



UNIVERSITÀ DI PISA

DIPARTIMENTO DI INFORMATICA

Laurea Triennale in Informatica

## Certificazione trustless di documenti tramite NFT minting su Blockchain

Relatore:

**Prof: Laura Ricci**

Tutore Aziendale:

**Dott: Luca Secci**

Candidato:

**Tommaso Mangiavacchi**

---

ANNO ACCADEMICO 2023/2024



# Indice

<b>1</b>	<b>Introduzione</b>	<b>7</b>
1.1	Esperienza aziendale . . . . .	7
1.2	Panoramica . . . . .	7
<b>2</b>	<b>Background</b>	<b>9</b>
2.1	Blockchain . . . . .	9
2.1.1	Principi Fondamentali . . . . .	9
2.1.2	Struttura delle Blockchain . . . . .	10
2.1.3	Protocolli di consenso . . . . .	11
2.1.4	Bitcoin . . . . .	11
2.1.5	Ethereum . . . . .	12
2.1.6	Address in Ethereum . . . . .	12
2.2	Sicurezza delle Blockchain . . . . .	13
2.2.1	Attacchi del 51% . . . . .	13
2.2.2	Attacchi double spending . . . . .	14
2.3	Smart Contract . . . . .	15
2.3.1	Gas . . . . .	16
2.4	Token . . . . .	17
2.4.1	Token fungibili . . . . .	18
2.4.2	Non-fungible token (NFT) . . . . .	18
2.4.3	Standard dei Token . . . . .	19
2.5	InterPlanetary File System (IPFS) . . . . .	19
2.6	Rollup . . . . .	21
2.6.1	Optimistic Rollups . . . . .	21
2.6.2	zkRollups . . . . .	22
2.7	API (Application Programming Interface) . . . . .	22
<b>3</b>	<b>Il problema affrontato</b>	<b>25</b>
3.1	Digital Product Passport (DPP) . . . . .	25
3.2	DPP su Blockchain . . . . .	26
3.3	NFT e Luxury Fashion . . . . .	27

<b>4</b>	<b>Implementazione</b>	<b>29</b>
4.1	Strumenti e servizi utilizzati . . . . .	29
4.1.1	ExpressJS . . . . .	29
4.1.2	Hardhat . . . . .	29
4.1.3	EthersJS . . . . .	29
4.1.4	Infura . . . . .	30
4.1.5	Linea . . . . .	31
4.1.6	React . . . . .	31
4.1.7	Material UI (MUI) . . . . .	32
4.1.8	ERC-721A . . . . .	33
4.2	Back-end . . . . .	34
4.2.1	API Tokenize . . . . .	34
4.2.2	API Verify . . . . .	35
4.3	Realizzazione dello Smart Contract . . . . .	35
4.3.1	Strutture Dati . . . . .	35
4.3.2	SafeMint() . . . . .	36
4.3.3	Verify() . . . . .	38
4.4	Front-end . . . . .	38
4.4.1	Caratteristiche Principali . . . . .	39
4.4.2	Aspetto del Front-end . . . . .	39
<b>5</b>	<b>Risultati sperimentali</b>	<b>41</b>
5.1	Confronto tra Storage, Memory e Calldata . . . . .	41
5.2	Utilizzo di Gas . . . . .	42
5.3	Ottimizzazione dello Storage . . . . .	42
5.3.1	Svantaggi . . . . .	43
5.3.2	Ottimizzazioni Minori . . . . .	43
5.4	Conclusioni . . . . .	44
<b>6</b>	<b>Conclusioni e Sviluppi Futuri</b>	<b>45</b>
6.1	Conclusione . . . . .	45
6.2	Sviluppi Futuri . . . . .	45
6.3	Competenze Acquisite . . . . .	46
<b>A</b>	<b>Glossario dei Termini</b>	<b>47</b>
A.1	Mining . . . . .	47
A.2	Staking . . . . .	47
A.3	Minting . . . . .	47
A.4	Burning . . . . .	47
A.5	Batch Minting . . . . .	47
A.6	BaseURI . . . . .	48
A.7	Layer 2 . . . . .	48
A.8	Trustless . . . . .	48
A.9	Provider . . . . .	48

---

A.10 Back-End . . . . .	48
A.11 Server . . . . .	48
A.12 Front-End . . . . .	48
A.13 Reactive . . . . .	48
A.14 HTML . . . . .	49
A.15 CSS . . . . .	49
A.16 JavaScript . . . . .	49
A.17 Web App . . . . .	50
A.18 Framework . . . . .	50
A.19 Node.js . . . . .	50



# Capitolo 1

## Introduzione

Questo testo ha lo scopo di illustrare l'architettura e le peculiarità del "Sistema di Certificazione Documentale" ideato e sviluppato presso l'azienda *Advinser*.

### 1.1 Esperienza aziendale

Ho avuto l'opportunità di svolgere il mio tirocinio presso Advinser, un'azienda di medie-piccole dimensioni situata nella provincia di Siena, con la quale avevo già collaborato durante uno stage formativo alle scuole superiori. Dopo averli contattati per sapere se fossero disponibili ad inserirmi nello sviluppo di uno dei loro progetti, ho partecipato ad un colloquio con le risorse umane, durante questo incontro, mi è stato presentato un argomento che ha suscitato il mio interesse. Successivamente, nel progredire della conversazione, ho appreso i dettagli implementativi del sistema da sviluppare e ho confermato che avrei preso in carico lo sviluppo del progetto.

Il tirocinio è stato svolto alternando giornate in presenza e giornate in remoto, per una durata complessiva di circa un mese e mezzo. Durante questa esperienza, ho avuto l'opportunità di immergermi nel contesto lavorativo e di contribuire attivamente allo sviluppo di un progetto. Sotto la guida del mio tutor aziendale, ho avuto modo di acquisire competenze pratiche nel campo dello sviluppo web e delle blockchain.

### 1.2 Panoramica

Il progetto prevede, tramite tecnologie web e blockchain, lo sviluppo di un sistema di certificazione documentale. Questo sistema distribuito, basato su blockchain, consente a chiunque possieda una chiave di licenza valida di certificare un documento digitale. Dopo la certificazione, l'utente può verificare l'autenticità del documento tramite l'interfaccia utente fornita dall'azienda.

La scrittura di smart contract è un elemento fondamentale del progetto. Questi smart contract gestiscono la certificazione dei documenti e offrono funzionalità di storicizzazione delle versioni. Ogni volta che un documento viene certificato, una

nuova versione viene registrata sulla blockchain. Questo rende possibile la tracciabilità delle modifiche e garantisce che tutte le versioni precedenti rimangano immutabili e verificabili.

Un altro aspetto chiave del progetto è l'implementazione di un servizio backend dedicato per interfacciarsi con gli smart contract. Il servizio back-end, sviluppato usando il framework Express.js, funge da intermediario tra gli utenti e la blockchain.

Nel contesto di questo progetto, il servizio back-end facilita la comunicazione tra l'applicazione web e la blockchain. Tuttavia, è importante sottolineare che solo il proprietario del contratto può eseguire le operazioni di certificazione e aggiornamento. Questo garantisce un ulteriore livello di sicurezza, assicurando che solo utenti autorizzati possano certificare i documenti.

Un ulteriore punto di rilevanza è l'integrazione del sistema con il concetto di Digital Product Passport (DPP) europeo. Il DPP è un'iniziativa volta a tracciare e documentare la storia, la composizione e la conformità dei prodotti lungo tutto il loro ciclo di vita. Utilizzando la blockchain per la certificazione dei documenti, il nostro sistema fornisce una soluzione ideale per implementare il DPP, specialmente nel settore del luxury fashion.

Il Digital Product Passport consente ai produttori di documentare dettagli cruciali su ogni prodotto, come la provenienza dei materiali, le pratiche di produzione sostenibili, le certificazioni di autenticità e i record di manutenzione. Implementando questi passaporti digitali sulla blockchain, possiamo garantire che le informazioni siano immutabili, facilmente accessibili e verificabili da qualsiasi parte interessata.



# Capitolo 2

## Background

### 2.1 Blockchain

Le blockchain sono reti decentralizzate, possono essere considerate come un registro pubblico e immutabile di transazioni digitali, distribuito tra una rete di nodi o computer interconnessi. Una funzionalità cardine delle blockchain come Bitcoin e Ethereum è la possibilità di scambiare denaro in modo anonimo e difficilmente rintracciabile, senza essere limitati da vincoli geografici. Inoltre le applicazioni delle blockchain vanno ben oltre il mondo delle criptovalute, spaziando in settori come Internet of things (IoT), immobiliare, certificazione digitale, tracciabilità delle merci, collezionismo e investimenti d'arte. Tuttavia, nonostante i numerosi vantaggi, le blockchain affrontano ancora sfide significative, come la scalabilità e le questioni normative. Inoltre, il processo di mining dei blocchi per le blockchain basate su Proof of Work, solleva preoccupazioni riguardo all'impatto ambientale, data la necessità di massiccia potenza di calcolo.

#### 2.1.1 Principi Fondamentali

- **Decentralizzazione:** Contrariamente ai sistemi tradizionali che dipendono da un'autorità centrale per la gestione e la validazione delle transazioni, una blockchain opera in modo decentralizzato. Le transazioni vengono verificate e registrate da una rete di nodi distribuiti, eliminando così la necessità di una terza parte fiduciaria.
- **Immutabilità:** Una volta che una transazione è stata registrata su una blockchain e accettata dalla rete, diventa praticamente impossibile modificarla o cancellarla. Questo è dovuto alla natura crittografica della tecnologia blockchain, che rende estremamente difficile alterare i dati dei blocchi precedenti senza invalidare l'intera catena.
- **Trasparenza:** Tutte le transazioni effettuate su una blockchain sono visibili a tutti i partecipanti della rete. Questa trasparenza contribuisce a garantire l'integrità del sistema e a prevenire frodi o manipolazioni.

### 2.1.2 Struttura delle Blockchain

In ogni blockchain, i dati sono organizzati in blocchi, ciascuno dei quali contiene un insieme di transazioni. Questi blocchi sono concatenati in modo sequenziale, formando una catena in cui ogni blocco è collegato al blocco precedente tramite un riferimento crittografico, solitamente un hash del blocco precedente. Il collegamento crittografico garantisce l'integrità e l'immutabilità della catena, poiché una modifica a un blocco altererebbe l'hash dei blocchi successivi, rendendo l'alterazione evidente. Questo meccanismo fondamentale è alla base di tutte le blockchain, indipendentemente dalle variazioni specifiche o dalle implementazioni particolari che possono differire tra diverse piattaforme blockchain.

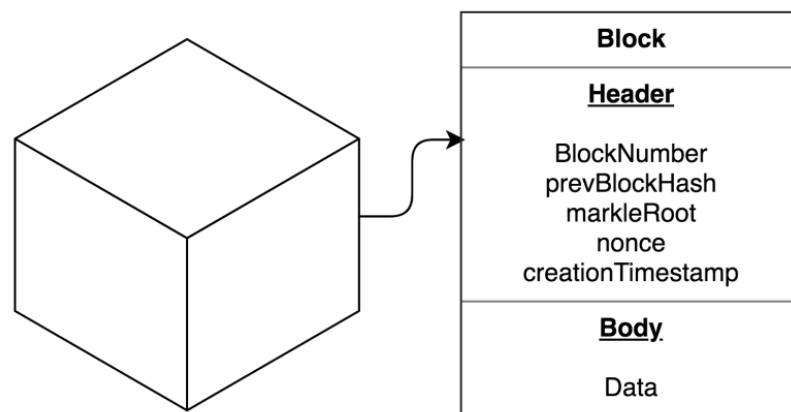


Figura 2.1: Esempio di informazioni contenute in un blocco

- **Block Number:** Un identificatore univoco per il blocco all'interno della catena.
- **Previous Hash:** L'hash del blocco precedente nella catena, che garantisce il collegamento crittografico tra i blocchi.
- **Nonce:** Un valore arbitrario che viene variato durante il processo di mining per trovare un hash valido per il blocco.
- **Merkle Root:** L'hash radice dell'albero di Merkle che rappresenta tutte le transazioni incluse nel blocco.
- **Timestamp:** La data e l'ora in cui il blocco è stato creato.

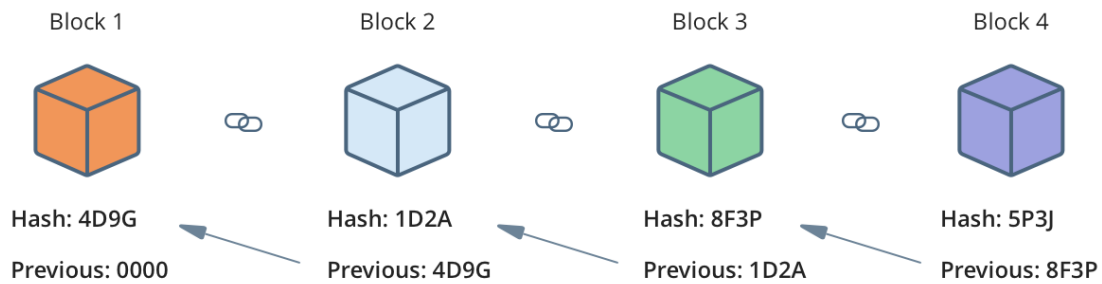


Figura 2.2: Esempio di una blockchain

### 2.1.3 Protocolli di consenso

La **Proof of Work** (PoW) e la **Proof of Stake** (PoS) sono due protocolli di consenso fondamentali utilizzati nelle più famose blockchain per garantire l'integrità e la sicurezza delle transazioni. La Proof of Work è il protocollo originale utilizzato da Bitcoin e altre blockchain. Coinvolge i "minatori" che competono per risolvere problemi crittografici complessi, impiegando una grande quantità di potenza di calcolo per confermare le transazioni e creare nuovi blocchi. Questo processo richiede una quantità significativa di energia elettrica, sollevando preoccupazioni riguardo all'impatto ambientale. D'altra parte, la Proof of Stake è una variante più recente che mira a risolvere i problemi di consumo energetico associati alla PoW. In PoS, la conferma delle transazioni e la creazione di nuovi blocchi sono basate sulla quantità di criptovaluta posseduta e "bloccata" dai partecipanti anziché sulla potenza di calcolo. Questo riduce notevolmente il consumo energetico complessivo del processo di consenso. Entrambi i protocolli hanno vantaggi e svantaggi unici. PoW è considerato più robusto dal punto di vista della sicurezza, poiché richiede un lavoro significativo per confermare le transazioni. Tuttavia, PoS è più efficiente dal punto di vista energetico e può essere considerato più equo in quanto la potenza di calcolo non è determinante. Non manco di citare altri protocolli di consenso che, per ragioni di importanza non andremo ad approfondire:

- Delegated Proof of Stake (**DPoS**)
- Byzantine Fault Tolerance (**BFT**)
- Proof of History (**PoH**)
- Proof of Authority (**PoA**)

### 2.1.4 Bitcoin

Bitcoin, rilasciato nel 2009, rappresenta la prima applicazione pratica della tecnologia blockchain. Creato da un individuo o un gruppo di individui sotto lo pseudonimo

di Satoshi Nakamoto, Bitcoin ha introdotto un nuovo modo di effettuare transazioni finanziarie in maniera decentralizzata, anonima e trasparente. La sua rete permette lo scambio di criptovaluta senza la necessità di intermediari come banche o governi. Questa tecnologia garantisce che tutte le transazioni siano verificabili e immutabili, prevenendo frodi e doppie spese.

### 2.1.5 Ethereum

Ethereum[7], lanciato nel 2015 da Vitalik Buterin, è spesso descritto come "the world computer"[1] estende il concetto di blockchain oltre le transazioni finanziarie, offrendo una piattaforma programmabile per creare e distribuire applicazioni decentralizzate (dApp). A differenza di Bitcoin, che si concentra principalmente sullo scambio di valore, Ethereum offre una piattaforma programmabile dove gli sviluppatori possono scrivere codici per gestire diverse operazioni direttamente sulla blockchain. Questo è reso possibile dalla Ethereum Virtual Machine (EVM), un ambiente di esecuzione che interpreta e esegue i contratti intelligenti (Smart Contracts). La criptovaluta nativa di Ethereum, Ether (ETH), è utilizzata per pagare le commissioni di transazione e per i servizi di calcolo sulla rete. Il 15 settembre 2022, Ethereum ha completato la transizione dal protocollo di consenso Proof of Work (PoW) al Proof of Stake (PoS), una fase cruciale nel suo aggiornamento denominato "Merge". Questa transizione ha segnato un cambiamento significativo nel modo in cui la rete Ethereum verifica e aggiunge nuovi blocchi alla blockchain.

### 2.1.6 Address in Ethereum

Nell'ecosistema Ethereum, gli address sono cruciali per l'interazione con la blockchain. Un address Ethereum è una sequenza alfanumerica di 42 caratteri che inizia con "0x", ad esempio "0x32Be343B94f860124dC4fEe278FDCBD38C102D88", e rappresenta un'identità univoca sulla rete. Gli address sono derivati dalle chiavi pubbliche tramite un algoritmo crittografico e vengono utilizzati per inviare e ricevere ether (ETH) e token.

#### Tipi di Address

Gli address possono essere di due tipi principali:

- **Account esterni (EOA):** Controllati da chiavi private, utilizzati per inviare transazioni e detenere ether.
- **Smart Contract:** Address che rappresentano contratti intelligenti distribuiti sulla blockchain.

#### Utilizzo degli Address

Gli address Ethereum vengono utilizzati per una varietà di funzioni essenziali sulla blockchain. Prima di tutto, servono per le transazioni: un utente può inviare ether o

token da un address a un altro, ad esempio, trasferendo fondi a un amico o pagando per un servizio. Inoltre, gli address sono fondamentali per l'interazione con gli smart contract. Quando un utente vuole invocare una funzione di un contratto, utilizza l'address del contratto per eseguire l'operazione.

Un altro uso importante degli address è per l'identificazione. Ogni address è unico, il che garantisce che gli account e i contratti siano distinti l'uno dall'altro. Questo è essenziale per la sicurezza e l'integrità delle transazioni sulla blockchain.

### Sicurezza degli Address

Una pratica cruciale per garantire la sicurezza di un address è la protezione delle chiavi private, che devono essere custodite con estrema attenzione. Se una chiave privata viene compromessa, il controllore dell'address perde l'accesso e il controllo sui fondi associati.

Verificare attentamente l'address del destinatario durante le transazioni è un altro aspetto critico. Gli attacchi di phishing possono mirare a ingannare gli utenti affinché inviino fondi a address sbagliati.

## 2.2 Sicurezza delle Blockchain

Le blockchain, sia basate su Proof of Work (PoW) che su Proof of Stake (PoS), offrono una sicurezza intrinseca grazie alla loro struttura decentralizzata e alla crittografia avanzata. Tuttavia, esistono diverse vulnerabilità e rischi che possono compromettere l'integrità e la sicurezza delle reti blockchain.

### 2.2.1 Attacchi del 51%

Gli attacchi del 51% rappresentano una delle maggiori minacce per le blockchain PoW e PoS. In entrambi i casi, un attaccante cerca di ottenere il controllo della maggioranza della potenza computazionale o della maggioranza delle risorse monetarie sulla rete, il che gli consente di manipolare le transazioni e compromettere la sicurezza complessiva della blockchain.

- **Variante Proof of Work:** Nel contesto della PoW, un attacco del 51% si verifica quando un singolo attaccante o un gruppo di attaccanti controlla più della metà della potenza di calcolo della rete.
- **Variante Proof of Stake:** Nei sistemi PoS, un attacco del 51% si verifica quando un attaccante possiede la maggioranza delle risorse monetarie o del "peso" sulla rete.

Nell'eventualità che si verifichi un attacco del 51%, il valore della criptovaluta subirebbe un crollo significativo poiché la fiducia degli utenti e degli investitori verrebbe gravemente compromessa. Un singolo individuo o un gruppo che detiene la

maggioranza del potere di calcolo o delle risorse di staking potrebbe manipolare le transazioni, invertire pagamenti e creare doppie spese. Questo scenario non solo metterebbe in discussione l'integrità della blockchain stessa, ma porterebbe anche a una perdita di fiducia nel mercato. Gli utenti, vedendo la vulnerabilità della rete, potrebbero ritirare i loro investimenti, vendere i loro asset digitali e cercare alternative più sicure, accelerando ulteriormente il deprezzamento della valuta. Da notare che la probabilità di successo di questo tipo di attacchi diminuisce significativamente nelle blockchain mature, stabili e ampiamente distribuite, dove il costo e la complessità di ottenere il controllo della rete diventano ridicolmente enormi.

### 2.2.2 Attacchi double spending

Un attacco di double spending può verificarsi quando un attaccante riesce a creare una versione alternativa della blockchain, iniziando a minare un ramo parallelo dal blocco precedente alla transazione da invalidare. Se l'attaccante riesce a produrre blocchi più rapidamente della rete originale, la sua catena diventerà la più lunga e sarà accettata come quella valida dalla maggioranza dei nodi, annullando la transazione originale.

Per esempio, l'attaccante potrebbe inviare una transazione a un venditore per acquistare un bene o servizio, ma allo stesso tempo iniziare a minare una catena alternativa che non include questa transazione. Se l'attaccante accumula abbastanza potenza computazionale per superare la catena legittima, può creare un blocco più lungo che esclude la transazione iniziale, permettendogli di riutilizzare gli stessi fondi in un'altra transazione. In conclusione, sebbene sia importante essere consapevoli dei potenziali rischi di attacchi, le attuali misure di sicurezza e i protocolli avanzati adottati dalle blockchain riducono significativamente la probabilità di successo di tali minacce.

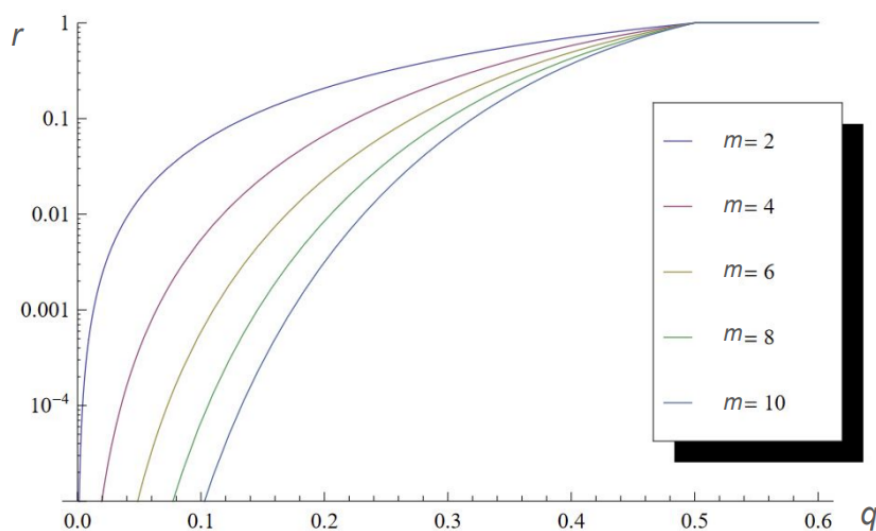


Figura 2.3: Probabilità di successo di un attacco di double spending, con  $q$  come potenza di calcolo dell'attaccante e  $m$  come numero di conferme richieste.

## 2.3 Smart Contract

Fondamentalmente, gli smart contract sono porzioni di codice. Quando vengono soddisfatte le condizioni per adempiere a un determinato contratto, gli smart contract sfruttano l'infrastruttura blockchain per condurre vari tipi di operazioni. Su Ethereum, gli smart contract sono scritti in Solidity[13], un linguaggio di programmazione specificamente progettato per questo scopo. Invece di richiedere a uno sviluppatore di supervisionare e gestire costantemente una piattaforma software, i contratti intelligenti agiscono come accordi automatizzati e eseguibili autonomamente che vengono attivati una volta che il codice rileva la presenza delle sue condizioni operative.

Ad esempio, un contratto intelligente può essere facilmente utilizzato per automatizzare i pagamenti degli stipendi in un'azienda, inviando ai dipendenti il loro stipendio attraverso la blockchain, in criptovalute, in una data specifica ogni mese. Gli elementi chiave del funzionamento degli smart contract includono:

- **Autorizzazione:** gli smart contract possono includere meccanismi di controllo degli accessi che definiscono chi può eseguire specifiche funzioni del contratto, garantendo che solo gli utenti autorizzati possano interagire con esso.
- **Esecuzione:** una volta attivato, lo smart contract esegue le azioni pre-programmate, che possono includere il trasferimento di fondi, l'aggiornamento dei registri, o l'attivazione di altri contratti.
- **Immutabilità:** il codice del contratto, una volta sulla blockchain, non può essere alterato, garantendo che le regole delineate nel contratto rimangano intatte e a prova di manomissione.
- **Trasparenza:** ogni interazione con il contratto viene registrata sulla blockchain, creando una traccia di controllo immutabile e trasparente.
- **Trustless:** elimina la necessità di fidarsi di una terza parte per garantire l'esecuzione di un contratto, che è garantita dalla blockchain stessa.

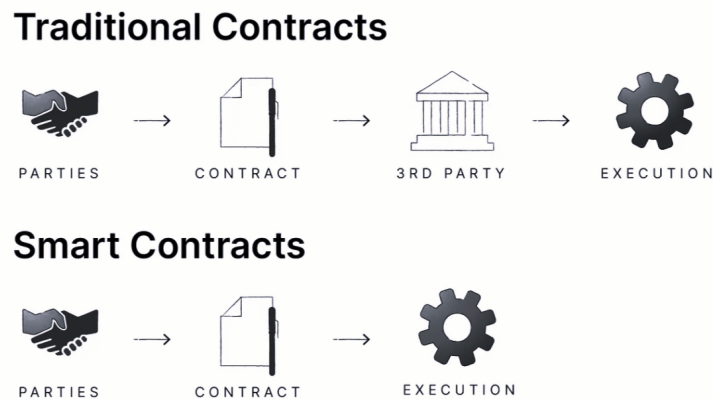


Figura 2.4: L'immagine illustra la differenza tra un contratto tradizionale, che richiede un intermediario fidato, e uno smart contract, che opera autonomamente senza necessità di terze parti.

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.25;
3 contract SimpleStorage {
4     uint storedData;
5     function set(uint x) public {
6         storedData = x;
7     }
8     function get() public view returns (uint) {
9         return storedData;
10    }
11 }
```

Listing 2.1: Questo è un semplice smart contract per la memorizzazione di un valore.

### 2.3.1 Gas

Il gas è un concetto fondamentale all'interno delle blockchain, particolarmente importante nel contesto di piattaforme come Ethereum. Si tratta di un'unità di misura che rappresenta il costo computazionale necessario per eseguire operazioni all'interno della rete blockchain. Tipicamente, per denotare il costo di una singola unità di gas, si tende ad usare il "wei", che è la più piccola unità di misura di Ether (ETH). Ecco una panoramica delle unità di misura di Ether per fare maggiore chiarezza:

- **Wei:**  $10^{-18}$  ETH
- **Gwei:**  $10^9$  wei
- **Ether:**  $10^{18}$  wei



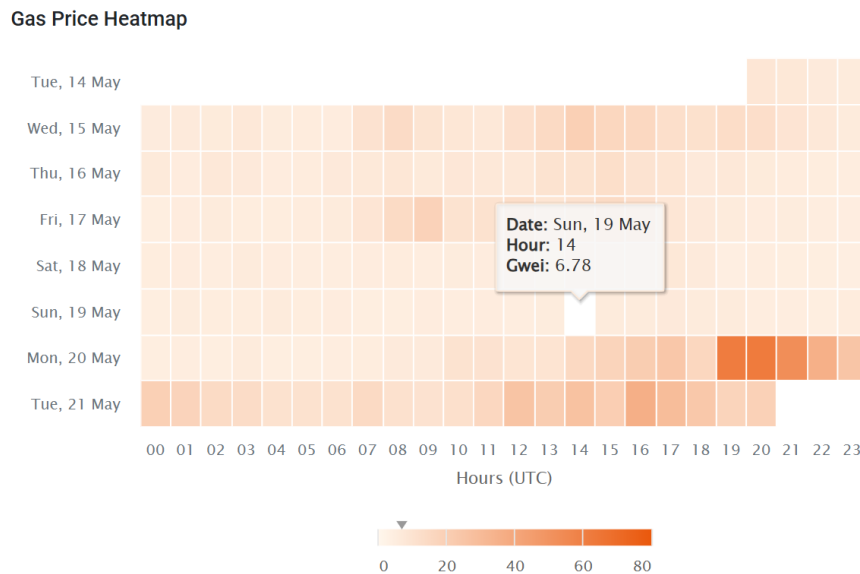


Figura 2.5: Heatmap dello storico del prezzo del gas di Ethereum nell'ultima settimana. Si può osservare come il prezzo di una singola unità di gas abbia superato i 60 gwei il 20 maggio alle ore 19 (UTC). Fonte: <https://etherscan.io/gastracker>

Ogni operazione o transazione richiede una certa quantità di gas per essere eseguita e il suo costo è determinato dalla complessità dell'operazione stessa. Il gas svolge diverse funzioni cruciali all'interno di una blockchain, serve a incentivare i validatori a includere le transazioni nella blockchain, poiché essi vengono ricompensati con le commissioni in gas per il lavoro svolto. Le transazioni sulla blockchain di solito includono una determinata quantità di gas insieme alla commissione in ether, la criptovaluta nativa di Ethereum, che viene pagata ai validatori. Se la quantità di gas fornita non è sufficiente per completare l'operazione, la transazione viene interrotta e gli eventuali gas spesi vengono restituiti all'utente. D'altra parte, se il gas fornito è eccessivo rispetto alle effettive necessità dell'operazione, l'eccesso di gas non utilizzato viene restituito all'utente. Il prezzo del gas è determinato dal mercato e può variare in base alla domanda e all'offerta nella rete. Questo meccanismo consente di mantenere il funzionamento efficiente della rete, incentivando gli utenti a pagare tariffe di transazione competitive per garantire l'inclusione rapida delle loro transazioni nei blocchi. In sintesi, il gas è un meccanismo cruciale di Ethereum ed è buona norma scrivere contratti dal leggero peso computazionale e con il minor numero possibile di letture/scritture sulla blockchain.

## 2.4 Token

I token sono unità digitali che rappresentano un asset o una utilità e possono essere trasferiti tra individui su una blockchain. Esistono diversi tipi di token, ognuno con caratteristiche e utilizzi specifici. In generale, i token possono essere classificati in due categorie principali: fungibili e non fungibili.

### 2.4.1 Token fungibili

I token fungibili, una delle categorie principali di token digitali, sono caratterizzati dalla loro intercambiabilità e identità di valore e funzione. Simili alle valute tradizionali, ogni unità di token fungibile è indistinguibile dalle altre dello stesso tipo e può essere scambiata o sostituita senza perdita di valore.

Un aspetto distintivo dei token fungibili è la loro uniformità di valore. Ad esempio, nel caso delle criptovalute come Bitcoin ed Ether, ogni unità di queste valute digitali ha lo stesso valore di qualsiasi altra unità della stessa criptovaluta. Questo rende i token fungibili particolarmente adatti per essere utilizzati come mezzo di scambio, riserva di valore o unità di conto, poiché garantiscono che il valore sia costante e uniforme tra tutte le unità.

### 2.4.2 Non-fungible token (NFT)

Gli NFT, acronimo di "Non-Fungible Tokens" (Token Non Fungibili), rappresentano una forma di token digitale che viene utilizzata per rappresentare la proprietà di oggetti digitali unici e non sostituibili. A differenza dei token fungibili come le criptovalute tradizionali, come ad esempio il Bitcoin o l'Ethereum, dove ciascuna unità è intercambiabile con un'altra unità dello stesso valore, gli NFT sono progettati per rappresentare oggetti digitali unici e unici, ognuno con proprietà distintive e identità indipendenti.

Gli NFT sono implementati su blockchain, come ad esempio Ethereum, utilizzando standard come ERC-721 o ERC-1155, che definiscono le regole e le funzionalità per la creazione, il trasferimento e la gestione di token non fungibili. Ogni NFT è rappresentato da un identificatore unico e immutabile sulla blockchain, che ne determina la proprietà e l'autenticità. Questo identificatore può essere collegato a metadati che descrivono l'asset digitale, come un link a un'opera d'arte o un file audio, aggiungendo un ulteriore livello di informazioni e valore.

Gli NFT possono rappresentare una vasta gamma di asset digitali unici, tra cui opere d'arte digitali, video, musica, tokenizzazioni di proprietà fisiche, certificati di autenticità, oggetti da collezione virtuali, terreni virtuali in metaversi, e molto altro. La proprietà di un NFT consente all'utente di trasferirlo, venderlo o conservarlo come un asset digitale unico, mentre il registro immutabile della blockchain garantisce la provenienza e l'autenticità dell'asset.







FUNGIBLE	NON-FUNGIBLE
Dollar 	Digital Art 
Bitcoin 	Physical Art 
Gold 	Property 

Figura 2.6: Esempio di fungibilità e non fungibilità.

### 2.4.3 Standard dei Token

Esistono vari standard per la creazione e gestione dei token su blockchain. Tra i più comuni ci sono:

- **ERC-20:** Uno standard per token fungibili su Ethereum. Definisce una serie di funzioni comuni che ogni token deve implementare, come il trasferimento di token e il saldo.
- **ERC-721:** Uno standard per token non fungibili su Ethereum. Ogni token creato utilizzando questo standard è unico e può essere associato a un asset specifico.
- **ERC-1155:** Uno standard versatile che consente la creazione di token sia fungibili che non fungibili all'interno dello stesso contratto intelligente.

## 2.5 InterPlanetary File System (IPFS)

IPFS[9], acronimo di InterPlanetary File System, è un protocollo e una rete peer-to-peer progettati per creare un metodo distribuito e decentralizzato per archiviare e condividere file. A differenza dei tradizionali sistemi centralizzati di archiviazione dei file, dove i dati sono conservati su server specifici gestiti da entità centralizzate, IPFS sfrutta una rete di nodi distribuiti che collaborano per archiviare e distribuire i file in modo più efficiente e sicuro.

Uno dei principi chiave di IPFS è l'indirizzamento dei contenuti anziché dei percorsi. Invece di localizzare un file in base al suo percorso (ad esempio, un URL che punta a un server specifico), IPFS utilizza un hash crittografico del contenuto del file stesso come indirizzo. Questo hash, noto come Content Identifier (CID), è un identificatore univoco che rappresenta il contenuto del file. Se il contenuto del file cambia, cambia anche il suo hash, il che garantisce l'integrità e l'immutabilità dei dati.

IPFS è una rete peer-to-peer, il che significa che i file non sono ospitati in un singolo server centrale, ma sono distribuiti tra i nodi partecipanti alla rete. Ogni nodo nella rete IPFS può memorizzare una parte o l'intero file, facilitando il recupero dei dati da più fonti. Questo aumenta la resilienza del sistema contro guasti e censura. Tra i principali vantaggi di IPFS ci sono la decentralizzazione, l'efficienza e la persistenza dei dati. IPFS elimina la necessità di server centralizzati, riducendo i punti di guasto singoli e aumentando la resistenza alla censura. Poiché i file sono distribuiti tra molti nodi, è più difficile per gli attori malintenzionati censurare o manipolare i dati. La distribuzione dei contenuti tramite una rete peer-to-peer può essere più efficiente rispetto ai tradizionali server centralizzati, permettendo ai file di essere recuperati da più nodi simultaneamente, riducendo i tempi di download e la latenza. Inoltre, grazie alla natura distribuita di IPFS, i file possono essere conservati in modo permanente, senza dipendere da una singola entità per l'archiviazione. Gli utenti possono contribuire alla rete conservando copie dei file, garantendo la disponibilità a lungo termine. IPFS trova applicazione in vari contesti, tra cui l'archiviazione distribuita, la condivisione di file e il web decentralizzato. Supporta la creazione di siti web completamente decentralizzati, dove le pagine web sono indirizzate tramite il loro CID e servite dalla rete peer-to-peer. È utile anche per creare backup distribuiti e ridondanti dei dati, migliorando la sicurezza e la disponibilità.

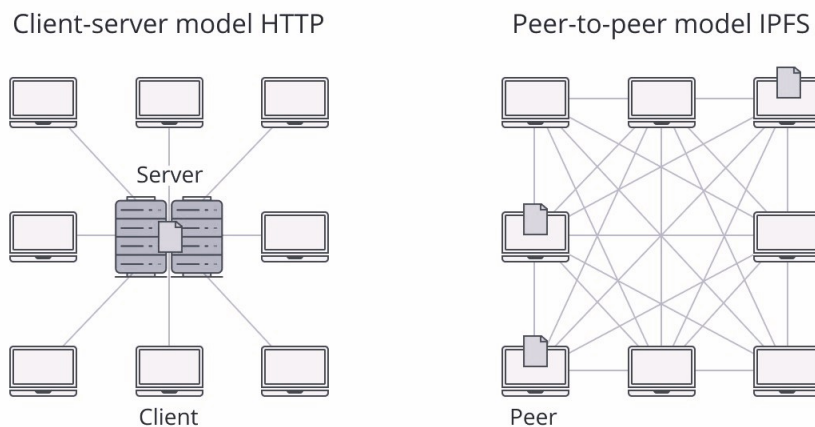


Figura 2.7: A sinistra è mostrata un'architettura client-server con il file archiviato su un server centrale, mentre a destra è rappresentata l'architettura peer-to-peer di IPFS, dove il file è salvato in modo decentralizzato su più nodi.

## 2.6 Rollup

I Rollup rappresentano una significativa innovazione nel contesto della scalabilità delle blockchain, offrendo soluzioni di secondo livello che risolvono efficacemente i problemi di congestione e i limiti di capacità delle reti blockchain. In particolare, per piattaforme come Ethereum, dove la richiesta di transazioni è in costante aumento, l'introduzione dei Rollup ha portato un notevole progresso nella gestione del carico di lavoro e dei costi associati alle transazioni. Queste soluzioni consentono di mantenere l'integrità e la sicurezza della blockchain principale mentre migliorano l'efficienza del sistema. Il loro funzionamento si basa sull'aggregazione di numerose transazioni in un'unica transazione ottimizzata, che viene quindi registrata sulla blockchain principale. Questo processo consente di ridurre drasticamente il numero di operazioni che devono essere eseguite direttamente sulla blockchain principale, alleggerendo così il carico di lavoro e riducendo i costi di elaborazione.

Un altro vantaggio significativo dei Rollup è la loro capacità di migliorare la velocità di transazione. Poiché molte transazioni possono essere elaborate off-chain e poi consolidate in una singola transazione on-chain, i Rollup possono aiutare a mitigare i problemi di latenza che affliggono le blockchain ad alta domanda. Inoltre, i Rollup possono facilitare una maggiore inclusione finanziaria, poiché le riduzioni nei costi di transazione possono rendere l'uso della blockchain più accessibile a una più ampia gamma di utenti, inclusi quelli con risorse finanziarie limitate. È importante notare che esistono diversi approcci ai Rollup, tra cui Optimistic Rollup e ZK Rollup, ciascuno con le proprie caratteristiche e vantaggi distintivi.

### 2.6.1 Optimistic Rollups

Gli Optimistic Rollup, come il nome suggerisce, operano con l'assunto ottimistico che le transazioni sono valide per default. Esse aggregano le transazioni fuori dalla catena principale e le trasmettono in blocchi compatti alla blockchain principale. Un elemento chiave degli Optimistic Rollup è il meccanismo di sfida. Se qualcuno sospetta che una transazione sia fraudolenta, può avviare una sfida che, se verificata, invalida la transazione e punisce il malintenzionato. Questo approccio riduce il carico computazionale poiché non tutte le transazioni devono essere verificate immediatamente.

Inoltre, gli Optimistic Rollup sono particolarmente vantaggiosi in termini di scalabilità, poiché consentono una notevole riduzione dei dati che devono essere memorizzati sulla blockchain principale. Questo approccio può anche favorire una maggiore adozione da parte degli utenti e degli sviluppatori, offrendo un equilibrio tra efficienza e sicurezza.

### 2.6.2 zkRollups

I ZK Rollup (Zero-Knowledge Rollup) utilizzano prove crittografiche chiamate zero-knowledge proofs per verificare la validità delle transazioni. Ogni lotto di transazioni aggregate è accompagnato da una prova crittografica che garantisce la validità di tutte le transazioni nel lotto. Questa prova è poi registrata sulla blockchain principale, che può verificare la prova senza dover elaborare ogni singola transazione. I ZK Rollup sono estremamente efficienti in termini di spazio e offrono sicurezza rigorosa, ma la generazione delle prove può essere computazionalmente intensiva.

Nonostante la complessità tecnica della generazione delle zero-knowledge proofs, i ZK Rollup rappresentano una soluzione promettente per le applicazioni che richiedono un alto livello di sicurezza e integrità dei dati. Inoltre, grazie alla loro capacità di compressione dei dati, i ZK Rollup possono ridurre significativamente i costi di archiviazione e migliorare la velocità delle transazioni, rendendoli ideali per una vasta gamma di applicazioni blockchain.

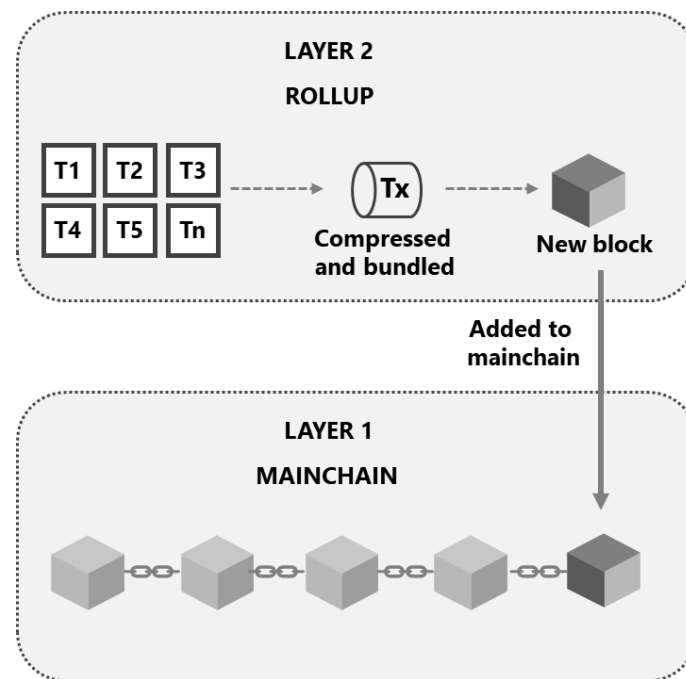


Figura 2.8: Le transazioni effettuate nella chain secondaria vengono raggruppate in blocchi ottimizzati per ridurre i costi per singola transazione; successivamente, un blocco aggregato viene aggiunto alla chain principale

## 2.7 API (Application Programming Interface)

Nel contesto dello sviluppo software, le API, fungono da ponte tra diverse applicazioni, consentendo loro di scambiarsi informazioni e di interagire in modo coordinato. Possono essere considerate un'interfaccia tra due componenti software, dove un'applicazione può fare richieste specifiche tramite le API e ricevere risposte in un

formato predefinito. Definiscono i metodi e i formati dei dati per la comunicazione tra le applicazioni, facilitando l'integrazione e l'interoperabilità dei sistemi software.

Le API sono essenziali per l'integrazione delle applicazioni, consentendo lo sviluppo di sistemi complessi e distribuiti. Grazie alle API, le applicazioni possono accedere alle funzionalità offerte da altri software, servizi o piattaforme senza dover comprendere o modificare il codice sottostante. Questo promuove la riusabilità del codice e accelera lo sviluppo delle applicazioni.

Un'applicazione di social media, ad esempio, potrebbe utilizzare le API di un servizio di geolocalizzazione per mostrare la posizione degli utenti o le API di un servizio di pagamento per elaborare transazioni finanziarie.

Le API consentono alle applicazioni di essere progettate in modo modulare, facilitando la manutenzione e l'aggiornamento del software nel tempo.

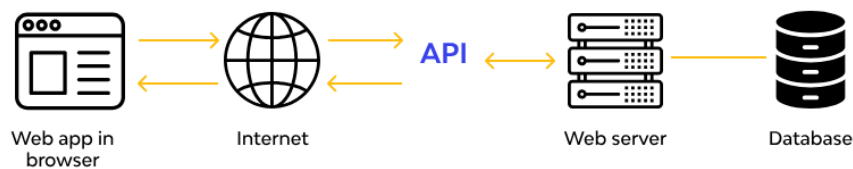


Figura 2.9: Illustrazione del funzionamento delle API. L'API funge da punto di contatto tra il client e il server, consentendo la comunicazione e l'interazione tra le due parti.





## Capitolo 3

# Il problema affrontato

Questo progetto nasce dalla necessità di fornire ai clienti un modo trasparente per certificare qualsiasi documento digitale. La certificazione di un documento consiste nella prova verificabile che il documento possedeva un determinato contenuto in un certo istante temporale. Tuttavia, l'obiettivo è andare oltre la semplice certificazione.

### 3.1 Digital Product Passport (DPP)

L'Unione Europea introdurrà presto il passaporto digitale dei prodotti in vari settori industriali. Questo documento digitale contiene informazioni dettagliate sul ciclo di vita di un prodotto, inclusi i materiali di produzione, l'impatto ambientale, la proprietà, le modalità di smaltimento responsabile e dati su garanzia e manutenzione.

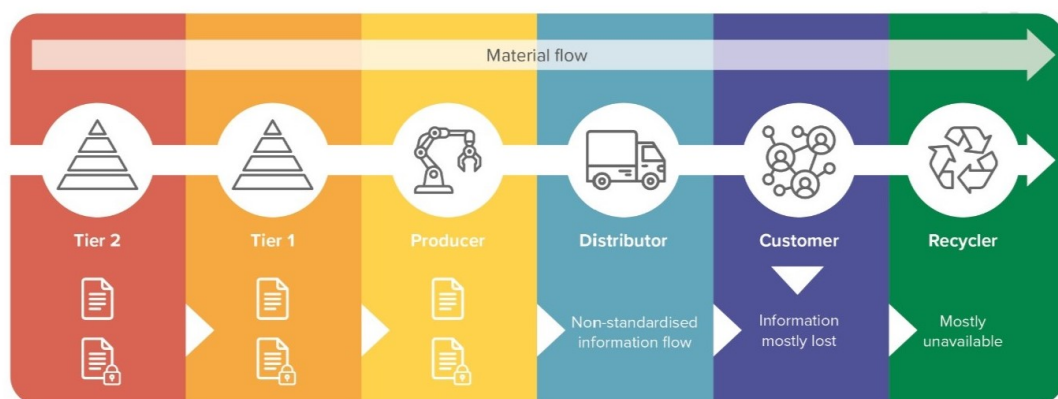


Figura 3.1: Diagramma del ciclo di vita di un prodotto lungo la catena di approvvigionamento. Il suo Digital Product Passport conterrà le informazioni di ogni fase della sua produzione.

In conseguenza di ciò, abbiamo deciso di sviluppare un certificatore digitale che sia flessibile al punto da fungere, oltre che da semplice certificatore documentale, anche da digital product passport, così che il campo di applicazione sia molto più estensivo. Considerando che il ciclo di vita di un prodotto è composto da diversi step produttivi e che le informazioni di ogni fase devono essere verificate e immutabili per i produttori successivi, è possibile creare una catena di certificazioni per ciascun prodotto, garantendo che le informazioni relative a ciascun passaggio produttivo, certificate in un dato momento, rimangano inalterate nel tempo. In pratica, questo approccio ci assicura che le informazioni in ogni fase siano autentiche e non soggette a modifiche, fornendo una cronologia affidabile e verificabile per ogni prodotto.

## 3.2 DPP su Blockchain

L'implementazione di un Digital Product Passport su una blockchain offre numerosi vantaggi rispetto ai metodi tradizionali. Utilizzando la tecnologia blockchain, si possono ottenere certificazioni digitali che sono trustless, verificabili autonomamente e autentiche. Ecco alcuni benefici di questa soluzione:

- **Trasparenza e Tracciabilità:** Ogni transazione registrata sulla blockchain è visibile a tutti i partecipanti della rete. Questo significa che le informazioni contenute nel DPP, come i materiali utilizzati, l'impatto ambientale e le modalità di smaltimento, sono accessibili e tracciabili da chiunque. La tracciabilità delle informazioni lungo l'intera catena di approvvigionamento consente di monitorare ogni fase del ciclo di vita del prodotto, migliorando la fiducia e la responsabilità.
- **Integrità dei Dati:** Una volta registrati sulla blockchain, i dati non possono essere alterati o cancellati. Questa immutabilità garantisce che le informazioni nel DPP siano accurate e affidabili. Ogni step produttivo certificato è protetto contro qualsiasi tentativo di manomissione, assicurando che i dati rimangano intatti e autentici. Questo livello di sicurezza è fondamentale per creare una catena di certificazioni inalterabili per ogni prodotto.
- **Verificabilità Trustless:** Il concetto di "trustless" si riferisce alla capacità di un sistema di funzionare senza la necessità di fidarsi di un'autorità centrale o di terze parti. In un DPP basato su blockchain, la fiducia è decentralizzata e garantita dalla rete stessa. Gli utenti possono verificare in totale autonomia le informazioni contenute nel DPP attraverso la blockchain, senza dover essere costretti a fidarsi di chi rilascia il certificato.
- **Certificazioni tramite NFT:** Ogni certificazione è rappresentata da un NFT (Non-Fungible Token). Questi NFT sono generati tramite uno smart contract basato sullo standard ERC-721. Gli NFT compatibili con ERC-721 garantiscono la proprietà e l'unicità delle certificazioni, rendendole facilmente trasferibili

e verificabili. L'uso degli NFT permette anche di arricchire i DPP con metadati aggiuntivi che, come discusso nel capitolo successivo, sono di fondamentale importanza.

### 3.3 NFT e Luxury Fashion

Nel settore del luxury fashion, le certificazioni digitali tramite NFT (Non-Fungible Token) rivestono un ruolo fondamentale nell'assicurare l'autenticità e il valore dei prodotti di alta moda. Uno dei vantaggi chiave degli NFT è la loro capacità di agire come attestato immutabile, consentendo ai brand di dimostrare in modo sicuro e incontestabile l'autenticità e l'originalità di ciascun articolo. Questo meccanismo riduce il rischio di contraffazione e frode. Inoltre, un aspetto fondamentale considerato è l'ownership dei prodotti. Grazie alla natura degli NFT, è possibile trasferire l'ownership di un articolo digitalizzato. Ciò significa che un NFT può attestare non solo l'autenticità, ma anche la proprietà effettiva di un determinato bene. Questo offre un ulteriore livello di sicurezza e tracciabilità nella gestione dei beni di lusso, consentendo ai proprietari di confermare in modo inequivocabile la loro legittima proprietà.

Tuttavia, uno dei problemi fondamentali nell'implementazione di questo sistema è l'associazione precisa tra l'NFT e l'oggetto fisico corrispondente. Poiché la sicurezza e l'autenticità dipendono dall'accuratezza di questa associazione, è essenziale sviluppare procedure affidabili e sistemi tecnologici appropriati per garantire che ogni NFT sia correttamente associato al suo prodotto di lusso corrispondente. Questo implica l'utilizzo di tecnologie come RFID, NFC o QR code, insieme a protocolli sicuri di registrazione e verifica sulla blockchain. Una gestione accurata di questa associazione è cruciale per garantire l'integrità del sistema e la fiducia degli acquirenti nei prodotti di lusso certificati tramite NFT.



# Capitolo 4

## Implementazione

### 4.1 Strumenti e servizi utilizzati

Di seguito sono riepilogati gli strumenti e i framework utilizzati nello sviluppo del progetto.

#### 4.1.1 ExpressJS

ExpressJS[6] è un framework minimalista per Node.js, utilizzato per costruire applicazioni web e API. Si distingue per la sua flessibilità e la sua capacità di adattarsi a una vasta gamma di casi d'uso, rendendolo una scelta popolare per lo sviluppo di applicazioni web e API. La caratteristica che rende ExpressJS un'ottima scelta per lo sviluppo del back-end risiede nella semplicità con cui è possibile scrivere le API e la sua minimalità.

#### 4.1.2 Hardhat

Hardhat[8] è uno strumento di sviluppo per smart contract basati su Ethereum. È utilizzato per compilare, distribuire, testare e fare il debug dei contratti. Hardhat è stato fondamentale nella gestione e nell'implementazione degli smart contract. Questo strumento ha migliorato notevolmente il processo di sviluppo, riducendo i tempi necessari per testare e distribuire i contratti sulla blockchain. È stato decisamente più conveniente testare tutto in locale piuttosto che caricarlo sulle diverse reti di test (testnet).

#### 4.1.3 EthersJS

Ethers.js[5] è una libreria JavaScript per interagire con la blockchain di Ethereum. È progettata per essere semplice da usare e fornisce strumenti essenziali per sviluppare applicazioni decentralizzate (dApp) e interagire con smart contract. Consente di connettersi a nodi Ethereum, inviare transazioni e leggere dati dagli smart contract.

Ethers.js facilita la gestione delle chiavi private e dei portafogli Ethereum, permettendo agli sviluppatori di creare nuovi portafogli, importare quelli esistenti e firmare transazioni direttamente dal client. Inoltre, semplifica l'interazione con gli smart contract tramite l'uso di ABI (Application Binary Interface), consentendo di invocare funzioni, leggere stati e ascoltare eventi.

La libreria è compatibile con vari provider di nodi Ethereum, come Infura e Alchemy, e include utilità per la gestione di dati Ethereum, conversioni di unità, hashing e firme digitali. Ethers.js è leggera e modulare, permettendo di includere solo le parti necessarie, riducendo l'ingombro del codice. La sua API intuitiva e ben documentata rende facile per gli sviluppatori iniziare rapidamente.

#### 4.1.4 Infura

Infura è una piattaforma di infrastruttura come servizio (IaaS) che fornisce accesso a diversi servizi blockchain, principalmente la rete Ethereum. Tuttavia, verrà utilizzato il rollup Linea per ottimizzare le operazioni sulla blockchain. Offrendo nodi Ethereum come servizio, Infura consente agli sviluppatori di connettersi facilmente alla blockchain senza dover gestire manualmente un nodo. Infura fornisce API che facilitano l'interazione con la rete Ethereum, permettendo di inviare transazioni, leggere dati dalla blockchain e accedere ai contratti intelligenti. Uno dei principali vantaggi di Infura è la sua capacità di scalare automaticamente per gestire grandi volumi di richieste, rendendolo ideale per applicazioni che necessitano di alta affidabilità e disponibilità. Inoltre, Infura supporta anche altre blockchain e servizi correlati, inclusi IPFS (InterPlanetary File System) che sarà mostrato più avanti nel capitolo.

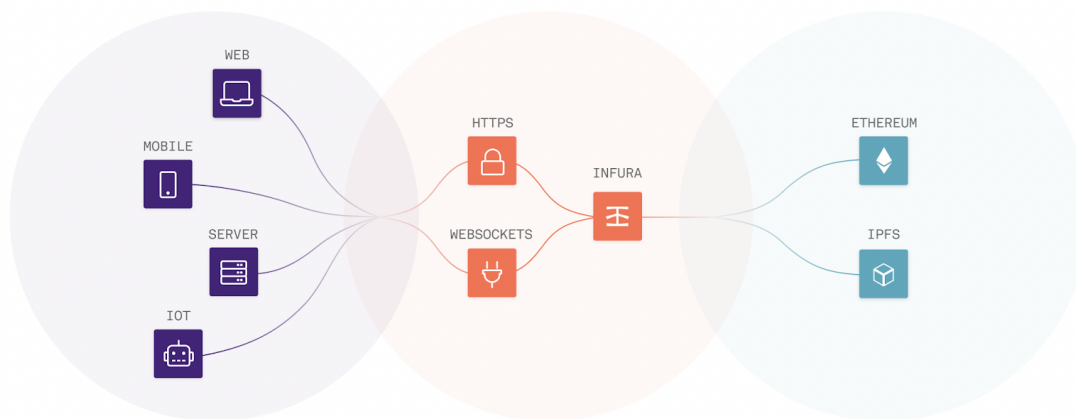


Figura 4.1: Schema di utilizzo di Infura per interfacciarsi con Ethereum e IPFS.

### 4.1.5 Linea

Linea[10] è una soluzione di scalabilità layer 2 che ha come scopo quello di far scalare la blockchain Ethereum.

Linea sfrutta i rollup per aggregare numerose transazioni fuori dalla catena principale e registrare il risultato netto sulla blockchain. Questo metodo mantiene la sicurezza e la decentralizzazione della blockchain principale, riducendo al contempo i costi delle transazioni e migliorando i tempi di conferma.

Linea risulta particolarmente vantaggiosa per applicazioni che hanno un volume molto grande di transazioni, il costo del gas basso ci permette di costruire applicazioni molto più complesse rispetto a quelle su Ethereum.

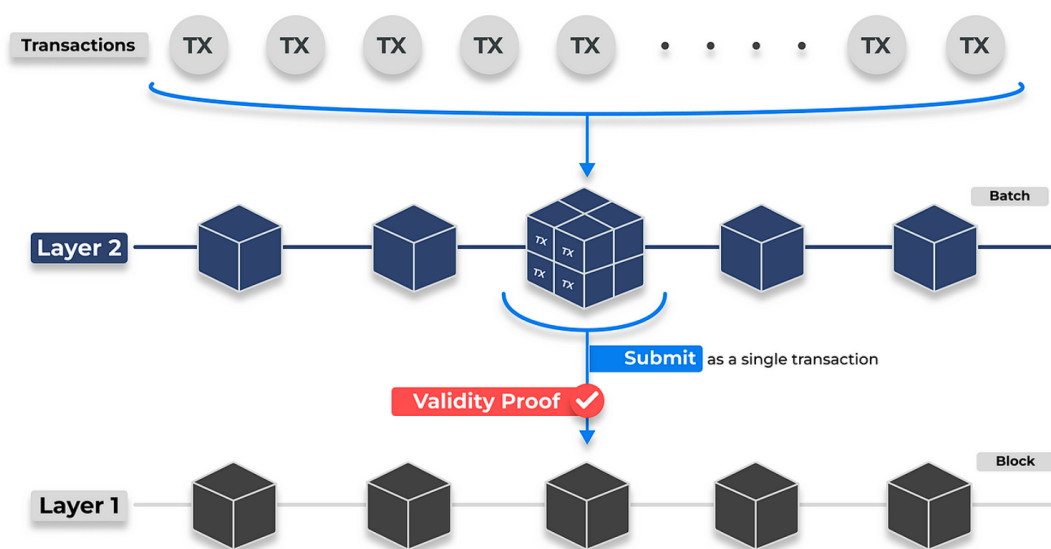


Figura 4.2: Meccanismo di scaling Layer 2: le transazioni sono raggruppate in batch, verificate con una prova di validità e registrate sulla blockchain Layer 1.

### 4.1.6 React

React[12] è una libreria JavaScript utilizzata per la creazione di interfacce utente dinamiche e reattive. Originariamente sviluppata da Facebook, React si basa sull'approccio component-based, il che significa che le interfacce utente vengono costruite utilizzando una serie di componenti riutilizzabili. Questo modo di scrivere codice ne promuove la modularità e l'organizzazione, facilitando l'implementazione di nuove funzionalità e il mantenimento del software nel tempo.

Uno dei principali vantaggi di React è la sua efficace gestione dello stato dell'applicazione. Questo approccio alla gestione dello stato rende più agevole la sincronizzazione tra i dati dell'applicazione e la loro rappresentazione visuale nell'interfaccia utente. Inoltre, la modularità e la riutilizzabilità dei componenti in React favoriscono lo sviluppo efficiente di web application complesse, consentendo agli sviluppatori di costruire e mantenere applicazioni scalabili e robuste nel tempo. Questa flessibilità

nell'organizzazione e nella struttura del codice promuove una maggiore produttività nello sviluppo di applicazioni web moderne.

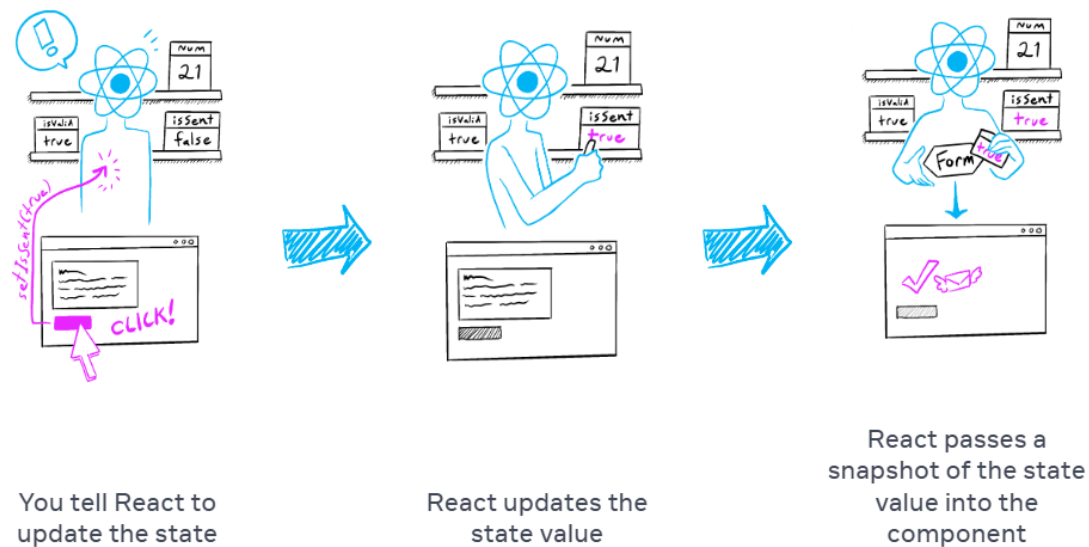


Figura 4.3: L'utente innesca un cambio di stato tramite un bottone, che viene gestito da React. Successivamente, React aggiorna la variabile di stato e aggiorna la vista del componente corrispondente.

#### 4.1.7 Material UI (MUI)

Material UI (MUI)[11] è una libreria di componenti React che implementa il design system di Google Material[2]. È stata utilizzata per stilizzare l'interfaccia utente del front-end, offrendo componenti predefiniti e personalizzabili che rispettano le linee guida del Material Design. MUI garantisce un'interfaccia utente coerente e uniforme con i principi del Material Design, assicurando un'esperienza utente intuitiva.

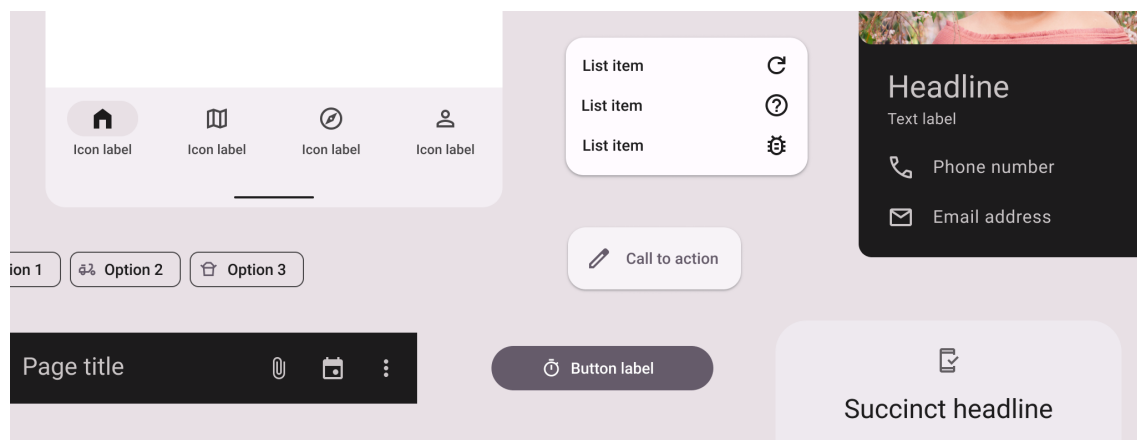


Figura 4.4: Esempio di design compliant alle linee guida del Material Design di Google.



### 4.1.8 ERC-721A

ERC-721A[4] è una variante dello standard ERC-721[3], introdotto per affrontare alcune limitazioni delle implementazioni delle interfacce IERC721 e IERC721Enumerable di Openzeppelin, in particolare per rendere il minting (creazione) di NFT più efficiente in termini di gas. Il vantaggio di ERC-721A è quello di poter fare minting di più NFT nella stessa transazione con costi decisamente più contenuti, rispetto a quello a ERC-721 di Openzeppelin. Quindi è possibile implementare operazioni di Batch Minting, ovvero di creazione raggruppata e ottimizzata di NFT. Nonostante le ottimizzazioni, ERC-721A mantiene la compatibilità con lo standard ERC-721, il che significa che i contratti che utilizzano ERC-721A possono interagire con le stesse applicazioni che supportano ERC-721 senza problemi. Le modifiche effettuate a ERC-721 e ERC-721Enumerable si sono concentrate sull'eliminazione di strutture dati ridondanti. La modifica più significativa è stata quella di aggiornare i dati del proprietario una sola volta per ogni richiesta di minting batch, invece di farlo per ogni NFT mintato.

#100 Owner: Alice	#101 Owner: <not set>	#102 Owner: <not set>	#103 Owner: Bob	#104 Owner: <not set>
-------------------------	-----------------------------	-----------------------------	-----------------------	-----------------------------

Figura 4.5: Supponiamo che Alice faccia il mint dei token 100, 101 e 102, e Bob faccia il mint dei token 103 e 104. Per determinare il proprietario del token 102, non è necessario che Alice sia esplicitamente impostata come proprietaria di ciascun token individualmente. Basta infatti controllare a ritroso finché non troviamo l'ultimo proprietario registrato, risalendo così alla proprietà di Alice.

Ecco come ciò si traduce in un utilizzo di gas drasticamente ridotto:

NFT generati	ERC-721Enumerable	ERC-721A	Gas risparmiato
1	154.814	76.690	78.124
2	270.339	78.819	191.520
3	384.864	80.948	303.916
4	501.389	83.077	418.312
5	616.914	85.206	531.708

Tabella 4.1: Analisi del costo del gas per il minting di NFT utilizzando ERC-721Enumerable di OpenZeppelin e ERC-721A.

## 4.2 Back-end

Il componente back-end dell'architettura del servizio è costituito da un server basato su ExpressJS. Nel file `index.js`, che viene eseguito dal server, sono definite diverse API, ovvero endpoint, che possono essere chiamati per interagire con il provider Infura, il quale si occupa poi di chiamare le funzioni dello smart contract. Per brevità, da ora in avanti diremo che il server contatta lo smart contract, anche se ciò avviene indirettamente tramite l'interazione con Infura. Le principali API messe a disposizione sono *Tokenize* e *Verify*.

### 4.2.1 API Tokenize

L'API *Tokenize* permette di certificare un documento caricandolo su IPFS e creando un NFT (Non-Fungible Token) con i metadati associati. I seguenti passaggi sono:

1. **Verifica e Caricamento del File:** L'API controlla che un file sia stato effettivamente caricato e che i parametri di minting siano presenti nella richiesta. Se queste condizioni sono soddisfatte, il file viene salvato temporaneamente sul server.
2. **Calcolo dell'Hash:** L'hash del file caricato viene calcolato utilizzando l'algoritmo SHA-256, che genera un identificatore univoco per il documento.
3. **Generazione dei Metadati:** Vengono creati i metadati del documento, che includono informazioni come il nome del documento, il suo hash e altre proprietà specifiche. Questi metadati sono salvati in un file JSON.
4. **Caricamento su IPFS:** Il file e i metadati sono caricati su IPFS per garantirne la conservazione decentralizzata. Gli hash risultanti dall'upload su IPFS vengono utilizzati per referenziare il contenuto.
5. **Chiamata alla funzione `safeMint()`:** Vengono codificati i dati di input e invocata la funzione `safeMint()` dello smart contract, a seconda dei parametri, viene eseguita un'operazione di certificazione o di aggiornamento. In ogni caso, viene creato un nuovo NFT con i metadati associati. Questo passaggio include l'interazione con il provider Infura per gestire le operazioni sulla blockchain.
6. **Risposta al Client:** L'API restituisce al client l'hash del file e l'hash della transazione sulla blockchain, confermando l'avvenuta creazione dell'NFT.
7. **Pulizia dei File Temporanei:** I file temporanei creati sul server durante il processo sono eliminati per mantenere pulito l'ambiente di esecuzione e ridurre la probabilità che vengano compromessi.

### 4.2.2 API Verify

L'API *Verify* consente di verificare se esiste una certificazione associata al documento specificato nel corpo della richiesta HTTP. Per ridurre l'utilizzo della banda, il file non viene caricato direttamente nel corpo della chiamata HTTP, invece, viene incluso l'hash del documento, già calcolato localmente nella macchina dell'utente. I passaggi seguenti descrivono il funzionamento dell'API:

1. **Chiamata a `verify()`:** L'hash del documento viene codificato e la funzione *verify()* viene chiamata con questo input.
2. **Decodifica Output:** Il risultato ricevuto dall'output della funzione *verify()* viene decodificato e convertito nel formato appropriato.
3. **Invio risposta:** La risposta decodificata viene inviata al client.

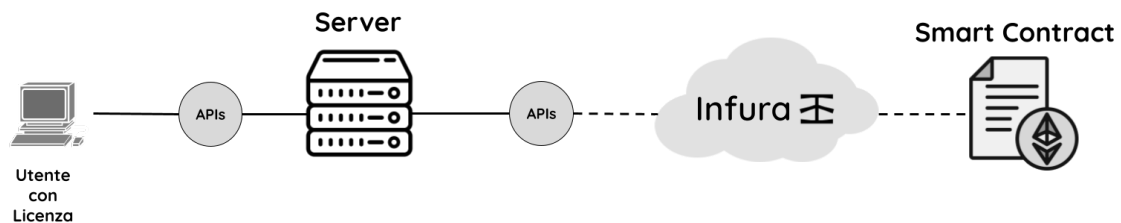


Figura 4.6: Architettura del progetto. L'utente interagisce con il server tramite le API messe a disposizione. Il server comunica con il provider Infura per interagire con lo smart contract.

## 4.3 Realizzazione dello Smart Contract

Lo smart contract implementa la versione ERC-721A. Le principali funzionalità esposte dal contratto sono *safeMint()* e *verify()*. *safeMint()* contiene le istruzioni per la certificazione e l'aggiornamento di una certificazione esistente. La funzione *verify()* permette di effettuare la verifica della certificazione relativa a un documento.

### 4.3.1 Strutture Dati

Le strutture dati impiegate nella memorizzazione dei dati sulla blockchain sono le seguenti:

- **tokenMap:** `mapping(uint64 tokenId => TokenObject)` che memorizza le associazioni tra l'identificativo di un NFT e la struttura *TokenObject*. La struttura *TokenObject* è definita come segue:

```

1 struct TokenObject {
2     bytes32 uri;
3     bytes32 cid;
4     uint256 timestamp_index;
5 }

```

Listing 4.1: Definizione della struttura TokenObject

Dato un *tokenId*, è possibile risalire alle seguenti informazioni:

- **URI**: l'indirizzo del file JSON contenente i metadati, salvato su IPFS.
  - **CID**: l'hash del documento.
  - **Timestamp**: l'istante di creazione del token.
  - **Index**: chiave per accedere al mapping *historyMap*, che conserva gli storici delle certificazioni.
- **historyMap**: mapping(uint128 index => uint256[ ]) Dato una certa chiave, è possibile risalire a un array di tokenId che rappresenta, in ordine cronologico, le certificazioni correlate, utilizzato per definire le versioni precedenti di un determinato documento.
  - **cidMap**: mapping(bytes32 cid => uint64[ ]) che associa un CID (hash di un documento) ai tokenId dello stesso documento.

### 4.3.2 SafeMint()

La funzione *safeMint()* prende in input un array di strutture *BufferItem*.

```

1 struct BufferItem {
2     address toAddress;
3     bool append;
4     bytes32[] tokenUri;
5     bytes32[] hash;
6     uint64[] obsoleteTknId;
7 }

```

Listing 4.2: Definizione della struttura BufferItem

Ogni elemento dell'array rappresenta un insieme di richieste per un determinato indirizzo Ethereum. La funzione itera su tutti gli elementi dell'array, eseguendo controlli preliminari per garantire l'integrità dei dati del contratto.

Per ogni richiesta, *safeMint()* verifica se si tratta di una nuova certificazione o di un aggiornamento di una certificazione esistente. La funzione *safeMint()* è protetta dal modifier **onlyOwner**, il che significa che solo il proprietario del contratto può invocarla. Questo controllo di accesso assicura che solo utenti autorizzati possano eseguire operazioni di minting, garantendo la sicurezza e l'integrità del processo.

- **Nuova certificazione:** le informazioni del documento vengono memorizzate per creare un NFT alla fine del processo. Poiché viene utilizzato lo standard ERC-721A, è possibile creare più token in modo efficiente. Una volta elaborate tutte le richieste per un singolo indirizzo, viene invocata la funzione `_safeMint()`, che prende come parametri l'indirizzo destinatario degli NFT e la quantità di token da generare. Questa funzione è molto più economica rispetto alla versione standard fornita da OpenZeppelin, che esegue il mint di un solo token per volta. Grazie a ERC-721A, si ottimizzano i costi di gas associati al minting multiplo.
- **Aggiornamento certificazione:** vengono seguiti ulteriori passaggi. La vecchia certificazione viene inserita in uno storico, e l'NFT associato alla vecchia certificazione viene bruciato (chiamata della funzione `burn()`). In questo modo, si mantiene un registro organizzato delle certificazioni aggiornate e si evita la duplicazione delle certificazioni stesse.

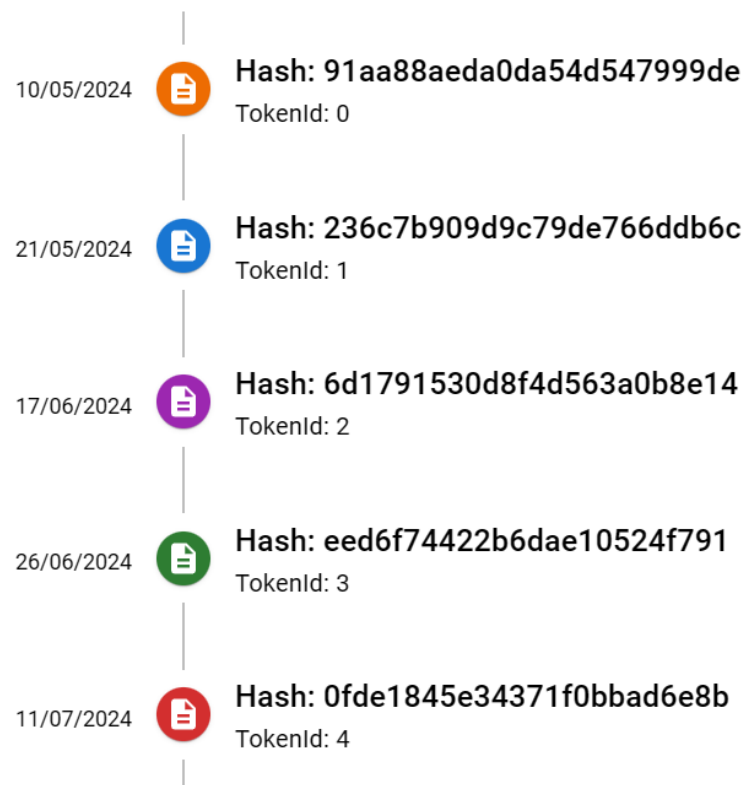


Figura 4.7: Esempio di una catena di certificazioni costituite da NFT. Nel contesto di un Digital Product Passport (DPP), questi NFT vengono concatenati per formare una cronologia, dove ogni NFT rappresenta una specifica certificazione o fase di produzione del prodotto. Ad esempio, per un paio di calze, il tokenId 0 potrebbe corrispondere a un documento sui materiali utilizzati nella prima fase di produzione, mentre un tokenId successivo potrebbe documentare il luogo di tessitura, la distribuzione, la vendita e infine il riciclo.

### 4.3.3 Verify()

La funzione *verify()* prende in input l'hash (CID) di un documento e restituisce una ricevuta composta rappresentata da un oggetto *CompoundReceipt*.

```
1 struct CompoundReceipt {  
2     uint256 timestamp;  
3     uint256 tokenId;  
4     string cid;  
5     string tokenUri;  
6     DecodedHistory[] history;  
7 }
```

Listing 4.3: Definizione della struttura *CompoundReceipt*

L'oggetto *CompoundReceipt* rappresenta una ricevuta della certificazione corrispondente al hash passato in input e contiene, oltre alle informazioni principali della certificazione, il campo *history*. Questo campo è un array di oggetti *DecodedHistory*, che rappresentano le varie versioni storiche della certificazione.

```
1 struct DecodedHistory {  
2     uint256 timestamp;  
3     uint256 tokenId;  
4     string cid;  
5     string tokenUri;  
6 }
```

Listing 4.4: Definizione della struttura *DecodedHistory*

Il campo *history* all'interno di *CompoundReceipt* permette di mantenere traccia della cronologia delle certificazioni. Ogni oggetto *DecodedHistory* all'interno di questo array contiene informazioni su una versione precedente della certificazione, inclusi il *timestamp* della certificazione storica, il *tokenId* del token NFT associato alla versione storica, il *cid* della versione storica e l'*tokenUri* che punta ai metadati del documento salvato su IPFS.

## 4.4 Front-end

Il front-end del progetto è responsabile dell'interfaccia utente e delle interazioni con il back-end tramite le API esposte. L'utente può caricare documenti, verificare certificazioni esistenti e visualizzare le informazioni relative agli NFT certificati.

Il front-end è stato sviluppato utilizzando tecnologie moderne come React.js per la costruzione dell'interfaccia utente. L'utilizzo di React è stato motivato dalla necessità di avere una interfaccia che muta al variare degli eventi e delle informazioni.

### 4.4.1 Caratteristiche Principali

- **Caricamento Documento:** L'utente può caricare un documento attraverso un'area di drag-and-drop. Esegue il calcolo dell'hash del documento in locale e lo invia al server back-end.
- **Verifica Certificazioni:** L'utente può verificare se un documento è già stato certificato inserendo l'hash del documento. Il sistema restituisce la ricevuta di certificazione e la cronologia delle versioni.
- **Visualizzazione NFT:** Le informazioni sugli NFT certificati, inclusi i metadati e la cronologia delle versioni, vengono visualizzate in modo chiaro e strutturato.

### 4.4.2 Aspetto del Front-end

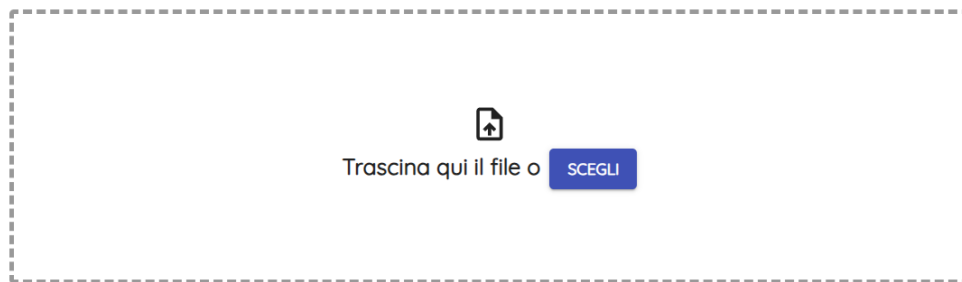


Figura 4.8: La figura rappresenta la pagina principale.

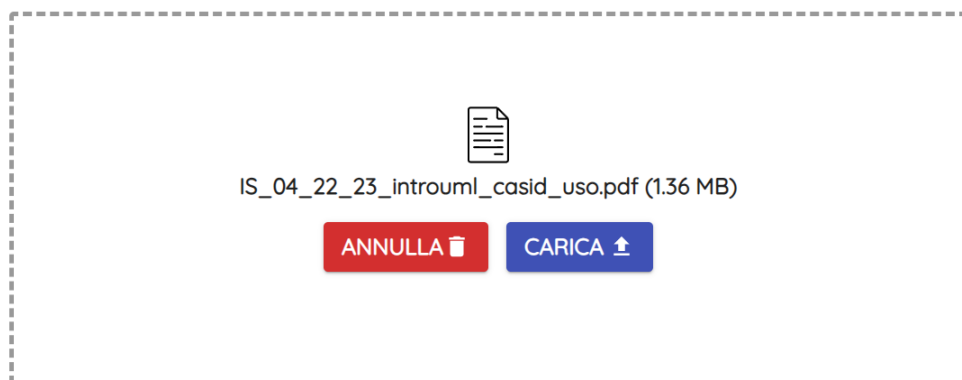


Figura 4.9: Esempio di caricamento di un file. L'area di caricamento mostra informazioni come il nome del file e la sua dimensione.

### Ricevuta

TokenID: 4  
TokenURI: <https://ipfs.io/ipfs/ce960ce61fd952f1e83f18c39e5e10cd8d79dc18f277cbf6bc42d543f8de7e3d>  
CID: b4e3cc9d38284ad3aca9eb3428433e34d4254a7c9977c648404d63092038d121  
Data: 3/6/2024, 16:24:00

### Cronologia

TokenID: 4  
TokenURI: <https://ipfs.io/ipfs/ce960ce61fd952f1e83f18c39e5e10cd8d79dc18f277cbf6bc42d543f8de7e3d>  
CID: b4e3cc9d38284ad3aca9eb3428433e34d4254a7c9977c648404d63092038d121  
Data: 3/6/2024, 16:24:00

TokenID: 3  
TokenURI: <https://ipfs.io/ipfs/ce960ce61fd952f1e83f18c39e5e10cd8d79dc18f277cbf6bc42d543f8de7e3d>  
CID: 0e33b083968134e7bedd9980e426fa4b66d281358551cbb8211cac54bdb725b9  
Data: 3/6/2024, 16:23:49

TokenID: 2  
TokenURI: <https://ipfs.io/ipfs/ce960ce61fd952f1e83f18c39e5e10cd8d79dc18f277cbf6bc42d543f8de7e3d>  
CID: fece356bf8801c9b5c3d560442e17ac8e8123f05e57606c355b8ac7aa377914e  
Data: 3/6/2024, 16:23:16

TokenID: 1  
TokenURI: <https://ipfs.io/ipfs/ce960ce61fd952f1e83f18c39e5e10cd8d79dc18f277cbf6bc42d543f8de7e3d>  
CID: acf150b4b26896dd734f9f4eb3ad4a16293609d7e829a58360f4738c7f9e544f  
Data: 3/6/2024, 16:22:59

Figura 4.10: Illustrazione di una *CompoundReceipt* associata a un documento. La ricevuta contiene informazioni sulla certificazione attuale del documento e la cronologia delle versioni precedenti, come i tokenId, gli URI dei token, i CID e le date di certificazione.



## Capitolo 5

# Risultati sperimentali

In questo capitolo vengono presentati i risultati ottenuti durante lo sviluppo e la sperimentazione del progetto. Si analizzano l'impatto dei diversi tipi di memoria utilizzati in Solidity (storage, memory e calldata), il consumo di gas per le operazioni di minting e burning, e l'efficienza della strategia di utilizzo di un singolo slot di memoria da 32 byte per archiviare quattro valori uint64 (quindi da 64 bit ciascuno).

### 5.1 Confronto tra Storage, Memory e Calldata

In Solidity, la scelta tra storage, memory e calldata ha un impatto significativo sulle prestazioni e sui costi delle operazioni contrattuali.

- **Storage:** Lo storage viene utilizzato per memorizzare dati in modo persistente sulla blockchain. Ogni operazione su una variabile di storage rappresenta una modifica dello stato della blockchain, il che comporta un costo di gas molto elevato. Di conseguenza, è essenziale ottimizzare le operazioni di scrittura e lettura su storage per ridurre i costi.
- **Memory:** La memoria viene utilizzata per i dati temporanei all'interno delle funzioni. Le operazioni di lettura e scrittura sono più economiche rispetto allo storage, ma i dati in memoria non persistono tra le chiamate delle funzioni.
- **Calldata:** Calldata è una memoria temporanea non modificabile che contiene i dati degli argomenti della funzione. È la più economica in termini di gas, ma può essere utilizzata solo per leggere dati, non per modificarli.

Durante lo sviluppo del progetto, le operazioni contrattuali sono state ottimizzate scegliendo attentamente tra storage, memory e calldata per ridurre il consumo di gas.

## 5.2 Utilizzo di Gas

L'efficienza del contratto è stata una priorità, e ciò si riflette nell'ottimizzazione del consumo di gas. Di seguito sono riportati i costi in gas per le operazioni di minting e burning:

- **10 mint:** Il costo medio per mintare 10 NFT è stato di 25 centesimi.
- **10 mint + burn:** Il costo medio per mintare 10 NFT e bruciare 10 NFT associato a vecchie certificazioni è stato di 40 centesimi.

Questi costi evidenziano l'efficienza raggiunta nell'operazione di minting, grazie all'utilizzo del contratto ERC-721A, che consente la creazione di più token in modo ottimizzato rispetto alla versione standard.

## 5.3 Ottimizzazione dello Storage

Nel progetto è stata utilizzata una strategia di ottimizzazione dello storage che prevede l'archiviazione di quattro valori in un singolo slot di 32 byte costituito da variabili uint256. Questa scelta riduce il numero di operazioni di scrittura necessarie, portando a una diminuzione dei costi complessivi di gas.

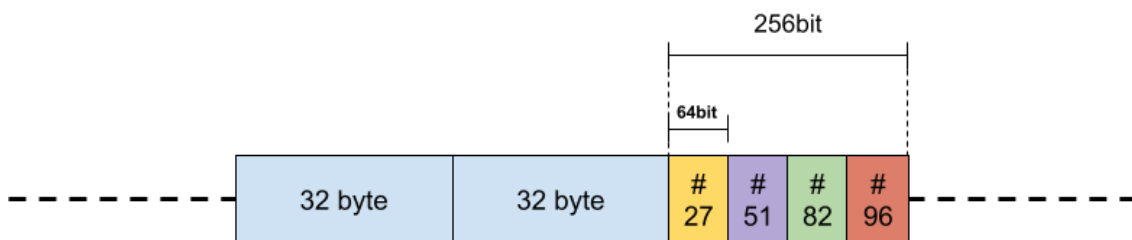


Figura 5.1: I tokenId 27, 51, 82 e 96 vengono archiviati in un singolo slot di memoria.

Questa tecnica consente di eseguire un'unica operazione di scrittura e tre operazioni di sovrascrittura invece di quattro scritture, riducendo significativamente il consumo di gas e occupando meno slot di memoria. Questa ottimizzazione è stata impiegata nella gestione degli array relativi agli storici delle certificazioni.

Operazione	Costo in Gas
Scrittura	20,000
Sovrascrittura	5,000
Lettura	200

Tabella 5.1: Costi in gas per operazioni di scrittura, lettura e aggiornamento di una variabile storage

### 5.3.1 Svantaggi

Nonostante i vantaggi in termini di efficienza del gas, l'approccio di utilizzare un singolo uint256 per memorizzare quattro valori uint64 presenta alcuni svantaggi:

- **Limitazione nel numero di NFT generabili:** Utilizzando uint256 per archiviare valori uint64, il numero totale di NFT generabili è ridotto da  $2^{256} - 1$  a  $2^{64} - 1$ , poiché i valori sono troncati a 64 bit ciascuno. Questo significa che il contratto non può gestire un numero "praticamente infinito" di NFT come potrebbe teoricamente fare utilizzando uint256 senza questa ottimizzazione.
- **Manutenzione del codice:** La gestione di dati compressi in un singolo slot di memoria può complicare il codice, rendendolo meno leggibile e più difficile da mantenere. Le operazioni di scrittura e lettura diventano più complesse, poiché richiedono il calcolo degli offset corretti all'interno dello slot di memoria.
- **Potenziale per errori:** La manipolazione manuale degli slot di memoria aumenta il rischio di errori di programmazione, che possono portare a bug difficili da diagnosticare e risolvere.

### 5.3.2 Ottimizzazioni Minori

Oltre all'ottimizzazione principale dello storage, sono state implementate diverse ottimizzazioni minori per migliorare ulteriormente l'efficienza del contratto:

- **Blocco unchecked:** L'uso del blocco `unchecked` per operazioni aritmetiche riduce il costo del gas eliminando i controlli di overflow integrati.

```
1      uint256 result;  
2      unchecked {  
3          result = value + 1;  
4      }  
5
```

Listing 5.1: Esempio di incremento di un valore in un blocco `unchecked`, questa tecnica consente di risparmiare gas

- **Uso di mapping rispetto ad array:** L'uso di mapping per gestire collezioni di dati invece di array permette un accesso più efficiente, evitando la necessità di iterare attraverso gli elementi e riducendo il costo del gas per operazioni di lettura e scrittura.
- **Caching dei dati di storage:** Cache dei dati di storage nelle variabili locali per ridurre il numero di letture dallo storage, che sono costose in termini di gas.

- **Verifica precoce dell'integrità dei dati:** I controlli per l'integrità dei dati vengono eseguiti all'inizio delle funzioni. Questo consente di effettuare un *revert* precoce in caso di errore, evitando di sprecare gas su operazioni che verrebbero comunque annullate.
- **Inizializzazione ritardata degli array storici:** Gli array storici vengono inizializzati solo al primo aggiornamento di una certificazione. In questo modo, si evita di allocare spazio di storage non necessario per i documenti certificati per la prima volta, poiché non hanno ancora uno storico.
- **Uso di `calldata` per input delle funzioni:** L'uso di `calldata` per gli input delle funzioni riduce il costo del gas, poiché `calldata` è più economico di `memory` per i dati in ingresso.
- **Funzione `view` per la verifica:** La funzione `verify()` è stata definita come `view` poiché è di sola lettura e non modifica lo stato del contratto, riducendo il costo delle chiamate a questa funzione.
- **Memorizzazione ottimizzata delle URI:** Le URI dei documenti vengono memorizzate come `bytes32` anziché come stringhe. Ad esempio, la base URI `"https://ipfs.io/ipfs/"` viene memorizzata *hard-coded*, mentre solo l'hash `"ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad"` viene salvato, occupando un solo slot di memoria.

## 5.4 Conclusioni

I risultati sperimentali mostrano che l'ottimizzazione delle operazioni e l'uso efficiente dello storage possono portare a una significativa riduzione dei costi di gas. La scelta di utilizzare `uint256` per archiviare più valori `uint64` si è dimostrata vantaggiosa, riducendo il numero complessivo di operazioni di scrittura. Inoltre, l'adozione di ERC-721A ha consentito un minting più efficiente, abbassando ulteriormente i costi di gas per operazioni multiple. Questi risultati sottolineano l'importanza di scrivere codice efficiente e della gestione dello storage per migliorare le prestazioni e ridurre i costi delle transazioni su blockchain.

Tuttavia, è importante tenere presente gli svantaggi associati a queste ottimizzazioni, in particolare la riduzione del numero massimo di NFT generabili e la maggiore complessità nella manutenzione del codice.

## Capitolo 6

# Conclusioni e Sviluppi Futuri

Il progetto ha raggiunto importanti risultati nella riduzione dei costi di certificazione e aggiornamento oltre all'ottimizzazione dell'uso dello storage sulla blockchain. Tuttavia, durante la revisione finale, l'azienda ha deciso di non utilizzare il contratto che fa utilizzo di ERC-721A. Questa scelta è stata fatta per garantire una maggiore manutenibilità. Il contratto scelto per l'uso effettivo è basato sullo standard ERC-721 di OpenZeppelin senza l'estensione Enumerable. Questo approccio, sebbene meno efficiente in termini di gas, offre una struttura di codice più semplice e manutenibile. Ogni nuovo utente che richiede una licenza avrà un contratto dedicato, evitando la mescolanza di dati tra diversi utenti. Quindi, il nuovo contratto non supporta il minting multiplo, potendo gestire al massimo una singola richiesta di certificazione o aggiornamento alla volta.

- **1 mint:** Il costo medio per mintare 1 NFT è ora di 30 centesimi.
- **1 mint + burn:** Il costo medio per mintare 1 NFT e bruciare l'NFT associato alla vecchia certificazione è ora di 50 centesimi.

### 6.1 Conclusione

Il progetto ha dimostrato l'importanza dell'ottimizzazione delle operazioni sullo smart contract, dell'implementazione di tecniche di batch minting e dell'uso minimo dello storage per ridurre i costi di gas. Tuttavia, la leggibilità e la manutenibilità del codice sono state prioritarie per l'azienda, garantendo uno smart contract sicuro e minimale.

### 6.2 Sviluppi Futuri

In futuro, questo smart contract sarà utilizzato per la certificazione di documenti nel settore del luxury fashion. Creando NFT che fungono da Digital Product Passport (DPP), il contratto permetterà di autenticare e tracciare ogni fase del ciclo di vita dei

prodotti di lusso. Questo approccio consentirà di certificare l'origine, l'autenticità e altre informazioni cruciali sui prodotti, migliorando la trasparenza e la fiducia dei consumatori. Inoltre, faciliterà la prevenzione della contraffazione, rendendo l'intero processo di certificazione più sicuro e affidabile.

## 6.3 Competenze Acquisite

Durante il tirocinio, ho acquisito competenze pratiche nel campo dello sviluppo Web e delle blockchain. Ho imparato a scrivere e ottimizzare smart contract, a integrare soluzioni blockchain con servizi backend e a gestire risorse in modo efficiente. Un aspetto fondamentale di questa esperienza è stato il problem solving. Ho affrontato momenti in cui non avevo la minima idea di quale soluzione adottare per aderire agli obiettivi del progetto. Le specifiche del progetto sono cambiate man mano che procedevamo con lo sviluppo. Dopo aver completato una versione complessa e poco manutenibile, abbiamo deciso di tornare a una versione più semplice e manutenibile. Questa esperienza mi ha insegnato l'importanza della flessibilità e dell'adattabilità nel lavoro.

Inoltre, durante il tirocinio, ho riconosciuto quanto sia cruciale avere la capacità di setacciare informazioni su internet. Nell'ambito dello sviluppo e del coding, questa competenza è fondamentale per individuare risorse affidabili e pertinenti, e per capire quali dati siano rilevanti in base alle necessità. La capacità di scorporre e interpretare le informazioni su Internet non è solo un vantaggio, ma un requisito imprescindibile per lo sviluppo software. Inoltre, una buona conoscenza dell'inglese è stata fondamentale per comprendere la documentazione tecnica e rimanere aggiornato sulle ultime tendenze del settore. La creatività e la voglia di sperimentare sono state essenziali per esplorare nuove soluzioni e ricercare la soluzione più bilanciata.

# Appendice A

## Glossario dei Termini

In questa sezione, vengono spiegati alcuni dei termini tecnici utilizzati nel testo che potrebbero non essere familiari al lettore.

### A.1 Mining

Il mining è il processo di validazione e aggiunta di nuove transazioni a una blockchain, utilizzando la potenza computazionale per risolvere problemi crittografici complessi.

### A.2 Staking

Lo staking è il processo di partecipazione attiva alla validazione delle transazioni su una blockchain proof-of-stake (PoS) bloccando una certa quantità di criptovaluta come garanzia.

### A.3 Minting

Il minting è il processo di creazione di nuovi token o NFT sulla blockchain.

### A.4 Burning

Il burning è il processo di rimozione permanente di token dalla circolazione, riducendo la quantità totale disponibile.

### A.5 Batch Minting

Il batch minting è il processo di creazione di più token o NFT in una singola transazione, per ottimizzare i costi del gas.

## A.6 BaseURI

La baseURI è l'URL di base utilizzato per formare gli URL completi dei metadati associati ai token NFT.

## A.7 Layer 2

Layer 2 si riferisce a soluzioni che operano sopra una blockchain esistente (Layer 1) per migliorare la scalabilità e la velocità delle transazioni.

## A.8 Trustless

Un sistema trustless è un sistema in cui i partecipanti non devono fidarsi l'uno dell'altro o di una terza parte, grazie all'uso della tecnologia blockchain che garantisce la sicurezza e l'integrità dei dati.

## A.9 Provider

Un provider, come Infura, è un servizio che consente alle applicazioni di interagire con la blockchain senza dover gestire un nodo completo.

## A.10 Back-End

Il back-end si riferisce alla parte server di un'applicazione, dove avvengono l'elaborazione dei dati, la logica di business e l'interazione con il database.

## A.11 Server

Un server è un sistema che fornisce risorse, dati, servizi o programmi ai client, che lo richiedono tramite una rete.

## A.12 Front-End

Il front-end si riferisce alla parte client di un'applicazione, con cui l'utente interagisce direttamente. Utilizza tecnologie come HTML, CSS e JavaScript.

## A.13 Reactive

Nel campo dello sviluppo front-end, un'interfaccia reattiva (reactive) si riferisce alla programmazione che risponde automaticamente agli eventi. Quando i dati cambiano,



l'interfaccia utente si aggiorna in modo efficiente per riflettere questi cambiamenti. Questo approccio migliora l'interattività e l'esperienza dell'utente.

## A.14 HTML

HTML (HyperText Markup Language) è il linguaggio standard utilizzato per creare e strutturare contenuti sul web. Utilizzando tag ed elementi, HTML definisce la struttura di una pagina web, come paragrafi, intestazioni, link, immagini e altri tipi di contenuti multimediali.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>My Web App</title>
5 </head>
6 <body>
7   <h1>Welcome to My Web App</h1>
8   <p>This is a simple front-end example.</p>
9 </body>
10 </html>
```

Listing A.1: Esempio di semplice front-end in HTML

## A.15 CSS

CSS (Cascading Style Sheets) è il linguaggio utilizzato per descrivere l'aspetto e la formattazione di un documento HTML. CSS consente di separare la presentazione dal contenuto, migliorando la manutenibilità e la flessibilità del codice.

## A.16 JavaScript

JavaScript è un linguaggio di programmazione utilizzato per creare contenuti web interattivi. È spesso utilizzato per manipolare il DOM (Document Object Model) e gestire eventi.

```
1 document.addEventListener('DOMContentLoaded', (event) => {
2   const button = document.createElement('button');
3   button.textContent = 'Clicca qui per un messaggio di
  benvenuto';
4   document.body.appendChild(button);
5
6   button.addEventListener('click', () => {
7     alert('Benvenuto nella mia pagina web!');
8   });
9 });
```

```
9 });
```

Listing A.2: Esempio di codice JavaScript

## A.17 Web App

Un'applicazione web (web app) è un'applicazione software che gira su un server web e può essere accessibile tramite un browser.

## A.18 Framework

Un framework è un insieme di strumenti e librerie che facilitano lo sviluppo di software. Esempi comuni includono React, Angular, Vue.js, Svelte e Next.js.

## A.19 Node.js

Node.js è un runtime JavaScript che permette di eseguire codice JavaScript lato server. È utilizzato per creare applicazioni back-end.

# Bibliografia

- [1] Andreas M Antonopoulos and Gavin Wood. *Mastering ethereum: building smart contracts and dapps*. O'reilly Media, 2018.
- [2] Material Design. Documentazione. <https://m3.material.io/>.
- [3] ERC-721. Documentazione. <https://docs.openzeppelin.com/contracts/5.x/api/token/erc721>.
- [4] ERC-721A. Documentazione. <https://chiru-labs.github.io/ERC721A/#/>.
- [5] EthersJS. Documentazione. <https://docs.ethers.org/v6/>.
- [6] ExpressJS. Documentazione. <https://expressjs.com/en/5x/api.html>.
- [7] Ethereum Foundation. Documentazione. <https://ethereum.org/it/developers/docs/>.
- [8] Hardhat. Documentazione. <https://hardhat.org/hardhat-runner/docs/>.
- [9] IPFS. Documentazione. <https://docs.ipfs.tech/>.
- [10] Linea. Documentazione. <https://docs.lineascan.build/>.
- [11] MUI. Documentazione. <https://mui.com/material-ui/>.
- [12] React. React. built-in hooks. <https://react.dev/reference/react/>.
- [13] Solidity. Documentazione. <https://docs.soliditylang.org/en/v0.8.26/>.



# Ringraziamenti

*Desidero innanzitutto ringraziare la professoressa Laura Ricci per la guida nella stesura della tesi.*

*Ringrazio l'azienda Advinser(That's it) per avermi ospitato durante lo svolgimento del tirocinio e per la cordialità con il quale mi hanno accolto al tavolo dove pranzavano.*

*Ringrazio di cuore i miei genitori, che mi hanno aiutato a raggiungere questo importante traguardo e mi hanno sempre supportato nei momenti di bisogno. Vi voglio bene e spero di avervi resi orgogliosi di me.*

*Un grazie di cuore agli amici del Praticelli, che hanno reso il mio soggiorno alla residenza meno solitario e più vivace. Vi ringrazio per essermi stati vicini e non vedo l'ora di rivedervi tutti.*

*Ringrazio anche i miei amici del Gruppo Studio, che mi hanno tenuto compagnia lungo questo viaggio e che mi hanno regalato dei bellissimi attimi di spensieratezza, ripenso ancora alle serate che abbiamo trascorso in via Ettore Sighieri e non posso che provare nostalgia per un periodo che è lontano ma eppure così vicino.*

*Sono felice di aver avuto Matteo, Stefano e Emanuele come coinquilini e sinceramente penso che non sarei potuto capitare con persone migliori. È stato bello, per quello che è durato, trovare la cucina gremita di amici dopo una giornata all'università.*