tutti i movimenti che avvengono, a valle del processo produttivo, nella *post supply-chain*, ossia dal rilascio finale del bene da parte di un *brand* di lusso. In questo scenario, dunque, operano sia attori universalmente riconoscibili che detengono un certo livello di affidabilità, come aziende produttrici e rivenditori ufficiali, sia figure con cui, a priori, non è possibile stabilire un rapporto di fiducia, quali piattaforme di *e-commerce* e rivenditori privati di seconda mano. In ogni caso, tramite il sistema sviluppato, si è cercato di trasferire la "fonte" della fiducia dall'entità in sè con cui si interagisce alla solidità crittografica del protocollo di comunicazione sottostante, considerando tutti gli attori alla stessa stregua ed eliminando, di fatto, la dipendenza da terze parti.

Per garantire una totale chiarezza del caso di studio, nella fasi iniziali del lavoro è stato presentato un *excursus* sui principi fondanti delle blockchain e sullo stato dell'arte del loro utilizzo ai fini del tracciamento.

Terminata la fase introduttiva, sono stati raccolti ed analizzati i requisiti funzionali che hanno guidato lo sviluppo del software; a tal proposito, è stato utilizzato l'apposito linguaggio di modellazione *i**, il quale ha permesso di evidenziare gli obiettivi dei principali attori individuati e di capire come questi potessero essere delegati al sistema, realizzando, dunque, dei diagrammi dei casi d'uso.

Giungendo nella fase di progettazione, sulla base degli attori modellati sono stati ideati dei moduli software che hanno costituito l'assetto architetturale di partenza. Dopodiché, tramite un'analisi del rischio, sono stati individuati gli asset fondamentali da proteggere e, di conseguenza, l'attenzione si è spostata sui requisiti non funzionali attinenti alla sicurezza, in particolare quelli di autenticità ed integrità. Tale processo ha avuto un riscontro diretto sull'architettura del sistema perché è emersa la sovrapponibilità con l'approccio decentralizzato delle blockchain.

Avendo chiarito con precisione cosa il software dovesse fare e quale fosse l'organizzazione strutturale più adatta per soddisfare le esigenze di trasparenza, autenticità ed immutabilità, è stato possibile procedere all'implementazione vera e propria; l'ultima fase del lavoro, infatti, ha previsto la scrittura di due *smart contract* che realizzano la logica del sistema sulla blockchain, e lo sviluppo di un'interfaccia web, tramite la quale gli utenti possono interagire, in maniera semplificata, con le funzionalità previste.

In particolare, un primo smart contract consente di creare la controparte digitale di un prodotto reale e, di conseguenza, utilizza lo standard *ERC-721* per la generazione di NFT. Tale asset digitale, a cui è stato dato il nome di "LuxuryChain Certificate", racchiude, al suo interno, tutti i dati in grado di attestare la qualità di un prodotto, tra cui le materie prime utilizzate, l'impatto ambientale dei processi produttivi, i controlli soddisfatti e le certificazioni ottenute. Oltre a ciò, il token può memorizzare anche tutte quelle caratteristiche fisiche nelle quali risiede l'unicità del prodotto; infatti, tali tratti distintivi possono essere utilizzati per riprodurre il "digital twin" del bene di lusso reale anche a livello grafico, ricalcandone l'aspetto.

La sicurezza del sistema sviluppato deriva dal fatto che l'NFT, in quanto creato e scambiato sulla blockchain, costituisce un oggetto digitale unico che non può essere copiato o sostituito; i suoi dati, infatti, sono a prova di falsificazione perché permanentemente trascritti su un registro distribuito ed, in più, possono essere crittograficamente verificati in qualsiasi momento, senza la necessità di intermediari.

Un cliente, dunque, al momento dell'acquisto di un prodotto di lusso, riceve anche l'NFT corrispondente che, avendo il ruolo di certificato di autenticità, consente di consolidare ed incrementare il valore del prodotto stesso. Ciò è particolarmente utile anche nel caso di una futura cessione; difatti, se un prodotto di seconda mano è stato certificato, a monte, tramite NFT, allora, verosimilmente, aumenteranno le probabilità di rivendita e, al contempo, il prodotto non subirà una svalutazione eccessiva, proprio perchè il token può dimostrarne la qualità, assicurando un prezzo equo nel tempo. Per questo motivo, se affiancato da un'adeguata campagna di promozione e sensibilizzazione, il sistema sviluppato può risultare efficace anche per ridurre gli acquisti volontari di merce contraffatta.

Un altro punto di forza è che il token di un prodotto, oltre che un attestato di autenticità, costituisce, a tutti gli effetti, anche un certificato di proprietà, utile per risolvere eventuali controversie o per identificare più agevolmente prodotti rubati. Infatti, tramite la blockchain, ogni passaggio di consegne dell'NFT viene permanentemente tracciato, così da poter ricostruire, in maniera inequivocabile, la storia del prodotto associato. Ciò, come facilmente intuibile, è fondamentale per risalire all'origine del bene e rilevare eventuali anomalie rispetto a quanto dichiarato; ad esempio, le contraffazioni possono essere individuate in maniera più semplice se un utente non è in grado di dimostrare la proprietà dei prodotti rivendicati.

La facoltà di gestire la versione digitale di un prodotto sotto forma di NFT, inoltre, può risultare una *feature* fortemente attrattiva per l'utente, andando a rendere ancora più personalizzata e gratificante l'esperienza di acquisto di un bene di lusso. In quest'ottica, le potenzialità dell'applicazione possono far breccia anche, e soprattutto, sulla cosiddetta "generazione Z", incline fortemente sia al settore della moda e del lusso, sia alle novità in campo tecnologico.

Dal punto di vista operativo, l'amministrazione del "digital twin" di un prodotto regolarmente acquisito è stata supportata per mezzo di un secondo smart contract che ha il compito di regolamentare il trasferimento dei certificati, parallelamente al processo di compravendita. Tale contratto, inoltre, risolve anche la gestione automatizzata del reso di un prodotto e del relativo certificato, focalizzando costantemente l'attenzione sull'efficienza computazionale, nell'obiettivo di limitare i costi delle transazioni sulla blockchain e ridurre i tempi di *processing*.

Ad ogni modo, va sottolineato che, per le finalità del lavoro, il *deploy* definitivo degli smart contract è stato effettuato su una *TestNet* pubblica; reti di questo tipo, pur consentendo di simulare uno scenario reale in modo sicuro e privo di costi, sono altamente soggette a congestione, con tempi di risposta piuttosto variabili. In ottica futura, dunque, dovrà essere valutata attentamente la tipologia di rete blockchain da utilizzare; ad esempio, può essere presa in considerazione la piattaforma *Hyperledger Besu*, la quale sta diventando sempre più

di riferimento per l'Unione Europea e particolarmente adatta per contesti consorziali.

Relativamente agli svilupppi auspicabili, inoltre, può essere sicuramente ampliato e migliorato, in particolar modo, il *front-end* dell'applicazione. Un obiettivo, a tal proposito, può essere quello di nascondere il più possibile, agli occhi dell'utente, le interazioni esplicite con il proprio wallet, in modo da rendere più fluida la navigazione. Alla luce di ciò, andrebbero progettate delle apposite API per gestire, dietro le quinte, la creazione di un nuovo wallet, il recupero tramite *seed phrase*, la cifratura delle chiavi private e l'inoltro delle transazioni. Sempre nella direzione di una maggiore fruibilità dell'interfaccia utente, inoltre, sarà importante limitare l'utilizzo diretto degli indirizzi blockchain, ad esempio includendo le informazioni di contatto all'interno di *QR-Code*. Gli aspetti citati, infatti, essendo strettamente connessi a nozioni proprie del mondo blockchain, possono risultare ostiche ad utenti non esperti.

Nell'ottica della flessibilità, poi, la pagina dedicata al *brand* per la creazione di un nuovo NFT dovrebbe consentire di specificare, in maniera puntuale, le caratteristiche rilevanti da certificare, in relazione allo specifico prodotto in questione. Nell'attuale implementazione, infatti, i dati da inserire nell'NFT sono stati codificati all'atto del *deploy* del relativo smart contract.

Dopodichè, l'area dedicata al *brand* può essere ulteriormente estesa con la possibilità di inoltrare una chiamata API al database aziendale, tramite cui recuperare le informazioni su un prodotto che deve essere certificato.

A livello pratico, dopo aver generato il certificato, il *brand* deve farsi carico di apporre sui prodotti fisici dei tag NFC o dei *QR-Code* che, una volta scansionati tramite smartphone, consentono ai rivenditori e ai consumatori di visualizzare l'NFT che ne dimostra l'autenticità.

Un'ulteriore funzionalità accattivante in ottica futura, specialmente per beni di lusso personali, è la possibilità di sfruttare gli NFT all'interno del metaverso, dando la possibilità ai consumatori di provare "virtualmente", tramite i propri avatar, i prodotti che sono in procinto di acquistare.

Infine, a livello più generale, si può pensare di integrare dei moduli di *machine learning* per funzionalità quali il rilevamento di anomalie, il *clustering* e la classificazione dei consumatori in relazione alle abitudini di acquisto, nonchè l'assegnazione di *reward* o penalità sulla base del comportamento degli utenti.

Il sistema proposto, in virtù di ciò, può costituire delle solide fondamenta per portare ad un livello superiore l'ecosistema dei *marketplace*, in cui tutti gli utenti possono contare sulla validità delle informazioni che circolano e, al tempo stesso, nessuno è in grado di acquisirne il totale controllo.

Bibliografia

AGARWAL, U., RISHIWAL, V., TANWAR, S., CHAUDHARY, R., SHARMA, G., BOKORO, P. N. e SHARMA, R. (2022), «Blockchain Technology for secure supply chain management: a comprehensive review», *IEEE Access - Survey*. (Cited at page 26)

ALTAGAMMA, BAIN e COMPANY (2022), «Renaissance in Uncertainty: Luxury Builds on Its Rebound», in «Altagamma–Bain worldwide luxury market monitor 2022», . (Cited at page 39)

AZZI, R., CHAMOUN, R. K. e SOKHN, M. (2019), «The power of a blockchain-based supply chain», *Computers and Industrial Engineering*. (Cited at page 31)

GARTNER (2019), «Blockchain Technology: What's Ahead?», *Gartner [online]*. (Cited at page 30)

LO, S. K., XU, X., CHIAM, Y. K. e LU, Q. (2017), «Evaluating Suitability of Applying Blockchain», in «2017 International Conference on Engineering of Complex Computer Systems», . (Cited at page 77)

PHILIPS, W. e WICAKSANA, A. (2022), «HYBRID APPROACH OF QUICK RESPONSE CODE AND NON-FUNGIBLE TOKEN IN PRIVATE PERMISSIONED BLOCKCHAIN FOR ANTI-COUNTERFEITING», *International Journal of Innovative Computing, Information and Control*. (Cited at page 33)

SCRIBER, B. A. (2018), «A Framework for Determining Blockchain Applicability», *IEEE Software*. (Cited at page 77)

SOMMERVILLE, I. (2015), *Software Engineering*, Pearson. (Cited at page 73)

TIAN, F. (2016), «An agri-food supply chain traceability system for China based on RFID and blockchain technology», in «International conference on service systems and service management», . (Cited at page 27)

TROFA, F. L. (2021), «Blockchain per la tracciabilità: lo scenario evolutivo del controllo della filiera», *Tech4Future [online]*. (Cited at page 38)

TURKI, M., CHEIKHROHOU, S., DANMAK, B., BAKLOUTI, M., MARS, R. e DHAHBI, A. (2022), «NFT-IoT Pharma Chain : IoT Drug traceability system based on Blockchain and Non Fungible Tokens (NFTs)», *Journal of King Saud University – Computer and Information Sciences*. (Cited at page 35)

WANG, G., SHI, S., WANG, M., QIAN, C., ZHAO, C., DING, H., XI, W. e ZHAO, J. (2022), «RF-Chain: Decentralized, Credible, and Counterfeit-proof Supply Chain Management with Commodity RFIDs», *Association for Computing Machinery*. (Cited at page 33)

Siti web consultati

- Chainlink: The Industry-Standard Web3 Services Platform – `https://chain.link`
- Common Attack Pattern Enumerations and Classifications (CAPEC™) – `https://capec.mitre.org/`
- Mitre ATT&CK – `https://attack.mitre.org/`
- Solidity Programming Language – `https://soliditylang.org/`
- Hardhat | Ethereum Development Environment for Professionals – `https://hardhat.org/`
- OpenZeppelin – `https://www.openzeppelin.com/`
- IPFS Powers the Distributed Web – `https://ipfs.tech`
- Pinata | IPFS API & IPFS Gateway – `https://pinata.cloud`
- Next.js by Vercel - The React Framework – `https://nextjs.org/`
- TheGraph – `https://thegraph.com/`

Ringraziamenti

Il passo che mi appresto a compiere è frutto di un percorso indubbiamente stimolante ma, al tempo stesso, intenso e talvolta tortuoso, non lo nego. Di certo, però, questi anni universitari hanno consentito di formarmi come persona, oltre che a livello di competenze; di conseguenza, desidero dedicare delle righe a tutti coloro che mi hanno accompagnato in questo viaggio e, in un modo o nell'altro, hanno contribuito ad arricchirne il valore.

Innanzitutto, vorrei ringraziare il Prof. Ursino che mi ha guidato, con grande disponibilità e precisione, in questi ultimi mesi e nella scrittura della tesi. Sono certo che la dedizione e la passione che riserva nei confronti del suo lavoro vengano trasmesse a tutti i suoi studenti, così come lo è stato per me. Un ringraziamento particolare va anche al Prof. Spalazzi e al dottorando Gianluca Bonifazi, per l'interesse mostrato e i preziosi consigli forniti durante l'attività di tirocinio.

Proseguendo, non potrò mai ringraziare abbastanza i miei genitori che, con i loro sforzi, mi hanno sempre messo nelle migliori condizioni possibili affichè potessi affrontare gli studi in serenità e senza pressioni.

Dopodichè vorrei ringraziare il resto della mia famglia, per l'affetto costante mostrato.

Un grazie anche ad Ancona, per avermi dato l'occasione di conoscere persone divenute ben presto importanti.
Ringrazio quindi Michele, Beatrice, Angelo Andrea, Nicola, Laura, Angela e Denis.

Un ringraziamento speciale va a Fabio, un ragazzo che tutti vorrebbero nella propria compagnia.

Desidero ringraziare altrettanto calorosamente Lello e Maria Pia, per la fiducia che hanno mostrato nei miei confronti.

Un ringraziamento va anche a Gabriele e Lorenzo, amici dai tempi delle medie con i quali resiste un bel rapporto.

Dopodichè, non posso trascurare tutto ciò che è connesso ad un'altra grande parte della mia vita, la musica.

Innanzitutto, quindi, ringrazio i compagni fraterni delle "Vie delle Indecisioni", Paride, Marco, Ludovico e Fabrizio.

Ringrazio poi Emiliano, Alessandro (Zio Fox), Isacco, Aurora e Luca, de "I conti del bar Mario".

Vorrei ringraziare, inoltre, Raffaele, con cui sono in grado di suonare senza neanche una prova.

Un ringraziamento va anche agli amici dei "Manduria", Carlo, Stefano, Giovanni e Lorenzo.

Riservo volutamente l'ultimo sentito ringraziamento ad una ragazza speciale, Silvia, perchè mi supporta ormai da anni ed è in grado di capirmi come pochi. Fai tanto per me anche con la sola presenza, grazie!

Grazie a tutti,

Federico