Università Politecnica delle Marche

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Triennale in Ingegneria Informatica e dell'Automazione

Tesi di Laurea

Studio di BlockChain, Smart Contract e Ricardian Contract; piattaforme BlockChain based messe a confronto

Study of BlockChain, Smart Contract and Ricardian Contract, BlockChain based platforms compared

Relatore

Prof. Luca Spalazzi

Candidato

Martina Pioli

Anno Accademico 2018-2019

Indice

Elenco delle figure

1

Introduzione

Nel 2009 nacque un nuovo sistema di pagamento mondiale, il Bitcoin.

Con il repentino lancio della criptovaluta e informazioni sulle blockchain nei principali media, siamo portati a credere che tutto questo progresso tecnologico sia stato fatto negli ultimi anni. In realtà, l'attuale conoscenza della criptovaluta è il risultato di molti anni di lavoro dietro le quinte. Alcuni dei concetti riguardanti le blockchain più popolari, come Proof of Work e gli Smart Contract, sono stati pensati da visionari digitali nei primi anni novanta.

Quasi tutti oramai, sentendo parlare di *Bitcoin*, sanno di cosa si tratta, ma in pochi sanno come funziona la rivoluzionaria tecnologia che ne sta alla base: la *BlockChain*. Studiando tale tecnologia, si è scoperto che è possibile sfruttare le sue caratteristiche per aumentare la privacy e la sicurezza su diverse operazioni eseguite tutti i giorni da molti utenti in rete.

1.1 Obiettivi della tesi

In questo documento si vuole parlare della tecnologia BlockChain, dapprima a livello generale, andando a toccare le componenti principali come i ledger, gli Smart Contract ed i Contratti Ricardiani; in seguito andando sullo specifico, facendo degli esempi pratici, confrontando tra loro alcune delle piattaforme che utilizzano tali tecnologie.

1.2 Struttura

La tesi è strutturata nel seguente modo:

Primo Capitolo: viene descritta la tecnologia blockchain con tutte le caratteristiche che hanno permesso la sua rapida espansione. Per cercare di spiegare meglio alcuni concetti si è fatto ricorso all'esempio che in tanti conoscono: la rete Bitcoin.

Secondo capitolo: viene illustrato lo smart contract in modo dettagliato per rendere un'idea del ruolo che quest'importante elemento va a ricoprire all'interno

di una rete basata su tecnologia blockchain.

Terzo capitolo: viene illustrato il Ricardian Contract, com'è nato ed il suo funzionamento, senza questo elemento non è possibile rendere legale un contratto.

Quarto capitolo: confronto tra diverse piattaforme che sono state sviluppate su base blockchain.

Quinto capitolo: caso specifico della piattaforma Corda, viene illustrato un esempio pratico.

2

BlockChain

2.1 Cos'è una BlockChain

In questo primo capitolo, per cercare di capire meglio il concetto di BlockChain, prenderemo come esempio la conosciuta tecnologia Bitcoin (scritto con la lettera maiuscola s'intende la tecnologia e la rete, con la lettera minuscola si fa riferimento alla criptovaluta in sé).

La BlockChain è un concetto complesso, ma è un'idea geniale che ci permette di risolvere diversi problemi a livello informatico.

Questa tecnologia viene definita *disruptive* cioè in evoluzione a livelli esponenziali. Si diffonderà su tanti ambiti diversi andando a cambiare il modo di vedere di molte cose, proprio perché è un enorme passo avanti rispetto allo stato attuale.

2.1.1 Caratteristiche

Le caratteristiche principali di una struttura blockchain sono:

- Non è possibile duplicare un'informazione; essendo il bitcoin identificato come mezzo di scambio, simile quindi ad una moneta, è importante evitare il problema della duplicazione per non intercorrere in un caso di falsificazione. Utilizzando un sistema di blockchain, il bitcoin è una valuta *infalsificabile*.
  Per la prima volta è possibile avere un'informazione informatica non duplicata. È infatti facilissimo fare delle copie di un qualsiasi tipo di file, ma non di un'informazione inserita in una blockchain.
- Registro pubblico (un database) *distribuito*. Significa che non è presente un solo registro in un unico server, ce ne sono migliaia duplicati su un'infinità di computer in tutto il mondo, cioè tutti quelli che appartengono alla rete, come nel caso preso in esempio. Questo sembrerebbe una contraddizione alla caratteristica precedente, ma in realtà i concetti sono diversi. Prima si parlava di duplicazione a livello d'informazione, mentre in questo caso si parla di duplicazione a livello di blockchain. Ogni nodo appartenente alla rete ha una copia identica nel proprio spazio locale.

- *Inviolabile*. Tale caratteristica deriva dal fatto che è un registro pubblico distribuito. Si potrebbe andare a modificare la propria blockchain in locale, ma tali modifiche saranno visibili solamente a chi si collega in quella macchina. Grazie al tipo di rete utilizzata i nodi hanno tutti la stessa importanza all'interno della rete, quindi per poter manipolare dei dati per l'intera comunità sarebbe necessario modificarli nel 50%+1 delle blockchain che si trovano in locale. Nel caso di Bitcoin è impossibile considerando la moltitudine di nodi presenti (dal 2009, anno in cui è stata creata questa rete, sono stati fatti tantissimi tentativi di violare la blockchain ma nessuno c'è mai riuscito).
- Consente di gestire transazioni abbastanza veloci ed abbastanza economiche:
  - *Abbastanza* veloci: in Bitcoin le transazioni richiedono circa 10 minuti per essere aggiunte alla blockchain, il tempo che impiega ogni nuovo blocco per essere aggiunto alla catena. Altre criptovalute che usano la stessa tecnologia sono molto più veloci, ad alcune bastano pochi minuti o pochi secondi;
  - *Abbastanza* economiche: in Bitcoin proprio il fatto che i blocchi siano relativamente pochi, fa sì che le transazioni che possono essere inserite non sono molte, quindi ultimamente viene richiesta una commissione, un costo per inserire una transazione all'interno di un blocco. Ma esistono comunque altre criptovalute che funzionano diversamente, a volte le transazioni sono anche gratuite.

In sintesi, la blockchain è pubblica, chiunque può averla, chiunque può maneggiarla ma nessuno può modificarla. Ogni nodo è un *full node*, cioè contiene tutta la blockchain.

2.1.2 Rete

Le blockchain girano su un software che si basa su una rete ***Peer to Peer*** [P2P], cioè una rete fatta da tanti computer tutti pari tra loro, con la stessa importanza. Questo è proprio il significato di rete distribuita che, a differenza di un sistema centralizzato, non ha un server principale o uno che abbia più importanza rispetto agli altri. Non esistendo quindi tale nodo, per poter modificare dati o attaccare il sistema sarebbe necessario trovare e modificare i dati per il 50%+1 dei nodi della rete, come già anticipato precedentemente.

Nel caso di Bitcoin, ognuno può installare un full node in locale, è per questo che parleremo di rete pubblica. Collegandoci al sito www.blockchain.info è possibile vedere gli ultimi nodi che sono stati creati ed accedere alle loro informazioni. La stragrande maggioranza dei dati presenti sono le transazioni salvate all'interno di quel blocco: da dove sono partite, dove sono arrivate e quanti bitcoin sono stati trasferiti. Cliccando sul mittente o il destinatario è possibile anche vedere il wallet, che tradotto letteralmente sarebbe il portafoglio del nodo, sapere quanti bitcoin possiede, quante e quali transazioni ha eseguito. La possibilità di accedere anche a queste informazioni ci fa capire che è tutto pubblico, ma, non è comunque possibile sapere direttamente l'identità di del proprietario del wallet. Anzi, ogni utente potrebbe avere più wallet all'interno della rete. Bitcoin è anonima.

2.1.3 Ledger condiviso

Un ledger condiviso è un libro mastro dove vengono salvati tutte le transazioni del sistema.

La tecnica che viene utilizzata per creare la blockchain è molto complessa. Prevede la creazione di blocchi, la cui integrità è garantita dall'uso di primitive crittografiche, che vengono concatenati l'uno all'altro. Tale concatenazione avviene attraverso delle chiavi anch'esse crittografate ed immodificabili.

Per garantire la coerenza tra le varie copie, l'aggiunta di un nuovo blocco è globalmente regolata da un protocollo condiviso. Una volta ricevuta l'autorizzazione ogni nodo aggiorna la sua copia privata.

Per fare un esempio più vicino al mondo reale consideriamo una banca. Ogni banca ha dei server privati all'interno dei quali sono salvate tutte le transazioni dei clienti, spostamenti di denaro da un conto ad un altro, prelievi o depositi. I dipendenti hanno accesso al server e potrebbero modificare queste informazioni. Non dovrebbe mai succedere ma tutto è basato nella buona condotta del personale della banca. Se invece di utilizzare un registro centralizzato utilizzassero la tecnologia blockchain, non ci sarebbe questo timore.

2.1.4 Come funziona

Le transazioni eseguite vengono inserite su blocchi che vengono confermati con la chiave crittografica e con il consenso degli altri partecipanti alla rete. Da quel momento vengono aggiornate tutte le blockchain in tutti i nodi della rete, le transazioni diventano irreversibili ed i blocchi inviolabili. Questo lo rende uno strumento straordinario e potentissimo, soprattutto considerando quello che abbiamo detto precedentemente, è una tecnologia distribuita, non esiste un proprietario o un gestore che possa intervenire per modificare i dati che contiene una volta che vengono accettati dal sistema. Chiunque la usa non può fare niente per modificare il funzionamento o le informazioni, deve accettarla così com'è. [Fig 2.1]

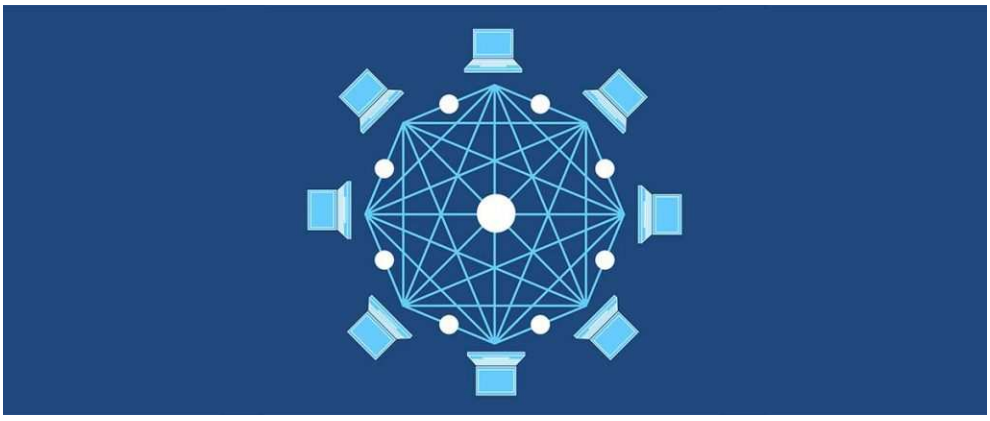

Figura 2.1. Come funziona una blockchain? - https://blog.smarteventi.it/blockchain-cose-funziona-guida-semplice.html

2.1.5 Perchè è così tanto rivoluzionaria?

Tutte le caratteristiche che abbiamo illustrato sono applicabili a qualsiasi registro pubblico. È infatti possibile sfruttare la tecnologia blockchain non solo a livello di Bitcoin o a livello di criptovaluta in generale, ma anche per tantissimi altri esempi che ritroviamo nella vita di tutti i giorni. Degli esempi possono essere: il catasto, il pubblico registro automobilistico (PRA), fino ad arrivare a pensare gli sviluppi della borsa attraverso transazioni e blocchi.

Per scendere ancora di più nel dettaglio, [12] la borsa di Sydney ha implementato, nel 2018, dopo due anni di test, l'adozione della tecnologia blockchain per le operazioni di liquidazione e regolamento degli scambi. Le ragioni alla base di questo cambiamento sono diverse, prime fra tutte la blockchain consente operazioni più rapide, più sicure e più economiche. Garantisce ai clienti tariffe più basse, ma tanti sono anche i vantaggi da parte del settore che vede un risparmio di diversi miliardi di dollari all'anno, grazie:

- allo snellimento delle procedure di liquidazione e regolamento che spesso ancora richiedono di porre fisicamente firme su documenti cartacei;
- alla riduzione del margine d'errore;
- alla riduzione delle tempistiche per ogni transazione;
- alla conseguente diminuzione del personale del settore, diverse mansioni svolte dai broker non sarebbero più utili.

Sarà possibile applicare questa tecnologia su tantissimi ambiti, ogni volta che si ha a che fare con un registro pubblico, metterlo online utilizzando una blockchain è diventato relativamente semplice ma estremamente potente.

Per poter funzionare, una blockchain ha bisogno di alcune procedure al suo interno. Ci troviamo di fronte agli Smart Contract, che sono appunto una delle parti fondamentali per il corretto funzionamento di questa tecnologia.

3

Smart Contract

Gli Smart Contract stanno emergendo insieme alle blockchain e stanno prendendo sempre più piede nell'ambito dei contratti tra più parti. L'argomento è molto vasto ed in via di sviluppo. Cercheremo di toccare tutti i punti di vista di questa tecnologia, partendo dalla parte operativa fino ad arrivare a quella legale.

3.1 Terminologia

Letteralmente Smart Contract significa contratto intelligente, la traduzione ci porta ad immaginare subito ad una tecnologia che riesca ad eseguire delle istruzioni in modo automatico, senza l'input di un utente esterno.

Tale termine viene utilizzato per diverse definizioni ma nel 2016 l'avvocato Josh Stark [7], che è anche capo delle operazioni presso una società di consulenza e sviluppo di blockchain, Ledger Labs, per fare un po' di ordine diede una panoramica per quanto riguarda i modi in cui viene definito uno Smart Contract più comunemente:

- Significato operativo: vengono coinvolti in questo caso degli agenti software, non necessariamente ledger collegati a piattaforme blockchain. La parola "contract" in questo caso indica che tali agenti stanno adempiendo a determinati obblighi ed esercitando determinati diritti, cioè ci aspettiamo che abbia la funzione che dovrebbe avere un contratto tradizionale tra più parti. Stark rinomina questa definizione *Smart Contract Code*;
- Significato legale: si focalizza su come i contratti legali possono essere espressi ed implementati in un software, riguarda quindi la scrittura e l'interpretazione per renderli applicabili anche a livello giuridico da un tribunale classico nel caso in cui s'intercorra in una situazione di controversia. In questo ambito rientra anche un tipo di contratto chiamato Ricardian Contract di cui tratteremo in seguito. Stark rinomina questa definizione *Smart Legal Contract*.

Dato che non esiste una definizione univoca sulla terminologia che deve essere usata, per essere chiari, in questo documento considereremo con smart contract l'insieme di queste definizioni, mentre, quando scenderemo nel dettaglio useremo i nomi dati da Stark così come definiti nel paragrafo precedente.

Un contratto deve essere applicabile in ogni ambito di utilizzo, negli smart legal contract si parlerà di diritti e obblighi abbastanza complessi, mentre negli smart contract code si parlerà di azioni del codice più semplici.

3.2 Automazione

Considereremo l'automazione dello smart contract definito in un ledger condiviso, cioè in una tecnologia blockchain.

Per essere smart, deve essere necessariamente automatizzabile, che è diverso dall'essere automatizzato. Infatti, in un contratto, è molto probabile che ci siano diverse parti nell'accordo, che richiedono l'intervento umano. Un esempio di come l'automazione possa essere raggiunta in uno smart legal contract è un Ricardian Contract (o Contratto Ricardiano). Rappresenta un mix tra scritture legali, parametri e codice collegati tra loro grazie a delle coppie nome-valore.

I parametri, formati dalle coppie nome-valore, serviranno per informare il codice dei dettagli finali delle operazioni. Esisterà un framework istanziato in un ledger condiviso che procederà nella creazione di contratti, conformemente con le leggi, nel momento in cui viene richiesta una nuova transazione nella piattaforma.

Il codice in questione potrebbe essere adatto solamente per la piattaforma specifica, ma possiamo immaginare che in un futuro ci sarà bisogno di una standardizzazione che porterà tali contratti ad essere compatibili con tutte le piattaforme blockchain.

3.3 Esecuzione

Continuando sulla stessa linea, uno smart contract, per essere automatico deve quindi essere eseguibile. Pensando ad un contratto nel mondo reale, ad esempio ad un contratto cartaceo che possiamo trovarci di fronte in qualsiasi momento, delle domande sorgono spontanee:

- Quali sono le parti che diventeranno eseguibili?
- Come devono essere eseguite per essere conformi alle leggi?

3.3.1 Cosa eseguire

Le parti eseguibili di un contratto cambiano in base a quale definizione di un contratto si tiene in considerazione:

- Per uno smart contract code: il requisito fondamentale è che il codice deve essere eseguito entro una scadenza di tempo ragionevolmente breve. La piattaforma blockchain ha il completo controllo delle azioni eseguite da questi codici, è importante che vengano portate a termine in modo esatto e senza ritardo. Immaginando che il codice sia esatto, le problematiche che potrebbero verificarsi sotto il punto di vista operativo saranno per lo più esterne alla piattaforma: prendendo in esempio un distributore automatico, potrebbe essere dovuto ad un

mal funzionamento del distributore. Quindi la parte eseguibile ricade sulle procedure che devono essere sviluppate da contratto e sul loro controllo: le monete inserite dal cliente vengono contate, viene elaborato l'inserimento di un numero come input per poter arrivare ad erogare il bene richiesto;
- Per uno smart legal code: la situazione è molto più complessa in quanto si tira in ballo anche il discorso legale. È necessario, infatti, includere diritti e obblighi che spettano alle parti che hanno firmato l'accordo. Tali diritti e obblighi sono espressi in scritture legali complesse e possono riguardare azioni individuali e non. È possibile trovare anche obblighi imperativi secondo i quali un'azione non fatta possa essere considerata un'esecuzione sbagliata del contratto e portare a riscontri legali.

3.3.2 Come vengono fatti eseguire

L'esecuzione di un contratto può avvenire attraverso metodi tradizionali o non tradizionali:

- Tradizionale: s'intende il ricorso ai tribunali. C'è un organo giuridico stabilito ed i metodi con cui le parti possono risolvere le controversie sono ben noti. Per atti illegali, hanno il potere di multare, sequestrare beni ecc. Hanno molta esperienza anche per quanto riguarda contratti non rispettati sotto ogni punto di vista, atti che non sono avvenuti (come ad esempio un pagamento) o avvenuti in modo errato. In questi casi, dopo che si è verificata una controversia, i tribunali attribuiscono danni o altri rilievi, a seconda delle situazioni;
- Si possono immaginare altri metodi di applicazione non tradizionali, ad esempio c'è attualmente un dibattito e una sperimentazione sul rinforzo degli smart contract code a livello network senza il bisogno della risoluzione delle controversie. Questo tipo di applicazione viene chiamato anche tamper-proof, cioè a prova di manomissione. Si assume che si abbia l'implementazione perfetta del sistema e che quindi le performance sbagliate o non eseguite non si verificheranno. In altre parole, le controversie non possono verificarsi perché è tutto gestito dal codice dello smart contract. Il codice deve essere definito per intraprendere la giusta azione in risposta alle varie dinamiche di stato che possono verificarsi. Ma si assume che il codice, una volta avviato, non possa più essere interrotto o modificato e, in un sistema veramente inarrestabile, tutte queste possibilità dovrebbero essere anticipate, per questo ci troviamo in una situazione ideale.

L'applicazione tamper-proof è l'obbiettivo finale che le piattaforme che gestiscono transazioni attraverso blockchain si sono imposte. La parte più complicata è il fatto che una volta che uno smart contract code viene istanziato e mandato in esecuzione, non è più possibile tornare indietro. Nel mondo reale, infatti, capita spesso che le disposizioni di un accordo varino dinamicamente, ad esempio per permettere ad un cliente di ritardare o stoppare momentaneamente un pagamento. Questi casi non sono prevedibili e non sarebbe possibile inserirli nel codice.

A livello ideale è possibile immaginare un sistema a prova di manomissione, passando poi alla pratica ci si rende conto che l'obbiettivo è inarrivabile ma resta comunque un punto cardine al quale fare riferimento.

3.4 Smart Contract Template

Uno smart contract template fornisce un framework per supportare accordi legali per strumenti finanziari.

Seguendo i Ricardian Contract, si utilizzano dei parametri per collegare la prosa legale al codice corrispondente, con lo scopo di iniziare a costruire anche delle basi per lo smart legal contract [Fig 3.1].

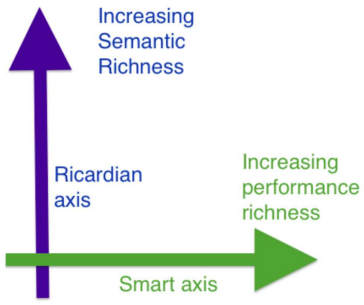

Figura 3.1. L'asse y rappresenta l'aumento della semantica a livello legale (Ricardian Contract), l'asse x rappresenta l'aumento delle prestazioni (Smart Contract)

Uno smart contract deve avere un codice standardizzato per semplificare la fase di controllo; i parametri, solitamente identificati come coppie nome-valore, sono la chiave per dirigere il comportamento che poi viene anche valutato guardando agli input consumati e agli output emessi.

Un accordo viene completamente istanziato in un template standard. Vengono quindi tradotti in codice i risultati delle trattative tra le parti che partecipano al contratto seguendo il più possibile questo modello standard.

I parametri sono la parte critica del contratto perché devono essere estratti dalla prosa legale, ma per sceglierli servono alcuni criteri, la cosa più importante è che bisogna tenere in considerazione che alcuni valori potrebbero non aver impatto nella parte operativa del contratto. È necessario capire quali sono quelli realmente utili allo sviluppo e scartare invece quelli che potrebbero creare confusione.

Una volta effettuata la scelta dei parametri, vengono istanziati, inizializzati ed usati all'interno dello smart contract su tre documenti diversi: nel template, nello smart contract code e nello smart legal code.

Nel primo dei tre, il template, un parametro può non avere un valore definito, mentre in un contratto, nel momento in cui viene inizializzato, deve essere associato necessariamente ad un valore.

Per poter sviluppare uno smart contract template si seguono i seguenti passi:

- *Ricerca a lungo termine*: gli avvocati stanno disegnando contratti legali, collaborando tra loro per poter trovare la miglior soluzione possibile.
  Una volta ottenuto il disegno con la base del contratto, un gruppo operativo con competenze per quanto riguarda il codice, ispeziona il tipo di contratto per identificare i parametri che devono essere trasformati in linguaggio leggibile dalle

macchine. Per poter fare un buon lavoro, il gruppo si pone delle domande: il contratto riesce a rispecchiare le intenzioni? Il funzionamento della parte operativa e i relativi parametri sono corretti?

- *Raffinamento dei parametri*: la maggior parte dei parametri sono semplici, con tipi base, ma se ne possono avere anche di più complessi, creati da espressioni. Generalizzando il problema si tende a complicare la struttura e ci si rende conto che, quelli rappresentati da espressioni, fanno riferimento ad altri semplici.
  Questi parametri ricavati da espressioni vengono chiamati *parametri di ordine superiore*. Vengo ricavati dalla prosa legale ma si punta ad eliminare la complessità sostituendo espressioni verbali complicate con espressioni logiche o aritmetiche. Ciò porterà a ridurre le ambiguità e gli errori.
- *Aumentare l'uso del codice standardizzato*: il passaggio dei parametri allo smart contract code è necessario se vogliamo standardizzare il codice. È molto importante rendere scalabile il codice di un contratto attraverso un template, per poter far riferimento ad una base ogni volta che il sistema dovrà crearne uno nuovo. Quando si va a creare un template, per poterlo rendere utilizzabile in più casi possibili, si tende a generalizzare, a rendere cioè il modello più complesso.
  Il problema più grande al momento è che le banche che iniziano ad utilizzare la tecnologia blockchain, usufruiscono ognuna di piattaforme e framework diversi tra loro. È difficile riuscire a trovare un template standard utilizzabile da tutti i software che troviamo, quindi sono in corso diversi studi per renderlo possibile passando per uno step intermedio: delle funzioni condivise.
  Il risultato finale dovrebbe vedere applicata una logica commerciale comune che permetta di condividere contratti anche tra piattaforme diverse[Fig 3.2].

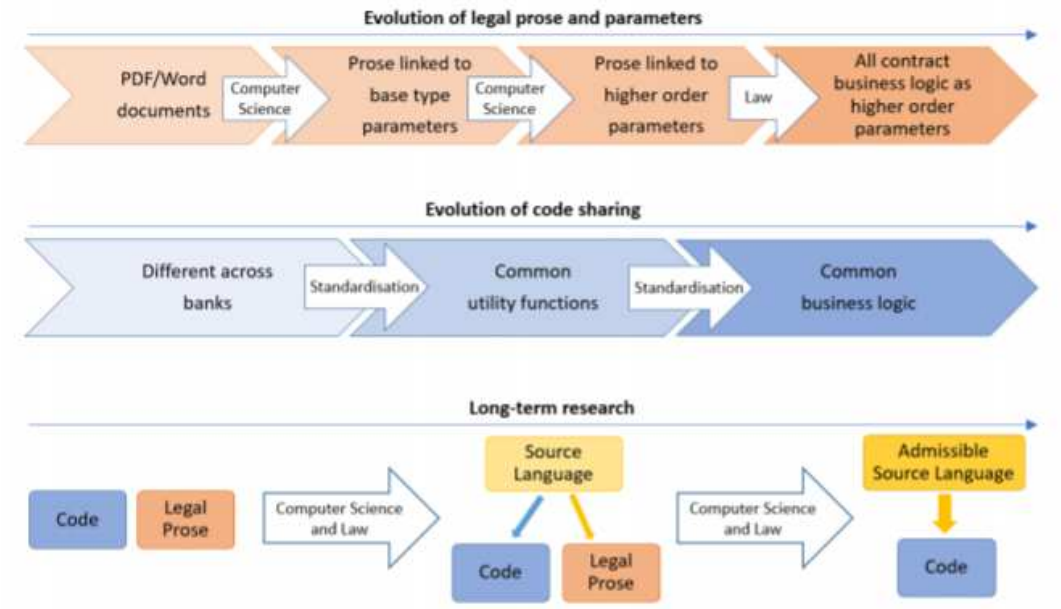

Figura 3.2. le tre evoluzioni di cui abbiamo parlato sopra

4

Ricardian Contract

Già accennato nel capitolo precedente, lo smart legal contract è una parte fondamentale in una blockchain. Sta a rappresentare la parte legale dei contratti che vengono scritti in rete e senza di essi non sarebbe possibile applicarli nel mondo di tutti i giorni anche dal punto di vista legislativo. In particolare, si analizzerà il Contratto Ricardiano, preso sott'occhio dalla società Block.one, leader per le soluzioni blockchain, per poter essere implementato nella piattaforma da loro sviluppata la EOS [19].

4.1 Storia

Viene definito come un documento digitale che specifica i termini e le condizioni di un'interazione tra due o più parti, è scritto in modo leggibile e firmato attraverso una firma digitale crittografata.

È stato creato nel 1995 da Ian Grigg per diventare parte del sistema di pagamento Ricardo, dal quale prende poi il nome. Ian Grigg è uno dei pionieri della crittografia finanziaria, un programmatore esperto che sviluppò i Contratti Ricardiani mentre stava studiando per il titolo di "Master in Business Administration".

Il primo approccio al mondo della moneta virtuale è stato grazie ad un suo amico che lavorava in un'azienda, la DigiCash. Stava sviluppando un sistema per creare pagamenti elettronici. L'azienda fallì, ancora il mondo non era pronto ad accogliere una simile tecnologia, ma Grigg capì che bisognava puntare l'attenzione non sul concetto di soldi virtuali ma su strumenti finanziari in generale.

Nel tentativo di trovare il miglior modo per implementare queste nuove scoperte, Grigg partì provando ad emettere dei legami tra più nodi in una rete internet. Studiando poi da vicino tali legami, si rese conto che si trattava, come tutti gli strumenti finanziari, di un contratto. Si è messo al lavoro per cercare quindi di digitalizzare un contratto, consapevole del fatto che, una volta raggiunto quell'obbiettivo, sarebbe riuscito a digitalizzare completamente ogni strumento e attività finanziaria.

Nel periodo di Grigg ancora non erano disponibili le giuste soluzioni tecnologiche per rendere queste idee realtà, ma al giorno d'oggi abbiamo a disposizione più tecnologie: in particolare la blockchain, un mondo che si sta sviluppando in modo esponenziale, è la struttura perfetta per implementare l'idea di partenza di Ian.

4.2 Come funziona un Contratto Ricardiano

4.2.1 Obiettivi

Gli obiettivi di un contratto ricardiano sono:

- In caso di controversie deve provvedere alla risoluzione del problema. A volte può capitare che il codice restituisca un output non aspettato che causa un danno ad un utente, come ad esempio la perdita di una somma di denaro. I programmi sono complicati e non è possibile calcolare precedentemente tutti gli scenari possibili in cui ci si potrebbe ritrovare. Se capita, appunto, una situazione inaspettata, il contratto ricardiano ha il compito di restituire una descrizione abbastanza dettagliata di quello che ci si sarebbe aspettato dal codice. In questo modo sarà possibile trovare più facilmente una soluzione;
- Deve essere leggibile dall'uomo dato che tratta la parte legale di un contratto, che è quella in cui si riscontrano la maggior parte delle difficoltà, si fa capo a queste scritture in caso di controversie. Inoltre le parti devono aver ben chiaro quale sia lo spirito dell'accordo e per cosa stanno firmando. Una volta sistemata la parte scritta sarà possibile cercare le parole chiave per inserire dei markup e rendere il contratto applicabile dal punto di vista del software.

Tenendo conto di queste necessità Grigg realizzò che i contratti si dividevano quindi in due parti fondamentali: la parte legale, formata dalla prosa, ed i parametri. In poche parole, si tratta di un documento scritto come se fosse un contratto tradizionale, con il quale è possibile presentarsi in tribunale, che poi però viene elaborato e crittografato con una funzione hash per essere utilizzabile da un software.

4.2.2 Sicurezza

Un Contratto Ricardiano, una volta scritto ed emesso nella rete, viene firmato con una firma digitale crittografata. Ogni documento è univoco grazie al suo hash, non può essere modificato una volta che è stato approvato e firmato dalle parti partecipanti al contratto. Questo protegge i proprietari dai tentativi di manomissione.

4.2.3 Gli arbitri

Attraverso l'integrazione dei contratti ricardiani, gli utenti avranno un accordo prestabilito che sarebbero in grado di presentare agli "arbitri della rete". Tali arbitri sono rappresentati da dei nodi all'interno della nostra piattaforma blockchain. Sono chiamati anche notai e devono essere imparziali per poter valutare in modo oggettivo le intenzioni e i doveri originali dell'accordo. Saranno loro poi ad allineare il contratto ricardiano allo smart contract operativo per poi accettarne o meno la pubblicazione.

4.3 Rappresentazione Grafica

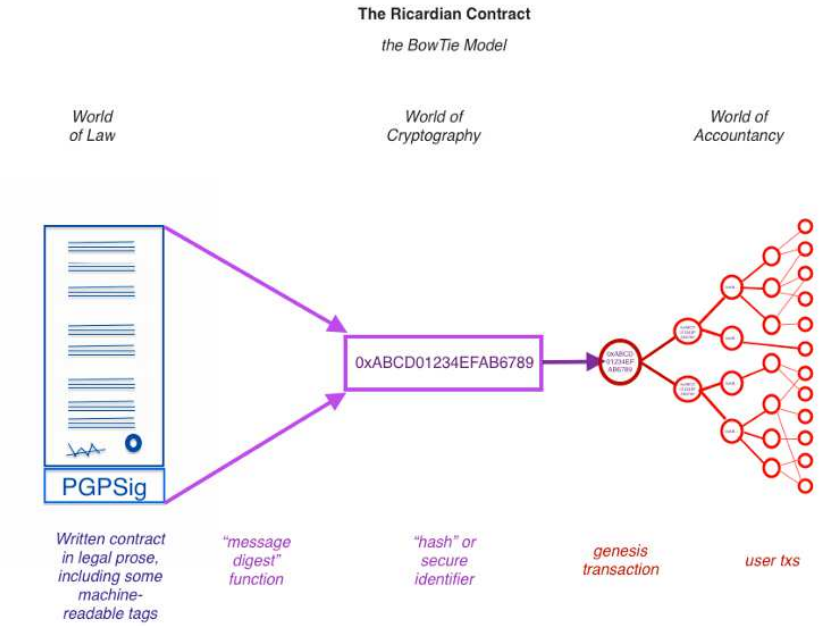

Figura 4.1. Rappresentazione grafica di un contratto ricardiano

4.4 EOS Toolkit

Block.one ha sviluppato il contratto ricardiano per applicarlo alle transazioni che avvengono nella sua piattaforma. È possibile vedere un esempio nel sito di EOS Toolkit: https://eostoolkit.io/transfer

Per poter entrare nella blockchain EOS ci sono delle regole da rispettare, pubblicate nel sito attraverso alcuni articoli. Leggiamo tra questi anche uno che cita:

"Users who call regproducer agree to, and are bound by, the regproducer Ricardian Contract."

Che tradotto significa: Ogni utente che chiama il *blocco produttore*, per poter creare una transazione, accetta il Contratto Ricardiano creato dallo stesso blocco ed è vincolato da questo.

Nel mondo EOS, la transazione basilare è quella del trasferimento di token da un utente ad un altro. Per ogni trasferimento verrà creato un contratto ricardiano apposito, secondo una scrittura di default, un template.

I parametri richiesti sono quelli illustrati nella fig 4.2:

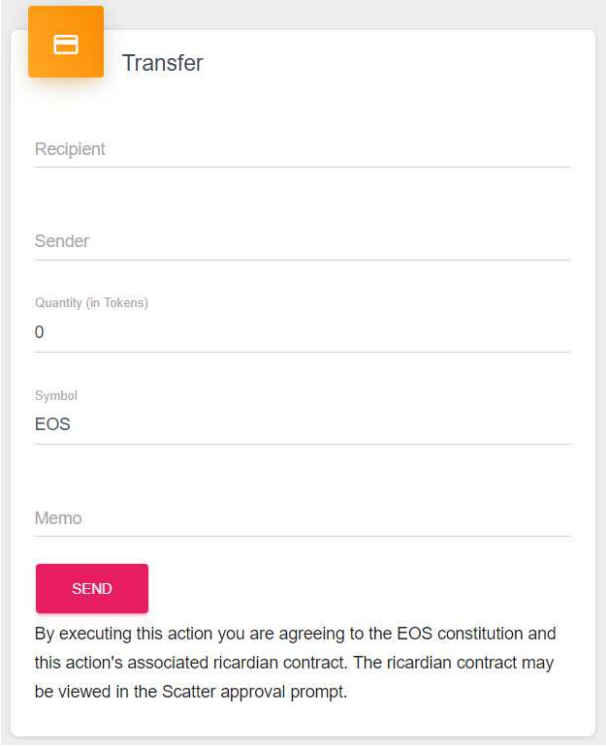

Figura 4.2. Parametri richiesti per una transazione su EOS

Una volta data la conferma dell'invio appare il contratto ricardiano in fig 4.3.

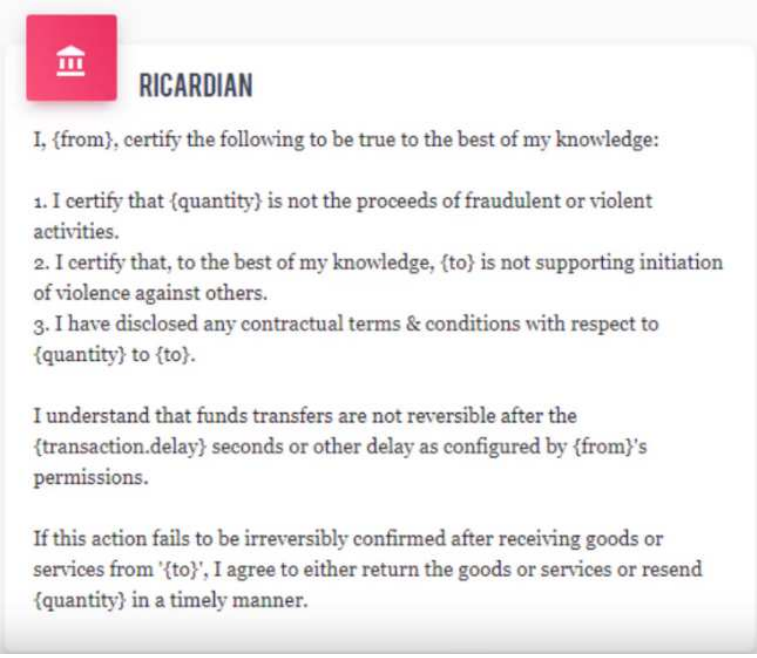

Figura 4.3. Parametri richiesti per una transazione su EOS

Nel momento in cui il contratto verrà approvato, sarà inserito nella rete sostituendo ai placeholder, fra parentesi graffe, i valori che sono stati inseriti nel modulo di trasferimento precedente. È importante far notare che questi placeholder sono i parametri di cui abbiamo parlato in precedenza. Verranno quindi utilizzati anche in fase di elaborazione dagli smart contract code.

Per un computer trasferire somme di denaro o token da un account ad un altro è molto semplice, ma non lo è allo stesso modo nel mondo umano: le truffe sono sempre dietro l'angolo. Il passaggio che include la creazione del contratto ricardiano serve proprio per eludere ogni tentativo di frode.

4.5 Caso d'uso

Nel mondo reale è possibile vedere un'altra applicazione dei contratti ricardiani. Stiamo parlando di OpenBazaar, una piattaforma e-commerce peer-to-peer. Supporta anche pagamenti con la criptovaluta e utilizza i contratti di Grigg come strumento per tracciare le responsabilità delle parti quando scambiano merci tra loro.

Ogni volta che viene conclusa una trattativa su OpenBazaar, esiste un contratto ricardiano per poter tracciale la legittimità dell'accordo. Viene firmato dalle parti partecipanti per poter aumentare la sicurezza ed evitare truffe. Nel caso in cui s'intercorresse ugualmente in un caso di frode, questo contratto avrà validità anche in una corte legale e potrà essere presentato dalla parte vittima per far valere i suoi diritti [Fig 4.4].

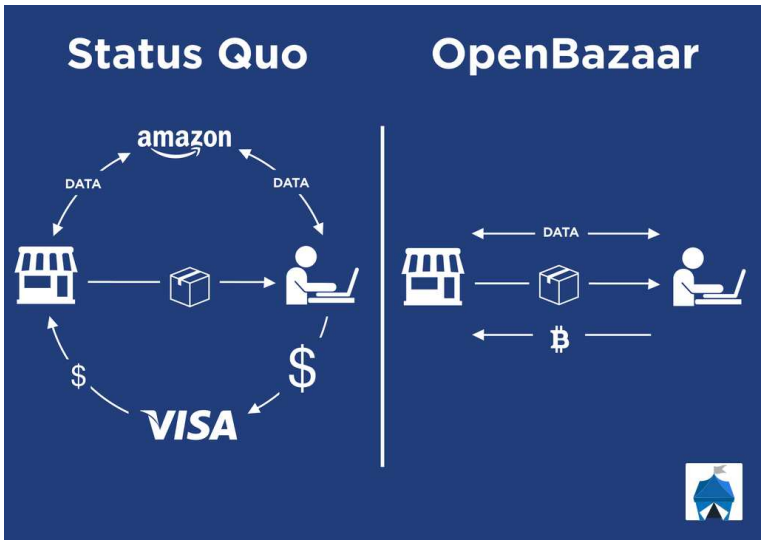

Figura 4.4. Pagamenti su OpenBazaar - https://openbazaar.zendesk.com/hc/en-us/articles/208020193-What-is-OpenBazaar-

5

Piattaforme BlockChain Based

In questo capitolo verranno riportati i risultati dello studio di diverse piattaforme basate sulla tecnologia blockchain. Mettendole a confronto è possibile trovare pregi e difetti di ognuna con lo scopo finale di riuscire a valutare in modo rapido e semplice quale sia la migliore scelta per implementare una rete blockchain che raggiunga gli obbiettivi che ci siamo preposti, tutto dipende dall'ambito in cui ci troviamo.

5.1 Introduzione alle piattaforme

La stragrande maggioranza dei servizi online, delle società e delle aziende, si basa su un sistema centralizzato. È un approccio che è stato utilizzato per secoli. È possibile trovare esempi anche nella vita quotidiana senza il bisogno di ricorrere alla tecnologia. Nonostante abbia diversi difetti, a volte è necessario usare un sistema di questo tipo quando i partecipanti alla rete non si fidano l'uno dell'altro.

Tale metodo centralizzato concentra tutto il controllo in una singola entità, anche con un certo margine d'inaffidabilità. Le app ed i server basati su questo sistema sono estremamente vulnerabili agli attacchi hacker dato che tutte le informazioni vengono salvate, solitamente, in un unico database, o al massimo con poche copie facilmente corruttibili.

Uno dei principali scopi delle piattaforme blockchain è proprio quello di rendere i dati immutabili e incorruttibili. Per arrivare a questo obiettivo hanno tutti adottato un sistema decentralizzato, completamente autonomo, non esiste alcuna sede centrale che regola il funzionamento di tutto il sistema. Un tipo di rete che supporta un meccanismo decentralizzato è la rete Peer-to-Peer, nella quale tutti i nodi partecipanti si trovano allo stesso livello.

Le piattaforme che andremo a confrontare adottano tutte il sistema decentralizzato ed una rete Peer-to-Peer.

È importante specificare anche che tutte supportano gli smart contract, anche se ognuna ha una visione leggermente diversa.

5.1.1 Ethereum

Ethereum è la piattaforma blockchain più importante, al giorno d'oggi, dopo Bitcoin. È stata creata a seguito di una serie di decisioni basate su necessità che non venivano soddisfatte da quest'ultima [1].

Il suo fondatore, Vitalik Buterin, voleva rendere l'esecuzione di smart contract e app decentralizzate autonoma. Infatti, è possibile archiviare registri, spostare fondi secondo precise istruzioni e molte altre cose, senza bisogno di intermediari che intervengano nelle trattative

5.1.2 Quorum

Quorum è una piattaforma Ethereum based, quindi le caratteristiche principali sono quelle di Ethereum, ma è stata creata con lo scopo di avere una maggior focalizzazione su casi d'uso finanziario. Per questo sono state fatte diverse modifiche a livello di gestione della privacy, per permettere agli utenti di poter scambiare anche dati sensibili senza il rischio di attacchi esterni.

5.1.3 Hyperledger

Hyperledger è uno dei progetti Linux Foudation [2]. È stato ideato per far avanzare le tecnologie blockchain per il business attraverso uno sviluppo collaborativo open source.

Gli obiettivi che gli sviluppatori sono:

- creare un framework open source per supportare le transazioni di mercato basate su un ledger distribuito;
- fornire infrastrutture aperte e neutrali;
- supportare la creazione di comunità tecniche per lo sviluppo blockchain;
- diffondere le opportunità di mercato derivanti dall'applicazione di tecnologie blockchain.

All'interno della community di Hyperledger sono già stati sviluppati diversi framework e tool per l'utilizzo delle tecnologie blockchain e in questo documento si è preso in considerazione Hyperledger Fabric.

5.1.4 Corda

Corda s'inserisce nel gruppo delle piattaforme blockchain con ledger distribuito nel 2016, è sviluppata da R3, un'azienda che guida un network di 200 partecipanti tra banche, istituti finanziari, enti regolatori, associazioni e aziende di tecnologia [15].

È nata per superare le difficoltà d'interoperabilità tra le diverse piattaforme che vengono utilizzate dai loro clienti. Infatti, ci si trova in una situazione in cui comunicare fra due reti che non sono del tutto compatibili, porta a inefficienze, rischi e costi. Partendo da queste problematiche di base, Corda è

" il risultato di oltre due anni di intenso lavoro di ricerca e sviluppo da parte di R3 per soddisfare i più alti standard dei servizi finanziari ed è oggi applicabile a qualsiasi scenario commerciale "

5.2 Struttura

5.2.1 Ethereum

Si presenta come una macchina a stati basata su transazioni, appunto, di stato. Quando si verifica una transazione nella blockchain, si passa da uno stato, che rappresenta la situazione prima del verificarsi dell'azione, ad un altro che diventa attuale.

Esistono due tipi di account:

- Account privato, o anche utente esterno, è controllato da chiavi private ed è visualizzabile tramite chiave pubblica. Può inviare o ricevere messaggi esternamente dal suo account verso un altro con una somma di Ether;
- Account contrattuale, controllato da smart contract. Anche questo tipo può ricevere o inviare messaggi, ma in modo automatico, dalle call.

Gli account contrattuali sono creati da una transazione proveniente da un account privato.

Lo stato è registrato in una struttura ad albero chiamata Patricia Tree.

Per quanto riguarda Quorum, il Patricia Tree viene diviso in due strutture: Public State Tree e Private State Tree.

5.2.2 Hyperledger Fabric

la struttura prevede un modello sicuro e robusto per l'identità, la verificabilità e la privacy.

È disegnato per supportare più reti con scopi diversi. Il protocollo deve permettere diversi tipi d'uso per poter spaziare da un mercato ad un altro, e diversi livelli di autorizzazione.

Il *chaincode* è l'elemento principale della piattaforma, si tratta di un programma "transazionale" decentralizzato che viene eseguito nei nodi validatori. È l'equivalente dello smart contract nelle altre piattaforme.

I chaincode sono scanditi da un timer, quelli che appartengono allo stesso gruppo di validatori, che hanno cioè le stesse restrinzioni, possono comunicare tra loro ad esempio attraverso delle call.

Lo stato del sistema è rappresentato dallo stato di ogni chaincode.

5.2.3 Corda

è un framework pensato a livello di mercato e d'industrie, non a livello di singole aziende.

L'elemento principale è lo *state object*: un documento digitale che contiene i dettagli e lo stato di un accordo fra più parti.

Il ledger condiviso è formato da un insieme di state object immutabili.

5.3 Esecuzione e linguaggio dei contratti

5.3.1 Ethereum

I contratti vengono eseguiti nella Ethereum Virtual Machine, in modo controllato ed esterno alla rete.

Il codice dei contratti è scritto in un linguaggio di basso livello: codice EVM (Ethereum Virtual Machine). Ogni byte rappresenta un'operazione eseguita.

5.3.2 Hyperledger Fabric

I chaincode sono eseguiti con dei servizi appositi, implementati per alleggerire il software: delle SandBox. Possono essere scritti in ogni linguaggio di programmazione tradizionale.

5.3.3 Corda

Per eseguire le transazioni, che modificano gli state object, viene utilizzata la sandbox di Java Virtual Machine. Il nucleo di Corda è scritto in linguaggio kotlin, un linguaggio simile a java ma molto più sintetico. È possibile sintetizzare interi blocchi di linee di codice in Java con una sola riga in Kotlin. Nonostante il nucleo abbia questo linguaggio è possibile comunque implementare contratti e stati in Java, dato che sono compatibili.

5.4 Pubbliche / Private e Permissioned / Non Permissioned

5.4.1 Ethereum

È una rete blockchain pubblica distribuita, non permissioned; in ogni macchina è possibile scaricare un client Ethereum, vedere i dati delle transazioni e partecipare alle validazioni delle transazioni (mining).

5.4.2 Quorum

Uno dei cardini di Quorum è proprio quello di aver introdotto la possibilità di avere transazioni private oltre che quelle pubbliche, diventando una piattaforma permissioned. Le reti utilizzate da Quorum non sono aperte a tutti, per permettere una maggiore focalizzazione su casi di utilizzo finanziario che supportino la privacy. È pensato per essere implementato tra i partecipanti che sono pre-approvati da un'autorità designata.

5.4.3 Hyperledger Fabric

L'idea è che le blockchain pubbliche possono rischiare di compromettere la privacy e la riservatezza di un partecipante, quindi Fabric, adotta una soluzione privata. Inoltre è permissioned, quindi è necessario avere un'autorizzazione per poter entrare in una rete.

5.4.4 Corda

Registra e gestisce accordi in modo privato ed adotta il modello permissioned, ma a differenza delle altre piattaforme con la stessa caratteristica, la piattaforma Corda è concepita per consentire a gruppi di più partecipanti di coesistere e interoperare attraverso la stessa rete aperta.

5.5 Operazioni

In ogni piattaforma le operazioni che è possibile eseguire sono simili ma vengono definite con nomi differenti.

5.5.1 Ethereum

In Ethereum troviamo le seguenti operazioni:

- Transazione: pacchetto di dati firmato, contiene un messaggio, è inviata da un account esterno;
- Messaggio: può essere creato da un contratto automatizzato, non da attori esterni, esiste solo nell'ambiente di Ethereum, si generano in particolare quando un contratto-codice esegue una call;
- Transazione di stato: passaggio da uno stato ad un altro attraverso una transazione (precedentemente definita).

5.5.2 Quorum

Per Quorum le operazioni sono le stesse con una divisione fra le transazioni:

- Transazioni pubbliche: il payload è visibile a tutti i partecipanti della rete, gestito con lo standard di Ethereum;
- Transazioni private: il payload è visibile solo ai partecipanti della rete le cui chiavi pubbliche sono specificate nel parametro che è stato aggiunto per Quorum [PrivateFor]. Non seguono lo stesso standard di Ethereum, sono gestite da Constellation [1].

[1] Constellation: sistema utilizzato per l'invio delle informazioni in modo sicuro all'interno di una rete di partecipanti

5.5.3 Hyperledger Fabric

Ci sono due tipi di transazioni che possono essere eseguite:

- Code-deploy: interagisce con una parte di chaincode;
- Code-invoking: è un API call ad una funzione chaincode.

5.5.4 Corda

Qualsiasi operazione su Corda viene eseguita attraverso *transazioni*, o flussi, che consumano state object e ne creano di nuovi formando catene di provenienza. Una transazione corrisponde al ciclo di vita di uno state object che può essere consumato e ricreato ad ogni funzione eseguita. Le funzioni che possono consumare uno state object o che possono crearlo (attraverso le transazioni appunto) sono specificate nei dati dello stesso. I flussi sono regolati dai *contract code* (simili agli smart contract), che specificano dei vincoli nell'evoluzione dei dati.

5.6 Valuta

5.6.1 Ethereum

I partecipanti sviluppano e gestiscono contratti utilizzando le risorse computazionali della rete. L'uso di queste risorse viene remunerato con una speciale moneta virtuale denominata Ether.

Ether ha di fatto un doppio ruolo: da una parte è essa stessa la potenza elaborativa necessaria per produrre i contratti e dall'altra rappresenta la criptovaluta che permette di "pagare" per la realizzazione dei contratti.

Ethereum conta poi su un Internal Transaction Pricing Mechanism denominato *Gas* che ha lo scopo di ottimizzare le risorse della rete, di prevenire lo spam e di allocare le risorse in modo proporzionato e corretto in funzione delle richieste.

Viene stabilito un costo base in Gas per ogni transazione, dove saranno poi da sommare i costi necessari per le operazioni che servono per interagire con il contratto. Si deve pagare ogni tipo di calcolo, ogni step costa 1 Gas, tranne per alcune operazioni computazionalmente più complesse come ADD che ne costa 3. Se non ci sono abbastanza Ether nell'account del richiedente, al fine di eseguire la transazione, questa non è considerata valida.

Su Quorum il prezzo del Gas è stato rimosso, rimane solo un parametro nelle transazioni (GasLimit) che tiene conto di un livello massimo di Gas. Serve come meccanismo per interrompere cicli infiniti di esecuzione e limitare il numero di transazioni incluse in un blocco.

5.6.2 Hyperledger Fabric

Gli sviluppatori di questa piattaforma si sono chiesti se le aziende hanno reale interesse ad entrare con i loro affari nel mondo della criptovaluta. La risposta è stata che a loro non interessa molto il metodo di pagamento, purché vengano pagati, ma,

considerando che ancora la moneta virtuale non è così tanto estesa, si accontentano di acquistare e vendere con dollari, euro, sterline ecc.

Quello che potrebbe interessare molto di più ad un'azienda è la tecnologia che ne sta alla base, appunto la blockchain. È per questo che non esiste una criptovaluta nativa per questa piattaforma anche se, avendo una filosofia di progettazione modulare estendibile, è sempre possibile implementarne una all'interno della propria rete.

5.6.3 Corda

L'approccio è simile a quello della piattaforma precedente, non esiste una moneta nativa, ma l'architettura consente l'implementazione di una criptovaluta per incentivare la partecipazione alla rete e pagare per i servizi. Tale moneta può essere sia a livello di piattaforma privata, sia specifica per un mercato particolare che opera sulla rete Corda più ampia.

5.7 Privacy e Identità

5.7.1 Ethereum

I dati per essere considerati validi devono essere condivisi ed ampiamente distribuiti. Ci si aspettava una situazione del genere considerando che si tratta di una piattaforma pubblica. La privacy in questo caso non viene considerata, anzi, in rete esistono dei metodi che permettono di diffondere i dati della blockchain anche agli utenti che non partecipano alla rete.

5.7.2 Quorum

Nello sviluppo di Quorum si è cercato di colmare il problema della privacy presente su Ethereum grazie all'implementazione dei contratti privati, ma non solo, anche gli smart contract hanno un livello di privacy in quanto potrebbero contenere strategie d'investimento, dati sulle transazioni o informazioni interne sensibili.

5.7.3 Hyperledger Fabric

Esiste un nodo, chiamato Registration Authority che registra tutte le identità che prendono parte alla rete. Tale nodo ha anche il permesso di gestire tutti gli account e le relative autorizzazioni. Il protocollo dietro a questo meccanismo coinvolge le firme digitali crittografate, maggiore è la richiesta della privacy più vengono richieste crittografie più avanzate. I dati confidenziali sono salvati criptando le transazioni in modo tale che solo gli stakeholder, cioè i diretti interessati, possano decrittografarli ed eseguirli. Essendo una piattaforma privata, le transazioni avvengono solo tra utenti che fanno parte dello stesso gruppo.

5.7.4 Corda

Tutti gli utenti devono accettare determinati parametri per poter comunicare con successo. Ogni partecipante alla rete deve avere un'identità unica, che si distingue da tutte le altre sia per il nome che per la chiave pubblica.

La firma dell'utente che viene utilizzata per firmare le transazioni ed i contratti, è legalmente vincolante.

5.8 Ruoli

5.8.1 Quorum

Su Ethereum non esistono i ruoli tra i nodi, hanno tutti le stesse funzionalità. Sono stati invece implementati su Quorum:

- Voter: i nodi che votano quale blocco sarà la prossima testa canonica, questo diventa valido quando riceve prima una certa soglia di voti;
- Maker: i nodi che creano blocchi inserendo la loro firma che poi servirà per la validazione;
- Voter e Maker: un nodo può avere entrambi i ruoli;
- Observer: un nodo che non ha nessuno dei ruoli precedenti, può solo ricevere blocchi o richiedere transazioni.

5.8.2 Hyperledger Fabric

I ruoli su una rete Hyperledger Fabric sono:

- Validating nodes: nodi che servono per convalidare i blocchi, simile ad un ruolo di notaio;
- Registration Authority: per ogni rete esisterà un unico nodo di questo tipo che serve per registrare e gestire le identità dei partecipanti;
- Partecipanti semplici: propongono le transazioni e partecipano alla rete condividendo il ledger.

5.8.3 Corda

I ruoli su una rete Corda sono:

- Transaction proposer: costruisce le transazioni proposte dai partecipanti in modo conforme ai vincoli dettati dai contratti;
- Identity Authority: registra e gestisce le identità degli utenti che prendono parte alla rete;
- Notary pools: garantiscono l'integrità e il consenso delle transazioni, se quest'ultime non vengono accettate da uno di questi nodi e non ricevono la loro firma digitale, non saranno considerate valide e non potranno essere inserite nel ledger;
- Partecipanti semplici: propongono le transazioni e partecipano alla rete condividendo il ledger.

5.9 Validazione

Se avessimo avuto un servizio centralizzato di fiducia, la validazione di un blocco o di una transazione sarebbe stato banale da implementare, tuttavia, parlando di sistemi decentralizzati c'è bisogno di un sistema di consenso al fine di assicurare che ognuno sia d'accordo sull'ordine delle transazioni.

5.9.1 Ethereum

Per controllare la validità di un blocco, Ethereum esegue il seguente algoritmo:

- Controlla se il blocco precedente esiste ed è valido;
- Controlla se la marca temporale del blocco è più grande di quello precedente ed inferiore a 15 min;
- Controlla che i valori associati all'operazione siano validi;
- Controlla se la POW[2] è valida sul blocco;
- Si ha S[0] lo stato finale del blocco precedente;
- Si genera un errore se viene superato il GASLIMIT;
- S[n] è lo stato finale, con l'aggiunta della ricompensa per il blocco al miner;
- Controlla se lo state root del Merkle Tree sia uguale allo state root finale fornito nell'intestazione del blocco.

Su Quorum la validazione segue lo stesso algoritmo tranne per una piccola modifica che permette di supportare le transazioni private.

5.9.2 Hyperledger Fabric

Le transazioni vengono validate da dei nodi che hanno questo ruolo specifico chiamati Validating Node. La validazione inoltre, è la terza fase del meccanismo di consenso che vedremo in seguito.

5.9.3 Corda

Per Corda la validazione è molto semplice, è riposto tutto nelle mani degli smart contract che sono implementati nella rete. Ogni contratto ha infatti dei vincoli che devono essere rispettati nel momento in cui avviene una transazione, lui stesso negherà la validità se uno di questi non si verifica.

[2] Proof of work: è una misura economica per scoraggiare attacchi o abusi di servizio.

5.10 Algoritmo di Consenso

Uno degli algoritmi nati per il consenso delle transazioni e dei blocchi in un sistema basato su blockchain, è il Proof Of Work. Non sempre però è ritenuto il più vantaggioso, infatti, le piattaforme di cui ci occupiamo, hanno implementato degli algoritmi di consenso diversi.

5.10.1 Ethereum

Utilizza un protocollo chiamato Casper che non sostituirà del tutto POW, ma getterà le basi per il meccanismo di Proof Of Stake [3], si tratta quindi di un ibrido. Nelle fasi iniziali Ethereum userà il primo metodo per elaborare le richieste ma passerà poi al secono per convalidare i checkpoint periodicamente [fig 5.1].

In altre parole, i blocchi verranno estratti tramite POW, ma dopo ogni 50esimo blocco ci sarà un punto di controllo POS in cui interverrà una rete di validatori.

Si vuole far notare che una volta che verrà implementato il protocollo di consenso POS al posto dell'attuale sistema, una rete avrà molti benefici: minore utilizzo di energia elettrica, miglioramento della scalabilità, maggiore sicurezza e minor incentivi di creazione di nuovi blocchi (con il POW corrispondevano a ricompense in Ether che ora non esistono più).

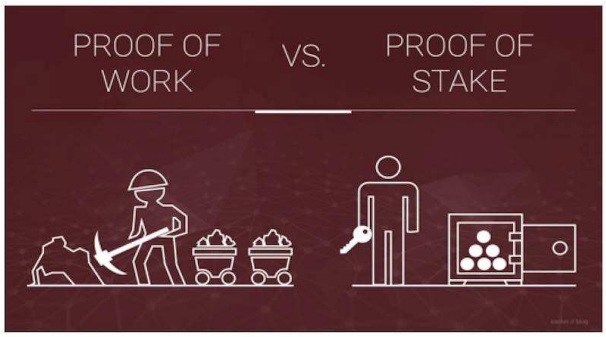

Figura 5.1. POW vs POS: https://www.lecriptovalute.org/2018/09/10/ethereum-casper-protocol-ultimo-aggiornamento-da-pow-a-pos/

5.10.2 Quorum

L'algoritmo di consenso implementato su Quorum è l'Istanbul Byzantine Fault Tollerance (Istanbul BFT).

[3] Proof of Stake: un utente deve mostrare di possedere di un certo numero di valute digitali per l'estrazione o la convalida delle transazioni a blocchi. Il protocollo di consenso PoS non richiede l'uso di potenti computer, eliminando così i costi energetici. Le valute del PoS sono mille volte più convenienti

Prima d'iniziare a costruire blocchi, i nodi scelgono attraverso un algoritmo round-robin, uno di loro come proponente. L'algoritmo si sviluppa nei seguenti stati:

- New Round: raccoglie le transazioni in attesa, propone un blocco e lo trasmette agli altri nodi;
- Pre-Prepare: ogni validatore entra in questo stato quando riceve il messaggio di pre-prepared;
- Prepared: quando il validatore riceve 2F+1 messaggi di prepare provenienti da validatori noti, manda un messaggio di convalida ed è pronto ad inserire il blocco nella catena;
- Committed: quando un validatore riceve 2F+1 messaggi di commit entra in questo stato e può tentare d'inserire il blocco nella catena firmandolo con i 2F+1 messaggi ricevuti;
- Final Committed: blocco inserito nella catena, si cerca un nuovo proponente;
- Round Change: quando si verifica qualche problema negli altri stati, il validatore aspetta che gli vengano recapitati i messaggi giusti per poter tornare allo stato di new round.

5.10.3 Hyperledger Fabric

Il consenso viene raggiunto attraverso 3 step:

- Approvazione della transazione da parte dei partecipanti;
- Ordinazione, in seguito all'approvazione della transazione in se, viene approvata anche la posizione in cui questa verrà inserita nel ledger;
- Validazione, viene preso un blocco di transazioni approvate ed ordinate e viene convalidata la correttezza dei risultati, comprendendo anche il controllo della politica del double-spending, cioè che quel denaro non sia già stato impiegato per un'altra transazione.

Ogni rete Fabric può implementare diversi modelli di approvazione, ordinazione e validazione in base alle proprie esigenze; non esiste un algoritmo preimpostato. Questo tipo di ideologia viene chiamato *pluggable*.

5.10.4 Corda

Anche Corda ha un consenso pluggable, ma in questa piattaforma il consenso viene ottenuto in particolare controllando se gli stati consumati da una transazione sono già stati consumati da un'altra. Non è necessario ottenere il consenso di una transazione perché queste vengono già controllate dagli smart contract dove ci sono i vincoli necessari per renderle valide o meno.

Nella rete troveremo un notaio o un gruppo notarile che autentica le transazioni con una firma che approva la scrittura nel ledger. Nel caso in cui si abbia un gruppo di notai a disposizione, ogni utente può decidere a quale affidarsi.

6

Corda

Analisi del funzionamento della piattaforma Corda a livello pratico.

6.1 Preparazione

Per fare qualche test si è utilizzato il sistema operativo Windows 10. Attraverso i passaggi qui descritti, sarà possibile seguire passo passo la configurazione che è stata eseguita nella macchina [6].

Per funzionare, la nostra macchina dovrà per prima cosa aver installato Java che è strettamente collegato con Kotlin, il linguaggio base di Corda.

6.1.1 IntelliJ IDEA

È un ambiente di svilluppo integrato (IDE) per Java, è sviluppato da JetBrains. Gli sviluppatori Corda consigliano il suo utilizzo dato che supporta molto bene anche il linguaggio Kotlin.

È importante avere un'accortezza durante la fase d'installazione nel momento in cui ci troveremo nella finestra "Installation Options": nella voce "Create Associations" spuntare l'opzione ***.kt*** che ci permetterà d'installare anche il tool che ci permette di lavorare con Kotlin.

Sarà inoltre possibile configurare Git all'interno di IntelliJ IDEA per utilizzarli in modo integrato.

Configurazione di Corda

Per scaricare il source e configurare la piattaforma vera e propria eseguire utilizzando Git, il comando:

git clone https://github.com/corda/corda.git

6.1.2 Gradle

È un software per l'automazione dello sviluppo. Gradle è stato progettato per automatizzare il processo di generazione di un progetto, cercando di tenere traccia della fase attuale dello stesso e determinare ciò che dovrebbe essere fatto successivamente per migliorarlo [4].

Si tratta di uno strumento open source che cerca di trarre vantaggio dall'aiuto di più sviluppatori che tutt'oggi ci stanno lavorando, ma questo non significa che non è ancora all'altezza di affrontare gli incarichi per il quale è stato creato.

In questo progetto verrà nominato più volte e dovremo sempre tenerlo aggiornato prima di mandare in esecuzione il nostro codice. Non è però necessario fare l'installazione in quanto, nel momento in cui viene scaricata la piattaforma Corda, nello step precedente, viene fornito un wrapper che può essere eseguito dalla riga di domando.

Per configurarlo, posizionarsi nella cartella *corda* ed eseguire dalla linea di comando:

gradlew.bat [taskName]

Dopo questi semplici passaggi è possibile aprire IDEA, fare una configurazione se è la prima volta che viene utilizzato, e lanciare il progetto da "Open" scegliendo la cartella "corda" che è stata precedentemente clonata con Git.

Nel momento in cui avremo aperto il nostro progetto uscirà un pop-up in basso a destra: "Event Log". Cliccandoci apparirà una finestra "Import Gradle Project". Se la configurazione Java è corretta, verrà impostato automaticamente un valore per la variabile "Gradle JVM", altrimenti significa che c'è qualche errore nella configurazione precedente.

Nota: un errore che si potrebbe fare è quello di non aver impostato le variabili d'ambiente Java.

6.2 Tender

Una volta pronto l'ambiente di sviluppo Corda, è possibile seguire dei tutorial che spiegano come implementare una CorDapp che possa essere mandata in esecuzione in una rete.

Basandoci su degli esempi già esistenti nel sito R3, è nato Tender un'applicazione che sia in grado di gestire delle aste in modo decentralizzato.

Su Corda, gli elementi più importanti sono: i nodi, gli stati, i contratti e i flussi. Ognuna di queste parti è implementata con file in modo separato, possiamo trovarli scritti sia in Java che in Kotlin, in questo progetto useremo Kotlin.

Quest'asta potrà essere sia pubblica a tutti i nodi della rete, sia limitata ad un certo numero di partecipanti.

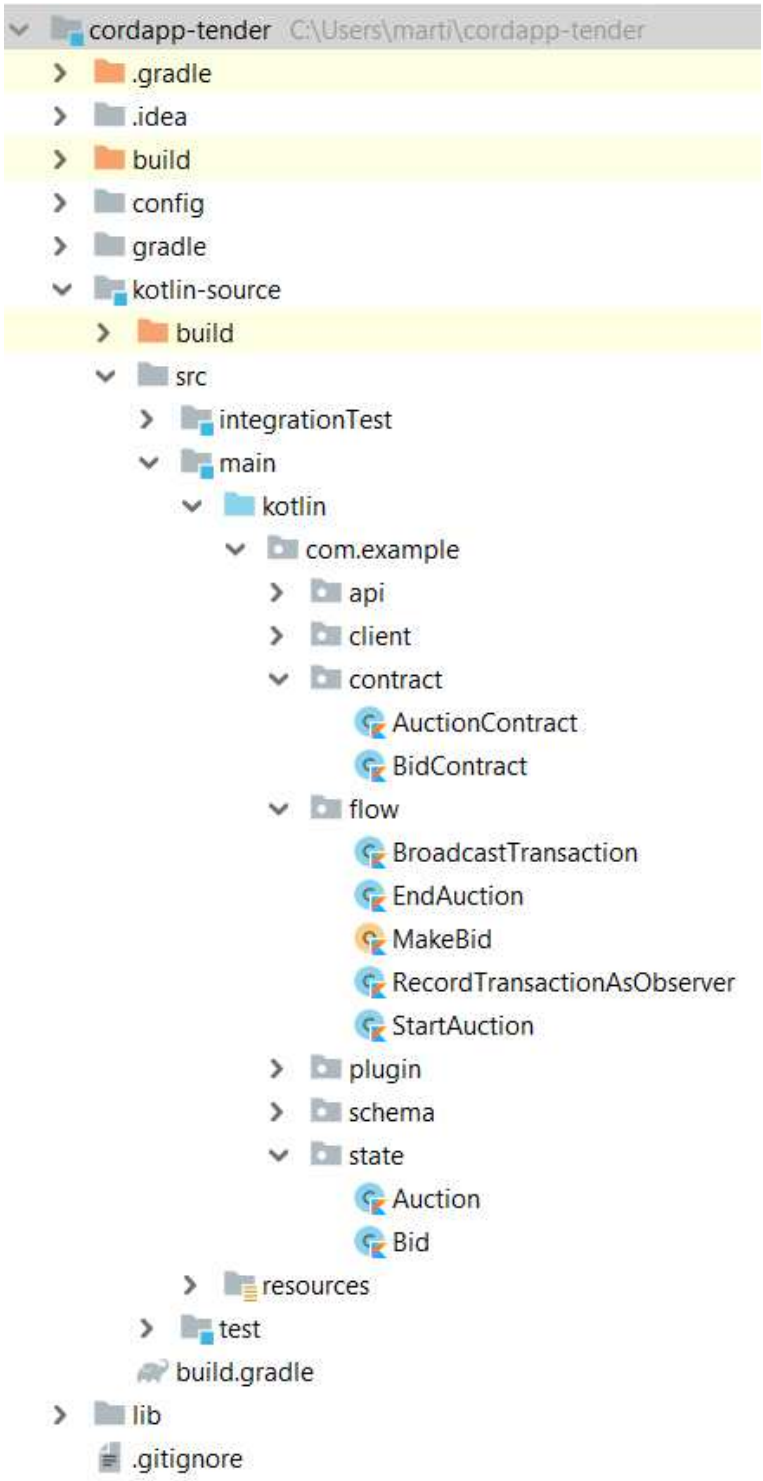

Figura 6.1. Directory map

6.2.1 Nodi

In ogni CorDapp, sarà necessario un file in cui siano dichiarati i nodi che prenderanno parte alla rete: build.gradle [Appendice].

Questo file deve essere eseguito prima di lanciare l'applicazione, serve infatti per generare i nodi che ne prenderanno parte.

Dopo aver eseguito il seguente comando, *gradlew deployNodes* , si creeranno delle nuove directory, quelle che nella mappa sono di colore giallo (Fig 6.1), dove verranno inserite tutte le configurazioni dei nodi che verranno mandati in esecuzione con il prossimo comando:

kotlin-source/build/nodes/runnodes

Una volta eseguito il secondo comando si apriranno dei terminali, due per ogni nodo. Il primo serve per interagire con la rete attraverso linea di comando (fig 6.2), mentre il secondo è il lato web; nella CorDapp Tender è infatti disponibile anche un'API, cioè un'interfaccia lato web, per rendere il servizio utilizzabile anche da utenti meno esperti.

Per interagire con l'applicazione attraverso l'interfaccia (fig 6.3) sarà necessario aprire un browser e digitare:

- Nodo A: localhost:10009/web/example/
- Nodo B: localhost:10009/web/example/
- Nodo C: localhost:10009/web/example/
- Auctioneer: localhost:10009/web/example/

Il valore del localhost corrisponde con la webport dichiarata nel file build.gradle.

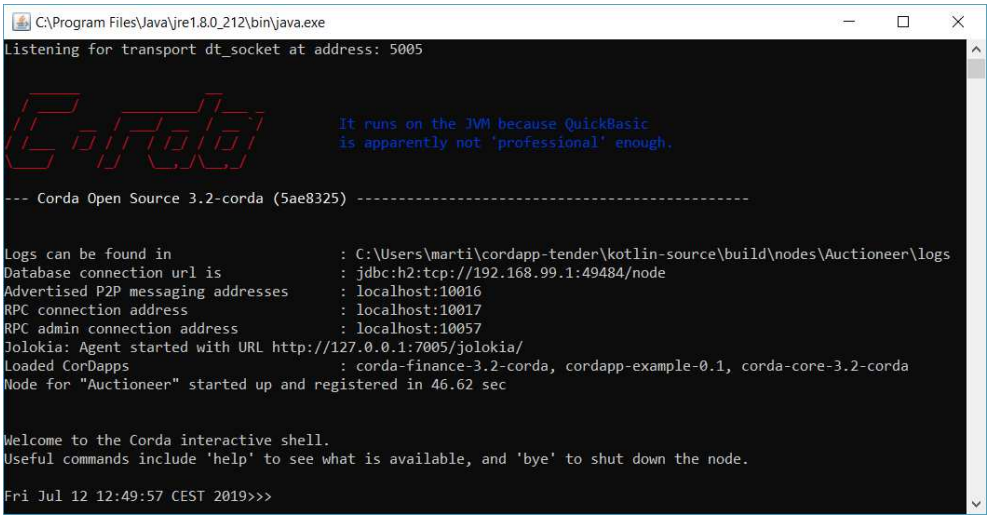

Figura 6.2. Terminale del nodo Aucioneer

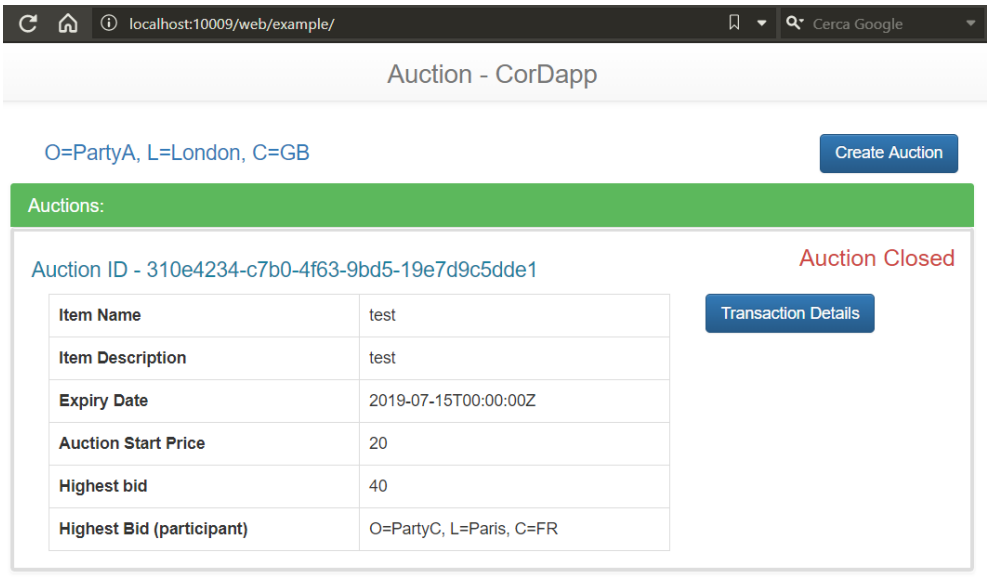

Figura 6.3. Interfaccia grafica del nodo Auctioneer

6.2.2 Stati

In Corda, tutti i dati che vengono salvati nel ledger condiviso sono rappresentati da stati.

Il primo passo per creare quest'applicazione è definire due nuovi tipi di stato per rappresentare un'asta e un'offerta: li chiameremo Auction [Appendice] e Bid.

Si è quindi creata la struttura in cui verranno salvate tutte le informazioni dello stato.

Prendendo come esempio Auction, avremo bisogno di salvare:

- nome dell'asta;
- descrizione;
- prezzo d'inizio;
- nome di chi ha proposto l'asta;
- nome di chi ha fatto l'offerta più alta, all'inizio sarà nullo e ad asta chiusa sarà il nome di chi ha vinto;
- offerta più alta;
- se l'asta è ancora attiva o meno, attraverso un booleano;
- lista dei partecipanti;
- un identificatore univoco.

6.2.3 Flussi

I flussi sono la parte operativa dell'applicazione, cioè dove sono scritte le funzioni che devono essere eseguite nel momento in cui viene richiesta un'azione.

È importante ricordare che ogni azione che viene eseguita in corda viene rappresentata sottoforma di transazioni, infatti in questi file, troveremo proprio il ciclo di vita della transazione.

Nel progetto trattato le transazioni più rilevanti sono:

- StartAuction: per inizializzare una nuova asta;
- MakeBid: gestisce una nuova offerta da parte di un partecipante;
- EndAuction: termina un'asta.

6.2.4 Contratti

Sotto la cartella *contract* troveremo invece i file che controllano che tutti i dati siano validi e che sia possibile eseguire le operazioni che vengono richieste dal sistema. Sono quindi la parte che gestisce, oltre alla parte funzionale, anche quella legale. I vincoli inseriti in questi file devono essere concordati con i partecipanti e rappresentano le regole che questi s'impegnano a rispettare per il corretto funzionamento del sistema. Per questo motivo troveremo tali vincoli sia in linguaggio semplice e leggibile, sia con la traduzione in codice leggibile dalla macchina.

Il sistema accederà a tali file ogni volta che viene creata una nuova asta o una nuova offerta per validare o meno l'azione.

7

Conclusioni

Attraverso questo studio si è approfondita la nuova tecnologia BlockChain che si sta sviluppando sempre di più in questi ultimi anni. Nata grazie ad una rete pubblica si vuole ora cercare di renderla sempre più presente nella vita di tutti i giorni trovando nuovi modi per utilizzarla.

Le sue caratteristiche permettono infatti di creare dei sistemi a prova di manomissioni, quindi sono molto più sicuri rispetto ad ogni altro meccanismo centralizzato che si possa adottare, ma oltre ad avere un livello di sicurezza elevato, la blockchain è anche molto flessibile e adattabile ad ogni tipo di situazione. Per questo secondo motivo infatti, è possibile adottarla per ogni scopo, partendo dalla criptovaluta, passando per sistema bancari e vendite online, fino ad aste semplici o più complesse come le gare d'appalto.

Nel momento in cui un contratto in una blockchain sarà scritto, in modo che sia accettabile anche sotto il punto di vista legale, è possibile diminuire il potere delle istituzioni tradizionali rendendo i privati più indipendenti e permettendo loro di diminuire i costi di gestione dei contratti, cioè saranno in grado di accordarsi direttamente con chi prende parte all'accordo. In caso di controversie, sarà più semplice la risoluzione del problema, l'obiettivo ideale è quello di dover non dover più far ricorso a tribunali classici.

Con il progetto qui sviluppato si è voluta gettare la base per aiutare l'implementazione di qualche progetto più oneroso in termini di risorse e tecnologie.

8

Appendice - Script Rilevanti

8.1 Configurazione Nodi

```
task deployNodes(type: net.corda.plugins.Cordform, dependsOn: ['jar']) {
    directory "./build/nodes"
    node {
        name "O=Notary,L=London,C=GB"
        notary = [validating : false]
        p2pPort 10006
        cordapps = ["$corda_release_group:corda-finance:$corda_release_version"]
    }
    node {
        name "O=PartyA,L=London,C=GB"
        p2pPort 10007
        rpcSettings {
            address("localhost:10008")
            adminAddress("localhost:10048")
        }
        webPort 10009
        cordapps = ["$corda_release_group:corda-finance:$corda_release_version"]
        rpcUsers = [[user: "user1", "password": "test", "permissions": ["ALL"]]]
    }
```

Figura 8.1. File build.gradle - nodo notaio e nodo A

8.2 Stato

```
data class Auction(
        val name: String,
        val description: String,
        val startPrice: Int,
        val ExpiryDate: Instant,
        val owner: Party,
        val auctionWinner: Party ?= null,
        val highestBid: Int,
        val AuctionActive: Boolean,
        val AuctionParticipants: List<Party>,
        override val linearId: UniqueIdentifier = UniqueIdentifier()
) : LinearState, SchedulableState {
    override val participants: List<AbstractParty> = if (auctionWinner!=null)
        listOf(owner,auctionWinner) else listOf(owner)

    override fun nextScheduledActivity(thisStateRef: StateRef,
                                       flowLogicRefFactory: FlowLogicRefFactory): ScheduledActivity? {
        return ScheduledActivity(flowLogicRefFactory.create(EndAuction::class.java,linearId.toString()), ExpiryDate)
    }
}
```

Figura 8.2. Auction.kt - Codice dello stato Aucion

8.3 Contratto

```
    private fun verifyCreate(tx: LedgerTransaction, signers: Set<PublicKey>) = requireThat { this: Requirements
        // Assert we have the right amount and type of states.
        "The can only one input state in an create bid transaction." using (tx.inputStates.size == 1)
        "There must be two output states in an create bid transaction." using (tx.outputStates.size == 2)

        val auctionInput = tx.inputsOfType<Auction>().single()
        val auctionOutput = tx.outputsOfType<Auction>().single()
        val bidOutput = tx.outputsOfType<Bid>().single()

        // Assert stuff about the bid in relation to the auction state.
        "The bid must be for this auction." using (bidOutput.auctionReference == auctionOutput.linearId)
        "The auction must be updated by the amount bid." using (bidOutput.amount == auctionOutput.highestBid)
        "The bid must be higher than start price" using (bidOutput.amount > auctionOutput.startPrice)
        "The bid must be higher than highest bid" using (bidOutput.amount > auctionInput.highestBid)

        // Assert correct signer.
        "The auction must be signed by the manager and bidder." using
                (signers.containsAll(listOf(auctionInput.owner.owningKey,bidOutput.bidder.owningKey)))
    }

}
```

Figura 8.3. Contratto - Creazione di una nuova offerta

Riferimenti bibliografici

1. Ethereum e Smart Contract: Come Funziona. https://www.fxempire.it/education/article/ethereum-e-smart-contract-come-funziona-137221.
2. Hyperledger Fabric: Scalabilità, riservatezza e prestazioni in un'architettura blockchain modulare. https://www.meditchain.com/la-rete/hyperledger-fabric/.
3. Ethereum: Libro Bianco - White Paper. http://www.ethereum-italia.it/?page_id=160, 2015.
4. Gradle: che cos'è? https://www.androidos-lab.it/2015/02/01/gradle-che-cose/, 2015.
5. Corda Technical Whitepaper. https://www.r3.com/reports/corda-technical-whitepaper/, 2016.
6. Getting set up. https://docs.corda.net/releases/release-M8.2/getting-set-up.html, 2016.
7. Making Sense of Blockchain Smart Contracts. https://www.coindesk.com/making-sense-smart-contracts, 2016.
8. The law and the legality of smart contracts. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2959166, 2016.
9. Che cos'è e quali sono gli ambiti applicativi di Ethereum. https://www.blockchain4innovation.it/esperti/cose-quali-gli-ambiti-applicativi-ethereum/, 2017.
10. Comparison of Ethereum, Hyperledger Fabric and Corda. https://pdfs.semanticscholar.org/00c7/5699db7c5f2196ab0ae92be0430be4b291b4.pdf, 2017.
11. Cos'è la blockchain? - Marco Cavicchioli. https://www.youtube.com/watch?v=jO-27dESTwE, 2017.
12. La borsa di Sydney è la prima al mondo ad adottare la tecnologia blockchain. https://www.agi.it/innovazione/borsa_sydney_blockchain-3212528/news/2017-12-08/, 2017.
13. Smart Contract Templates: foundations, design landscape and research directions. https://arxiv.org/pdf/1608.00771.pdf, 2017.
14. An Introduction to Hyperledger. https://www.hyperledger.org/wp-content/uploads/2018/07/HL_Whitepaper_IntroductiontoHyperledger.pdf, 2018.
15. Che cos'è Corda e come si differenzia dalle altre tecnologie blockchain. https://www.zerounoweb.it/software/blockchain/che-cose-corda-e-come-si-differenzia-dalle-altre-tecnologie-blockchain/, 2018.
16. Cos'è Ethereum e Come Funziona: Guida Introduttiva. https://libertex.com/it/blog/cosa-e-ethereum-e-come-funziona-guida-introduttiva, 2018.

17. Ethereum Casper Protocol ultimo aggiornamento da PoW a PoS. https://www.lecriptovalute.org/2018/09/10/ethereum-casper-protocol-ultimo-aggiornamento-da-pow-a-pos/, 2018.
18. Gli accounts di Ethereum e le Transazioni. https://nextgenerationcurrency.com/gli-accounts-di-ethereum-e-le-transazioni/, 2018.
19. Introduction to Ricardian Contracts. https://www.eoscanada.com/en/introduction-to-ricardian-contracts, 2018.
20. The Corda Platform: An Introduction. https://www.corda.net/content/corda-platform-whitepaper.pdf, 2018.
21. What Are EOS Ricardian Contracts? - EOS Tips. https://www.youtube.com/watch?v=qpUsxsYDoII, 2018.

Ringraziamenti

Il percorso di questa laurea triennale sta volgendo al termine ed è necessario ringraziare chi l'ha reso possibile.
Grazie alla mia famiglia, a mio fratello Michele ed ai miei genitori, Marco e Silvana, che, per quattro lunghi anni, oltre al sostegno economico, mi hanno sempre aiutato nelle scelte più importanti e nei momenti più difficili.
Grazie al professor Luca Spalazzi, che è stato una guida ed un riferimento per il tirocinio e per la stesura di questa tesi.
Grazie ai miei colleghi che hanno condiviso, chi più chi meno, con me questo percorso: Giulia, Angelo, Alessandro, Alessandro, Luca, Lucia e tanti altri.
Grazie ai miei amici, in particolare ad Alessia che non ha mai smesso di esserci.
Grazie a chi c'era quando ho iniziato questo percorso, ma soprattutto grazie a chi è riuscito a supportarmi in questi ultimi mesi, i più difficili.
Grazie alla pallavolo, alla Nova Volley, ad Albatros ed a Puma che mi hanno permesso di scaricare le tensioni accumulate con lo studio allenandomi e giocando partite importanti.
Grazie al Rettore, ai professori e all'UNIVPM tutta.