ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

Facoltà di INGEGNERIA

Corso di Laurea Triennale in Ingegneria e Scienze Informatiche

STAKESHARE, SVILUPPO DI UN APPLICATIVO COMPLETAMENTE DECENTRALIZZATO SU BLOCKCHAIN

Relatore:
Prof. Stefano Ferretti
Correlatore:
Prof. Vittorio Ghini
Dott. Samuele Medici

Candidato:
Nicolò Ghignatti

Sessione I
Anno accademico 2023/2024

Indice

Capitolo 1

Introduzione

L'obiettivo finale di questa tesi risiede nello sviluppo di un'applicazione interamente decentralizzata, costruita sulla blockchain nota come **Internet Computer Protocol** (ICP[1]). Questo progetto ambizioso mira a sfruttare in modo completo le caratteristiche innovative e all'avanguardia della blockchain ICP. La blockchain ICP è particolarmente rinomata per la sua capacità di combinare tre qualità essenziali: scalabilità, sicurezza ed efficienza energetica. Queste qualità sono fondamentali per creare un sistema che non solo funzioni efficacemente ma che riesca anche a superare le tradizionali limitazioni delle piattaforme centralizzate che dominano attualmente il panorama tecnologico.

Attraverso l'impiego della blockchain ICP, l'applicazione proposta si prefigge di offrire agli utenti un ambiente digitale in cui essi possano interagire direttamente, senza la necessità di intermediari. Questa interazione diretta garantisce una serie di benefici significativi, tra cui una maggiore privacy per gli utenti, una trasparenza superiore e un controllo totale sui propri dati personali. L'eliminazione degli intermediari, che sono spesso necessari nelle piattaforme centralizzate, non solo semplifica i processi operativi ma riduce anche i costi associati e i tempi di latenza delle operazioni. Questi miglioramenti congiunti portano a un'esperienza utente notevolmente migliorata, che è più fluida, veloce e conveniente.

Il presente lavoro si propone di dimostrare in modo chiaro e dettagliato come la tecnologia blockchain possa rivoluzionare il modo in cui le informazioni vengono memorizzate, condivise e verificate all'interno di sistemi digitali. La blockchain, con la sua struttura decentralizzata e sicura, offre soluzioni innovative che possono contribuire allo sviluppo di applicazioni digitali più sicure, efficienti e accessibili rispetto alle tecnologie attualmente in uso. In particolare, la blockchain ICP si distingue per la sua capacità unica di supportare applicazioni su larga scala, mantenendo al contempo un elevato livello di sicurezza. Questo è reso possibile grazie a meccanismi di consenso avanzati e a una crittografia robusta che protegge i dati e le transazioni da accessi non autorizzati e da attacchi informatici.

Inoltre, la ricerca condotta in questa tesi si concentra sulle molteplici sfide tecniche associate allo sviluppo di applicazioni decentralizzate. Queste sfide includono, ma non si limitano a, la gestione degli smart contract, la scalabilità della rete e la facilità d'uso per gli utenti finali. Ogni aspetto tecnico è analizzato in dettaglio, cercando di proporre soluzioni pratiche e innovative che possano essere applicate per superare questi ostacoli. La gestione degli smart contract, ad esempio, richiede una considerazione approfondita delle implicazioni di sicurezza e dell'efficienza esecutiva. Gli smart contract sono infatti fondamentali per l'automazione delle transazioni e per l'esecuzione di codice su piattaforme blockchain, e una loro gestione efficace è cruciale per il successo delle applicazioni decentralizzate.

La scalabilità della rete rappresenta un'altra area critica di attenzione all'interno di questa tesi. La capacità di una blockchain di gestire un numero crescente di transazioni è essenziale per il suo successo a lungo termine e per la sua adozione su larga scala. Il progetto esplora diverse tecniche avanzate per migliorare la scalabilità, come l'implementazione di sharding e soluzioni di layer 2. Queste tecniche hanno il potenziale di aumentare significativamente il throughput della rete, permettendo alla blockchain di gestire un volume di transazioni molto più elevato senza compromettere la sicurezza e l'integrità dei dati.

Il progetto principale su cui si basa questa tesi è l'applicazione denominata Stakeshare. Stakeshare è un'applicazione che facilita la gestione delle quote di partecipazione in diverse iniziative, sfruttando la notarizzazione su blockchain per garantire trasparenza e sicurezza. Scritta in Rust e compilata in WebAssembly per l'esecuzione su Internet Computer Protocol (ICP), questa soluzione consente agli utenti di registrare e scambiare quote di proprietà o partecipazione in maniera digitale e decentralizzata, con la certificazione legale tramite standard NFT (ICRC7). Attraverso l'uso di smart contracts, Stakeshare automatizza gli accordi tra le parti e la distribuzione dei proventi, rendendo il processo di gestione delle quote più semplice, sicuro e conforme alle normative.

Nello sviluppo di Stakeshare, ho dovuto affrontare numerose sfide tecniche e organizzative. La scelta del linguaggio di programmazione Rust e la sua compilazione in WebAssembly per l'ICP ha richiesto una profonda comprensione delle peculiarità di entrambi i sistemi. Ho lavorato sulla progettazione e l'implementazione di smart contracts in grado di gestire le transazioni in modo sicuro ed efficiente. Questo ha incluso la scrittura di codice per automatizzare la distribuzione dei proventi e la gestione delle proprietà, garantendo al contempo che tutte le operazioni fossero conformi alle normative vigenti.

Inoltre, ho affrontato le sfide legate alla scalabilità dell'applicazione, implementando tecniche avanzate per assicurare che il sistema potesse gestire un alto volume di transazioni senza compromettere le performance. La progettazione di un'interfaccia utente intuitiva e facile da usare è stata un'altra area critica di lavoro, assicurando che gli utenti potessero interagire con l'applicazione in modo semplice e diretto.

Infine, questo lavoro di tesi si propone di fornire una base teorica e pratica solida per futuri sviluppatori e ricercatori che sono interessati ad esplorare ulteriormente le possibilità offerte dalla

blockchain ICP. Contribuendo così all'avanzamento della conoscenza e all'innovazione nel campo delle tecnologie decentralizzate, la tesi non solo documenta in modo dettagliato il processo di sviluppo dell'applicazione ma include anche una serie di argomentazioni sui vantaggi dell'adozione di questa blockchain specifica. Inoltre, si discute in generale dell'utilizzo delle blockchain nello sviluppo di applicazioni, evidenziando i benefici e le potenzialità di queste tecnologie emergenti.

In conclusione, questo progetto rappresenta un contributo significativo alla comprensione e all'implementazione pratica delle applicazioni basate sulla blockchain ICP. La combinazione di ricerca teorica e sviluppo pratico fornisce un quadro completo e approfondito delle potenzialità di questa tecnologia innovativa. Promuovendo l'adozione di soluzioni decentralizzate, il progetto può trasformare vari settori, dalla finanza alla gestione dei dati personali, rendendo il mondo digitale più sicuro, efficiente e democratico. La speranza è che questo lavoro possa servire da guida e ispirazione per futuri sviluppi nel campo delle tecnologie decentralizzate, contribuendo a un progresso continuo e sostenibile in questo ambito affascinante e in rapida evoluzione.

Il resto della tesi è organizzato quanto segue:

- Capitolo 2: Background, qualche accenno a quello che è il background tecnologico della nostra applicazione, delineando i concetti chiavi alla base di quest'ultima.
- Capitolo 3: Tecnologie usate, in questo capitolo vengono descritte e motivate le tecnologie scelte per lo sviluppo dell'applicazione, analizzandone i vantaggi e come hanno contribuito al progetto.
- Capitolo 4: Stakeshare, capitolo dedicato all'applicazione sviluppata come progetto di tesi. Viene presentata una dettagliata descrizione di Stakeshare, inclusi gli obiettivi, le funzionalità principali e i risultati ottenuti durante il processo di sviluppo.

Capitolo 2

Background

2.1 Architetture decentralizzate

2.1.1 Introduzione

Le architetture decentralizzate rappresentano un paradigma di progettazione dei sistemi informatici che si distingue per la distribuzione delle risorse e delle responsabilità tra diversi nodi della rete, piuttosto che concentrarle in un unico punto centrale. A differenza delle architetture centralizzate, dove un unico server o un gruppo ristretto di server gestisce tutte le operazioni e i dati, le architetture decentralizzate permettono una distribuzione equa e autonoma delle operazioni tra molteplici nodi indipendenti.

Questo modello innovativo sta guadagnando crescente popolarità grazie alla sua capacità di offrire numerosi vantaggi rispetto alle architetture centralizzate tradizionali. Una delle principali caratteristiche delle architetture decentralizzate è la resilienza. Poiché non esiste un singolo punto di fallimento, il sistema è intrinsecamente più robusto e resistente agli attacchi, ai guasti hardware e alle interruzioni del servizio. La ridondanza dei dati e delle funzioni operative garantisce che il sistema possa continuare a funzionare anche in caso di problemi su uno o più nodi della rete.

La scalabilità è un altro importante vantaggio delle architetture decentralizzate. In un sistema decentralizzato, la capacità di elaborazione e archiviazione può crescere in modo organico aggiungendo nuovi nodi alla rete. Questo approccio permette di gestire un numero crescente di utenti e di richieste senza incorrere nei colli di bottiglia tipici delle architetture centralizzate. La scalabilità orizzontale consente di espandere le risorse in maniera flessibile e modulare, adattandosi alle esigenze del momento.

In termini di sicurezza, le architetture decentralizzate offrono un livello di protezione superiore. La distribuzione dei dati e delle operazioni rende più difficile per i malintenzionati compromettere l'intero sistema, poiché dovrebbero attaccare contemporaneamente una grande quantità di nodi.

Inoltre, l'uso di tecniche di crittografia avanzate e di algoritmi di consenso, come quelli utilizzati nelle blockchain, assicura che le transazioni e i dati siano validati e immutabili, incrementando la fiducia e la trasparenza del sistema.

Questi benefici rendono le architetture decentralizzate particolarmente adatte a una vasta gamma di applicazioni moderne, tra cui le criptovalute, le piattaforme di smart contract, i sistemi di gestione dei dati e le reti peer-to-peer per la condivisione di file. La capacità di migliorare la resilienza, la scalabilità e la sicurezza sta trasformando il modo in cui vengono progettati e implementati i sistemi informatici, spingendo sempre più organizzazioni a considerare l'adozione di modelli decentralizzati.

2.1.2 Principi di funzionamento

Le architetture decentralizzate sono progettate per distribuire compiti, dati e risorse tra molti nodi autonomi in una rete, eliminando la dipendenza da un singolo punto di controllo. Questo approccio è in netto contrasto con le architetture centralizzate, dove un singolo server o un gruppo di server centralizzati gestisce tutte le operazioni e detiene tutti i dati.

In un sistema decentralizzato, ogni nodo della rete può svolgere funzioni simili, agendo sia come client che come server. I nodi comunicano direttamente tra loro tramite protocolli peer-to-peer (P2P), scambiando informazioni e cooperando per raggiungere obiettivi comuni. Questa comunicazione diretta elimina la necessità di un intermediario centrale, rendendo la rete più robusta e resiliente.

Affinché quest'architettura sia applicabile, bisogna prestare attenzione a determinati punti critici:

- **distribuzione dei dati**: uno dei principi fondamentali delle architetture decentralizzate è la distribuzione dei dati tra tutti i nodi della rete. Questo significa che ogni nodo conserva una copia dei dati o una parte dei dati necessari al funzionamento del sistema. Ad esempio, nella blockchain, ogni nodo possiede una copia completa del registro delle transazioni. Quando una nuova transazione viene aggiunta, viene trasmessa a tutti i nodi, che la validano e la aggiungono al proprio registro.

- **algoritmi di consenso**: Per garantire che tutti i nodi della rete siano d'accordo sullo stato dei dati e sulle operazioni eseguite, le architetture decentralizzate utilizzano algoritmi di consenso. Questi algoritmi permettono ai nodi di raggiungere un accordo comune nonostante la loro distribuzione geografica e l'assenza di un'autorità centrale. Alcuni degli algoritmi di consenso più noti includono la *proof of work*, usata da Bitcoin, e la *proof of stake*, usata la Ethereum 2.0.

- **comunicazione e cooperazione**: Nei sistemi decentralizzati, la comunicazione tra i nodi è cruciale per il funzionamento efficace della rete. I nodi devono essere in grado di trasmettere

dati, aggiornamenti e richieste in modo rapido ed efficiente. Protocollo come BitTorrent, utilizzato per la condivisione di file, è un esempio classico di rete P2P dove i file sono suddivisi in piccoli pezzi e distribuiti tra tutti i partecipanti. Ogni nodo può scaricare e caricare contemporaneamente i pezzi del file, accelerando il processo complessivo di distribuzione.

2.1.3 Vantaggi

Le architetture decentralizzate presentano diversi vantaggi:

- **resilienza e affidabilità**: L'assenza di un singolo punto di fallimento rende i sistemi decentralizzati meno vulnerabili agli attacchi e ai guasti.
- **scalabilità**: L'aggiunta di nuovi nodi aumenta la capacità di calcolo e archiviazione della rete, permettendo di gestire un numero crescente di utenti e richieste.
- **sicurezza e trasparenza**: La distribuzione dei dati e l'uso di crittografia avanzata rendono più difficile per i malintenzionati compromettere il sistema, e ogni operazione è verificabile pubblicamente.
- **riduzione dei costi**: La decentralizzazione elimina la necessità di grandi infrastrutture centralizzate e distribuisce i costi tra i partecipanti alla rete.

2.1.4 Esempi

Vi sono vari esempi di architetture decentralizzate, famosi sono **BitTorrent**[2], che è una delle reti P2P più conosciute al mondo per la condivisione file, **Bitcoin**[3] che è la prima criptovaluta basata su blockchain o **IPFS** (InterPlanetary File System[4]) che è un protocollo di rete progettato per creare un sistema di archiviazione e confivisione file decentralizzato.

2.2 Le blockchain

2.2.1 Breve Storia

La storia della blockchain [5] è profondamente radicata nell'ambiente finanziario e tecnologico globale, originandosi come una soluzione alla crisi finanziaria del 2008. Questa crisi mise in evidenza i limiti e le vulnerabilità del sistema finanziario tradizionale, che era altamente centralizzato e soggetto a manipolazioni e corruzione. La blockchain venne concepita come una tecnologia che potesse offrire un meccanismo alternativo, più sicuro, trasparente e decentralizzato per gestire transazioni e dati digitali.

La nascita ufficiale della blockchain è datata 3 gennaio 2009, quando Satoshi Nakamoto, l'autore anonimo del Bitcoin White Paper [6], lanciò la prima criptovaluta, il Bitcoin [3]. Questo evento segnò l'inizio di un nuovo capitolo nella storia della tecnologia, dove la decentralizzazione e la crittografia divennero fondamentali per garantire la sicurezza e l'integrità delle transazioni digitali evitando di affidarsi a terze parti.

Dal 2009, la blockchain ha attraversato vari cicli di interesse e ipotesi, culminando in un periodo di grande entusiasmo nel 2016, noto come "Crypto Winter". Durante questo periodo, molte startup e progetti legati alla blockchain hanno ricevuto finanziamenti massicci attraverso le Initial Coin Offerings (ICO[7]), solo per poi affrontare un drastico calo dei prezzi delle criptovalute e numerosi problemi normativi.

Nel 2020, la situazione ha iniziato a cambiare con iniziative da parte di banche centrali e governi che hanno accelerato le sperimentazioni per emettere Central Bank Digital Currencies (CBDC[8]) e regolare i crypto-asset. Nel 2021, la finanza decentralizzata (DeFi) ha visto un significativo aumento di interesse, mentre i crypto-asset hanno guadagnato rilevanza grazie all'adozione di istituti finanziari e alla riscoperta degli NFT[9] (Non-Fungible Tokens).

Nel 2022, il termine Web3 è diventato popolare, indicando un futuro internet basato su blockchain e smart contracts, dove i dati e le proprietà digitali sono gestiti in modo decentralizzato. Questo periodo ha visto anche la crescita delle applicazioni costruite su piattaforme pubbliche, estendendo l'applicazione della blockchain ben oltre il settore finanziario.

Tutt'ora il mondo delle blockchain è in continuo sviluppo e con il migliorare delle prestazioni sta prendendo sempre più piede come base per diversi applicativi. Assieme a questo rapido sviluppo delle blockchain sussiste una continua nascita di nuove criptovalute, sempre più integrate tra di loro e che rispondono ad ogni singola esigenza dell'utente.

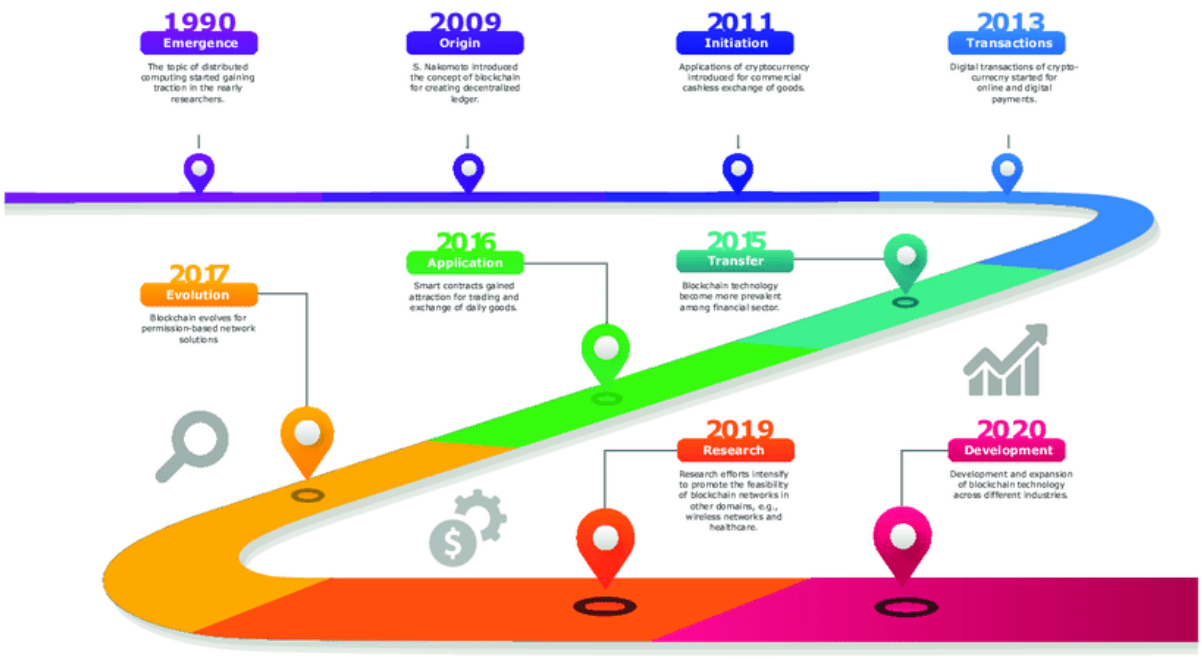

Figura 2.1: Linea del tempo sulla storia delle blockchain, immagine presa da researchgate

2.2.2 Come funziona

La blockchain, letteralmente "catena di blocchi", è una tecnologia che sfrutta le caratteristiche di una rete informatica di nodi (computer della rete che mantengono una copia del registro blockchain) per gestire e aggiornare in modo univoco e sicuro un registro contenente dati e informazioni in maniera aperta, condivisa e distribuita, senza la necessità di un'entità centrale di controllo e verifica. In pratica la blockchain può essere introdotta per disitermediare interazioni di varia natura, così da evitare di affidarsi ad enti terzi.

Il concetto alla base di questa "catena di blocchi" sono, appunto, i blocchi, un blocco non è altro che uno "spazio digitale" che ha una capienza limitata e contiene diverse informazioni, tra cui: il numero di blocco, i dati delle transazioni protetti con impornta hash, l'impronta hash del blocco precedente, il timestamp ed il nonce. Il nonce (number used once) è un numero che uno dei nodi della blockchain deve indovinare risolvendo un puzzle crittografico, che non è altro che un problema matematico per la cui risoluzione occorre un'enorme potenza di calcolo.

Il nodo che risolve il problema matematico assume il ruolo di "**miner**" e riceve una compensa (solitamente un'unità di criptovaluta) guadagnando sia dalla ricompensa ricevuta, sia percependo delle "**transaction fee**".

Per dimostrare di aver risolto il puzzle il miner deve dimostrare il suo lavoro, presentando la **Proof of Work**[10] (che può assumere nomi diversi in funzione di come viene svolta questa "verifica") ai vari nodi, i quali verificano e convalidano velocemente la correttezza della soluzione.

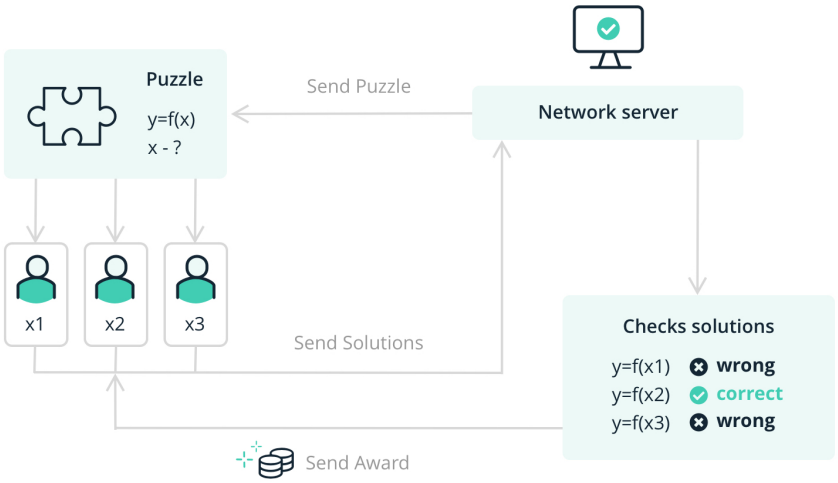

Figura 2.2: Esempio di come funziona il processo di verifica della correttezza, immagine presa da Ledger Academy

Grazie all'impronta hash e alla Proof of Work, la blockchain è considerata altamente sicura e inattaccabile. L'impronta hash del blocco precedente crea una connessione ininterrotta tra i blocchi, formando così una catena. Questo collegamento rende estremamente difficile alterare le informazioni di un blocco senza modificare anche tutti i blocchi successivi, il che richiederebbe un'immensa quantità di risorse computazionali. Inoltre, ogni nodo della rete possiede una copia del registro blockchain, il che significa che qualsiasi tentativo di manomissione sarebbe rapidamente identificato e respinto dalla rete.

Oltre alla Proof of Work, esistono altri meccanismi di consenso utilizzati nelle blockchain, come la Proof of Stake, che non richiede un'enorme potenza di calcolo ma piuttosto un impegno di risorse in termini di criptovaluta. La **Proof of Stake**(PoS[11]) è considerata una soluzione più ecologica rispetto alla Proof of Work, poiché riduce il consumo energetico necessario per mantenere la sicurezza della rete.

La blockchain trova applicazione in diversi settori oltre alle criptovalute, ad esempio, può essere utilizzata per la gestione delle identità digitali, la tracciabilità dei prodotti nella filiera logistica, la gestione dei diritti d'autore e dei brevetti. Per fare ciò è necessaria la creazione di smart contracts, contratti intelligenti che si auto-eseguono al verificarsi di determinate condizioni predefinite. Questa tecnologia ha il potenziale di trasformare radicalmente vari ambiti economici e sociali, promuovendo una maggiore trasparenza, sicurezza e decentralizzazione.

2.2.3 Esempi di blockchain

Nome	Data di nascita	Creatore
Algorand	2019	Silvio Micali
Aptos	2022	Aptos Foundation
Avalanche	2018	Emin Sirer, Maofan Tin e Kevin Sekniqi
Binance Smart Chain	2020	Binance
Bitcoin	2009	Satoshi Nakamoto
Cardano	2018	ADA
DESO	2021	Nader al-Naji
Dogecoin	2021	Dogecoin Foundation
EOS.IO	2017	Daniel Larimer e Brenden Blumer
Ethereum	2015	Ethereum Foundation
Hedera Hashgraph	2019	Leemon Baird
Hyperladger Fabric	2018	Linux Foundation
Internet Computer	2021	Dfinity Foundation
IOTA	2016	IOTA Foundation
Linea	2024	ConsenSys
Litecoin	2011	Charlie Lee
MazaCoin	2014	Payu Harris, AnonymousPirate
MobileCoin	2014	MobileCoin Inc.
Monero	2014	Joseph Liu, Howard C
Nano	2015	Colin LeMahieu
NEAR	2020	NEAR Foundation
Polkadot	2020	Gavin Wood,Peter Czaban e Robert Habermeier
Polygon PoS	2017	Polygon Technology
Primecoin	2013	Sunny King
Quorum	2016	JP Morgan, acquistato da ConsenSys
R3 Corda	2017	R3
Ripple	2012	Ripple Labs
Solana	2020	Anatoly Takovenko, Raj Gokal
Stellar	2016	Jed McCaleb, Joyce Kim
Terra 2.0	2022	Do Kwon e altri
Tezos	2018	Arthur and Kathleen Breitman
TRON	2018	Justin Sun
XinFin	2019	XinFin Fintech, XDC Foundation
Zcash	2016	Zooko Wilcox-O'Hearn

2.2.4 Applicazioni

La tecnologia blockchain è adottata su scala globale, con oltre 300 milioni di persone che ne fanno uso per le criptovalute, rappresentando il 3,9% della popolazione mondiale. Ci sono circa 82 milioni di utenti di portafogli blockchain in tutto il mondo, un numero che è significativamente cresciuto negli ultimi anni, passando da poco più di 10 milioni di utenti nel 2016 a oltre 80 milioni nel 2021.

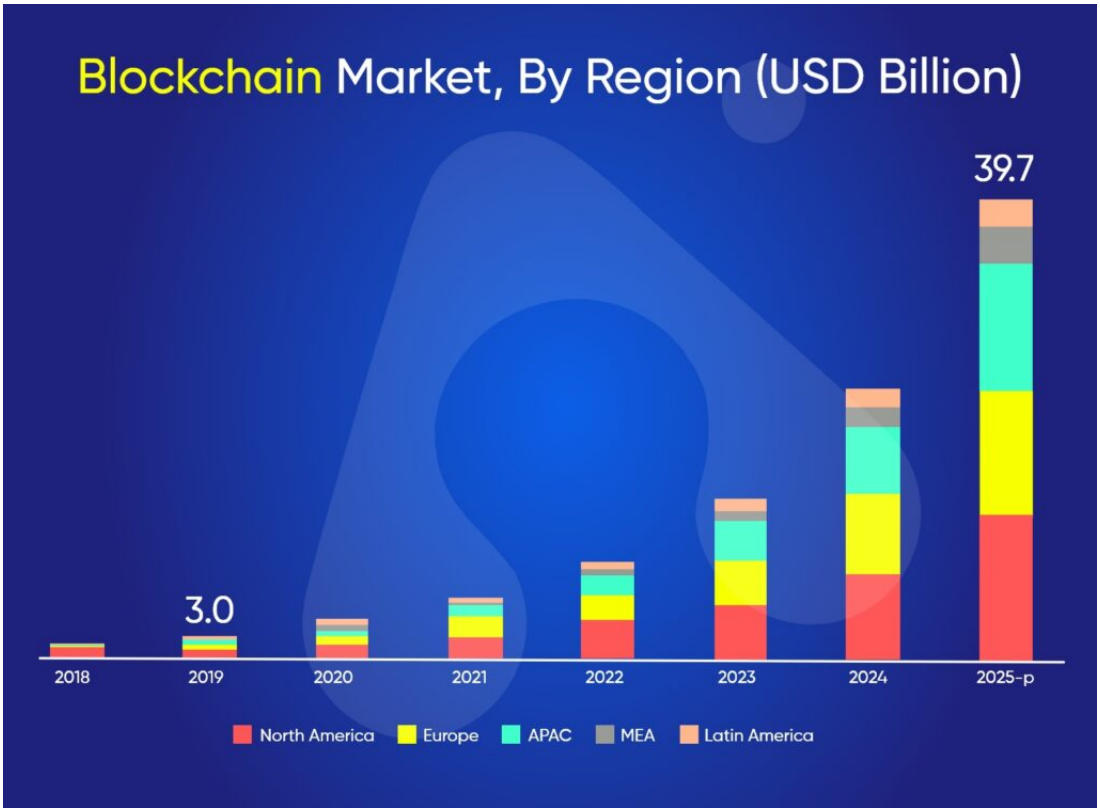

Figura 2.3: Mercato delle blockchain diviso in regioni, immagine presa da AppVenturex

Le applicazioni della blockchain coprono vari settori, tra cui servizi finanziari, retail, marketing e sanità, con prospettive future che includono la gestione della catena di approvvigionamento, l'archiviazione cloud, la cybersecurity e gli smart contract. L'adozione delle criptovalute è in aumento grazie al rapido aumento delle capacità delle blockchain.

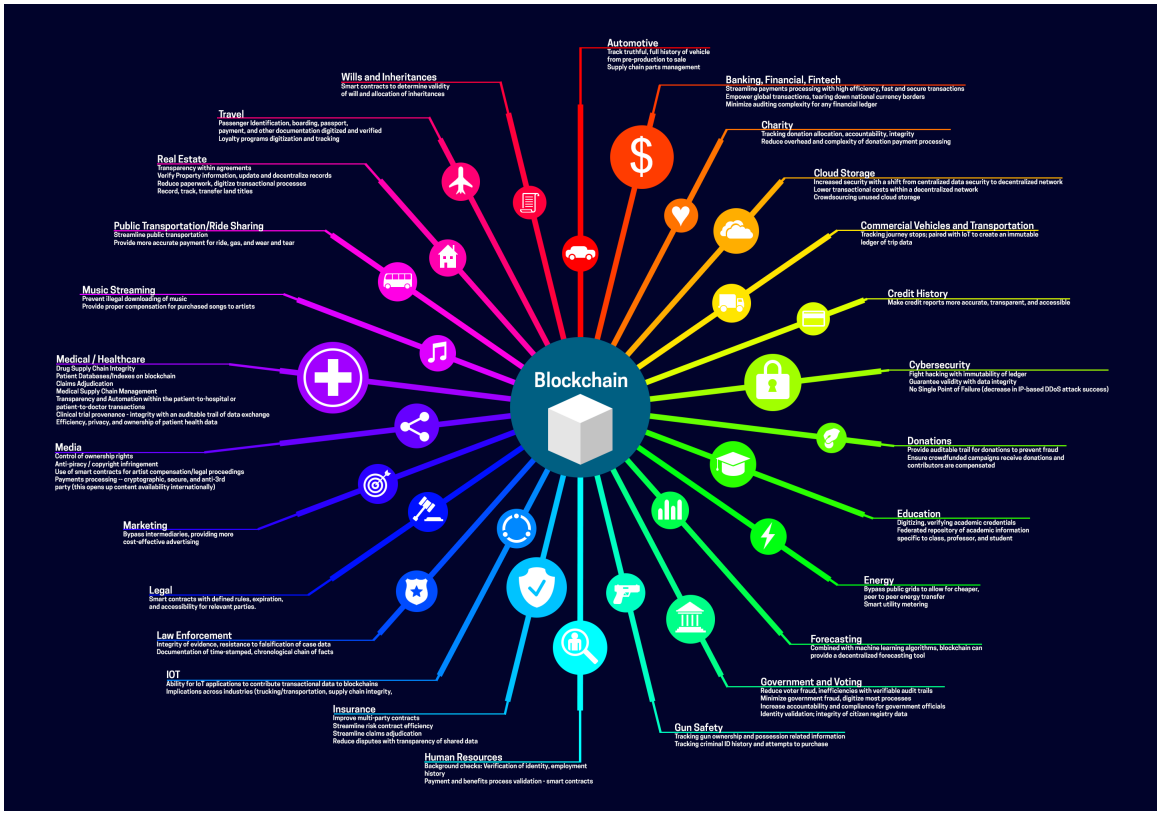

Figura 2.4: Applicazione delle blockchain nel mondo odierno, immagine presa da SlideShare

Capitolo 3

Tecnologie usate

3.1 Typescript

TypeScript[12] è un super-set di JavaScript[13] che aggiunge tipi statici e altre funzionalità allo standard JavaScript, rendendolo più robusto e mantenibile. Creato da Microsoft nel 2010, TypeScript è stato progettato per superare alcune delle limitazioni di JavaScript, offrendo agli sviluppatori strumenti per scrivere codice più sicuro e coerente. Con la sua crescente adozione e supporto da parte della community, TypeScript sta diventando una scelta popolare per lo sviluppo front-end e back-end.

Essendo un linguaggio di programmazione open-source che estende JavaScript, TypeScript aggiunge tipi statici e altre funzionalità che migliorano la leggibilità e la manutenibilità del codice. I tipi statici permettono di rilevare errori durante la fase di compilazione piuttosto che a runtime, riducendo così il rischio di bug e semplificando il processo di debugging. Questa caratteristica è particolarmente utile in progetti di grandi dimensioni, dove il controllo dei tipi può prevenire errori che altrimenti potrebbero diventare difficili da tracciare.

TypeScript viene compilato in JavaScript, il che significa che il codice TypeScript può essere eseguito in qualsiasi ambiente che supporta JavaScript. Questo rende TypeScript compatibile con tutte le librerie e framework JavaScript esistenti, mantenendo la portabilità e l'accessibilità del codice. Gli sviluppatori possono quindi beneficiare delle caratteristiche avanzate di TypeScript senza rinunciare all'ecosistema JavaScript.

Un altro vantaggio di TypeScript è la sua integrazione con gli strumenti di sviluppo moderni. Gli editor di codice, come Visual Studio Code, offrono un eccellente supporto per TypeScript, con funzionalità come il completamento automatico, il refactoring intelligente e la navigazione del codice, che migliorano significativamente la produttività degli sviluppatori. Inoltre, TypeScript supporta funzionalità avanzate come le interfacce, i moduli e le annotazioni di tipo, che permettono una migliore organizzazione e modularità del codice.

L'adozione di TypeScript è in costante crescita, non solo per i progetti di nuove applicazioni, ma anche per il refactoring di applicazioni esistenti. Molte aziende e progetti open-source hanno iniziato a migrare a TypeScript per sfruttare i suoi benefici, contribuendo così alla sua maturità e stabilità come linguaggio di programmazione. La sua capacità di migliorare la qualità del codice e la produttività degli sviluppatori lo rende una scelta sempre più diffusa per lo sviluppo di applicazioni moderne.

3.1.1 Utilizzo nel progetto finale

La scelta sull'utilizzo di typescript nel progetto è stato adottata solo per lo sviluppo di una semplice interfaccia grafica affinchè si potesse testare la comunicazione tra frontend e backend, lasciando delle linee guida per l'utilizzo dei canister di backend e come si dovrebbe instaurare la comunicazione con esso.

```
export async function createEvent() {
  const eventName = (document.getElementById("eventName") as HTMLInputElement)
    .value;
  const eventDescription = (
    document.getElementById("eventD") as HTMLInputElement
  ).value;
  // @ts-ignore
  const file = (document.getElementById("imageForEvent") as HTMLInputElement)
    .files[0];

  fetch(URL.createObjectURL(file))
    .then((response) => response.blob())
    .then((blobData) => {
      const reader = new FileReader();
      reader.onload = async (event) => {
        const arrayBuffer = event.target?.result as ArrayBuffer;
        await agent.fetchRootKey();
        await backend_webapp.create_event(eventName, eventDescription, {
          Blob: new Uint8Array(arrayBuffer),
        });
      };
      reader.readAsArrayBuffer(blobData);
    });
}
```

Figura 3.1: Esempio di utilizzo di typescript nel progetto

3.2 WebAssembly

WebAssembly[14] (Wasm) rappresenta un passo significativo avanti nella programmazione web, offrendo un nuovo modo per eseguire codice direttamente nel browser. A differenza degli approcci tradizionali che si affidano pesantemente al JavaScript, WebAssembly consente agli sviluppatori

di eseguire codice scritto in linguaggi come C, C++ e Rust, portando così un salto di qualità in termini di prestazioni e flessibilità.

WebAssembly è un formato binario di istruzioni progettato come obiettivo portatile per la compilazione di linguaggi di alto livello come C, C++ e Rust. Opera in modo simile al codice macchina ma è progettato per essere eseguito in modo efficiente dai moderni CPU. Il vantaggio chiave di WebAssembly rispetto al JavaScript è la sua velocità di esecuzione; il codice compilato in WebAssembly si esegue molto più velocemente rispetto al codice JavaScript equivalente, migliorando così le prestazioni delle applicazioni web complesse.

La scrittura di bytecode WebAssembly viene prima fatta in un linguaggio di alto livello come C o Rust che viene compilato in bytecode WebAssembly utilizzando una catena di compilatori. Il risultante bytecode è una rappresentazione compatta ed efficiente del codice originale, ottimizzata per un'ottima esecuzione nell'ambiente del browser. Quando una pagina web viene caricata, il modulo WebAssembly viene scaricato insieme alle risorse HTML, CSS e JavaScript. L'engine WebAssembly del browser interpreta il bytecode e lo esegue, consentendo all'applicazione di funzionare fluidamente e rapidamente.

3.2.1 Utilizzo nel progetto finale

L'utilizzo del bytecode WebAssembly è stato limitato alla compilazione degli smart contract scritti in Rust. Per quanto il suo utilizzo sia stato limitato, è risultata una conoscenza interessante in quanto alcune funzionalità di Rust sono state limitate, come ad esempio la generazione di numeri randomici, a causa della successivazione compilazione sul wasmtime, il runtime di WebAssembly.

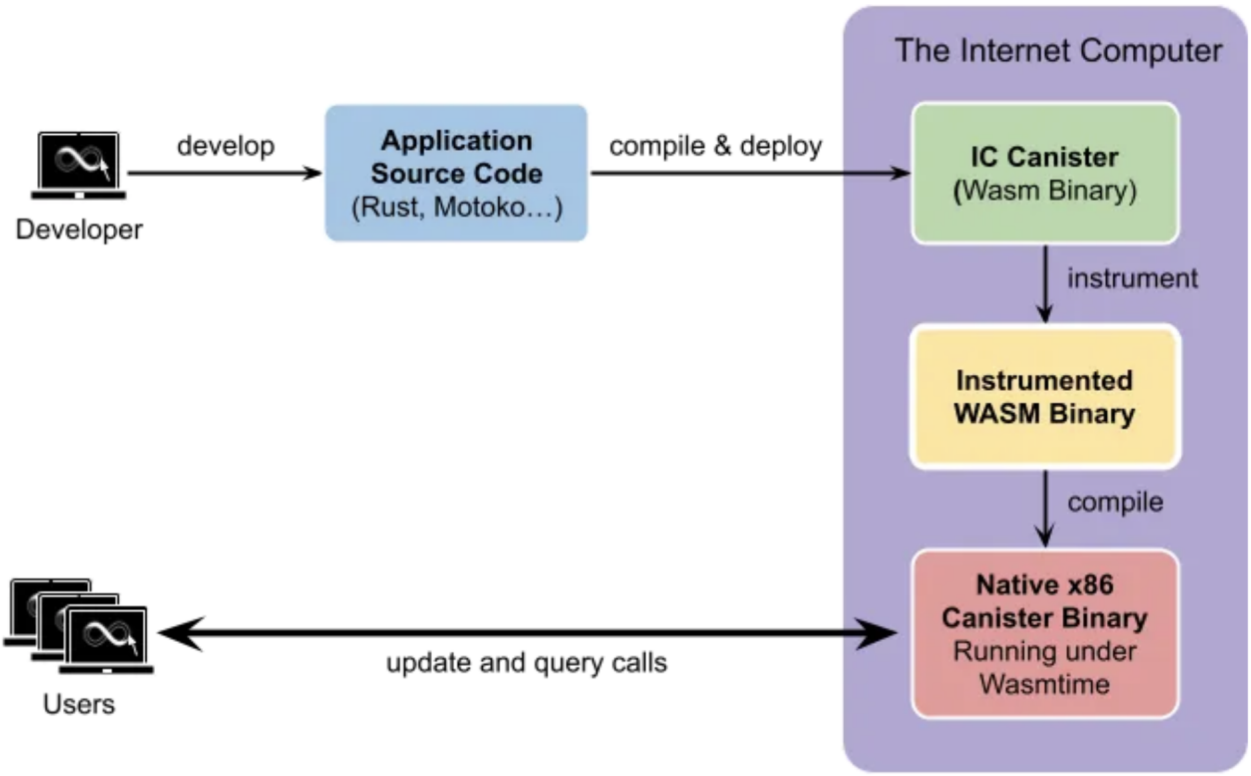

Figura 3.2: Come viene utilizzato il runtime di WebAssembly su ICP, immagine presa da Medium

3.3 Rust

Rust[15] è un linguaggio di programmazione funzionale, moderno e potente, progettato per essere sicuro, rapido e conciso. È stato sviluppato nel 2010 da Graydon Hoare, ex membro dello staff di Mozilla Research, con l'obiettivo di risolvere alcuni dei problemi comuni associati alla sicurezza e alla performance dei linguaggi di programmazione contemporanei. Rust combina elementi di linguaggi come C++ e Haskell, offrendo un ambiente di programmazione che favorisce la sicurezza e l'efficienza senza sacrificare la produttività.

Uno dei punti di forza principali di Rust è la sua enfasi sulla sicurezza della memoria. Grazie al sistema di proprietà e borrowing, Rust elimina molte delle categorie di bug legate alla gestione della memoria che affliggono i linguaggi come C e C++. Questo sistema permette di garantire che i puntatori siano sempre validi, evitando problemi di accesso concorrente e memory leaks, senza la necessità di un garbage collector. Questa caratteristica rende Rust particolarmente adatto per lo

sviluppo di sistemi embedded, software di sistema, e applicazioni dove le performance e la gestione efficiente delle risorse sono cruciali.

L'ecosistema di Rust è in rapida crescita, con una community attiva e numerose librerie e strumenti a disposizione degli sviluppatori. Il package manager e build system integrato, Cargo, semplifica la gestione delle dipendenze e la compilazione del codice, rendendo lo sviluppo con Rust più efficiente. Inoltre, Rust è noto per la qualità della sua documentazione e per l'approccio "documentation first", che assicura che gli sviluppatori abbiano accesso a risorse complete e ben strutturate.

Rust è adottato sempre più frequentemente in ambiti diversi, dalle applicazioni di sistema e di rete, alle applicazioni web e allo sviluppo di software embedded, questo seguirsi di vantaggi è stato alla base della scelta di adottare questo linguaggio alla base di questo progetto.

3.3.1 Vantaggi

- **Sicurezza**: Una delle caratteristiche distintive di Rust è la sua enfasi sulla sicurezza. Rust utilizza un sistema di **ownership** e **borrowing** che impedisce errori comuni come accessi non autorizzati, duplicazioni di memoria e violazioni di buffer. Questo sistema aiuta a prevenire molti tipi di bug che possono causare crash software o vulnerabilità di sicurezza.

- **Prestazioni**: Rust è progettato per essere un linguaggio performantico. La sua sintassi e le sue convenzioni di compilazione incoraggiano lo sviluppo di codice efficiente, minimizzando l'overhead e massimizzando l'utilizzo delle risorse hardware. Rust è particolarmente apprezzato per la sua velocità e la sua efficienza, tanto che è spesso usato per scrivere componenti critici in sistemi operativi, motori di database e altri software ad alte prestazioni.

- **Sistema di Tipi Robusto**: Rust possiede un sistema di tipi forte che aiuta a prevenire molti errori durante la fase di compilazione, migliorando la sicurezza e la stabilità del codice.

- **Flessibilità di Applicazione**: Rust è utilizzato in vari ambiti, dalle applicazioni di sistema e di rete, allo sviluppo web e di software embedded.

- **Compilazione e Tooling Avanzati**: Gli strumenti di compilazione di Rust sono avanzati e offrono un feedback rapido e dettagliato, facilitando il processo di sviluppo e debugging.

- **Scalabilità**: Rust è progettato per scalare bene sia in termini di complessità del progetto che di performance, rendendolo adatto per progetti di tutte le dimensioni.

3.3.2 Utilizzo nel progetto finale

Proprio per le sue due caratteristiche più importanti, la gestione della memoria e le sue prestazioni, è alla base del backend del nostro applicativo. Tutti i 3 smart contract alla base del progetto sono rigorosamente scritti in Rust seguento le convenzione vigenti.

3.4 ICP

La Blockchain ICP, nota anche come Internet Computer Protocol, è un progetto innovativo che mira a integrare la tecnologia blockchain direttamente nell'infrastruttura web globale. L'obiettivo principale dell'ICP è quello di portare la programmabilità del web e la decentralizzazione, tipiche delle blockchain, al livello di infrastruttura, permettendo agli sviluppatori di creare applicazioni web decentralizzate (dApps) senza dover ricorrere alle blockchain tradizionali come Ethereum o Bitcoin.

3.4.1 Come funziona

L'ICP utilizza un modello di rete peer-to-peer (P2P) per eseguire codice e memorizzare dati in modo decentralizzato. A differenza delle blockchain tradizionali, che eseguono principalmente transazioni finanziarie, l'ICP è progettato per supportare una vasta gamma di applicazioni web, compresi siti web, servizi cloud, e applicazioni mobile.

Questa blockchain, che si fa pioniera di un nuova architettura blockchain, introduce un nuovo paradigma per la costruzione di applicazioni web decentralizzate, offrendo una piattaforma che mira a rivoluzionare il modo in cui le applicazioni internet vengono create e distribuite, introducendo novità in molti aspetti dalle convenzionali blockchain, i campi in cui queste novità sono più marcate, sono i seguenti:

- **Architettura decentralizzata**:
  - **Canister**: A differenza delle tradizionali blockchain che utilizzano nodi pieni, la blockchain ICP utilizza "canister" come unità fondamentali di calcolo e storage. Un canister è un modulo software che contiene sia codice che stato, permettendo di eseguire logica computazionale e mantenere dati persistenti. Inoltre, i canister sono autonomi e possono comunicare tra loro attraverso messaggi, permettendo la creazione di sistemi complessi e interconnessi.
  - **Nodi**: sono le unità fondamentali che compongono l'infrastruttura fisica della rete ICP, questi nodi sono server distribuiti globalmente, gestiti da partecipanti indipendenti, che forniscono la potenza di calcolo e le risorse di storage necessarie per eseguire i canister.

I nodi lavorano insieme per formare subnet, che sono essenzialmente delle reti di nodi che collaborano per eseguire un gruppo di canister.

  - **Rete di nodi**: La rete ICP è composta da una vasta rete di nodi che ospitano i canister. Questi nodi sono gestiti da provider di infrastrutture indipendenti, assicurando una distribuzione geografica ampia e una resilienza alla censura.

- **Consensus e Sicurezza**:
  - **Chain-Key Cryptography**: La blockchain ICP utilizza un meccanismo di consenso chiamato Threshold Relay, che si basa sulla crittografia a chiave a catena. Questo metodo consente ai nodi di partecipare al processo di consenso in modo sicuro e efficiente, minimizzando la necessità di energia e migliorando la scalabilità.
  - **Sicurezza dei Dati**: I dati all'interno dei canister sono protetti attraverso meccanismi di crittografia robusti, garantendo che solo gli utenti autorizzati possano accedere alle informazioni sensibili.

- **Scalabilità e Performance**:
  - **Subnet**: La rete ICP è suddivisa in subnet, ciascuna delle quali può eseguire applicazioni e contratti intelligenti indipendentemente dalle altre. Questo design modulare permette alla rete di espandersi orizzontalmente, migliorando la scalabilità e la capacità di elaborazione.
  - **Efficienza Energetica**: A differenza di alcune blockchain che richiedono grandi quantità di energia per il mining, la blockchain ICP è progettata per essere energeticamente efficiente, riducendo drasticamente l'impatto ambientale.

- **Sviluppo di Applicazioni**
  - **Linguaggi di programmazione**: Oltre a poter sviluppare applicativi attraverso Motoko, linguaggio proprietario della Dfinity Foundation, è possibile sviluppare applicativi anche attraverso linguaggi come Rust, Python o Typescript.
  - **Interoperabilità**: La blockchain ICP supporta l'interoperabilità con altre reti blockchain e sistemi legacy, permettendo di integrare facilmente servizi esistenti e nuove applicazioni decentralizzate.

- **Governance**
  - **Neuron Network**: La governance della rete ICP è gestita attraverso un sistema di "neuroni", che sono essenzialmente token che rappresentano diritti di voto. I detentori di neuroni possono partecipare alle decisioni relative all'evoluzione della rete, inclusi aggiornamenti del protocollo e politiche economiche.

3.4.2 Architettura

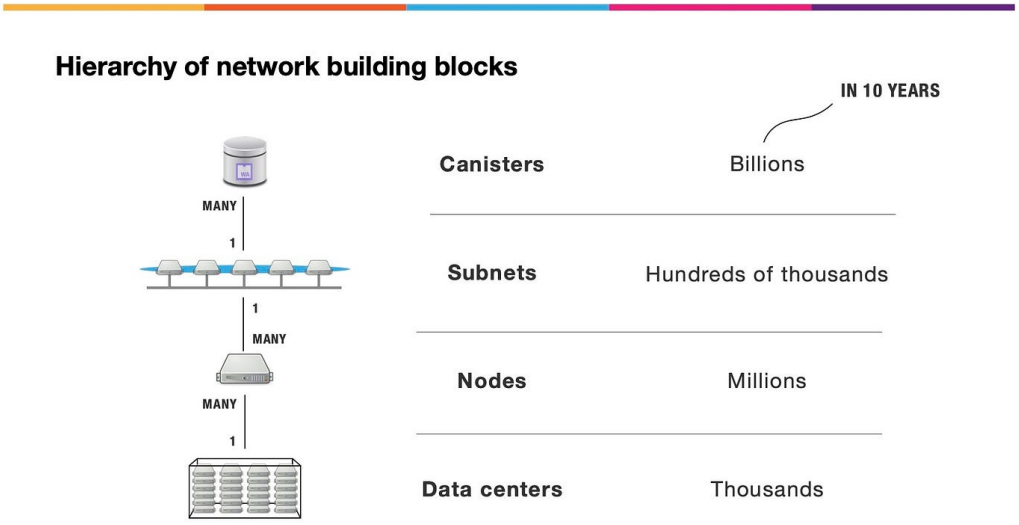

Figura 3.3: Struttura gerarchica di ICP, immagine presa da Medium

L'architettura fisica e topologica dell'Internet Computer Protocol (ICP) è progettata per massimizzare la scalabilità, la sicurezza e l'efficienza, supportando la visione di un "World Computer". Elementi chiave al suo interno sono i data center, i nodi, le subnet e i canister:

- **Data Centers**: L'ICP è ospitato in data centers situati in tutto il mondo. Questi data centers contengono i nodi che costituiscono la rete ICP.
- **Nodi**: I nodi sono server fisici che partecipano alla rete ICP. Ogni nodo esegue il protocollo ICP e contribuisce alla formazione e alla validazione della blockchain. I nodi sono selezionati dal Network Nervous System (NNS) per formare subnet.
- **Subnet**: Quando il NNS decide di creare una nuova subnet, seleziona un gruppo di nodi disponibili che hanno aderito all'IC ma non sono ancora stati allocati a nessuna subnet. Questi nodi iniziano a formare una nuova blockchain di subnet. L'ICP scala la sua capacità orizzontalmente creando nuove subnet che ospitano canister aggiuntivi, analogamente a come l'infrastruttura cloud tradizionale scala aggiungendo nuove macchine.
- **Canister**: I canister sono le unità fondamentali di calcolo e storage nell'ICP. Contengono sia il codice che lo stato, permettendo di eseguire logica computazionale e mantenere dati persistenti. I canister sono distribuiti attraverso le subnet. Ogni subnet può ospitare numerosi canister, e i canister possono comunicare tra loro attraverso le subnet.

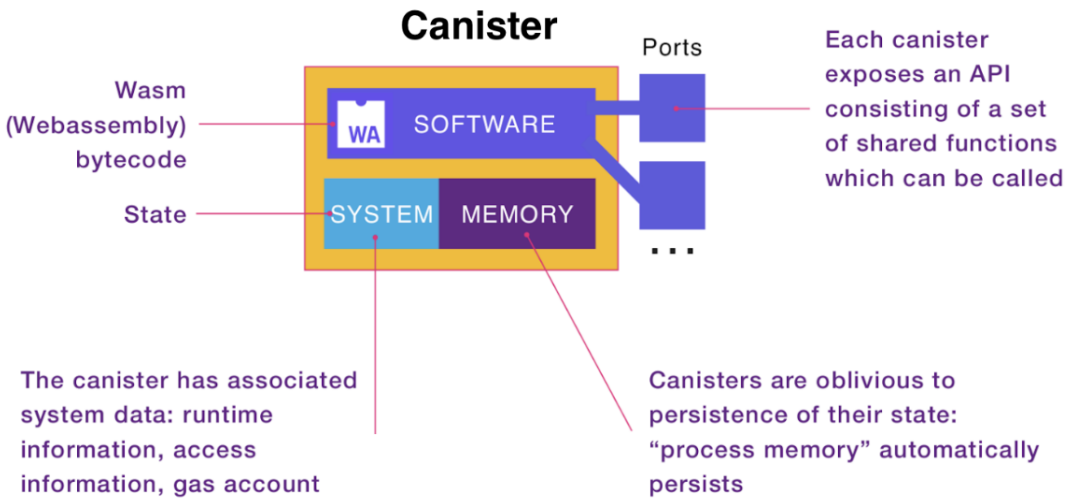

Figura 3.4: Architettura di un canister, immagine presa da CoinBureau

3.4.3 Perchè funziona

L'ICP funziona perché integra la decentralizzazione e la programmabilità direttamente nell'infrastruttura web, eliminando la necessità di intermediari e centralizzazioni. Questo approccio offre diversi vantaggi:

- **Scalabilità**: Grazie alla natura P2P e all'utilizzo di un motore di runtime ottimizzato, l'ICP può gestire un gran numero di transazioni e applicazioni simultaneamente senza sovraccaricare la rete.
- **Sicurezza**: Utilizzando tecniche di crittografia avanzate e un modello di rete resistente agli attacchi, l'ICP protegge i dati e le applicazioni dagli attacchi esterni.
- **Interoperabilità**: L'ICP è progettato per lavorare insieme a altre tecnologie web, come HTTP e HTTPS, facilitando l'integrazione con le applicazioni esistenti.

Inoltre i prezzi di storage sono veramente competitivi, in alcuni casi si può addirittura definire come più conveniente rispetto a soluzioni molto utilizzate, come ad esempio AWS.

Data Transfer	Internet Computer	AWS
Data IN cost	$9.56 / GB	Free
Data OUT costs	$0.0002733 / GB	$0.07 / GB
IN cost for 1 TB	$9,560	Free
OUT cost for 1 TB	$273.30	$70,000
Total cost for 1 TB	**$9,833.30**	**$70,000**

Internet Computer and AWS costs to save 1 TB of data and then read it 1,000 times.

Figura 3.5: Rapporto dei costi di ICP confrontati con AWS, immagine presa da X

Oltre agli ottimi prezzi lato storage, ICP utilizza un innovativo "Reverse Gas Fee model" che ottimizza i costi delle transazioni e dell'esecuzione degli smart contracts. Questo modello riduce la volatilità dei costi operativi per gli sviluppatori, permettendo una pianificazione finanziaria più prevedibile. Inoltre, il gas cycle model di ICP incentiva l'efficienza del codice, contribuendo a mantenere bassi i costi operativi e a garantire la sostenibilità economica delle applicazioni decentralizzate sulla piattaforma.

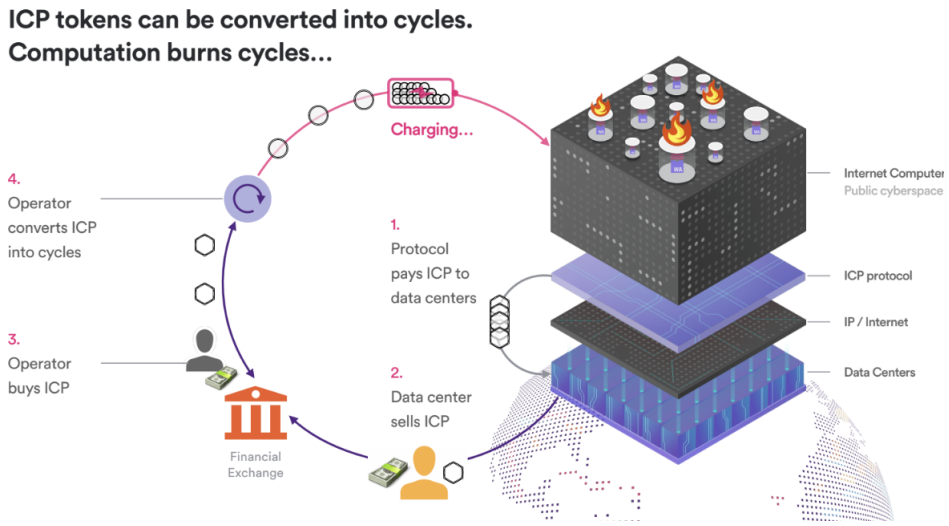

Figura 3.6: Modello "Reverse Gas Fee" introdotto da ICP, immagine presa da CoinHustle

3.4.4 Impatto e applicazioni

L'ICP ha il potenziale di rivoluzionare il modo in cui gli sviluppatori costruiscono e distribuiscono applicazioni web. Alcune delle applicazioni previste includono:

- **Applicazioni web decentralizzate (Dapp)**: Siti web e applicazioni che possono essere ospitate e distribuite direttamente sulle reti ICP, senza la necessità di un server centrale.
- **Serivizi cloud decentralizzati**: Piattaforme di hosting e storage che utilizzano l'ICP per fornire servizi cloud in modo decentralizzato e sicuro.
- **Contratti intelligenti**: Implementazione di contratti intelligenti direttamente nell'infrastruttura web, consentendo transazioni automatizzate e sicure.

3.4.5 Utilizzo nel progetto finale

La blockchain ICP è stata utilizzata come struttura alla base del progetto proprio per i suoi vantaggi di poter ospitare un applicativo interno in maniera completamente decentralizzata, per le sue elevate prestazioni e per la sicurezza che garantisce, senza considerare che i costi sono molto ridotti e competitivi.

Capitolo 4

Stakeshare

4.1 Introduzione

Stakeshare è un'applicazione che facilita la gestione delle quote di partecipazione in diverse iniziative, sfruttando la notarizzazione su blockchain per garantire trasparenza e sicurezza. In Rust e compilata in WebAssembly per l'esecuzione su Internet Computer Protocol (ICP). Questa soluzione consente agli utenti di registrare e scambiare quote di proprietà o partecipazione in maniera digitale e decentralizzata, con la certificazione legale tramite standard NFT (ICRC7). Attraverso l'uso di smart contracts, Stakeshare automatizza gli accordi tra le parti e la distribuzione dei proventi, rendendo il processo di gestione delle quote più semplice, sicuro e conforme alle normative

4.2 Analisi

Prima di iniziare lo sviluppo dell'applicazione, è stata condotta un'attenta fase di analisi e ricerca, durante la quale sono state formulate le idee su come dovesse essere strutturata StakeShare e sulle funzionalità che dovesse offrire. La parte di strutturazione del progetto è stata particolarmente interessante, poiché sono state affrontate diverse discussioni sulla composizione e suddivisione della logica dei canister, in collaborazione con il gruppo di Dfinity.

Oltre alla creazione della struttura del progetto, una fase particolarmente lunga è stata la ricerca delle tecnologie migliori da utilizzare. Questo compito è stato reso complesso dal fatto che molte tecnologie sono ancora in fase di sviluppo (come icrc7) e non sono ben documentate; alcune di esse non vengono nemmeno menzionate ufficialmente. Tuttavia, grazie a un forte interesse da parte della community, è relativamente semplice continuare a scoprire nuove tecnologie, convenzioni e standard che facilitano lo sviluppo in diversi settori. Durante questa fase di ricerca sono stati scoperti nuovi standard e nuove tecnologie come, ad esempio, quelli utilizzati nel progetto:

- **Stable Memory**: forma di memoria persistente utilizzata nei canister.
- **Icrc7**: standard per token non fungibili.

4.3 Sviluppo

L'applicativo è stato diviso in 3 canister per tutta le sezione di backend, ciascuno dei 3 ha funzioni completamente diverse, il primo è il backend vero e proprio dell'applicazione, il secondo serve per gestire una singola collezione di NFT associata ad un proprietario, mentre il terzo è un canister che implementa la factory per la creazione delle collezioni di NFT agli utenti. Inoltre esiste un canister che conterrà il frontend dell'applicazione, che è stato sviluppato solo per finalità di testing.

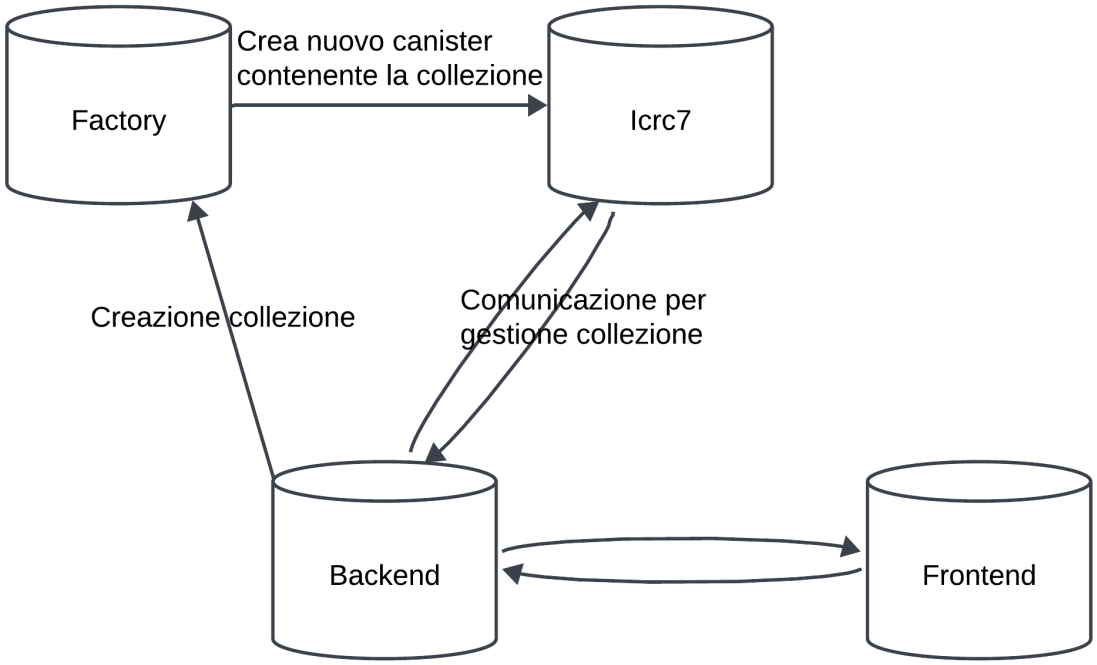

Figura 4.1: Comunicazione tra i canister dell'applicazione

4.3.1 Internet Identity

L'Internet Identity (II) di Internet Computer Protocol (ICP) rappresenta un sistema di autenticazione digitale altamente innovativo, progettato per migliorare significativamente la sicurezza, la

semplicità e il rispetto della privacy degli utenti. Questo sistema elimina la necessità di gestire password, offrendo un metodo avanzato e sicuro per identificarsi durante l'interazione con applicazioni decentralizzate (dApp) e servizi sulla piattaforma Internet Computer.

Il principale vantaggio dell'Internet Identity risiede nella sua capacità di sostituire le password tradizionali con un sistema di autenticazione basato su dispositivi di identità sicuri, come ad esempio i portachiavi hardware, i dispositivi biometrici o le chiavi crittografiche. Quando un utente desidera accedere a una dApp o a un servizio online, il sistema Internet Identity utilizza questi dispositivi per generare in modo sicuro delle credenziali di autenticazione uniche. Queste credenziali sono temporanee e specifiche per la sessione, il che significa che non possono essere riutilizzate o rubate da malintenzionati.

Il processo di autenticazione tramite Internet Identity si svolge come segue:

- **Registrazione dell'Identità**: Durante la fase iniziale, l'utente registra un dispositivo di identità sicuro con il sistema Internet Identity. Questo dispositivo può essere un portachiavi hardware, uno smartphone con funzionalità biometriche o una chiave di sicurezza FIDO2. Una volta registrato, il dispositivo viene associato in modo sicuro all'account dell'utente.

- **Generazione delle Credenziali**: Quando l'utente tenta di accedere a un servizio, il dispositivo di identità genera una coppia di chiavi crittografiche uniche e temporanee. La chiave privata rimane sul dispositivo e non viene mai condivisa, mentre la chiave pubblica viene utilizzata per verificare l'identità dell'utente.

- **Autenticazione Sicura**:Il servizio o la dApp verifica la chiave pubblica e consente l'accesso solo se la chiave privata corrispondente è valida. Questo processo è estremamente sicuro perché le chiavi crittografiche non possono essere indovinate, replicate o rubate facilmente come le password.

Questo metodo elimina numerosi problemi associati alle password tradizionali. Non ci sono password da ricordare o gestire, riducendo il rischio di phishing, furto di credenziali e attacchi di forza bruta. Inoltre, poiché le chiavi crittografiche sono specifiche per ciascuna sessione e non vengono mai trasmesse o archiviate sui server, la privacy dell'utente è ulteriormente garantita.

Lo sviluppo dell'Internet Identity all'interno di applicazioni basate su ICP ha comportato sfide interessanti, soprattutto in termini di configurazione e testing. Sebbene implementare questo metodo di login in un ambiente di produzione sulla mainnet sia relativamente semplice grazie al canister di autenticazione fornito da Dfinity (https://identity.ic0.app/), la fase di sviluppo e testing richiede un approccio diverso.

Per il testing locale, gli sviluppatori devono utilizzare un canister di autenticazione deployato in locale, fornito dalla casa produttrice. Questo canister può essere importato nel progetto e modificato secondo le necessità specifiche dello sviluppatore. La gestione del canister in un ambiente

locale permette di testare le funzionalità dell'Internet Identity in modo sicuro e controllato prima di distribuirle su scala più ampia sulla mainnet.

```
    "internet_identity": {
      "candid": "https://github.com/dfinity/internet-identity/releases/latest/download/internet_identity.did",
      "frontend": {},
      "remote": {
        "id": {
          "ic": "rdmx6-jaaaa-aaaaa-aaadq-cai"
        }
      },
      "type": "custom",
      "wasm": "https://github.com/dfinity/internet-identity/releases/latest/download/internet_identity_dev.wasm.gz"
    }
  },
```

Figura 4.2: Codice nel json per il deploy del canister dell'internet identity

4.3.2 Stable Memory

La memoria stabile dell'Internet Computer Protocol (ICP), nota anche come "Stable Memory", è una componente chiave del sistema ICP, che permette di memorizzare dati a lungo termine, separata dalla memoria heap del canister.Questa funzionalità è particolarmente utile per mantenere i dati attraverso gli aggiornamenti dei canister, poiché i dati memorizzati nella memoria stabile non vengono cancellati o rimossi quando il canister viene fermato o aggiornato. La memoria stabile è conservata durante tutto il processo, mentre qualsiasi altro stato WebAssembly viene scartato, Questo rende ICP una delle poche blockchain su cui è possibile aggiornare uno smart contract già deployato.

Per utilizzare la memoria stabile, è necessario indicare quali porzioni dei dati del canister si desidera persistere attraverso gli aggiornamenti nel codice del canister. Di default, la memoria stabile di un canister è vuota, ma può raggiungere un limite massimo di 400 GiB se il subnet su cui è distribuito il canister lo consente.

```rust
use crate::common::types::{Event, Group, RequestResult};

type Memory = VirtualMemory<DefaultMemoryImpl>;

thread_local! {
    static MEMORY_MANAGER: RefCell<MemoryManager<DefaultMemoryImpl>> =
        RefCell::new(MemoryManager::init(DefaultMemoryImpl::default()));

    static COLLECTIONS: RefCell<StableBTree<String, Group, Memory>> = RefCell::new({
        StableBTree::init(MEMORY_MANAGER.with(|m| m.borrow().get(MemoryId::new(0))))
    });

    static EVENT_COLLECTIONS: RefCell<StableBTree<String, Event, Memory>> = RefCell::new({
        StableBTree::init(MEMORY_MANAGER.with(|m| m.borrow().get(MemoryId::new(1))))
    });

    static TOKEN_COUNTER: RefCell<StableCell<u128, Memory>> = RefCell::new({
        StableCell::init(MEMORY_MANAGER.with(|m| m.borrow().get(MemoryId::new(2))), 0).unwrap()
    })
}
```

Figura 4.3: Utilizzo delle stable structure all'interno del progetto

Affinchè si possano salvare strutture create da noi all'interno della stable memory è necessario che implementino il trait `Storable` così che una struttura sia serializzabile e deserializzabile.

```rust
impl Storable for Group {
    fn to_bytes(&self) -> Cow<[u8]> {
        Cow::Owned(
            serde_json::to_string(self) Result<String, Error>
                .expect(msg: "failed to serialize to bytes") String
                .as_bytes() &[u8]
                .to_vec(),
        )
    }

    fn from_bytes(bytes: Cow<[u8]>) -> Self {
        let from_str: Result<Group, Error> = serde_json::from_str(
            String::from_utf8(bytes.to_vec()) Result<String, FromUtf8Error>
                .expect(msg: "failed to serialize from bytes") String
                .as_str(),
        );
        from_str.expect(msg: "failed to serialize from bytes")
    }

    const BOUND: Bound = Bound::Unbounded;
}
```

Figura 4.4: Implementazione del trait `Storable` per la struttura `Group`

La stable memory è strutturata come un array di byte, dove ogni byte può essere indirizzato e modificato singolarmente. Questo fornisce un alto grado di flessibilità per la gestione dei dati. L'accesso alla stable memory è effettuato tramite un'interfaccia API che permette operazioni di lettura e scrittura.

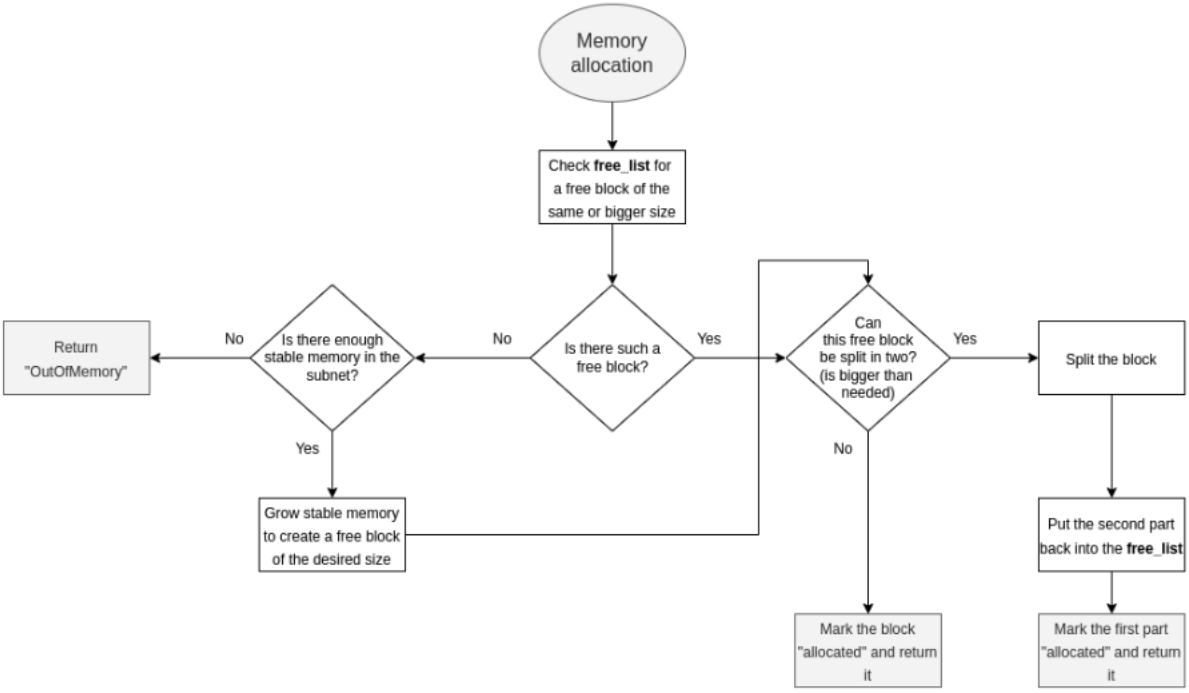

Figura 4.5: Funzionamento interno della stable memory, allocazione

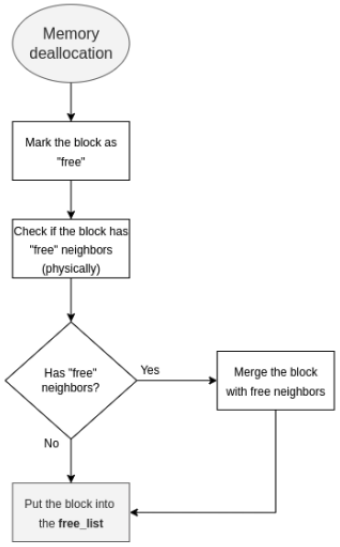

Figura 4.6: Funzionamento deallocazione della memoria

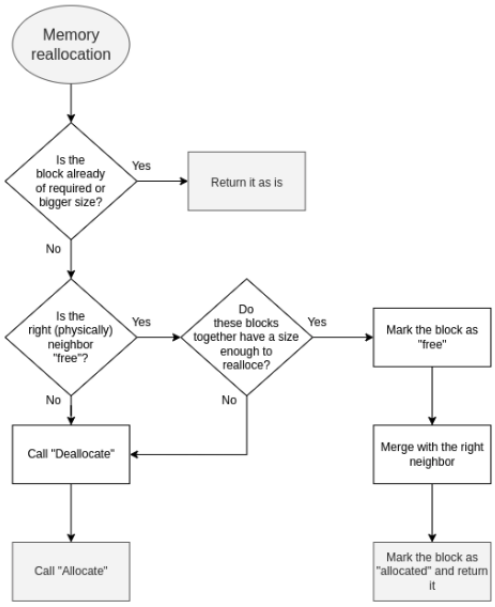

Figura 4.7: Funzionamento riallocazione della memoria

4.3.3 Chiamate Intra-canister

Essendo necessario permettere la comunicazione tra i diversi canister creati al fine di suddividere il più possibile l'incarico di ciascun canister, è stato cruciale implementare chiamate asincrone. Queste chiamate asincrone hanno reso possibile la comunicazione tra i canister, garantendo una maggiore modularità e scalabilità del sistema. Tuttavia, l'uso di chiamate asincrone ha comportato una riduzione delle prestazioni rispetto alle chiamate sincrone, poiché ogni interazione tra canister richiede tempo aggiuntivo per la gestione delle richieste e delle risposte. Nell'ecosistema dell'Internet Computer Protocol (ICP), le chiamate tra canister si dividono principalmente in due categorie: `query` e `update`.

```rust
pub async fn assign_nft_for_event(
        // updating minting authority, default is on the factory canister
        update_minting_authority(factory_id: factory_canister_id, owner.clone(), icrc7_canister_id).await;
        match mint_icrc7_for_user(
            owner.clone(),
            icrc7_canister_id,
            icrc7_name: Some(icrc7_name),
            icrc7_description.clone(),
            icrc7_logo: Some(event_id.clone()),
        ) impl Future<Output = Result<…, …>>
        .await
        {
            Ok(v: u128) => token_ids.push(v),
            Err(err: MintError) => {
                return RequestResult::new(
                    code: 499,
                    message: format!(
                        "Error minting NFT for user {} : {:?}",
                        member.name.clone(),
                        err
                    ),
                    body: vec![],
                )
            }
        };
    }
```

Figura 4.8: Esempio di come sono svolte le chiamate tra canister a livello di codice

Queste due tipologie di API esposte dai canister hanno caratteristiche molto diverse tra di loro:

- `query`: Le chiamate query sono utilizzate per recuperare dati dai canister in modo rapido e non permanente. Queste chiamate sono estremamente efficienti perché vengono eseguite in modo sincrono e non influenzano lo stato permanente del canister. Sono ideali per operazioni di lettura che richiedono bassa latenza, come la visualizzazione di informazioni agli utenti.
  - **efficienza**: Poiché non modificano lo stato, le query sono molto veloci.
  - **sicurezza**: Le chiamate query non necessitano di consenso tra i nodi, il che le rende meno sicure per operazioni critiche.
  - **uso tipico** : Recupero di dati, visualizzazione di informazioni.
- `update`: Le chiamate update, d'altra parte, sono utilizzate per modificare lo stato del canister. Queste chiamate sono più lente rispetto alle query perché devono essere propagate attraverso la rete, registrate e confermate dai nodi per garantire la coerenza e la persistenza dei dati.
  - **affidabilità**: Le update sono garantite dal consenso della rete, assicurando che ogni modifica sia replicata in modo sicuro.

– **latenza**: Sono più lente a causa del processo di consenso e della scrittura nello stato permanente.
– **uso tipico**: Modifiche ai dati, transazioni finanziarie, operazioni critiche.

Le chiamate tra canister sono un punto cruciale per quanto riguarda le prestazioni del nostro applicativo in quanto vengono classificate come chiamate update, nonstante non sempre vanno a modificare lo stato del canister. La soluzione è stata adottare un nuovo meccanismo di chiamate intra-canister, chiamato `composite-query`, utili qualora si è sicuri che nessun metodo chiamato vada a modificare lo stato del canister.

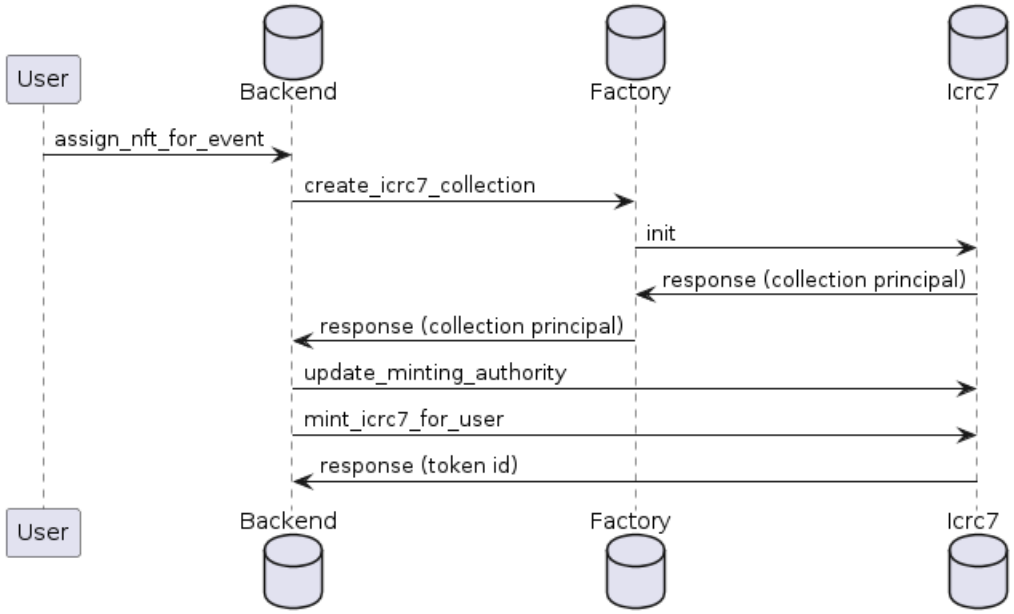

Figura 4.9: Diagramma di sequenza delle chiamate tra canister

4.3.4 ICRC7

ICRC7[16] è un protocollo in fase di sviluppo nell'ambito dell'Internet Computer Protocol (ICP) destinato a standardizzare la creazione e la gestione dei token non fungibili (NFT) sulla rete ICP. Questo protocollo mira a fornire una base tecnica per l'emissione, il trasferimento e la gestione degli NFT, assicurando interoperabilità e funzionalità avanzate tra le applicazioni decentralizzate (dApp) costruite sulla piattaforma ICP. L'acronimo "ICRC" sta per "Internet Computer Request for Comments", e rappresenta uno standard creato dal gruppo di lavoro di Internet Computer. Gli

standard ICRC possono essere utilizzati per creare qualsiasi cosa sull'ICP, non solo token fungibili come gli standard ICRC-1[17] e ICRC-2[18] per i token fungibili.

Lo standard ICRC-7 è progettato per essere uno standard minimale che permetta di distribuire una collezione di NFT sull'Internet Computer Protocol (ICP). In una collezione di NFT, ogni NFT può avere informazioni di metadati uniche. Questi metadati possono includere elementi distintivi come un'immagine unica, tratti o tag specifici, o una descrizione dettagliata che caratterizza l'NFT.

Lo standard ICRC-7 implementa diversi metodi API aggiuntivi rispetto allo standard DIP721[19], che è l'equivalente di ICP dello standard ERC721[20] di Ethereum. Tra le funzionalità avanzate offerte da ICRC-7 vi sono metodi di query in batch, che consentono di recuperare informazioni su più NFT contemporaneamente, metodi di aggiornamento in batch, che permettono di modificare i metadati di diversi NFT in una sola operazione, e metodi di approvazione della collezione, che facilitano la gestione e il controllo delle collezioni di NFT.

Nonostante l'avanzamento significativo, lo standard ICRC-7 non è ancora in produzione poiché è ancora in fase di bozza. Questo implica che, sebbene sia stato ampiamente discusso e considerato pronto per il voto nell'ambito del Network Nervous System (NNS) di ICP, non è ancora disponibile per l'uso nelle applicazioni reali. La bozza del documento di specifica di ICRC-7 fornisce ulteriori dettagli sulle funzionalità previste e le implementazioni proposte, delineando una visione chiara di come questo standard potrà evolvere una volta finalizzato.

Per lo sviluppo del progetto è stata necessaria l'implementazione di questo protocollo. Grazie al supporto della community di Dfinity e alle linee guida fornite dalle loro repository, il processo è stato molto soddisfacente. La community ha giocato un ruolo cruciale nel fornire feedback, suggerimenti e risorse, facilitando l'adozione e l'adattamento dello standard ICRC-7 alle specifiche esigenze del progetto.

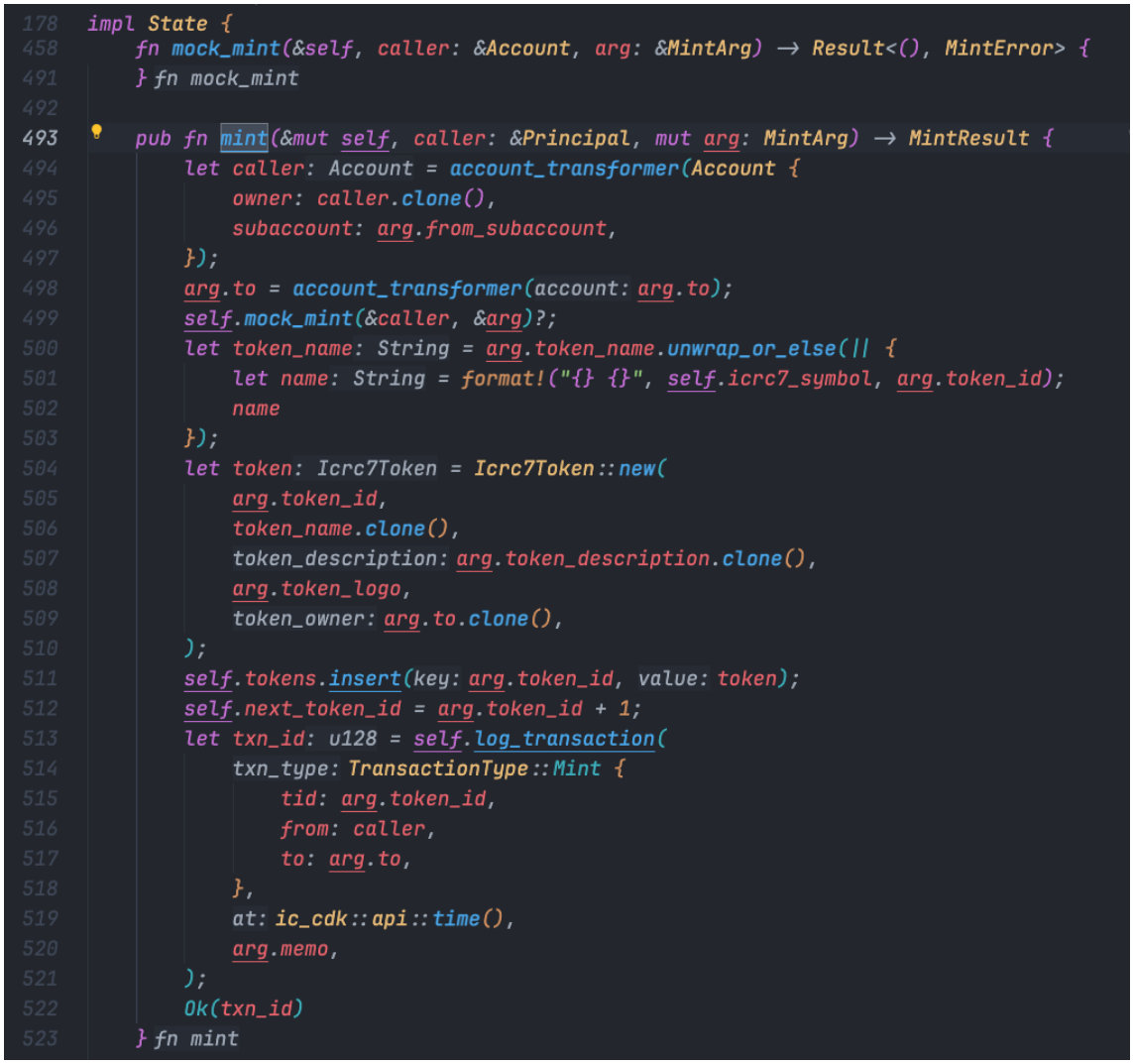

Figura 4.10: Esempio dell'implementazione della funzione di mint

4.3.5 NFT

Essendo che ogni canister che implementa `ICRC7` rappresenta una collezione di token (NFT), la gestione dei metadati di questi ultimi ha richiesto un approccio particolare. Per ottimizzare l'uso dello spazio di memoria e ridurre i costi, è stata implementata una relazione `1 a N` tra metadati e NFT. In pratica, più NFT possono condividere lo stesso metadato, minimizzando la quantità di dati duplicati memorizzati nel sistema.

Questa ottimizzazione viene realizzata identificando univocamente ogni evento associato a un metadato specifico. Nei metadati del token viene quindi inserito solo l'identificativo dell'evento, che funge da riferimento al metadato condiviso. In questo modo, invece di memorizzare separatamente i metadati per ogni NFT, si memorizza un singolo set di metadati che può essere referenziato da più token, risparmiando così spazio e costi di archiviazione.

Questa strategia ha un impatto sulla fase di recupero delle informazioni, rendendola leggermente più complessa. Per ottenere i metadati effettivi di un NFT, è necessario eseguire una query che utilizza l'identificativo dell'evento per recuperare il metadato associato. Tuttavia, dato che queste operazioni di recupero sono eseguite tramite query, le prestazioni rimangono elevate, garantendo un'esperienza utente fluida.

In sintesi, questo approccio offre un equilibrio ottimale tra efficienza e costo. Le prestazioni delle query assicurano che il recupero dei metadati sia rapido e senza intoppi, mentre l'uso condiviso dei metadati riduce significativamente lo spazio di memoria richiesto. Di conseguenza, il sistema può gestire un numero maggiore di NFT senza incorrere in costi aggiuntivi o problemi di scalabilità. Questo modello di gestione dei metadati rappresenta un esempio di come l'architettura di ICP possa essere utilizzata per creare soluzioni innovative e efficienti nel contesto degli NFT.

```
14  pub async fn assign_nft_for_event(
35      let mut token_ids: Vec<u128> = vec![];
36      for member: Member in members {
37          let icrc7_name: String = format!(…
40          );
41          let owner: Principal = string_to_principal(member.internet_identity.clone());
42          let icrc7_canister_id: Principal = create_icrc7_collection(…
49          .await;
50          // updating minting authority, default is on the factory canister
51          update_minting_authority(factory_id: factory_canister_id, owner.clone(), icrc7_canister_id).await;
52          match mint_icrc7_for_user(
53              owner.clone(),
54              icrc7_canister_id,
55              icrc7_name: Some(icrc7_name),
56              icrc7_description.clone(),
57              icrc7_logo: Some(event_id.clone()),
58          ) impl Future<Output = Result<…, …>>
59          .await
60          {…
73          };
74      }
75      RequestResult::new(code: 200, message: "All NFTs has been minted".to_string(), body: token_ids)
76  } fn assign_nft_for_event
77
```

Figura 4.11: Esempio di quando viene creato il token associato per il completamento dell'evento, dove si passa l'ID dell'evento

4.4 Risultati

4.4.1 Internet Identity

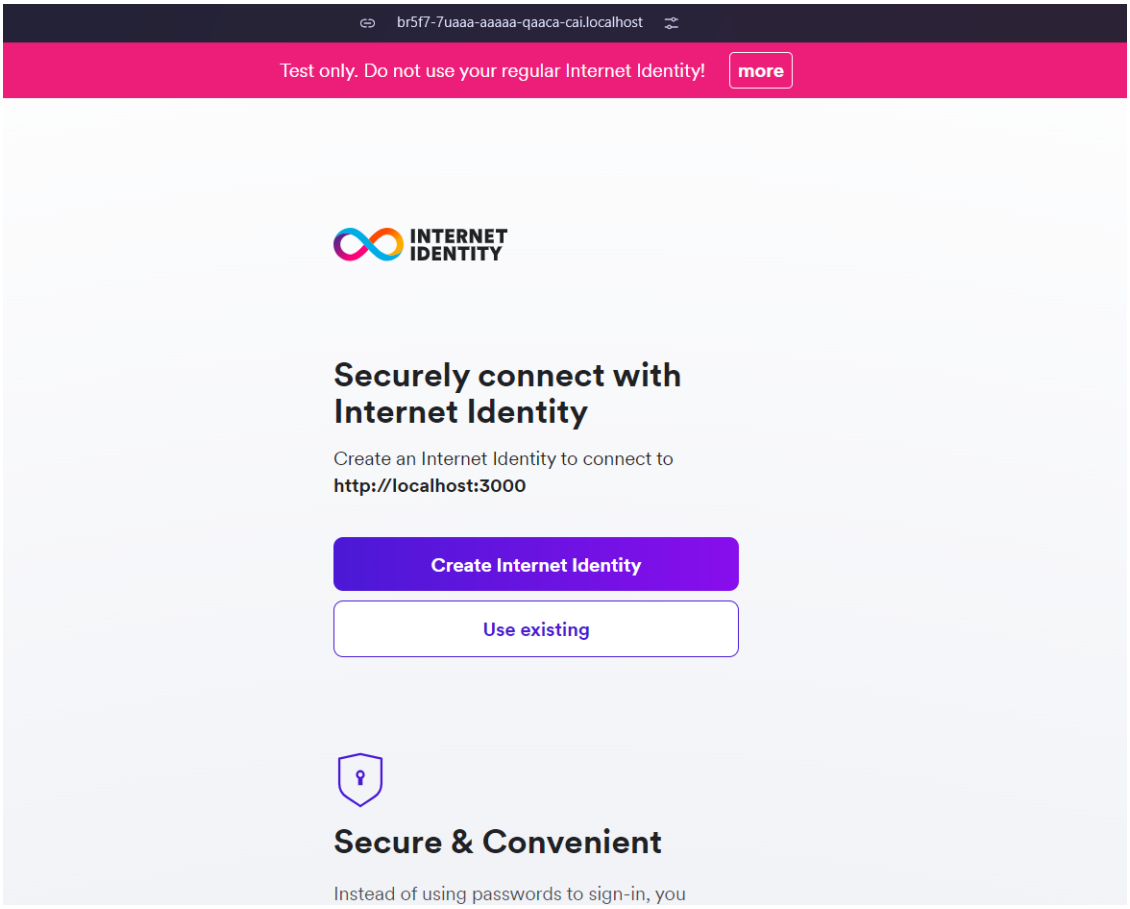

Figura 4.12: Interfaccia di login

L'implementazione dell'Internet Identity (II) di Internet Computer Protocol (ICP) all'interno del nostro progetto ha prodotto risultati significativi in termini di sicurezza, facilità d'uso e rispetto della privacy degli utenti. In particolare, i principali risultati ottenuti sono i seguenti:

- L'eliminazione delle password ha ridotto il rischio di attacchi di phishing e violazioni di dati.
- Il nuovo sistema di autenticazione ha semplificato l'esperienza dell'utente e il tempo medio necessario per completare il processo di login è diminuito significativamente.

L'implementazione dell'Internet Identity di ICP ha dimostrato di essere una soluzione efficace per migliorare la sicurezza e la facilità d'uso delle applicazioni decentralizzate. I risultati ottenuti durante il testing su localnet hanno confermato la validità del sistema e la sua adattabilità alle esigenze specifiche del progetto.

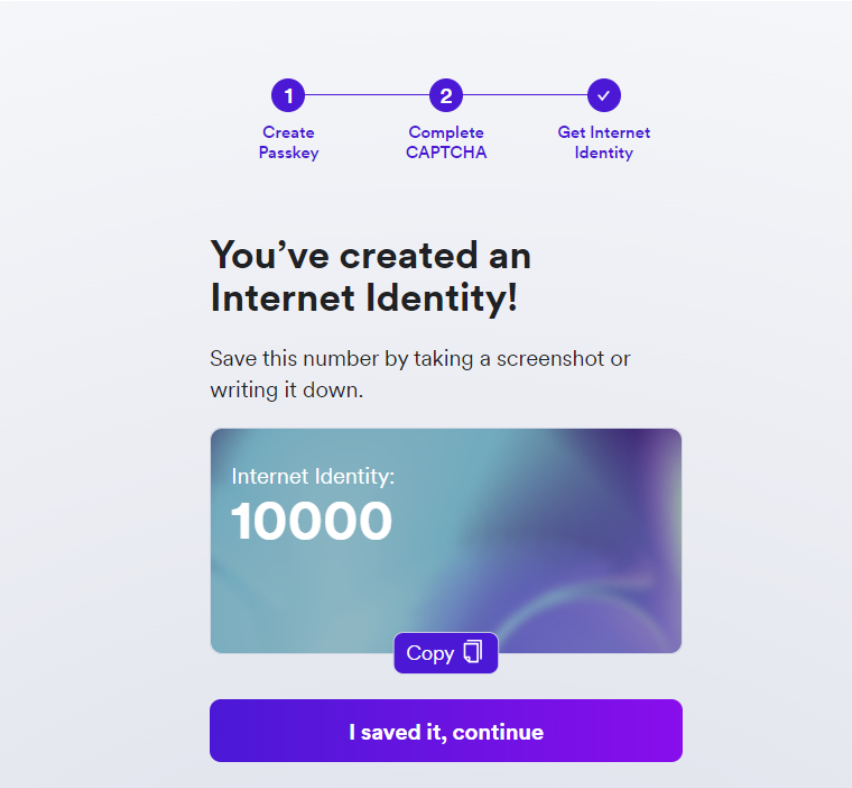

Figura 4.13: Interfaccia avvenimento della creazione dell'internet identity

4.4.2 Gestione dati

Il punto centrale del progetto è stato lo sviluppo di un backend solido ed efficiente. Per raggiungere questo obiettivo, uno degli aspetti più importanti è stato la costruzione di un'interfaccia esposta robusta e di un formato dei dati che permettesse agli sviluppatori di comprendere facilmente come utilizzare i dati e interpretare correttamente gli eventi che si sono verificati. Una chiara e ben definita interfaccia, insieme a un formato dei dati intuitivo, ha reso possibile una gestione più efficiente delle comunicazioni tra le diverse componenti del sistema, facilitando la diagnosi di eventuali problemi e migliorando l'interoperabilità complessiva.

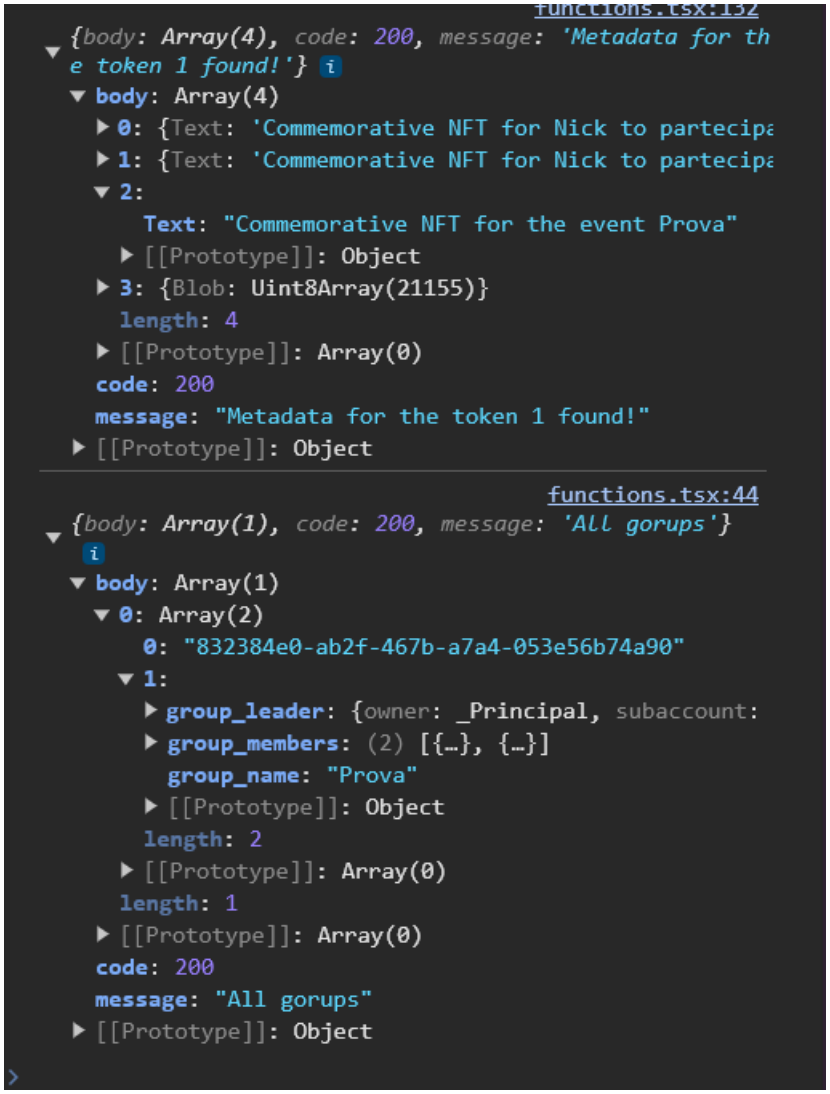

Figura 4.14: Esempio ricezione dati da frontend

Per la creazione del formato dati da scambiare tra il backend e il frontend della mia applicazione, mi sono ispirato al modello di messaggio proposto dallo standard HTTP[21]. Questo modello è stato scelto in quanto offre un'ottima comprensibilità, facilitando la struttura e la gestione delle comunicazioni tra le diverse componenti del sistema. Grazie alla chiarezza e alla standardizzazione dei messaggi HTTP, è possibile garantire una facile interpretazione e manipolazione dei dati, migliorando l'efficienza del processo di sviluppo e la robustezza delle interazioni tra il backend e il frontend.

Capitolo 5

Conclusioni

Il percorso di sviluppo di Stakeshare è stato un viaggio stimolante e arricchente, pieno di sfide e scoperte tecnologiche. Nonostante, magari, all'inizio l'obiettivo non fosse chiaro, è risultato interessante e impegnativo affacciarsi al mondo del Web3, comprendendone i vantaggi e le sfide che si pongono, capendo come mai stia avendo un grande sviluppo. La scelta di Rust come linguaggio di programmazione si è rivelata fondamentale per garantire performance e affidabilità, mentre la compilazione in WebAssembly ha permesso di sfruttare al meglio le capacità dell'Internet Computer Protocol (ICP).

L'integrazione dello standard per NFT ICRC-7 per la certificazione legale delle quote è stata una delle tappe più significative, assicurando che le transazioni fossero non solo sicure, ma anche conformi alle normative. Ogni fase del progetto, dalla progettazione iniziale all'implementazione dei smart contracts, ha comportato un approfondimento delle tecnologie blockchain e una crescente consapevolezza delle loro potenzialità. L'automazione degli accordi e della distribuzione dei proventi tramite smart contracts è stata una sfida tecnica che ha richiesto creatività e precisione, ma ha anche evidenziato quanto queste soluzioni possano semplificare processi complessi, mostrando le loro effettive potenzialità.

Guardando indietro, lo sviluppo di Stakeshare è stato un'opportunità straordinaria per mettere in pratica le competenze acquisite durante il mio percorso accademico, nonostante molte di esse non siano risultate necessarie, e per esplorare nuove frontiere tecnologiche. Questo progetto ha mostrato quanto sia in continuo movimento il panorama informatico globale. La costruzione di un'applicazione basata su ICP ha richiesto un continuo aggiornamento e adattamento alle nuove tecnologie emergenti, evidenziando l'importanza della formazione continua nel campo della tecnologia.

In conclusione, il viaggio attraverso lo sviluppo di Stakeshare non solo ha portato alla realizzazione di un'applicazione innovativa per la gestione delle quote di partecipazione, ma ha anche rappresentato una significativa crescita personale e professionale. La mia comprensione delle tec-

nologie blockchain e delle loro applicazioni pratiche si è notevolmente ampliata, arricchendo il mio bagaglio di competenze tecniche. Reputo sicuramente questo progetto interessante sotto molti punti di vista, soprattutto per quanto ha contribuito alla mia crescita come sviluppatore. La capacità di affrontare e superare le sfide tecniche incontrate durante lo sviluppo di Stakeshare mi ha preparato per affrontare futuri progetti con maggiore sicurezza e competenza. Questo viaggio mi ha mostrato quanto sia vasto e dinamico il campo della tecnologia blockchain, e sono entusiasta di continuare a esplorare le sue infinite possibilità

Bibliografia

[1] *Internet Computer Protocol*. URL: https://internetcomputer.org/.

[2] *BitTorrent website*. URL: https://www.bittorrent.com/it.

[3] *Bitcoin website*. URL: https://bitcoin.org/it/.

[4] *IPFS website*. URL: https://ipfs.tech/.

[5] *Blockchain wikipedia*. URL: https://it.wikipedia.org/wiki/Blockchain.

[6] *Bitcoin White Paper*. URL: https://bitcoinwhitepaper.co/.

[7] *Icon Coin Offering*. URL: https://www.investopedia.com/terms/i/initial-coin-offering-ico.asp#:~:text=Initial%20coin%20offerings%20(ICOs)%20are,have%20yielded%20returns%20for%20investors..

[8] *Central Bank Digital Currency*. URL: https://www.investopedia.com/terms/c/central-bank-digital-currency-cbdc.asp.

[9] *Non-Fungible Token wiki*. URL: https://it.wikipedia.org/wiki/Non-fungible_token.

[10] *Proof of Work*. URL: https://it.wikipedia.org/wiki/Proof-of-work.

[11] *Proof of Stake*. URL: https://it.wikipedia.org/wiki/Proof-of-stake.

[12] *Typescript Reference Documentation*. URL: https://www.typescriptlang.org/.

[13] *Javascript W3C introduction*. URL: https://www.w3schools.com/js/default.asp.

[14] *WebAssembly Documentation*. URL: https://webassembly.org/.

[15] *Rust Documentation Book, The Rustbook*. URL: https://doc.rust-lang.org/book/.

[16] *ICRC-7 linee guida*. URL: https://github.com/dfinity/ICRC/blob/icrc_7_and_37/ICRCs/ICRC-7/ICRC-7.md.

[17] *ICRC-1 token standard*. URL: https://internetcomputer.org/docs/current/references/icrc1-standard.

[18] *ICRC-2 token swap standard*. URL: https://internetcomputer.org/docs/current/references/samples/motoko/icrc2-swap/.

[19] *DIP-721 standard*. URL: https://internetcomputer.org/docs/current/references/samples/rust/dip721-nft-container/.

[20] *ERC-721 standard*. URL: https://ethereum.org/it/developers/docs/standards/tokens/erc-721/.

[21] *Standard HTTP*. URL: https://it.wikipedia.org/wiki/Hypertext_Transfer_Protocol#Esempi_di_messaggi_HTTP.

Ringraziamenti

Ora è il momento dei ringraziamenti. Non saranno troppo formali, ma desidero comunque esprimere la mia gratitudine in maniera decente.

In primis, desidero ringraziare i miei genitori, Diego ed Elena, che mi hanno sostenuto ampiamente durante il mio percorso di studi. In secondo luogo, vorrei ringraziare mio fratello Mattia, che riesce ogni giorno a ricordarmi quanto io sia più intelligente con tutto quello che dice. Messi da parte gli insulti, lo desidero ringraziare per tutto quello che ha fatto per me. Inoltre, desidero ringraziare tutta la mia famiglia, nonni e zii, per il supporto che mi hanno fornito durante questo periodo.

Un grazie gigante va a tutti gli amici che mi hanno sostenuto in questo percorso di studi, a partire dai membri del mio gruppo, in particolare coloro con cui ho stretto un legame più profondo, come Luca, Ettore, Giovanni, Mounir e Abdou. Un ringraziamento particolare va a Marco, che mi è sempre stato accanto e con cui condivido grandissimi ricordi, e un altro ringraziamento speciale a Margherita, che mi ha dato fantastici consigli e mi ha fatto molto ridere durante tutto questo periodo.

Ringrazio anche tutti i miei compagni, attuali e passati, di pallavolo, che mi hanno permesso di staccare dallo studio nel migliore dei modi.

Un ringraziamento va anche a tutte le amiche architette e biomediche, da chi ha contribuito di più a chi meno.

Infine, desidero ringraziare tutti i ragazzi di Quinck, che mi hanno aiutato durante lo sviluppo del progetto di tesi e mi hanno insegnato un sacco di cose, e soprattutto il mio relatore Stefano Ferretti, che si è dimostrato gentilissimo e super disponibile.