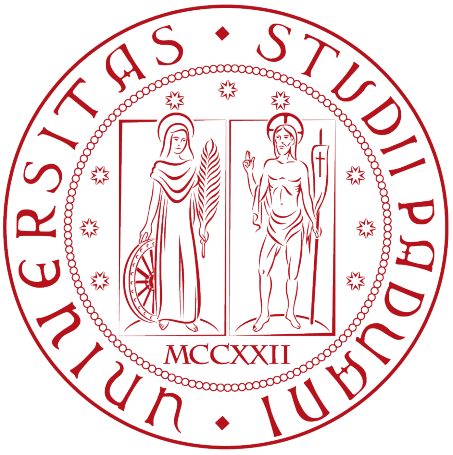
Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica



Titolo della tesi

Tesi di Laurea Triennale

Relatore

Prof.ssa Ombretta Gaggi

Laureando

Gabriel Rovesti

Anno Accademico 2022-2023

Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

— Oscar Wilde

Dedicato a ...

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, della durata di circa trecento ore, dal laureando Gabriel Rovesti presso l'azienda Sync Lab, sede di Padova. Gli obbiettivi da raggiungere erano molteplici.

Da scrivere

In primo luogo era richiesto lo sviluppo di ... In secondo luogo era richiesta l'implementazione di un ... Tale framework permette di registrare gli eventi di un controllore programmabile, quali segnali applicati Terzo ed ultimo obbiettivo era l'integrazione ...

"But poetry, beauty, romance, love, these are what we stay alive for."

— Dead Poets Society, Robin Williams

Ringraziamenti

Innanzitutto, vorrei esprimere la mia gratitudine alla Prof.ssa Ombretta Gaggi, relatrice della mia tesi, per l'aiuto e il sostegno fornitomi durante la stesura del lavoro.
Desidero ringraziare con affetto mia mamma per il sostegno, il grande aiuto e per essermi stata vicina in ogni momento durante gli anni di studio.
Vorrei inoltre ringraziare i molti amici, di corso, di università e di vita, che mi hanno aiutato e sostenuto in questi anni e per le tante conoscenze che hanno spinto ulteriormente la mia voglia di fare e di conoscere.

Padova, Aprile 2023 Gabriel Rovesti

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Introduzione al contesto applicativo.
Esempio di utilizzo di un termine nel glossario Application Program Interface (API).
Esempio di citazione in linea *Manifesto Agile*. URL: http://agilemanifesto.org/iso/it/.
Esempio di citazione nel pie' di pagina citazione[1]

1.1 L'azienda e motivazioni della scelta

Sync Lab è una *software house* italiana nata a Napoli nel 2002 che, grazie ad una progressiva maturazione delle sue competenze in ambito applicativo e tecnologico, è riuscita a diventare un punto di riferimento per le aziende che intendono innovare i propri processi di business, trasformandosi in un *System Integrator*. L'azienda si è fatta notare sul mercato grazie alla proposta di vari prodotti software, sviluppati all'interno del suo laboratorio di ricerca e sviluppo, conquistando importanti fette di mercato in ambito nazionale nei settori mobile, di sicurezza e videosorveglianza. Ad oggi, Sync Lab conta più di 150 clienti e oltre 200 dipendenti disolcati nelle proprie sedi. Queste sono situate a Napoli, Milano, Verona, Roma e Padova. L'obiettivo principale dell'azienda è quello di fornire soluzioni tecnologiche innovative e di qualità che possano soddisfare le esigenze dei propri clienti, garantendo un servizio di consulenza e assistenza continuo e di alto livello.

Il mio contatto con l'azienda è avvenuto autonomamente, tramite l'invio di una candidatura spontanea in base alle aziende presenti nella mappa di StageIt dell'anno 2022. Incuriosito dalle tematiche proposte e dai progetti presenti, ho incontrato subito la massima disponibilità del mio attuale tutor, l'ingegner Fabio Pallaro, che mi ha fornito da suito un supporto valido per la stesura del piano di lavoro e per la definizione degli obiettivi dello stage. Da parte sua e dell'azienda, vi è stata subito massima disponibilità e flessibilità nell'organizzazione del mio studio teorico e per la realizzazione del progetto, dando una guida sulle possibili tecnologie e lasciandomi approfondire in autonomia le tematiche che più mi interessavano.

Figura 1.1: Logo azienda Sync Lab

[1] Daniel T. Jones James P. Womack. *Lean Thinking, Second Editon.* Simon & Schuster, Inc., 2010.

1.2 Introduzione al progetto e idea dello stage

Da scrivere

1.3 Way of Working e strumenti

L'azienda Sync Lab adotta un modello di sviluppo Agile, con l'obiettivo di monitorare e controllare lo sviluppo del progetto in modo flessibile e continuo, suddividendo le attività in piccoli incrementi e con una collaborazione asincrona e distribuita. In particolare, il modello di sviluppo adottato è Scrum, che prevede la suddivisione del progetto in sprint, ovvero periodi di tempo di durata fissa, in cui vengono pianificate le attività da svolgere e i relativi obiettivi da raggiungere. Al termine di ogni sprint, viene effettuato su base settimanale un incontro interno approfondito con il tutor aziendale, per discutere lo stato di avanzamento del progetto e le attività da svolgere per il successivo sprint. L'obiettivo del modello è dare maggiore importanza al ciclo di vita del e dei processi correlati, piuttosto che al prodotto finale, con l'obiettivo di migliorare la qualità del prodotto stesso. Inoltre, grazie alla collaborazione con altri stagisti con progetti basati sugli stessi concetti e in generale con il team di sviluppo, è stato possibile condividere conoscenze e competenze, discutere problemi e trovare soluzioni comuni, con un approccio incrementale e iterativo.

Gli strumenti utilizzati per lo sviluppo del progetto sono stati i seguenti:

- ***Trello***, per la gestione delle attività e dei task da svolgere, condiviso con il tutor aziendale e verificato dallo stesso ad ogni incremento. All'interno di questo strumento è possibile monitorare le attività attraverso delle schede, simili a dei *post-it*, differenziando le attività per specifiche colonne di avanzamento. Le diciture riportate dalle colonne sono generalmente come segue:
  - **Backlog**: che contiene attività in corso di svolgimento, comprendenti schede con obiettivi di massima dello stage e delle sue singole parti e periodi;
  - **In corso**, ossia attività in esecuzione e da realizzare entro la fine dello *sprint*;
  - **In verifica**, ossia attività in attesa di verifica e considerate concluse, in attesa di essere spostate nella colonna *Terminati* a seguito della *sprint review*;
  - **Terminati**, cioè attività completate e verificate da parte del tutor aziendale.

  Un esempio di utilizzo di Trello è riportato in Figura 1.2.

- ***Google Calendar***, per la pianificazione delle attività e degli incontri con il tutor aziendale, condiviso con il tutor stesso e con gli altri stagisti e dipendenti di Sync Lab, capendo così e in anticipo le attività da svolgere e gli incontri da effettuare;

- ***Visual Studio Code***, IDE utilizzato per la scrittura del codice sorgente e dei file di configurazione, con l'aggiunta di alcuni *plugin* per la formattazione del codice e per la gestione dei file di configurazione del progetto. All'interno di questo, è stato realizzato lo studio di esempi di codice e la realizzazione dell'intero progetto, con l'aggiunta di un sistema di *linting* per la formattazione del codice e di un sistema di *debugging* per il controllo del flusso di esecuzione del codice. Sempre tramite questo strumento, è stata scritto la tesi, attraverso l'utilizzo di un *plugin* per la scrittura in LaTeX;

- ***Diagrams.net***, per la realizzazione e la creazione delle immagini del documento, utilizzati per la documentazione e la strutturazione delle sezioni;

- ***StarUML***, per la realizzazione dei diagrammi Unified Modeling Language (UML) del documento, quindi per i diagrammi dei casi d'uso e dei diagrammi delle classi;

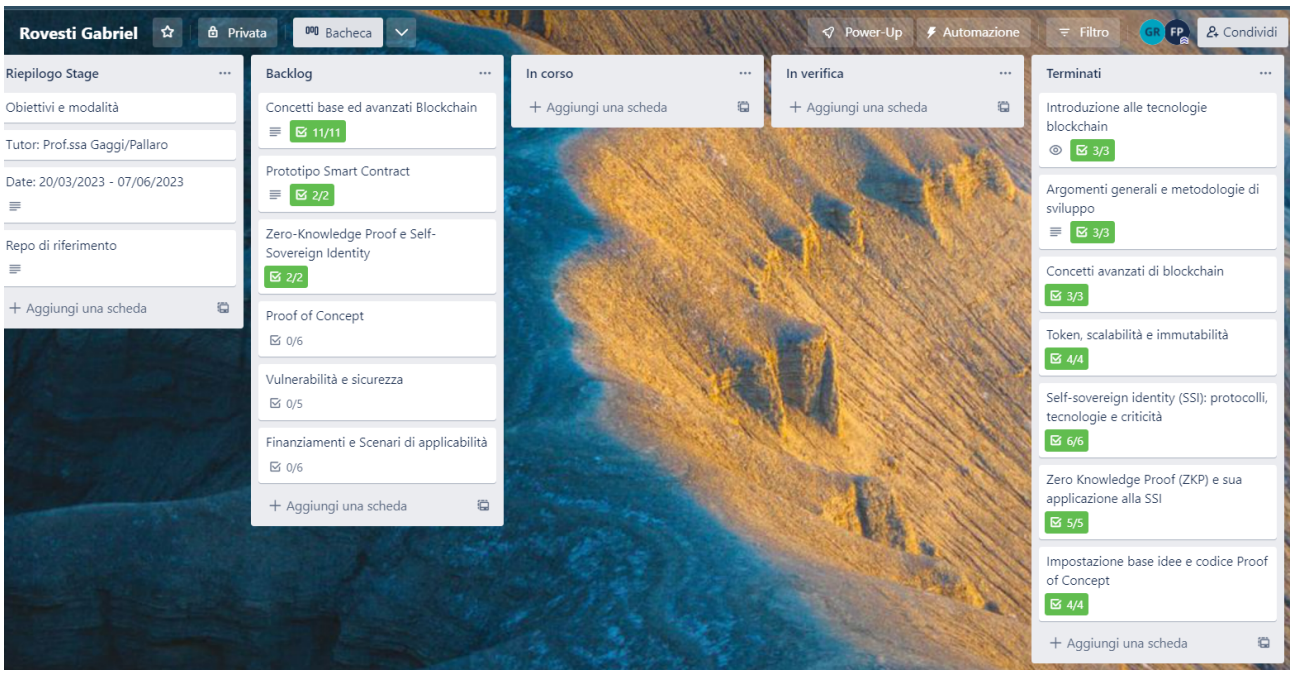


Figura 1.2: Esempio di utilizzo di Trello

- ***GitHub***, utilizzato internamente per la gestione del codice sorgente e dei file di configurazione, condiviso con il tutor aziendale e verificato dallo stesso ad ogni incremento. In particolare, è stato utilizzato il sistema di controllo versione Git, che permette di tenere traccia delle modifiche effettuate ai file e di gestire le varie versioni del codice e degli appunti interni da me realizzati, per una migliore organizzazione e condivisione delle informazioni. Inoltre, è stato possibile condividere il codice sorgente con gli altri stagisti e dipendenti di Sync Lab, per garantire il versionamento e la condivisione delle modifiche effettuate.
- ***Discord***, social network utilizzato per la comunicazione interna tra i dipendenti di Sync Lab, tramite la gestione di vari canali vocali e testuali divisi per argomento e per sedi interne aziendali. La suddivisione in gruppi e canali ha permesso di comunicare in modo efficace e veloce, condividendo informazioni e risorse utili per lo svolgimento del progetto, sia a livello di risorse didattiche che per la risoluzione di eventuali dubbi e problemi. Inoltre, è stato possibile comunicare con gli altri stagisti in modo semplice e veloce, condividendo conoscenze e competenze e discutendo problemi e soluzioni comuni.

1.4 Organizzazione del testo

1.4.1 Scrittura del documento

Il secondo capitolo descrive le tecnologie studiate e utilizzate per lo sviluppo del progetto.

Il terzo capitolo approfondisce ...

Il quarto capitolo approfondisce ...

Il quinto capitolo approfondisce ...

Il sesto capitolo approfondisce ...

Nel settimo capitolo descrive ...

1.4.2 Convenzioni tipografiche

Riguardo la stesura del testo, relativamente al documento sono state adottate le seguenti convenzioni tipografiche:

- gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento;
- per la prima occorrenza dei termini riportati nel glossario viene utilizzata la seguente nomenclatura: *parola*[g];
- i termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere *corsivo*.

Capitolo 2

Tecnologie di interesse

In questa sezione viene presentata una panoramica di base delle tecnologie oggetto del mio tirocinio, al fine di descrivere in modo chiaro e conciso i concetti di base e le caratteristiche principali delle tecnologie utilizzate nel progetto di stage e oggetto di studio autonomo e autodidatta.

2.1 Blockchain: concetti base

2.1.1 Introduzione

La blockchain è una tecnologia che permette di memorizzare dati in maniera decentralizzata e distribuita. Essa è una struttura dati che si comporta come un registro distribuito, salvando le informazioni in modo sicuro ed immutabile. La struttura è stata introdotta nel 2008 da Satoshi Nakamoto, che ha pubblicato il suo white paper *Bitcoin: A Peer-to-Peer Electronic Cash System*. Nel 2009 è stato pubblicato il primo software open source per la blockchain, Bitcoin, che ha permesso di creare una moneta digitale decentralizzata.

A tal fine, non si devono confondere le cosiddette criptovalute con la blockchain. Di fatto, quest'ultima è solo la struttura che permette lo scambio di beni di qualsiasi tipo, in modo sicuro, registrato ed immutabile. Una criptovaluta è invece una moneta digitale, che può essere scambiata con altre monete digitali o con beni fisici. Essendo lo standard blockchain *open source*, è possibile crearne di nuove con moltà facilità.

Normalmente, viene utilizzata per memorizzare transazioni finanziarie, ma può essere utilizzata per memorizzare qualsiasi tipo di informazione. Un altro suo nome è *distributed ledger technology* (DLT), che indica che le sue informazioni sono registrate come su un libro mastro, in cui le singole componenti della rete, definite nodi, possono accedere e modificare i dati, stabilendo se questi sono validi o meno.

2.1.2 Blocco

I dati delle blockchain sono strutturati in singoli blocchi, ciascuno contenente uno specifico set di informazioni. Ogni blocco è collegato al precedente tramite un hash, che ne garantisce l'immutabilità. I blocchi sono collegati in una catena, che viene aggiornata ogni volta che viene aggiunto un nuovo blocco. Nello specifico, possiamo dettagliare una struttura formata da:

- blocchi di dati;
- nonce, un numero generato casualmente alla creazione del blocco;
- l'hash del blocco precedente;
- il numero della transazione;
- *timestamp* di generazione del blocco (data e ora).

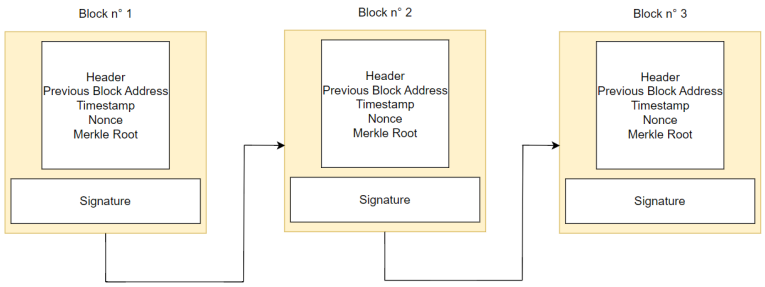


Figura 2.1: Blockchain vista come blocchi a catena

Per verificare l'integrità dei dati memorizzati, vengono usate delle strutture dati chiamati *Merkle trees*, in cui ogni foglia rappresenta l'hash della transazione. Le foglie vengono poi raggruppate in coppie, e l'hash di ogni coppia viene calcolato e memorizzato in un nodo superiore. Il processo si ripete, fino a raggiungere la radice dell'albero, che rappresenta l'hash di tutte le transazioni contenute nel blocco.

2.1.3 Transazione

Una transazione all'interno di una blockchain comporta il trasferimento di beni digitali, che possono essere valute, token, o qualsiasi altro tipo di informazione. In questo senso, possiamo individuare vari componenti del processo di transazione:

- gli utenti, che avviano le transazioni firmandole digitalmente con la propria chiave privata;
- i *miners*, che attraverso un processo specifico definito come *mining*, verificano la validità delle transazioni e le includono nel blocco successivo.
- i nodi, che convalidano i blocchi di transazioni inviati dai miners prima che vengano aggiunti alle blockchain.

Nello specifico, possiamo descrivere una transazione in questo modo:

1. l'utente avvia la transazione creando una firma digitale utilizzando la propria chiave privata. La firma dimostra che l'utente ha il diritto di inviare i beni;
2. la transazione viene trasmessa alla rete di nodi o computer che eseguono il software della blockchain. Ogni nodo riceve la transazione e la aggiunge a un pool di transazioni non confermate;
3. i nodi della rete convalidano la transazione per assicurarsi che il mittente abbia fondi sufficienti per completare la transazione e che questa sia conforme alle regole del protocollo blockchain;
4. una volta che un numero sufficiente di nodi ha convalidato la transazione, questa viene aggiunta a un nuovo blocco di transazioni, insieme ad altre transazioni convalidate di recente;
5. il blocco di transazioni viene aggiunto alla blockchain in un processo chiamato mining. L'estrazione comporta la risoluzione di complesse equazioni matematiche per creare un nuovo blocco, il che richiede una grande potenza di calcolo;
6. una volta aggiunto il nuovo blocco alla blockchain, la transazione viene considerata confermata e i beni vengono trasferiti dall'indirizzo del mittente a quello del destinatario. La transazione è ora registrata in modo permanente sul libro mastro della blockchain, che può essere visualizzato e verificato da chiunque abbia accesso alla rete;

2.1.4 Wallet

Un *wallet*, detto anche *portafoglio*, è un software che permette di memorizzare e gestire le chiavi private e pubbliche, e di inviare e ricevere transazioni. In particolare, possiamo più propriamente definirli portachiavi, in

quanto non contengono realmente i beni digitali, ma le chiavi utilizzate per accedervi. L'utente dispone in ogni momento di:

- una chiave pubblica, usata per inviare messaggi e ricevere pagamenti. È un codice univoco che identifica l'utente;
- una chiave privata, usata per firmare i messaggi e per accedere ai propri beni digitali. È un codice segreto che deve essere conservato in modo sicuro.

Ogni wallet dispone di una frase segreta, che contiene tutte le informazioni necessarie per recuperare ed accedere ai fondi del proprio portachiavi.

Inoltre, dispone di un proprio indirizzo, matematicamente derivato dalla stessa chiave pubblica mediante l'operazione di *hashing*, con una lunghezza di 160 bit. Ciascuno è *pseudonimo*, in quanto non appartiene nello specifico ad una persona, ma non è completamente anonimo.
È importante tenere la chiave privata in un luogo sicuro e non condividerla con nessuno, in quanto è l'unica cosa che garantisce l'accesso ai propri fondi. Distinguiamo due tipi di wallet:

- *hot wallet*, che sono i portafogli online, dunque più vulnerabili al rischio di *hacking*;
- *cold wallet*, che sono i portafogli offline, quindi considerati più sicuri, in quanto si collegano ad Internet principalmente per effettuare le transazioni.

2.1.5 Mining

Il processo di *mining* consente di creare nuovi blocchi sulla catena, al fine di convalidare le transazioni e ottenere nuove criptovalute come ricompensa per il proprio 'sforzo'. Questo obiettivo viene raggiunto attraverso un processo chiamato *consenso*, che prevede la risoluzione di complessi puzzle matematici utilizzando la potenza di calcolo. Il miner che riesce a risolvere il puzzle prima degli altri, vince il diritto di aggiungere il blocco alla blockchain.

Questo processo è composto da due fasi:

- *hashing*, che consiste nella risoluzione di un puzzle matematico, che consiste nel trovare un numero che, una volta applicata una funzione di hash, abbia un valore inferiore ad un valore prefissato. Il valore di questo numero viene chiamato *nonce*;
- *ricerca del consenso*, che consiste nella verifica della validità del blocco, che viene effettuata da tutti i nodi della rete. Se il blocco è valido, viene aggiunto alla blockchain.

I minatori (miners) utilizzano un software speciale per risolvere il problema matematico incredibilmente complesso di trovare un nonce che generi un hash accettato. Poiché il nonce è di soli 32 bit e l'hash di 256, ci sono circa quattro miliardi di possibili combinazioni nonce-hash che devono essere estratte prima di trovare quella giusta.

Quando un blocco viene estratto con successo, la modifica viene accettata da tutti i nodi della rete e il miner viene ricompensato finanziariamente. Il primo minatore che risolve il puzzle e aggiunge un nuovo blocco alla blockchain viene ricompensato con un blocco di criptovaluta di nuovo conio. Il processo richiede un software specializzato e una grande potenza di calcolo, che può essere ottenuta utilizzando un computer o un gruppo di computer con grosso dispendio di energia e risorse.

2.1.6 Algoritmi di consenso

Il meccanismo di ricerca di consenso prevede numerose varianti, che differiscono per il modo in cui i nodi della rete si accordano per aggiungere un nuovo blocco alla blockchain.
Possiamo principalmente distinguere:

1. *Proof of Work (PoW)*, basato nella ricerca di un hash (una stringa di numeri e lettere) che soddisfa una determinata condizione, detta target. La condizione target richiede che l'hash del nuovo blocco sia inferiore a un determinato valore. Questo valore viene stabilito in base alla difficoltà della blockchain, che viene regolata automaticamente per mantenere il tempo medio di creazione di un nuovo blocco costante.

   Il primo miner che trova correttamente l'hash del blocco viene ricompensato in criptovaluta per il lavoro svolto. Questo meccanismo garantisce sicurezza nella rete, perché rende difficile ad un attaccante malintenzionato compromettere la sicurezza della rete non avendo la stessa potenza di calcolo.

   Questa è attualmente utilizzata in *Bitcoin*, in cui il tempo medio della formazione di blocchi è di 10 minuti circa, oppure la piattaforma *Ethereum*, altro importante esempio di blockchain. Si consideri che ne esistono varie tipologie, in cui ciascuna cerca di ridurre il consumo energetico dando prova del lavoro svolto (*Meaningful*) oppure assicurandosi che i nodi abbiano calcolato correttamente dopo un certo tempo (*Delayed*). Il principale svantaggio del PoW è che richiede un elevato consumo energetico, che può essere risolto con l'utilizzo di algoritmi di consenso alternativi.

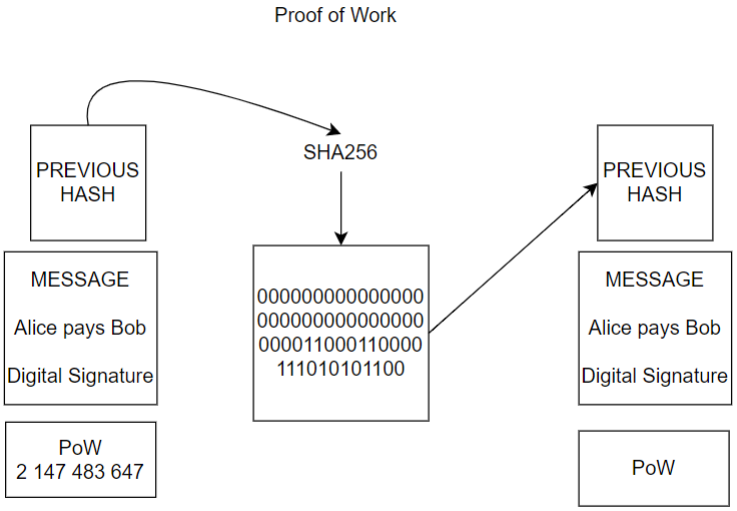


Figura 2.2: Processo di conferma in Proof of Work

2. *Proof of Stake (PoS)*, che si si basa sulla detenzione di una certa quantità di criptovaluta come garanzia per la validazione delle transazioni. L'alto consumo di energia dell'algoritmo precedente ha portato ad algoritmi come il *Proof of Stake*, i nodi della rete bloccano una certa quantità di criptovaluta come 'punteggio' per dimostrare che hanno un interesse nella corretta validazione delle transazioni. Questo punteggio viene utilizzato come base per la selezione del nodo che convalida la transazione successiva.

   A differenza del PoW, non ci sono miner coinvolti nel processo. Al loro posto, i partecipanti alla rete che vogliono essere coinvolti nella verifica della validità delle transazioni e nella creazione di blocchi nella rete devono detenere una certa quota nella rete, per esempio mettendo una certa quantità di moneta della rete in un portafoglio collegato alla sua blockchain. Questo processo è noto come 'placing a stake' o 'staking', che può essere tradotto come il fatto di mettere i propri interessi in gioco.

   Pertanto, le reti PoS sono basate su algoritmi deterministici, il che significa che i validatori dei blocchi sono eletti a seconda della natura della posta in gioco. Pur risultando meno intensiva in termini di energia rispetto alla PoW, il sistema risulta essere sbilanciato a favore dei produttori di blocchi oppure a favore degli utenri ricchi, poiché i validatori con più moneta hanno più possibilità di essere eletti. Alcune varianti di questo algoritmo prevedono infatti di scegliere dei nodi delegati per convalidare le transazioni e avere più controllo (*Delegated*), oppure di scegliere i validatori in base alla quantità di moneta che hanno investito (*Proof of Burn*).

2.1.7 Tipi

Una prima categorizzazione delle blockchain avviene distinguendole in base ai permessi. Nello specifico:

- *permissioned*, in cui i nodi della rete sono controllati da un ente centrale, che può essere un'azienda, un'organizzazione o un'istituzione. Esse limitano l'accesso alla rete a determinati nodi e possono anche limitare i diritti di tali nodi su tale rete. Le identità degli utenti di una blockchain autorizzata sono note agli altri utenti della blockchain autorizzata. Queste sono spesso utilizzate da aziende per la gestione di dati sensibili o convalida di transazioni;
- *permissionless*, che non richiedono alcun permesso per partecipare e permettono agli utenti di essere pseudoanonimi (usando uno pseudonimo non si rivela alcun dettaglio relativo all'identità personale) e non restringono i permessi dei nodi. Queste ultime tendono ad essere più sicure delle precedenti, avendo vari nodi che convalidano le singole transazioni. Di fatto, sono usate dalle varie blockchain, come ad esempio quella di Bitcoin.

Fatta tale premessa, possiamo distinguere:

- *public*, in cui chiunque abbia accesso alla rete può partecipare e diventare un nodo autorizzato nella blockchain. Di fatto sono sicure, ma non anonime e molto meno scalabili. Una loro futura implementazione sarà all'interno dei sistemi di votazioni elettroniche.
- *private*, in cui i singoli nodi sono controllati da un ente centrale, che opera in una rete chiusa e, come tali, restringono la natura di blockchain, dato che richiede controllo centralizzato dei singoli nodi per poter funzionare. Queste sono utilizzate da parte di alcune banche per la gestione dei loro sistemi di pagamento.
- *hybrid*, in cui alcuni nodi sono pubblici e altri sono privati, normalmente utilizzata per eseguire convalide delle transazioni presenti al suo interno. Microsoft utilizza una blockchain di identità digitale basata sul meccanismo ibrido.
- *consortium*, all'interno della quale i nodi sono controllati da un ente centrale, ma la rete è aperta a tutti i membri del consorzio, portando anche qui ad una privatizzazione e centralizzazione della rete. IBM sta utilizzando una pitattaforma basata sulla gestione dei pagamenti per banche ed aziende partner.

2.2 Blockchain: concetti avanzati

2.2.1 Token

Possiamo definire i token come unità di valore accettate da una comunità e costruite su una blockchain pre-esistente (dunque, non possono essere minati). Possono essere usati per rappresentare asset fisici come l'oro o l'immobiliare, oppure per rappresentare beni digitali come i dati o i diritti di accesso a servizi. Essendo registrati su una blockchain, sono immutabili e possono essere trasferiti in modo sicuro e trasparente da un proprietario all'altro.

Per utilizzare i token, gli utenti devono prima avere un portafoglio digitale, ovvero un'interfaccia che consente loro di accedere alla blockchain e interagire con essa. Una volta che un utente ha un portafoglio, può ricevere e inviare token. Per ricevere i token, l'utente deve fornire il proprio indirizzo del portafoglio al mittente, che poi invia i token all'indirizzo fornito. Per inviare i token, l'utente deve avere abbastanza token nel proprio portafoglio e deve conoscere l'indirizzo del destinatario a cui inviare i token.

Ogni bene viene considerato *asset*, in quanto non riproducibile e non falsificabile, esistente solo in forma digitale e quindi non fisica. Un token può essere lanciato sul mercato tramite un *initial coin offering* (ICO), che è un'offerta pubblica iniziale di token, al fine di finanziare nuovi progetti e comprenderne il nuovo valore.

Come per le blockchain, gli asset non sono regolamentati da un organo centrale, pertanto è possibile raccogliere fondi senza nuovi intermediari e così creare nuovi progetti, adeguatamente documentati tramite un *white paper* che descrive il progetto e il suo funzionamento. Il loro processo di creazione dei token nelle blockchain viene definito come *minting*, in cui un creatore mette a disposizione un certo numero di token, che possono essere acquistati dai partecipanti.

Principalmente, i token possono essere classificati in base al loro scopo:

- *Utility token*, che consentono di acquistare un determinato bene o servizio e conferisce al titolare un diritto di opzione per l'acquisto o somministrazione di cose o per la fornitura di servizi (attuali o futuri).
- *Security token*, che sono token che rappresentano un diritto di proprietà su un bene fisico o digitale.
- *Payment token*, più comunemente noti come 'crypto'. Come si può intuire, questi token sono utilizzati per acquistare e vendere beni e pagare le commissioni delle transazioni basate sulla blockchain senza la necessità di un intermediario.

I token, principalmente, sono caratterizzati dal fatto di essere spendibili e scambiabili con altri beni dal valore equivalente, ossia 'fungibili'. In questo senso, è possibile citare:

- *token fungibili (fungible tokens)*, che hanno la caratteristica di essere non uniche, divisibili e avere un chiaro valore di mercato, quindi scambiabili sempre con altri beni dello stesso valore;
- *token non fungibili (non fungible tokens NFT)*, beni fisici o digitali unici ed irripetibili. Questi vengono definiti come tali in quanto non possono essere scambiati con altri beni dello stesso valore, e per natura delle stesse blockchain, sono considerati immutabili. Infatti, l'utente, disponendo di un wallet, può creare in qualsiasi momento un bene considerabile come unico, pagando una commissione al momento dell'acquisto del bene. Questo particolare tipo di token è molto utilizzato per la vendita di beni come opere d'arte, musica, videogiochi, per loro natura non contraffabile dato lo scambio tramite blockchain.

La loro creazione è vincolata da alcuni standard, stabilità della comunità della blockchain Ethereum, che ne vincola la creazione tramite *smart contract*, che sono dei contratti che vengono eseguiti sulla blockchain e che possono essere scritti in vari linguaggi di programmazione.
Di seguito i principali standard di riferimento che stabiliscono precisamente i parametri della loro creazione:

- *ERC-20*, che è il più diffuso e utilizzato standard per i token, che consente di creare token che possono essere trasferiti tra gli utenti. Questo stabilisce che un contratto esponga un saldo, un'opzione di trasferimento, un'opzione di approvazione e un evento di trasferimento;
- *ERC-721*, che è uno standard per i token non divisibili (NFT), che rappresentano un bene unico non scambiabile in altri modi. Di base utilizza gli stessi parametri del precedente, ma possiede un campo specifico che indica chi è il proprietario;
- *ERC-1155*, che è uno standard per i token divisibili e non divisibili e riunisce a livello di dati un supporto comune con funzione di trasferimento e di approvazione anche in blocco.

2.2.2 Tokenizzazione

La tokenizzazione è il processo di rappresentazione di un bene o di un'attività in forma digitale attraverso l'emissione di un token su una blockchain. In altre parole, si tratta di convertire un bene fisico o immateriale in un asset digitale che può essere negoziato e scambiato in modo decentralizzato. La combinazione con la blockchain potrebbe aprire nuove prospettive per l'ottimizzazione dei processi aziendali, che includono più partner, e l'introduzione di nuovi modelli di business. Poiché la tokenizzazione è un processo decentralizzato, non è necessario un intermediario per la creazione e la gestione di un token e questo facilita la creazione di

nuovi beni, sapendo che la verifica viene effettuata da tutti i partecipanti della rete ed ogni bene è considerato unico.

Esistono diversi tipi di asset da considerare, i quali saranno poi successivamente convertiti in token:

- *Asset finanziari*, che rappresentano un diritto di proprietà su un bene fisico o digitale. Il concetto della finanza decentralizzata sta emergendo come *DeFi (Decentralized Finance)*, in cui i beni vengono spostati su blockchain a seconda della loro natura. Infatti, come per i token i beni acquistano una fungibilità e possono essere scambiati tra utenti;
- *Asset immobiliari*, che rappresentano un bene fisico, come un immobile, che può essere tokenizzato e quindi diventare un bene digitale. Questi beni sono fungibili e come tali hanno un valore tendenzialmente fisso, che può essere soggetto a variazioni nel tempo;
- *beni intangibili - intangibles*, come diritti di proprietà intellettuale o i citati beni collezionabili, e quindi acquisire grazie alla natura di blockchain un valore unico e verificato.

Si può quindi comprendere che la tokenizzazione può potenzialmente trasformare tutto in un bene con un valore, permettendo di avere dei prezzi equi stabiliti dalle vere esigenze del mercato, riducendo i costi di gestione essendo un sistema scalabile e sicuro che lo gestisce, permettendo l'aumento della liquidità, avendo sempre degli investitori pronti a scambiare i token, in modo trasparente e senza intermediari, riducendo notevolmente i tempi di scambio rispetto ai beni tradizionali.

2.2.3 Smart contract

Gli smart contract (o contratti intelligenti) sono programmi che automatizzano le azioni richieste in un accordo o contratto, considerate tracciate e irreversibili. Essi sono programmi informatici auto-eseguibili che vengono eseguiti su una blockchain, che automatizzano ed eseguono le clausole contrattuali in modo sicuro, trasparente e immutabile, senza la necessità di intermediari. Il loro funzionamento è regolato dal libro mastro centrale di blockchain, in cui i partecipanti devono firmare crittograficamente la loro partecipazione, fornendo precisamente i loro indirizzi di riferimento e codificando lo scambio delle informazioni secondo gli standard definiti nella sezione precedente.

Gli smart contract sono stati sviluppati per risolvere i problemi di affidabilità e sicurezza dei contratti tradizionali, che sono soggetti a errori umani, mancanza di trasparenza e vulnerabilità. Il loro funzionamento segue generalmente questo schema:

1. un utente avvia una transazione dal proprio portafoglio blockchain;
2. la transazione arriva al database distribuito, dove viene confermata l'identità del portafoglio dell'utente;
3. la transazione, che può essere un trasferimento di fondi e comprende il codice che definisce il tipo di transazione da eseguire, viene approvata;
4. questa viene eseguita come blocco all'interno della blockchain.

Principalmente, questi vengono utilizzati nella piattaforma Ethereum, dove i contratti vengono scritti in linguaggio di programmazione *Solidity*, che è un linguaggio di programmazione orientato agli oggetti basato su C++, eseguiti sulla piattaforma *Ethereum Virtual Machine (EVM)*, che ne consente l'esecuzione in modo scalabile regolata dagli standard precedenti. Ogniqualvolta venga eseguita una transazione, si ha un costo pagato in *gas*, costo in termini di energia necessaria per eseguire la transazione, che viene calcolato in base al numero di operazioni eseguite e pone un limite al numero di transazioni che possono essere eseguite in un dato periodo di tempo.

Possiamo citare alcuni esempi di applicazione di smart contract:

Figura 2.3: Funzionamento di uno Smart Contract

- *smart legal contracts*, legalmente applicabili e richiedono alle parti di soddisfare i loro obblighi contrattuali. Per creare alcuni contratti legali intelligenti, le parti coinvolte lavorano sul codice del contratto intelligente - o lo fanno i loro sviluppatori di software - finché non si accordano sui termini e sulle condizioni dell'accordo;
- *decentralized autonomous organizations (DAO)*, che sono organizzazioni che funzionano in modo decentralizzato e autonomo, senza un leader centrale e open-source, ma regolate da un contratto intelligente. All'interno delle blockchain vegono organizzate delle votazioni, che possono essere eseguite da tutti i partecipanti, che possono votare per o contro una proposta, oppure garantire il corretto scambio dei beni digitali all'interno della piattaforma, prevenendo possibili vulnerabilità;
- *crowdfunding*, che è un metodo di finanziamento di progetti e aziende, in cui le persone possono contribuire con denaro o beni, in cambio di un premio o di un prodotto finale;
- *logical applicative contracts*, che sono contratti che possono essere utilizzati per eseguire operazioni logiche, come la verifica di un dato e l'esecuzione di un'altra operazione, oppure gestire un sistema di pagamento o beni di scambio, garantendo la tracciabilità e la trasparenza di ogni bene alla consegna.

I contratti rimangono aggiornati grazie all'uso di *oracoli (oracles)*, che sono sistemi che permettono di ottenere informazioni esterne alla blockchain, come il valore di un bene o il valore di un'azione, attraverso l'uso di API o di altri sistemi di comunicazione. In questo modo, il contratto mantiene la sua funzionalità interagendo in maniera ibrida con il mondo esterno; questi vengono definiti come *hybrid smart contracts*, combinando un'infrastruttura *on-chain* con un'infrastruttura *off-chain*. Il loro utilizzo, per quanto utile, deve essere attentamente bilanciato, sia in termini di veridicità dei dati che di sicurezza, in quanto possono essere soggetti a compromissione o di scalabilità, per cui è necessario un'attenta valutazione dei rischi.

2.2.4 Scalabilità

Le blockchain sono preziose per il fatto di essere un sistema deterministico e con un grado di affidabilità molto elevato, generalmente neutro e verificabile da parte dell'utente finale. In questo contesto, è fondamentale parlare del cosiddetto *blockchain trilemma*, che è un problema di scelta tra tre caratteristiche fondamentali di una blockchain: decentralizzazione, sicurezza e scalabilità. Infatti, decentralizzare significa mantenere

un vasto numero di nodi sulla rete, senza però alcun controllo da parte di un ente centrale, mantenendo allo stesso modo un alto numero di transazioni in modo scalabile e una robustezza ai possibili partecipanti della rete.

In questo senso, si cerca di trovare un compromesso tra le tre caratteristiche, che è possibile ottenere l'aggiunta di ridondanza dei dati anche al di fuori della blockchain principale, cercando di bilanciare quanto possibile il consenso sui nodi distribuiti non intaccando il livello di libertà offerto ma anche aumentare il livello di dati trasmessi. Per questo, il nucleo scalabilità permette di introdurre numerose soluzioni al fine di migliorare le prestazioni della blockchain, dividendo però la blockchain in vari strati, definiti come *layer*.

- *Layer 0*, che è la blockchain principale, che contiene i dati e le transazioni;
- *Layer 1*, che è il livello responsabile della sicurezza e della scalabilità, che contiene i nodi che eseguono le transazioni e che sono responsabili della validazione;
- *Layer 2*, noto anche come livello di esecuzione, in cui si hanno i contratti intelligenti e le applicazioni che vengono eseguite sulla blockchain;
- *Layer 3*, noto anche come livello applicativo, che contiene le applicazioni decentralizzate (dApps), eseguite sulle blockchain e che interagiscono con gli utenti.

Comunemente, le soluzioni di scalabilità cercano di intervenire in vari modi:

- *Sharding*, che è una tecnica di scalabilità che consente di dividere la blockchain in più parti, chiamate *shards*, che possono essere gestite da nodi diversi e aumentando la dimensione dei singoli blocchi e la loro frequenza di creazione. Questa si effettua comunemente per il Layer 1.
- *Sidechains*, che sono blockchain secondarie che vengono utilizzate per eseguire transazioni parallele, che vengono poi sincronizzate con la blockchain principale. In questo ambito, soluzioni comuni includono i *canali di stato (state channels)*, che sono dei contratti intelligenti che permettono di eseguire transazioni tra due parti, senza che queste siano visibili sulla blockchain principale, e i *canali di pagamento (payment channels)*, che migliorano la comunicazione bidirezionale delle transazioni eseguendole sulla blockchain secondaria e dandone traccia sulla principale.

2.3 Self-Sovereign Identity

La *self-sovereign identity (SSI)* è un approccio all'identità digitale che dà agli individui il controllo sulle informazioni che usano per dimostrare chi sono a siti web, servizi e applicazioni in tutto il web. Senza l'SSI, gli individui con account (identità) persistenti su Internet devono affidarsi a una serie di fornitori terzi, come Facebook, Google e altri, che hanno il controllo delle informazioni associate alla loro identità.

Esistono molti modi per implementare l'SSI utilizzando le chiavi crittografiche e ne analizzeremo due.

- l'utilizzo di firme digitali, attraverso un processo di *firma digitale*, che permette di firmare un documento con una chiave privata, e di verificare la firma con la chiave pubblica;
- *Decentralized Identifier (DID)*, che è un identificatore univoco alfanumerico per un soggetto, che può essere utilizzato per identificare una persona, un'organizzazione, un dispositivo, un servizio, un documento, ecc.

Le parti coinvolte in questo processo sono principalmente tre:

1. *emittente*, detto anche *holder*, ossia l'entità che emette una credenziale, ad esempio un documento d'identità governativo;
2. *titolare*, detto anche *issuer*, ossia il proprietario della credenziale, cioè l'entità su cui l'emittente genera la credenziale;
3. *verificatore*, detto anche *verifier*, cioè l'entità che controlla la validità e l'autenticità della credenziale presentata dal titolare.

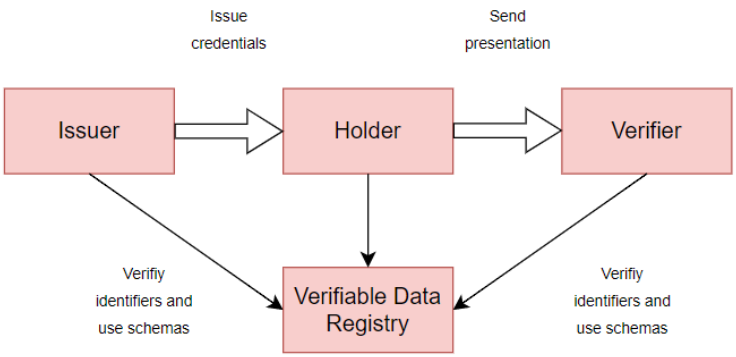


Figura 2.4: Processo di riconoscimento di una credenziale SSI

Lo scopo principale di questa tecnologia è consentire agli utenti un'esistenza indipendente da provider terzi, permettendo loro di controllare la propria identità in modo sicuro e accedendovi senza dover affidarsi a terzi. Inoltre, i sistemi e gli algoritmi che la supportano devono essere trasparenti, mantenendo le informazioni in modo trasparente e permanente, rendendo però semplice la portabilità e l'interoperabilità di queste all'interno delle varie piattaforme.

La tecnologia blockchain, per mezzo della sua natura decentralizzata, permette di creare un sistema di identità digitale che soddisfi questi requisiti. In particolare:

- il titolare della credenziale possiede il suo DID firmato dalla propria coppia di chiavi che certifica la sua identità;
- l'emittente fornisce delle *Verifiable Credentials (VC)*, che certificano in modo digitale e crittograficamente protetto la validità del proprio ruolo;

- il verificatore controlla che, tramite ciascun blocco, sia stata rilasciata una VC valida e che il titolare sia il legittimo possessore di quella VC.

L'obiettivo principale è quello di fornire un utilizzo delle tecnologie blockchain tali da permettere agli utenti di selezionare quali credenziali mostrare (*selective disclosure o divulgazione selettiva*), secondo opportuni standard definiti gradualmente dall'organizzazione internazionale *W3C*, tra cui il citato VC.

2.3.1 Tipi e applicazioni

Il concetto di SSI è stato introdotto nel 2015 da Sovrin Foundation, che ha sviluppato il protocollo Sovrin, organizzazione privata senza scopo di lucro che ha creato la prima rete di identità auto-sovrana, diventato poi standard W3C. Esso si basa sul protocollo è *Hyperledger Indy*, che è un framework open source per la creazione di SSI basato su blockchain che fornisce strumenti, librerie e componenti riutilizzabili per fornire identità digitali legate al mondo blockchain, come ad esempio la firma digitale e la crittografia.

Questo è il principale, ma possiamo definire un insieme di standard utilizzati al fine di garantire agli utenti di possedere e controllare in modo indipendente la propria identità digitale, gestendo come vengono condivise le proprie informazioni. La particolarità di questi protocolli è di comunicare gli uni con gli altri attraverso messaggi crittografati:

- *DID (Decentralized Identifier)*, che consente di creare identificativi digitali decentralizzati come citato supportano e permettono di identificare i soggetti in modo resistente alla falsificazione, autenticando le informazioni in modo sicuro;
- *VC (Verifiable Credentials)*, che consente di gestire attestazioni verificati, quindi insiemi di informazioni cfhe un individuo possiede o controlla, dimostrando che l'attestazione è stata rilasciata da un emittente autorizzato. Questo all'interno delle blockchain garantisce che ciascun nodo e parte in gioco sia chi dice di essere senza ledere la decentralizzazione alla base;
- *ZKP (Zero Knowledge Proof)*, che consente di dimostrare la conoscenza di una proprietà di un dato, senza rivelare alcun dettaglio relativo. Questa tecnologia è considerata a sé stante un'innovazione tecnologica, che permette attraverso l'individuazione di parti precise, la certezza che solo chi conosce l'informazione può correttamente rispondere alle domande poste senza rivelare i propri dati.

Attualmente, la tecnologia SSI prevede una graduale implementazione in sistemi di autenticazione e in generale di accesso ai servizi pubblici, permettendo all'utente un ulteriore controllo sui dati trasmessi rivelandoli solo a parti autorizzate. Questo risulta particolarmente utile nel settore sanitario, legislativo (garantendo certezza di identità e di voto) e finanziario, permettendo di ridurre i costi di gestione e di mantenimento dei dati, oltre che di ridurre i tempi di accesso ai servizi. Il principale organo promotore è la *Decentralized Identity Foundation (DIF)*, che ha come obiettivo quello di creare un ecosistema di identità digitali decentralizzate, promuovendo integrazioni, progetti e nuove idee in questo ambito.

2.4 Zero Knowledge Proof

Definita anche come prova a conoscenza zero, è una tecnologia che permette di dimostrare la conoscenza di una proprietà di un dato, senza rivelare alcun dettaglio relativo. Occorre definire le parti coinvolte:

- *Prover*, ossia l'entità che prova la conoscenza di alcune informazioni riservate. L'informazione segreta è il 'testimone' (*witness*) della prova e la presunta conoscenza del testimone da parte del prover stabilisce un insieme di domande a cui può rispondere solo chi conosce l'informazione.
- *Verifier*, ossia l'entità che verifica la correttezza della prova. Lui si occupa di scegliere a caso la sfida (*challenge*) e di verificare la risposta del prover. La risposta del prover permette al verificatore di controllare

se il primo ha davvero accesso al testimone. Per assicurarsi che il prover non stia indovinando alla cieca e non ottenga le risposte corrette per caso, il verificatore sceglie altre domande da porre.

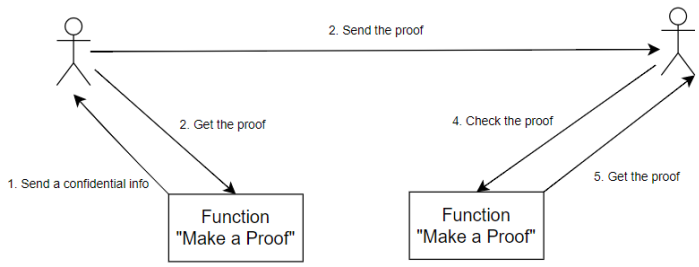


Figura 2.5: Processo di verifica di una ZKP

A livello di transazioni, la ZKP è utilizzata per garantire la sicurezza con certezza che le transazioni siano valide (*validity proofs*), tramite la presenza del testimone tipicamente basato su calcoli polinomiali o prossimità ad un insieme di valori, come ad esempio la distanza di Hamming. Le transazioni possono essere invalidate in ogni momento tramite le *fraud proofs*, che dimostrano che la transazione è stata effettuata in modo illegittimo. Queste ultime confrontano i Merkle trees delle transazioni e verificano che l'hash corrisponda a quella originale.

Di fatto, ogni prova deve dimostrare di essere completa, pertanto è necessario che il verificatore sia in grado di verificare che la risposta del prover sia corretta, dimostrando questa affermazione con solidità, in modo da non lasciare spazio a dubbi e a falsi positivi, senza trasmettere alcun dato (a conoscenza zero). L'idea principale della ZKP è presupporre di non fidarsi di nessuno (*zero trust*), al fine di garantire la sicurezza delle informazioni. In questo modo, ciascun utente viene monitorato a livello di privilegi, accessi e dati, senza che nessuno possa accedere senza dimostrare di averne il permesso.

2.4.1 Tipi e applicazioni

Esistono diverse implementazioni di ZKP, ognuna delle quali presenta un proprio compromesso in termini di dimensione della prova, tempo del prover, tempo di verifica e altro ancora. I principali tipi sono:

- *zk-SNARK*, acronimo di *Zero Knowledge Succinct Non-interactive Argument of Knowledge*, prova di dimensioni ridotte e facile da verificare. Essa è stata sviluppata dalle piattaforme *Zcash* e *Ethereum* utilizzando le curve ellittiche, che presuppongono che sia impossibile trovare il logaritmo discreto di un elemento casuale della curva ellittica a partire da un punto base pubblicamente noto. Il calcolo delle curve ellittiche è meno dispendioso dal punto di vista computazionale rispetto al calcolo delle funzioni di hashing.
- *zk-STARK*, acronimo di *Zero Knowledge Succinct Transparent Argument of Knowledge*, tipo di prova crittografica che richiede un'interazione minima o nulla tra il prover e il verificatore. I vantaggi principali delle STARK rispetto alle SNARK sono che hanno tempi di prover rapidi e sono più facili da scalare in quanto offrono una maggiore potenza di calcolo. Essa è stata sviluppata dalla piattaforma *Horizen* utilizzando le *curve ellittiche*, che presuppongono che sia impossibile trovare il logaritmo discreto di un elemento casuale della curva di Weierstrass a partire da un punto base pubblicamente noto. Il calcolo delle curve di Weierstrass è più dispendioso dal punto di vista computazionale rispetto al calcolo delle curve ellittiche.

All'interno delle blockchain, vengono utilizzati all'interno delle sidechain secondo il sistema delle *zk-Rollup*, che permettono di eseguire transazioni su una blockchain laterale (sidechain) senza doverle scrivere su questa. In altre parole, con uno zk-rollup, le transazioni vengono 'confezionate' in un'unica transazione che viene elaborata sulla blockchain, riducendo il carico di lavoro richiesto alla rete. Inoltre, grazie all'utilizzo delle Zero Knowledge

Proof, le transazioni vengono verificate in modo sicuro, senza rivelare alcuna informazione sulle transazioni stesse.

Principalmente, queste vengono utilizzate in sistemi di scalabilità layer-2, al fine di memorizzare solo il minimo numero previsto di transazioni (*rollup*), oppure memorizzando solamente un hash sulla catena per proteggere i dati del libro mastro (validium), oppure scegliere come salvarli per ogni transazine (*volition*). Alcuni casi d'uso possibili di applicazione di questa tecnologia possono essere:

- pagamenti anonimi
- protezione dell'identità
- autenticazione
- archiviazione di dati sensibili

L'utilizzo combinato con la Self-Sovereign Identity è particolarmente interessante perché consente di garantire la privacy e la sicurezza delle informazioni personali degli utenti. In particolare, è possibile garantire di conoscere le parti in gioco nella rete poiché verificate in modo sicuro e privato tramite credenziali certificate, scegliendo anzi quali informazioni rivelare e senza dover rivelare alcuno dei propri dati per dimostrare di possederli.

Capitolo 3

Descrizione dello stage

In questo capitolo viene descritto il contesto in cui si è svolto lo stage, il progetto di stage e gli obiettivi prefissati in base alla pianificazione iniziale.

3.1 Analisi preventiva dei rischi

Durante la fase di analisi iniziale sono stati individuati alcuni possibili rischi a cui si potrà andare incontro. Si è quindi proceduto a elaborare delle possibili soluzioni per far fronte a tali rischi.

Da scrivere, prendendo esempio da questo sotto

1. Performance del simulatore hardware
Descrizione: le performance del simulatore hardware e la comunicazione con questo potrebbero risultare lenti o non abbastanza buoni da causare il fallimento dei test.
Soluzione: coinvolgimento del responsabile a capo del progetto relativo il simulatore hardware.

3.2 Obiettivi preposti

Il tirocinio prevede lo svolgimento dei seguenti obiettivi, riportando questa notazione, come dal documento *Piano di Lavoro*:

- O per i requisiti obbligatori, vincolanti in quanto obiettivo primario richiesto dal committente;
- D per i requisiti desiderabili, non vincolanti o strettamente necessari, ma dal riconoscibile valore aggiunto;
- F per i requisiti facoltativi, rappresentanti valore aggiunto non strettamente competitivo.

Le sigle precedentemente indicate saranno seguite da una coppia sequenziale di numeri, identificativo del requisito.

- Obbligatori:
  - *O01*: Descrivere i concetti di base della blockchain, tra cui la sua architettura, i nodi della rete, la criptografia e il consenso distribuito;
  - *O02*: Analizzare il concetto di Smart contract e il linguaggio Solidity, con particolare attenzione alle vulnerabilità principali e alle tecniche per evitare errori di programmazione;
  - *O03*: Approfondire il funzionamento della firma asimmetrica delle transazioni su catena e la validazione dei blocchi, studiando le tipologie di consenso e le catene più conosciute;
  - *O04*: Studiare le tecniche di crittografia utilizzate per garantire la sicurezza e la privacy delle informazioni personali nell'ambito della Self-Sovereign Identity (SSI) e dei protocolli per la gestione delle identità digitali;

  - *O05*: Individuare casi d'uso reali per la SSI, analizzando i consorzi internazionali coinvolti in ambito di ricerca e i finanziamenti europei;
  - *O06*: Discutere le sfide e i problemi legati alla SSI, studiando un possibile scenario futuro di applicabilità e basato su Zero Knowledge Proof (ZKP).

- Desiderabili:
  - *D01*: Implementare una dApp tramite le librerie EthersJS/Web3JS, utilizzando un smart contract di esempio;
  - *D02*: Realizzare una UI (interfaccia utente) per la dApp utilizzando HTML, CSS e JavaScript;
  - *D03*: Utilizzare la SSI e i protocolli studiati per implementare funzionalità di autenticazione utente nella dApp;
  - *D04*: Testare e debuggare la dApp implementata;
  - *D05*: Discutere problemi e sfide relativi all'implementazione di applicazioni decentralizzate su blockchain Ethereum, con particolare attenzione alla scalabilità e alla sicurezza.

- Facoltativi:
  - *F01*: Approfondire l'utilizzo di altri linguaggi di programmazione per gli smart contract, come Vyper;
  - *F02*: Analizzare l'utilizzo di altre tecnologie blockchain, come Polkadot o Cardano, per implementare la SSI;
  - *F03*: Esplorare altre funzionalità delle librerie EthersJS/Web3JS, come l'invio di transazioni;
  - *F04*: Investigare i limiti e le sfide legate all'utilizzo della tecnologia blockchain.

3.3 Pianificazione

Da scrivere

Capitolo 4

Analisi dei requisiti

Breve introduzione al capitolo

4.1 Casi d'uso

Per lo studio dei casi di utilizzo del prodotto sono stati creati dei diagrammi. I diagrammi dei casi d'uso (in inglese *Use Case Diagram*) sono diagrammi di tipo UML dedicati alla descrizione delle funzioni o servizi offerti da un sistema, così come sono percepiti e utilizzati dagli attori che interagiscono col sistema stesso. Essendo il progetto finalizzato alla creazione di un tool per l'automazione di un processo, le interazioni da parte dell'utilizzatore devono essere ovviamente ridotte allo stretto necessario. Per questo motivo i diagrammi d'uso risultano semplici e in numero ridotto.

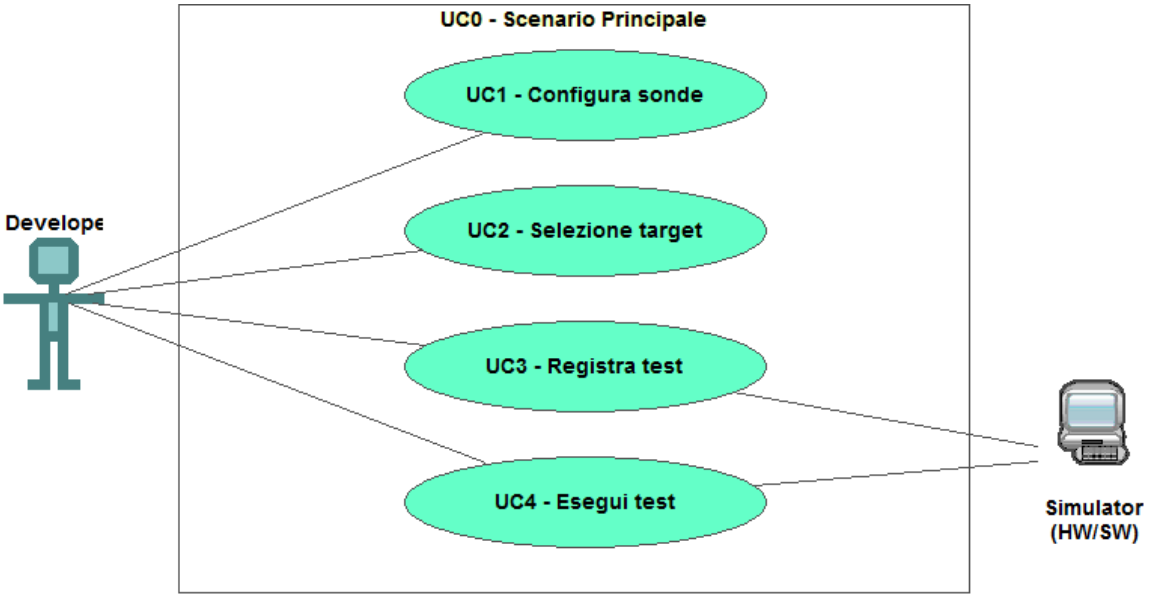


Figura 4.1: Use Case - UC0: Scenario principale

UC0: Scenario principale

Attori Principali: Sviluppatore applicativi.

Precondizioni: Lo sviluppatore è entrato nel plug-in di simulazione all'interno dell'IDE.

Descrizione: La finestra di simulazione mette a disposizione i comandi per configurare, registrare o eseguire un test.

Postcondizioni: Il sistema è pronto per permettere una nuova interazione.

4.2 Tracciamento dei requisiti

Da un'attenta analisi dei requisiti e degli use case effettuata sul progetto è stata stilata la tabella che traccia i requisiti in rapporto agli use case.
Sono stati individuati diversi tipi di requisiti e si è quindi fatto utilizzo di un codice identificativo per distinguerli. Il codice dei requisiti è così strutturato R(F/Q/V)(N/D/O) dove:

R = requisito

F = funzionale

Q = qualitativo

V = di vincolo

N = obbligatorio (necessario)

D = desiderabile

Z = opzionale

Nelle tabelle 4.1, 4.2 e 4.3 sono riassunti i requisiti e il loro tracciamento con gli use case delineati in fase di analisi.

Tabella 4.1: Tabella del tracciamento dei requisti funzionali

Requisito	Descrizione	Use Case
RFN-1	L'interfaccia permette di configurare il tipo di sonde del test	UC1

Tabella 4.2: Tabella del tracciamento dei requisiti qualitativi

Requisito	Descrizione	Use Case
RQD-1	Le prestazioni del simulatore hardware deve garantire la giusta esecuzione dei test e non la generazione di falsi negativi	-

Tabella 4.3: Tabella del tracciamento dei requisiti di vincolo

Requisito	Descrizione	Use Case
RVO-1	La libreria per l'esecuzione dei test automatici deve essere riutilizzabile	-

Capitolo 5

Progettazione e codifica

Breve introduzione al capitolo

5.1 Tecnologie utilizzate

In questa sezione, saranno elencate le tecnologie principali utilizzate durante lo sviluppo del sistema oggetto del tirocinio.

5.1.1 Codifica

Solidity

ethers.js

web3.js

5.1.2 Versionamento

GitHub

5.2 Ciclo di vita del software

5.3 Progettazione

Namespace 1

Descrizione namespace 1.

Classe 1: Descrizione classe 1

Classe 2: Descrizione classe 2

5.4 Design Pattern Utilizzati

5.5 Codifica

Capitolo 6

Verifica e validazione

Capitolo 7

Conclusioni

7.1 Consuntivo finale

7.2 Raggiungimento degli obiettivi

7.3 Conoscenze acquisite

7.4 Valutazione personale

Appendice A

Appendice A

Citazione

Autore della citazione

Bibliografia

Riferimenti bibliografici

James P. Womack, Daniel T. Jones. *Lean Thinking, Second Editon*. Simon & Schuster, Inc., 2010 (cit. a p. 1).

Siti web consultati

Manifesto Agile. URL: http://agilemanifesto.org/iso/it/ (cit. a p. 1).