Università degli Studi di Modena e Reggio Emilia

Dipartimento di Ingegneria “Enzo Ferrari”

Corso di Laurea in Ingegneria Informatica

Piattaforma Blockchain HyperLedger: il caso d’uso di un’asta di autovetture

Relatore:
Prof.ssa Sonia Bergamaschi

Candidato:
Marco Tagliavini

Anno Accademico 2016-2017

Keywords:

Blockchain
Bitcoin
Hyperledger
Distributed Ledger Technology
Smart Contract
Consensus

INDICE

1 Introduzione

All'interno di questo elaborato saranno illustrate le caratteristiche tecniche che permettono il funzionamento di Bitcoin, in particolare la Blockchain, il meccanismo alla base del suo funzionamento.

La blockchain, un registro distribuito e aperto a chiunque, tenendo traccia di tutte le transazioni risolve senza la necessita di un'autorità centrale i problemi tipici delle relazioni tra più persone o gruppi.

L'impossibilita di una sua manomissione è sicuramente l'elemento più innovativo del sistema, la cui applicazione alternativa a Bitcoin sembra in grado di rivoluzionare tutti i sistemi di gestione centralizzata a cui siamo abituati.

Verrà presentato *Hyperledger*, un progetto open source della Linux Foundation che si pone l'obiettivo di portare la tecnologia Blockchain nel settore industriale.

In particolare sarà utilizzato lo strumento *Hyperledger Composer*, il quale, sfruttando il framework blockchain *Hyperledger Fabric*, permette una rapida sperimentazione di applicazioni basate su questa tecnologia.

Mediante Composer, verrà modellata una rete di vendita autoveicoli all'asta, al fine di mostrare come è possibile inserire e trasferire beni tra i partecipanti all'interno di una business network basata su blockchain.

2 Bitcoin

Bitcoin è un progetto open source descritto per la prima volta nel 2009 da *Satoshi Nakamoto* [1], una o più persone delle quali non si conosce l'identità e rappresenta la prima realizzazione del concetto di *"cryptocurrency"* (cripto valuta), una nuova forma di denaro che viene creato e controllato attraverso la crittografia anziché da un'autorità centrale, come avviene normalmente con le valute tradizionali.

2.1 Cos'è

"Bitcoin è un'innovativa rete di pagamento e un nuovo tipo di denaro"

è la definizione che si trova nel sito ufficiale del progetto. [2] Infatti con lo stesso nome si fa riferimento sia alla rete ("Bitcoin" con iniziale maiuscola) sia alla moneta ("bitcoin" con iniziale minuscola).

Questo tipo di valuta elettronica è stata creata per risolvere i problemi di fiducia, trasparenza e responsabilità tra due parti nello scambio di denaro per beni e servizi su Internet, senza l'utilizzo di intermediari.

Bitcoin, infatti, rappresenta la prima rete di pagamento che utilizza una tecnologia distribuita peer-to-peer per operare senza un'autorità centrale: la gestione delle transazioni e l'emissione di denaro sono effettuati collettivamente dalla rete. Essendo il sistema distribuito**,** permette di tenere traccia di tutti i trasferimenti in modo da evitare il *problema del double spending* (spendere due volte). Tutti gli utenti sono a conoscenza di ciò che accade e pertanto non c'è bisogno di una autorità centrale che gestisca le transazioni.

Essendo una moneta anonima, i bitcoin possono essere paragonati ai contanti: si danno ad un'altra persona per pagare un bene o un servizio e dopo averlo fatto non li si possiede più, il tutto senza la necessità di una terza parte che faccia da intermediaria nello scambio.

Figura 1 - Logo Bitcoin

2.2 Come funziona

Dal punto di vista di un utente che utilizza Bitcoin come valuta, è sufficiente installare sul proprio dispositivo (cellulare, desktop, hardware [4] o web) un'applicazione *"portafoglio Bitcoin"*. Una volta che il portafoglio è stato installato, verrà generato un indirizzo Bitcoin, ovvero un identificatore di 26-35 caratteri alfanumerici, non contenente nessun riferimento ai dati reali dell'utilizzatore.

Questo indirizzo potrà essere inviato ad altri utenti, i quali lo potranno utilizzare per inviare pagamenti all'utente. Allo stesso modo, l'utente potrà inviare denaro ad altri utenti semplicemente conoscendo i loro indirizzi.

Figura 2 - Una transazione è possibile mediante lo scambio dell'indirizzo Bitcoin.

I bitcoin vengono inviati (o registrati) da un indirizzo a un altro. Ogni utente può possedere uno o più indirizzi. Ogni transazione viene trasmessa alla rete ed inclusa in una catena di blocchi in modo che gli stessi bitcoin non possano essere spesi due volte. Dopo un'ora o due, ogni transazione è bloccata nel tempo dall'enorme quantità di potenza di calcolo che continua ad estendere la catena. Utilizzando queste tecniche, Bitcoin fornisce una rete di pagamento veloce, estremamente affidabile ed utilizzabile da chiunque.

2.3 La tecnologia

La tecnologia che permette il funzionamento della rete Bitcoin e in realtà una combinazione di più tecnologie già esistenti. Infatti vengono utilizzate la crittografia a chiave pubblica, la rete peer-to-peer e la proof-of-work per elaborare e verificare i pagamenti.

2.3.1 Indirizzo Bitcoin: Crittografia a chiave pubblica

L'indirizzo Bitcoin che viene scambiato per inviare e ricevere pagamenti è la *chiave pubblica* del sistema di *crittografia asimmetrica*. [5] La crittografia è necessaria per garantire la privacy del mittente e del destinatario degli scambi rendendo incomprensibili a soggetti non autorizzati le informazioni, essendo le transazioni pubbliche.

Nella crittografia a chiave pubblica, ogni utente possiede una coppia di chiavi:

- la chiave pubblica, usata per codificare le informazioni, la quale sarà condivisa con il destinatario;
- la chiave privata, che invece resta segreta e che viene utilizzata per la decodifica delle informazioni ricevute.

Ad esempio, se Alice vuole ricevere un messaggio segreto da Bob, manda a Bob la sua chiave pubblica. Bob usa la chiave pubblica di Alice per cifrare un messaggio che manda ad Alice, la quale è l'unica ad avere la chiave per decifrarlo. Chiunque può veder passare il messaggio, ma non può decifrarlo, in quanto non ha la chiave privata necessaria per decodificare il contenuto. Alice deve però tenere al sicuro la chiave privata, infatti chiunque possiede questa chiave è in grado di decodificare il messaggio.

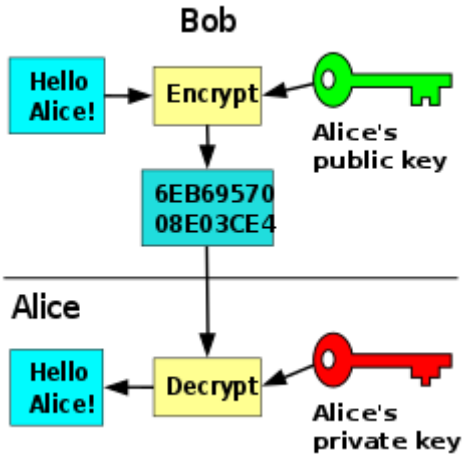


Figura 3 - Crittografia a chiave pubblica: vengono utilizzate chiavi diverse per la cifratura e decifratura.

2.3.2 Portafoglio: la Blockchain

Il *portafoglio Bitcoin* non è altro che un file contenente tutte le transazioni confermate avvenute dalla creazione della moneta virtuale e memorizzate come una catena di blocchi concatenati tra loro, da qui nasce il nome *blockchain*.

La *blockchain* pertanto è il registro pubblico e condiviso sul quale si basa l'intera rete Bitcoin. In questo modo, i portafogli possono calcolare il loro bilancio spendibile e le nuove transazioni possono essere verificate, al fine di dare la possibilità al proprietario di poter spendere solamente bitcoin che sono effettivamente nella sua disponibilità. L'integrità e l'ordine cronologico della blockchain sono protetti attraverso crittografia.

2.3.3 Transazioni

Una *transazione* è un trasferimento di valori tra portafogli Bitcoin e viene inclusa in un blocco nella blockchain una volta confermata. I portafogli Bitcoin contengono la *chiave privata*, che viene utilizzata per firmare le transazioni fornendo la prova matematica che esse provengono dal portafoglio del proprietario. La firma impedisce anche che la transazione, una volta eseguita, possa essere alterata da malintenzionati. La chiave privata resta segreta ed è memorizzata all'interno del dispositivo su cui è installata l'applicazione portafoglio.

Tutte le transazioni vengono trasmesse in modo broadcast ai nodi ed in genere iniziano ad essere confermate dalla rete nei 10 minuti successivi, attraverso un processo chiamato *mining*.

2.3.4 Processo di *mining*

Il *mining* (estrazione dei dati) è un *sistema di consenso distribuito* utilizzato per confermare la validità delle transazioni in attesa includendole nella blockchain.

Il mining viene svolto dai *miner* (minatori), ovvero alcuni soggetti partecipanti alla rete peer-to-peer che si occupano della verifica delle transazioni. Sono in genere calcolatori o reti di calcolatori con enormi potenze di calcolo, i quali svolgono i calcoli crittografici necessari al fine di generare nuovi blocchi all'interno dei quali memorizzare le transazioni in attesa di conferma. Questo processo è chiamato *proof-of-work* (prova di lavoro, vedere il paragrafo 2.3.6) e serve a mantenere un ordine cronologico nella blockchain, proteggere la

neutralità della rete e consentire ai diversi nodi di concordare sullo stato del sistema.

2.3.5 Timestamp Server

È necessario adottare un metodo per far sì che il beneficiario sappia che i precedenti proprietari non abbiano firmato alcuna transazione precedente a quella che lo riguarda, al fine di evitare il problema della doppia spesa. Generalmente questa operazione è svolta da un'autorità fiduciaria centrale, la quale conosce e controlla tutte le transazioni. Per fare lo stesso ma senza un'autorità di fiducia, le transazioni devono essere annunciate pubblicamente, e attraverso l'utilizzo di un *Timestamp Server* (Server di marcatura temporale) i partecipanti potranno concordare sull'ordine in cui esse sono state ricevute.

Per implementare un server di marcatura temporale distribuito su base peer-to-peer, viene utilizzata una soluzione simile ad *Hashcash* [6], un algoritmo di tipo *proof-of-work* proposto nel 1997 da Adam Back ed usato per rendere costoso in termini di tempo di elaborazione, e quindi di energia consumata, l'invio di spam via email. Il problema ha la caratteristica di essere difficile e dispendioso da risolvere, ma una volta trovata la soluzione risulta molto facile verificarne la validità.

2.3.6 Proof-of-Work

La *proof-of-work*, ovvero il lavoro svolto dai minatori, consiste nel calcolo del codice *hash* dell'intestazione di un blocco, rispettando determinate caratteristiche.

L'*hash* è una funzione crittografica, ovvero un algoritmo matematico che mappa dei dati di lunghezza arbitraria (il messaggio) in una stringa binaria di dimensione fissa chiamata valore di hash. Tale funzione è progettata per essere unidirezionale, ovvero non invertibile: l'unico modo per ricreare i dati di input dall'output di una funzione di hash ideale è quello di tentare una ricerca di forza-bruta. [7]

In Bitcoin viene utilizzato prevalentemente l'algoritmo *SHA-256* [8], sia nella generazione degli indirizzi che per la *proof-of-work*.

Il blocco include, tra le altre cose, un riferimento al blocco precedente, un hash di un insieme di transazioni e un "*nonce*".

Il *nonce* (number used once [9]) in crittografia indica un numero, generalmente casuale o pseudo-casuale, che ha un utilizzo unico. Questo *nonce* dovrà essere

incrementato ripetutamente fino a quando la funzione hash calcolata sul blocco non genera un valore che inizia con un certo numero di zero bit. Questa operazione viene svolta milioni di volte al secondo da ciascun minatore. Quando un minatore trova il nonce corretto che verifica la condizione, gli altri minatori controllano se la soluzione è esatta e solo quando un certo numero di nodi ha verificato la validità del nuovo blocco, questo viene reso valido ed il minatore riceve una ricompensa in bitcoin per il lavoro compiuto. Il valore della ricompensa è programmato per dimezzarsi ogni 4 anni. Alla creazione del sistema era di 50 bitcoin per ogni blocco generato, nel 2012 scese a 25, mentre dal 2016 vale 12,5.

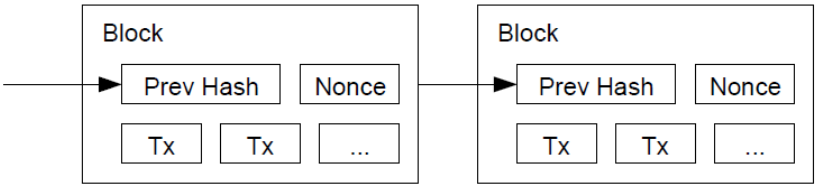


Figura 4 - Blocchi concatenati nella blockchain Bitcoin: nel blocco attuale viene incluso l'hash del blocco precedente, in modo da rendere non modificabili i blocchi senza ricalcolare gli hash dei blocchi successivi.

Il lavoro medio necessario per svolgere i calcoli è esponenzialmente proporzionale al numero di zero bit richiesti. Per compensare l'aumento della velocità dell'hardware e il variare dell'interesse dei nodi operativi col tempo, la difficoltà della proof-of-work è determinata da una media mobile che ha come obiettivo la creazione di un numero medio di blocchi ogni ora (circa un blocco ogni 10 minuti). Se i blocchi vengono generati troppo velocemente, la difficoltà aumenta.

Queste regole impediscono che qualunque blocco precedente venga modificato, perché ciò invaliderebbe tutti i blocchi successivi. Infatti, per modificare un blocco passato, un utente malintenzionato dovrebbe rifare la proof-of-work del blocco e di tutti i blocchi successivi ad esso fino a raggiungere e superare il lavoro dei nodi onesti che nel frattempo continuano a generare nuovi blocchi. È dimostrato che la probabilità che un utente malintenzionato raggiunga il lavoro dei nodi onesti diminuisce in modo esponenziale a mano a mano che vengono aggiunti blocchi successivi alla catena.

2.3.7 Privacy

Il modello bancario tradizionale garantisce un certo livello di privacy limitando l'accesso alle informazioni solamente alle parti coinvolte e alla terza parte fidata. In Bitcoin non è possibile applicare tale metodo poiché le transazioni sono tutte annunciate pubblicamente. La privacy pertanto viene mantenuta tenendo anonime le chiavi pubbliche. In questo modo il pubblico può vedere che qualcuno sta mandando un certo importo a qualcun altro, ma non vi sono informazioni che collegano la transazione ad un utente specifico. Come ulteriore livello di protezione, sarebbe preferibile generare una nuova coppia di chiavi per ogni singola transazione, in questo modo sarebbe ancora più difficile risalire al rispettivo proprietario.

3 Blockchain

La blockchain (catena di blocchi) è una lista in continua crescita di record, chiamati *blocks*, che sono collegati tra loro e resi sicuri mediante l'uso della crittografia. [10]

Ogni blocco della catena contiene un *puntatore hash* come collegamento al blocco precedente, un *timestamp* e i dati della transazione.

Una volta registrati, i dati in un blocco non possono essere retroattivamente alterati senza che vengano modificati tutti i blocchi successivi ad esso, il che necessiterebbe il consenso della maggioranza della rete.

Blockchain, è fondamentalmente un registro aperto e distribuito che può registrare le transazioni tra due parti in modo efficiente, verificabile e permanente.

La tecnologia Blockchain ha attirato l'attenzione essendo alla base delle criptovalute come Bitcoin. Infatti questa valuta elettronica, creata per risolvere i problemi di fiducia, trasparenza e responsabilità tra due parti nello scambio di denaro per beni e servizi su Internet, rappresenta la prima implementazione di questa tecnologia.

La blockchain Bitcoin è un registro pubblico e distribuito di tutte le transazioni Bitcoin che hanno avuto luogo dal 2009, ovvero dalla creazione del sistema, e che sono state approvate dal 50%+1 dei nodi. Ciò garantisce che tutti coloro che partecipano alla rete Bitcoin abbiano accesso a tutte le transazioni avvenute fino a quel momento e quindi tutti potranno concordare sul modo in cui ciascuna di tali transazioni ha avuto luogo.

In altre parole, la blockchain rappresenta il libro contabile (o libro mastro) di questo sistema, ossia il registro sul quale sono riportati tutti gli scambi tra le parti. Inoltre, questo registro è immutabile grazie alla particolare concatenazione dei blocchi, quindi nessuno può cambiare un dato registrato in esso senza modificare tutto il registro.

Estrapolata dal suo contesto, questa tecnologia può essere utilizzata in tutti gli ambiti in cui è necessaria una relazione tra più persone o gruppi. Ad esempio può garantire il corretto scambio di titoli e azioni, può sostituire un atto notarile e può garantire la bontà delle votazioni, può certificare i passaggi di un prodotto all'interno di una filiera produttiva, proprio perché ogni transazione viene sorvegliata da una rete di nodi che ne garantiscono la correttezza senza l'ausilio di intermediari. La tecnologia blockchain, chiamata anche *Distributed Ledger Technology* (DLT), potrebbe rivoluzionare l'industria ed il commercio e

guidare il cambiamento economico su scala globale perché è immutabile, trasparente e ridefinisce il concetto di fiducia, consentendo soluzioni sicure, veloci, affidabili e trasparenti, sia pubbliche che private.

3.1 Distributed Ledger Technology

La tecnologia blockchain funziona come un libro mastro distribuito (*Distributed Ledger*) visibile a tutte le parti coinvolte nella transazione. Attraverso un meccanismo di consenso, il libro mastro è garantito per essere coerente.

Un *Distributed Ledger* è un tipo di database condiviso, replicato e sincronizzato tra tutti i membri di una rete. In questo database vengono registrate le transazioni (come lo scambio di beni o informazioni) tra i partecipanti alla rete. I dati non sono memorizzati su un solo computer ma su più macchine collegate tra loro attraverso una rete peer-to-peer.

Ciascun nodo è autorizzato ad aggiornare e gestire il libro contabile distribuito in modo indipendente, ma sotto il controllo consensuale degli altri nodi.

Gli aggiornamenti non sono più gestiti, come accadeva tradizionalmente, sotto il controllo rigoroso di un'autorità centrale, ma sono invece creati e caricati da ciascun nodo in modo appunto indipendente. In questo modo ogni partecipante è in grado di processare e controllare ogni transazione ma, nello stesso tempo ogni singola transazione, essendo gestita in autonomia, deve essere verificata, votata e approvata dalla maggioranza dei partecipanti alla rete. Qui nasce il concetto alla base dei Distributed Ledgers, ovvero il *Consenso*.

I registri dei vari nodi vengono infatti aggiornati solamente al raggiungimento di un consenso sulle operazioni che vengono svolte. L'aggiornamento viene trasmesso a ciascun partecipante affinché tutti i registri siano coerenti. Grazie alle tecniche criptografiche e di hashing che caratterizzano la tecnologia blockchain, ogni operazione rimane memorizzata in modo indelebile ed immutabile su ogni singolo nodo.

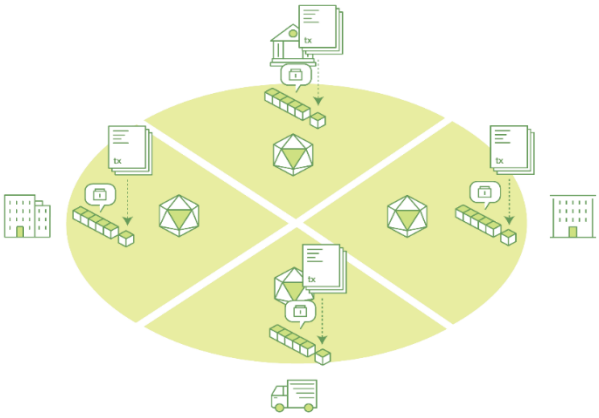


Figura 5 - Ogni nodo inserisce transazioni su una copia locale della blockchain.

3.1.1 Consenso

Il consenso è il processo che permette di mantenere sincronizzate le transazioni attraverso la rete. Questo processo è necessario per garantire che i registri vengano aggiornati mantenendo ordinate le informazioni, solo quando le transazioni sono approvate dagli altri partecipanti.

Esistono diversi algoritmi di gestione del consenso, ciascuno con i propri vantaggi e svantaggi, che li rendono utilizzabili solamente in certi casi.

Gli algoritmi di consenso più utilizzati sono:

- **Proof of Work**: usato in Bitcoin ed Ethereum, richiede ai validatori (miners) di risolvere complessi problemi crittografici.
  Pro: funziona nelle reti di tipo pubblico, quindi non fidate;
  Contro: estremamente oneroso dal punto di vista energetico ed è lento nella conferma delle transazioni.

- **Proof of Stake:** probabilmente verrà usato in futuro in Ethereum, richiede ai validatori di depositare della valuta come garanzia.
  Pro: funziona nelle reti non fidate;
  Contro: richiede necessariamente che il sistema sia basato su una criptovaluta ed esiste il problema "nothing at stake" dove, in caso di fallimento del consenso, ad un nodo non costa nulla votare per più ramificazioni della stessa blockchain, il che impedisce al consenso di risolversi.

- **Solo:** usato in Hyperledger Fabric v1.0, le transazioni vengono validate da un solo nodo, senza di fatto richiedere il consenso ad altri.
  Pro: Molto veloce, adatto in fase di sviluppo;
  Contro: non c'è consenso, usabile solo per test.

- **Kafka:** usato in Hyperledger Fabric v1.0, sono presenti dei servizi di ordinamento che distribuiscono ai vari nodi i blocchi ordinati.
  Pro: efficiente e tollerante ai guasti;
  Contro: non c'è una verifica contro attività malevoli.

- **PBFT (Practical Byzantine Fault Tolerance):** viene aggiunto un nuovo blocco se più dei 2/3 di tutti i nodi di convalida sono concordi.
  Pro: abbastanza efficiente e fornisce un minimo controllo contro i nodi malevoli;
  Contro: i validatori sono conosciuti e connessi tra di loro.

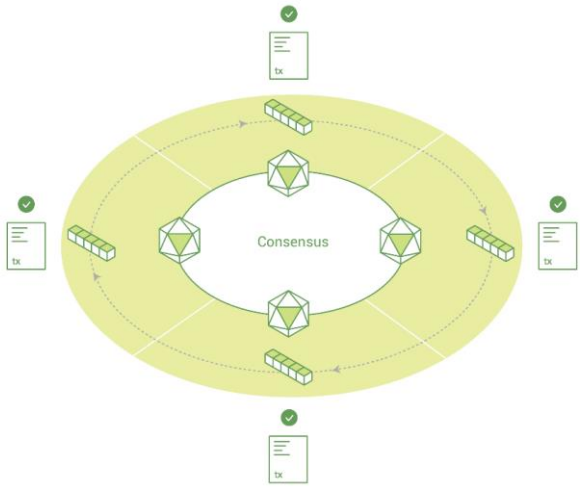


Figura 6 – Grazie al Consenso tutte le transazioni vengono ordinate e sincronizzate. La blockchain e coerente su tutti i nodi

Il consenso potrà essere gestito diversamente a seconda che si parli di blockchain di tipo *Pubblico* o di tipo *Privato*.

3.1.2 Distributed Ledger di tipo Pubblico o Privato

Le blockchain di tipo *Pubblico*, come ad esempio Bitcoin o Ethereum, sono aperte, non hanno una proprietà o un attore di riferimento e sono concepite per non essere controllate.

Questo modello di blockchain impedisce ogni forma di censura, tutti i partecipanti possono vedere il registro e nessuno è nella condizione di impedire che una transazione possa avvenire e che possa essere aggiunta al libro mastro una volta che ha conquistato il consenso necessario tra tutti i nodi partecipanti alla blockchain.

In genere le blockchain di tipo *Pubblico* sono anche *Unpermissioned* (senza permessi), ovvero permettono a ciascun nodo di partecipare alla rete e di contribuire all'aggiornamento dei dati sul libro mastro attraverso il processo del consenso.

Le blockchain di tipo *Privato*, a differenza delle precedenti, possono essere controllate e dunque possono avere una "proprietà". Per questo motivo le blockchain di tipo Privato spesso sono anche *Permissioned* (con permessi).

Infatti permettono di definire speciali regole per l'accesso alla rete e per la visibilità dei dati delle transazioni. Di fatto viene introdotto nella blockchain un concetto di *Governance* e di definizione di regole di comportamento. Quando un nuovo dato o record viene aggiunto, il sistema di approvazione non è vincolato alla maggioranza dei partecipanti, bensì a un numero limitato di membri autorizzati che sono definibili come *Trusted* (fidati).

Questo tipo di Blockchain possono essere gestite ed utilizzate da entità note, come istituzioni e grandi imprese, le quali devono gestire filiere con una serie di attori, fornitori e subfornitori, banche, società di servizi, operatori nell'ambito della vendita al dettaglio.

Tecnicamente le blockchain di tipo *Permissioned* sono anche più performanti e veloci di quelle di tipo *Permissionless.* Essendo i membri conosciuti e pertanto affidabili, non occorre utilizzare algoritmi di consenso energeticamente dispendiosi (come la proof-of-work di Bitcoin) per scoraggiare eventuali nodi disonesti.

	Permissionless (Qualsiasi utente può partecipare alla rete e validare le transazioni)	**Permissioned** (Solo alcuni utenti hanno il diritto di validare le transazioni)
Pubblica (Tutti gli utenti possono leggere i dati)	• Ogni utente può leggere i dati delle transazioni. • Ogni utente può validare le transazioni. Esempio: **Bitcoin** ed **Ethereum**	• Ogni utente può leggere i dati delle transazioni. • Solo utenti autorizzati possono validare le transazioni. Esempio: una blockchain Permissioned in cui si decide di rendere pubblici i dati delle transazioni
Privata (Accesso ai dati limitato ad utenti autorizzati)	• Solo utenti autorizzati possono leggere i dati delle transazioni. • Ogni utente può validare le transazioni. Esempio: **Ethereum** permette di creare istanze private	• Solo utenti autorizzati possono leggere i dati delle transazioni. • Solo utenti con diritti speciali possono validare le transazioni. Esempio: **Hyperledger Fabric**

Tabella 1 - Possibili tipi di reti blockchain.

3.1.3 Smart Contracts

Uno *Smart Contract* è la traduzione (o trasposizione) in codice di un contratto in modo da verificare in automatico l'avverarsi di determinate condizioni e di eseguire automaticamente azioni nel momento in cui le condizioni concordate tra le parti sono raggiunte e verificate.

In altre parole lo *Smart Contract* è un algoritmo software che "legge" sia le clausole che sono state concordate sia le condizioni operative nelle quali devono verificarsi le condizioni e si auto-esegue nel momento in cui i dati riferiti alle situazioni reali corrispondono ai dati riferiti alle condizioni e alle clausole concordate.

Lo *Smart Contract* ha bisogno di un supporto legale per la sua stesura, ma non ne ha bisogno per la sua verifica e per la sua attivazione. Ai contraenti spetta il compito di definire condizioni, clausole, modalità e regole di controllo e azione, ma una volta che il loro contratto è diventato codice (dunque uno *Smart Contract*) e viene accettato, gli effetti non dipendono più dalla loro volontà.

Questo punto è estremamente rilevante perché rappresenta la "certezza di giudizio oggettivo", escludendo qualsiasi forma di interpretazione.

All'interno di una blockchain, gli *Smart Contracts* vengono integrati per supportare l'aggiornamento coerente delle informazioni e per realizzare le

funzionalità tipiche del libro contabile, come transazioni, interrogazioni ed elaborazione dati.

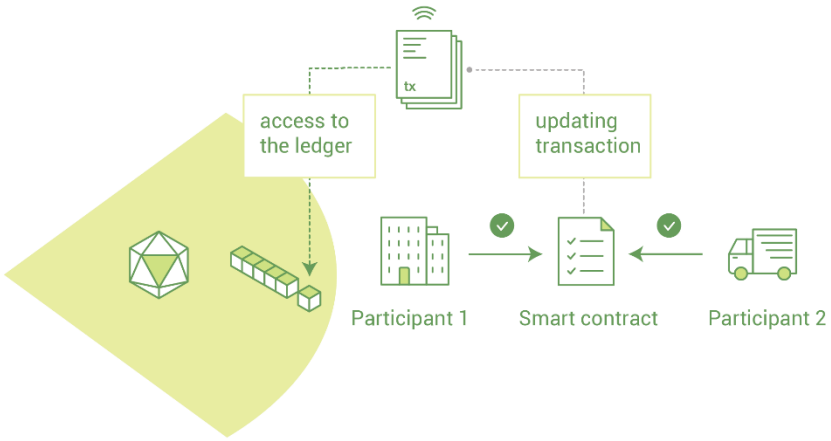


Figura 7 - Smart Contracts

4 HyperLedger Project

Hyperledger [17] è un progetto open source fondato nel 2015 dalla Linux Foundation per supportare lo sviluppo collaborativo di *Distributed Ledgers* basati su tecnologia blockchain.

L'obiettivo del progetto è far progredire la collaborazione intersettoriale sviluppando blockchain e registri distribuiti, con particolare attenzione al miglioramento delle prestazioni e dell'affidabilità di questi sistemi in modo che siano in grado di supportare transazioni commerciali dalle principali società tecnologiche, finanziarie e della distribuzione a livello globale.

Il progetto integra standard e protocolli open ed indipendenti, attraverso un framework composto da moduli specifici a seconda degli usi. Tra questi moduli vi sono blockchains con funzionalità di consenso e storage, servizi per la gestione delle identità, il controllo degli accessi e Smart Contracts.

Attualmente nel progetto sono coinvolti più di 180 membri, tra cui diversi sviluppatori indipendenti di software blockchain (Blockchain, ConsenSys, Digital Asset, R3, Onchain), società tecnologiche (Cisco, Fujitsu, Hitachi, IBM, Intel, NEC, NTT DATA, Red Hat, VMware), aziende di servizi finanziari, software aziendali (SAP) ed altri.

Figura 8 - Logo del progetto.

4.1 Hyperledger Fabric

Tra le varie piattaforme del progetto *Hyperledger*, è presente *Hyperledger Fabric,* un framework blockchain open source originariamente sviluppato da IBM sotto il nome di *IBM Open BlockChain* (OBC) e successivamente portato all'interno del progetto *Hyperledger.*

Hyperledger Fabric presenta un'architettura modulare e permette una divisione dei ruoli tra i nodi dell'infrastruttura, l'esecuzione di *Smart Contracts* (chiamati

"*chaincode*" in *Fabric*) e la possibilità di configurare il consenso ed i servizi di appartenenza (*Membership Services*). [18]

Una rete *Fabric* comprende:

- Nodi Peer, che eseguono *chaincode*, accedono ai dati del registro, approvano le transazioni e si interfacciano con le applicazioni.
- Nodi Ordinatori, che assicurano la coerenza della blockchain e consegnano le transazioni approvate ai peer della rete
- Servizi *Membership Service Provider* (*MSP*), generalmente implementati come Autorità di Certificazione, gestiscono i certificati utilizzati per autenticare l'identità ed i ruoli dei membri.

Lo scopo di *Fabric* è di essere integrato in progetti in cui è richiesta una *Distributed Ledger Technology* (DLT), pertanto come servizio offre solamente un'SDK (Software Development Kit) per Node.js, Java e Golang.

Fabric supporta *chaincode* scritto in Golang, Javascript e Java, per questo risulta più flessibile rispetto ad altri framework che utilizzano linguaggi proprietari per la scrittura degli Smart Contracts.

Ciò che rende *Hyperledger Fabric* diverso da altri sistemi blockchain è il fatto che offre la possibilità di creare una rete privata e *permissioned*, ovvero permette di gestire le autorizzazioni. Al contrario delle reti open e *permissionless* (senza autorizzazioni), che permettono a qualsiasi identità sconosciuta di collegarsi e partecipare alle transazioni, i membri di una rete *Hyperledger Fabric* possono essere ammessi attraverso un *Membership Service Provider* (MSP), il quale gestirà le identità dei partecipanti. Essendo i membri della rete tutti conosciuti, saranno superflui protocolli come il Proof-of-Work per garantire la sicurezza della rete e scoraggiare utenti non onesti.

Hyperledger Fabric permette diverse personalizzazioni. Ad esempio i dati nel registro possono essere memorizzati in più formati, i meccanismi di consenso possono essere attivati o disattivati e sono supportati diversi *Membership Service Provider*.

Fabric offre anche la possibilità di creare canali, consentendo a un gruppo di partecipanti di creare un registro separato per le loro transazioni. Questa è un'opzione particolarmente importante per le reti in cui alcuni partecipanti potrebbero essere concorrenti e non vogliono che ogni transazione eseguita sia visibile agli altri partecipanti. Se due partecipanti formano un canale, allora solamente loro avranno copie del registro di quel canale.

4.1.1 Registro distribuito

Il registro (ledger) di *Hyperledger Fabric* è formato da due componenti: il *world state* ed il *transaction log*. Ogni partecipante ha una copia del registro di ogni rete *Hyperledger Fabric* a cui appartiene.

Il componente *world state* (stato globale) descrive lo stato del registro in un dato momento. Rappresenta il database del ledger distribuito.

Il componente *transaction log* (registro delle transazioni) registra tutte le transazioni che hanno portato allo stato corrente il *world state*. In pratica è la cronologia degli aggiornamenti del database *world state*.

Il ledger è quindi una combinazione del database *world state* e della cronologia *transaction log*.

Il database del *world state* può essere sostituito. Di default è un database LevelDB (leveldb.org) di tipo key-value. Il *transaction log* invece non necessita di essere sostituibile, visto che semplicemente tiene traccia delle modifiche al database utilizzato dalla rete blockchain.

4.1.2 Smart Contracts

Gli Smart Contracts di *Hyperledger Fabric* sono scritti all'interno di *chaincode* e sono invocati da una applicazione esterna alla blockchain quando essa deve interagire con il registro. Nella maggior parte dei casi *chaincode* interagisce solamente con il *world state* del registro (ad esempio per eseguire query), e non con il *transaction log*.

Chaincode può essere implementato in diversi linguaggi di programmazione. Attualmente sono supportati il linguaggio Go (golang.org), Java e JavaScript.

4.1.3 Privacy

Mediante i canali privati è possibile creare transazioni riservate tra specifici sottogruppi di membri della rete. Tutti i dati riguardanti il canale, inclusi transazioni, membri e informazioni sul canale, sono inaccessibili ed invisibili a tutti coloro che non sono in possesso di specifici diritti d'accesso.

4.1.4 Consenso

Le transazioni devono essere scritte nel registro nell'ordine in cui si verificano, anche se queste potrebbero trovarsi in diversi gruppi di partecipanti all'interno

della rete. Affinché questo accada, è necessario stabilire l'ordine delle transazioni e predisporre un metodo per rifiutare quelle errate che sono state inserite per sbaglio o maliziosamente.

Esistono diversi modi per realizzare il consenso, ciascuno con i propri compromessi. Per esempio, PBFT (Practical Byzantine Fault Tolerance) può fornire un meccanismo per permettere ai file replicati di comunicare tra loro, al fine di mantenere coerente ogni copia, anche in caso di corruzione di un nodo. In alternativa, in Bitcoin, l'ordinamento avviene attraverso il processo di mining, il quale consiste nella soluzione di un problema crittografico.

Hyperledger Fabric è stato progettato per consentire agli utenti della rete di scegliere un meccanismo di consenso che rappresenti al meglio le relazioni esistenti tra i partecipanti. Come con la privacy, ci possono essere diverse necessità: da reti altamente strutturate nelle relazioni a quelle più peer-to-peer. I meccanismi attualmente supportati sono *SOLO* e *Kafka*, prossimamente sarà aggiunto anche *SBFT* (Simplified Byzantine Fault Tolerance).

4.2 Hyperledger Composer

Hyperledger Composer [19] è un set di strumenti open source sviluppato da IBM ed inserito nel progetto *Hyperledger* per semplificare lo sviluppo di applicazioni blockchain.

L'obiettivo principale di questo strumento è di velocizzare i tempi di realizzazione e facilitare l'integrazione delle applicazioni blockchain con i sistemi aziendali esistenti.

Hyperledger Composer si appoggia al framework blockchain di *Hyperledger Fabric* e permette agli utenti di modellare rapidamente una business network contenente risorse e transazioni ad esse correlate.

Le risorse possono essere beni materiali, immateriali, servizi o proprietà.

Per modellare la rete, occorre definire le transazioni che possono interagire con le risorse. Le transazioni sono implementate utilizzando semplici funzioni JavaScript.

Le business network comprendono anche i partecipanti che interagiscono attraverso esse, ognuno dei quali viene associato a un'identità unica che può essere utilizzata anche in più reti diverse.

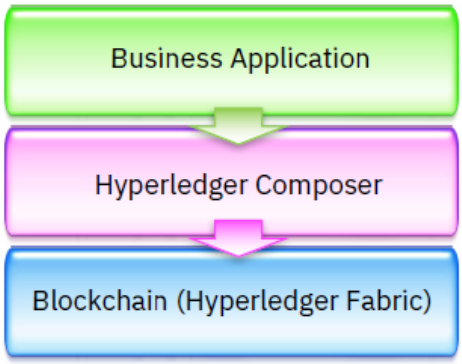


Figura 9 - Hyperledger Composer si pone come strato intermedio tra il framework Fabric e le applicazioni.

Una volta modellata la rete attraverso il *Composer*, sarà possibile testarla e renderla accessibile ad applicazioni esistenti attraverso API, oppure esportare le definizioni in un package *Business Network Archive* (.BNA) che può essere eseguito su un'istanza di Hyperledger Fabric.

Il package .BNA contiene al suo interno:

- File dei Modelli (.CTO)
- Funzioni delle Transazioni (.JS)
- Access Control Lists (.ACL)
- Query statiche (.QRY)
- Documentazione (.MD)

L'applicazione client non viene inclusa nel package.

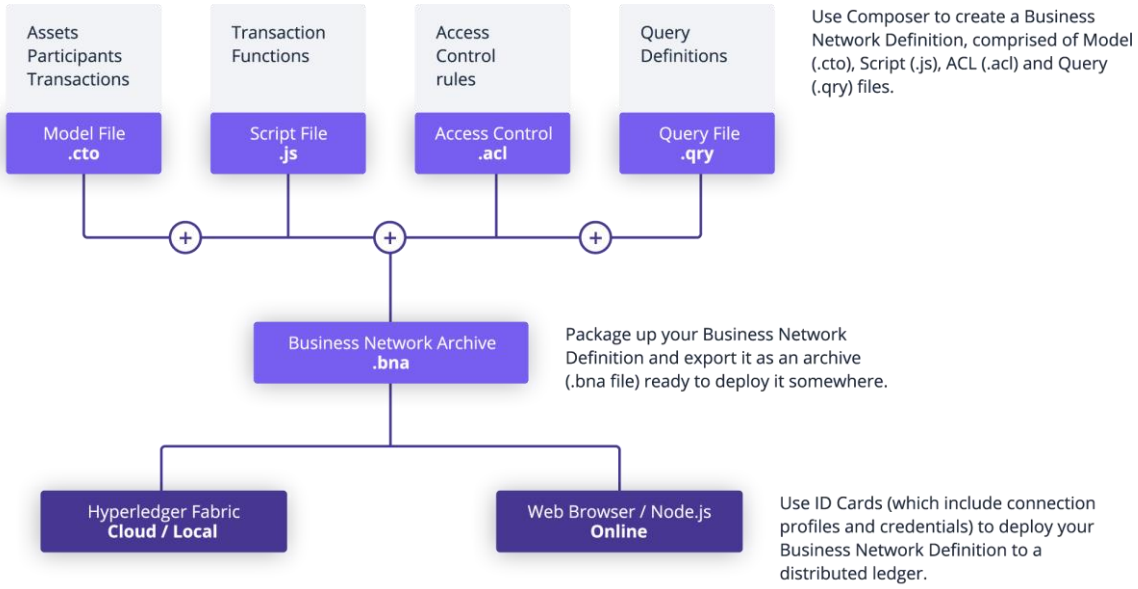


Figura 10 – Contenuto del package Business Network Archive (.BNA).

4.2.1 Concetti chiave

Blockchain State Storage: tutte le transazioni inviate attraverso la rete sono memorizzate nel ledger blockchain, in particolare lo stato corrente di risorse e membri partecipanti è memorizzato nel database *world state*. La blockchain distribuisce il registro ed il database di stato ad un insieme di nodi e garantisce, attraverso l'uso di un algoritmo di consenso, che gli aggiornamenti siano coerenti tra tutti i peer.

Connection Profiles: *Hyperledger Composer* utilizza dei profili di connessione per connettersi a un programma in esecuzione. Un profilo di connessione è un documento JSON che risiede nella directory home dell'utente (o proviene da una variabile d'ambiente) e viene richiamato quando si utilizzano le API *Composer* o gli strumenti a riga di comando. L'utilizzo dei profili di connessione garantisce che il codice e gli script siano facilmente trasferibili da un'istanza d'esecuzione all'altra.

Assets: le risorse sono beni (tangibili o non tangibili), servizi o proprietà che sono archiviati nei registri. Le risorse possono rappresentare quasi qualsiasi cosa in una business network, ad esempio una casa in vendita, un catalogo, il certificato di catasto, i documenti assicurativi per quella casa possono essere tutti beni appartenenti ad una o più reti. Le risorse devono avere un identificativo univoco e possono contenere qualsiasi altro attributo necessario. Una risorsa può essere collegata ad altre risorse o a membri della rete.

Participants: i partecipanti sono i membri di una business network. Possono possedere risorse e inviare transazioni. È possibile definire tipi di partecipanti e, come le risorse, devono possedere un identificatore e possono contenere attributi.

Identities e ID cards: all'interno di una business network, i membri possono essere associati a un'identità. Le *ID cards* sono una combinazione di un'identità, un profilo di connessione e altri metadati. Le *ID cards* semplificano il processo di connessione a una business network ed estendono il concetto di identità anche al di fuori della rete.

Transactions: le transazioni sono il meccanismo con cui i membri interagiscono con le risorse. Questa interazione potrebbe essere semplice, come un partecipante che piazza un'offerta su una risorsa in un'asta, oppure qualcosa di più elaborato, come un banditore che chiude l'asta ed automaticamente viene trasferita al miglior offerente la proprietà della risorsa.

Query: le query vengono utilizzate per ottenere informazioni dal database *world state* della blockchain. Sono definite all'interno della business network e

possono includere parametri variabili per semplificare la personalizzazione delle ricerche. Utilizzando le query, i dati possono essere facilmente estratti dalla rete blockchain. Possono essere lanciate utilizzando le API di *Hyperledger Composer*.

Events: Gli eventi sono dichiarati nella definizione della business network allo stesso modo delle risorse o dei membri. Una volta che gli eventi sono stati definiti, possono essere avviati dalle funzioni che processano le transazioni, per avvisare i sistemi esterni che qualcosa di importante è avvenuto nel registro. Tramite API le applicazioni possono selezionare gli eventi dai quali ricevere gli avvisi.

Access Control: le business networks possono contenere una serie di regole di controllo degli accessi. Queste regole consentono un controllo preciso su quali risorse ed a quali condizioni i partecipanti possono accedere. Il linguaggio di controllo degli accessi è abbastanza ricco per interpretare anche dichiarazioni sofisticate. Separare il controllo accessi dalle funzioni che processano le transazioni facilita la revisione, il debug, lo sviluppo e la manutenzione.

Historian registry: il registro storico è un registro specializzato che memorizza le transazioni che hanno successo, compresi i membri e le identità che le hanno richieste. Le transazioni vengono memorizzate come risorse di tipo *HistorianRecord*, definite all'interno di *Composer*.

4.2.2 Hyperledger Composer Playground

È possibile utilizzare *Hyperledger Composer* attraverso un'interfaccia utente web chiamata *Hyperledger Composer Playground*.

Playground è disponibile sia come versione online ospitata come servizio cloud sulla piattaforma *Bluemix* di IBM (nessuna installazione necessaria) oppure attraverso una installazione locale, per il test di reti offline. Per utilizzare a pieno le funzionalità di sviluppo offerte da Hyperledger Composer in locale, è necessario installare anche i Developer Tools.

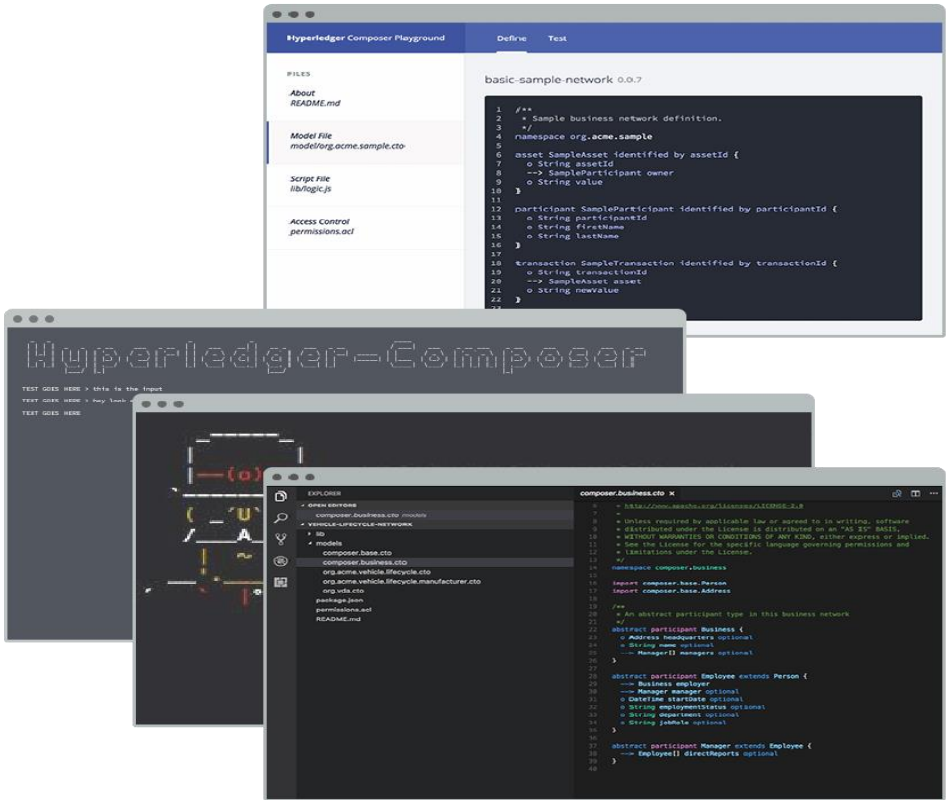


Figura 11 - Hyperledger Composer Playground e Developer Tools

Mediante l'interfaccia grafica web *Playground*, è possibile modellare la business network e definire quali oggetti di valore (*Assets*) sono scambiati, chi partecipa (*Participants*) allo scambio, come l'accesso è reso sicuro (*Access Control*), quale business logic (*Transactions*) realizza il processo. Una volta completate le definizioni, è possibile inserire delle istanze di beni e partecipanti ed effettuare transazioni allo scopo di testare la business network appena modellata.

L'interfaccia di *Playground* si divide in due schermate: *Define* e *Test*.

La schermata *Define* (definisci) viene utilizzata per creare, modificare e aggiornare la rete.

Nella parte sinistra della finestra, è visualizzato un elenco di tutti i file che definiscono le funzionalità e permettono di modellare la rete. Selezionando un file, esso verrà aperto nell'editor nella parte centrale della schermata.

I file sono di tipo *models*, *script*, *access control* o *query* e ciascuno di essi definirà specifiche funzionalità della rete.

Una volta caricati o modificati i file, premendo il pulsante "*Update"* la rete viene aggiornata con le modifiche apportate, le quali saranno visibili nella schermata *Test*.

Con il pulsante "*Export"* è possibile scaricare la rete corrente in un file di tipo *Business Network Archive* (.bna).

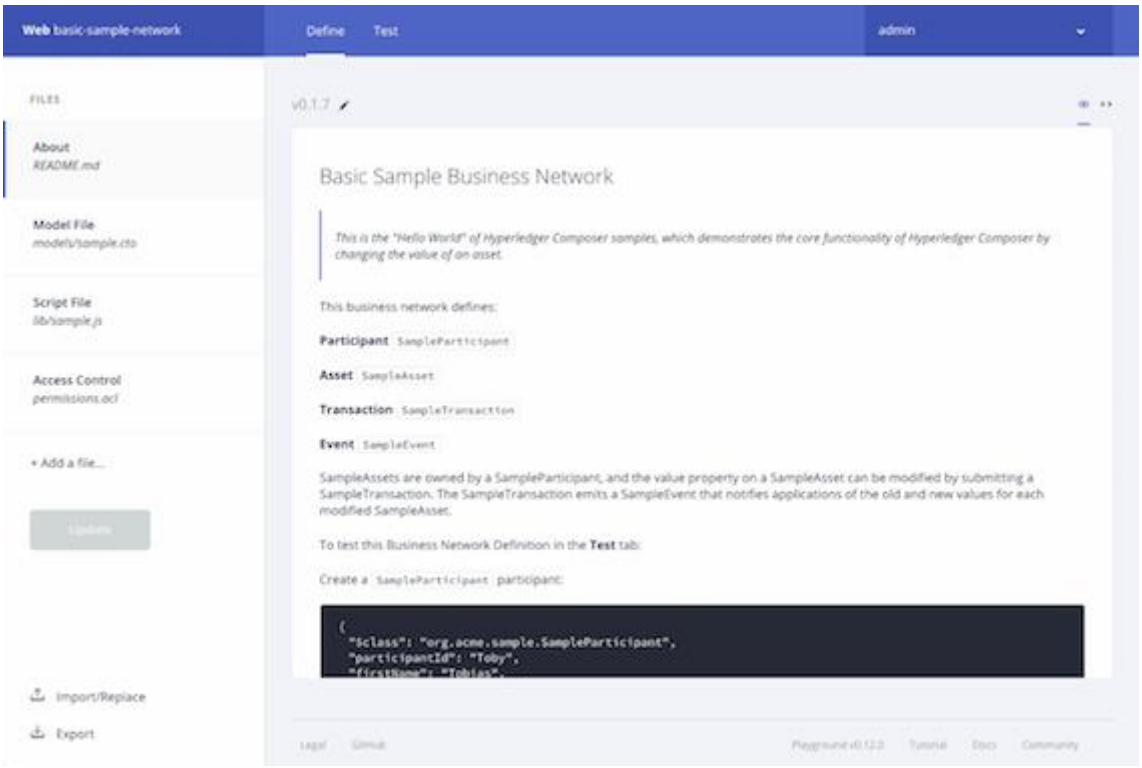


Figura 12 - Schermata Define di Playground

La scheda *Test* viene utilizzata per testare la business network andando a definire istanze di membri e beni precedentemente definiti nel file .cto dalla scheda *Define*.

Sulla sinistra vengono visualizzati i membri ed i beni della rete divisi per tipo. Cliccando su uno di essi viene visualizzato un registro contente tutte le istanze. Da ogni registro è possibile creare nuove istanze dei rispettivi tipi (assets o participants). Per inserire una transazione occorre premere il pulsante *"Submit Transaction"* e selezionare da un menu a tendina il tipo di transazione che si vorrà creare.

Il registro *All Transactions* rappresenta l'*Historian Registry* di *Hyperledger Composer* (vedre paragrafo 4.2.1) nel quale i records rappresentano ogni

transazione avvenuta nella rete, incluse quelle causate dagli eventi (come ad esempio la creazione di un partecipante o di un bene).

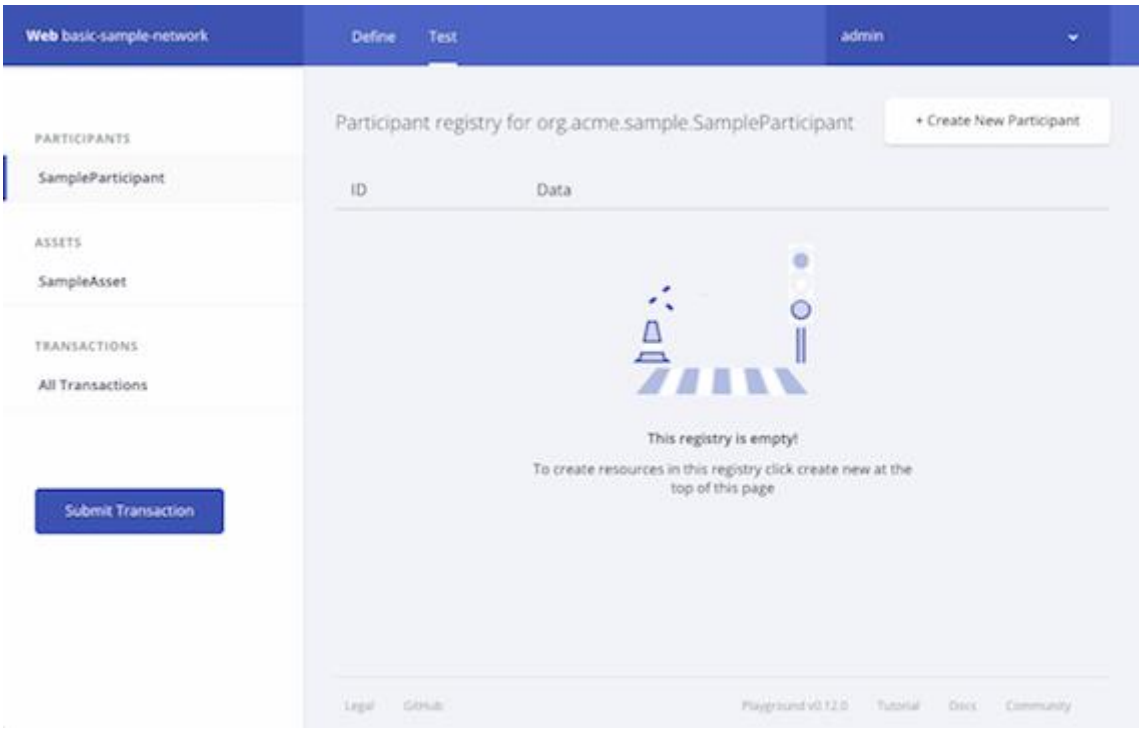


Figura 13 - Schermata Test di Playground

5 Asta di Autoveicoli basata su Blockchain

Lo scopo di questo esempio è di implementare i concetti della blockchain mostrando come vengono trasferiti dei beni tra i partecipanti all'interno di una business network opportunamente modellata.

5.1 Introduzione

L'esempio consiste nell'implementare una semplice asta di autoveicoli. Vi sarà un partecipante che metterà in vendita la propria automobile. Altri partecipanti piazzeranno offerte per cercare di aggiudicarsi il veicolo. Alla chiusura dell'asta, colui che avrà effettuato l'offerta più alta diventerà il nuovo proprietario del veicolo ed allo stesso tempo l'importo offerto verrà trasferito al venditore. Il tutto avverrà senza l'intermediazione di terze parti, ma semplicemente attraverso regole (*Smart Contracts*) definite al momento della creazione della rete.

La rete è composta da un insieme di partecipanti noti (compratori e venditori), beni (auto ed annunci di vendita), e transazioni (piazzare offerte e chiudere le aste).

Fondamentalmente, gli acquirenti e i venditori di queste risorse potrebbero essere riuniti in una blockchain, senza bisogno di terze parti di fiducia. Tuttavia, un banditore potrebbe essere utilizzato, se necessario, per svolgere compiti di amministrazione della rete.

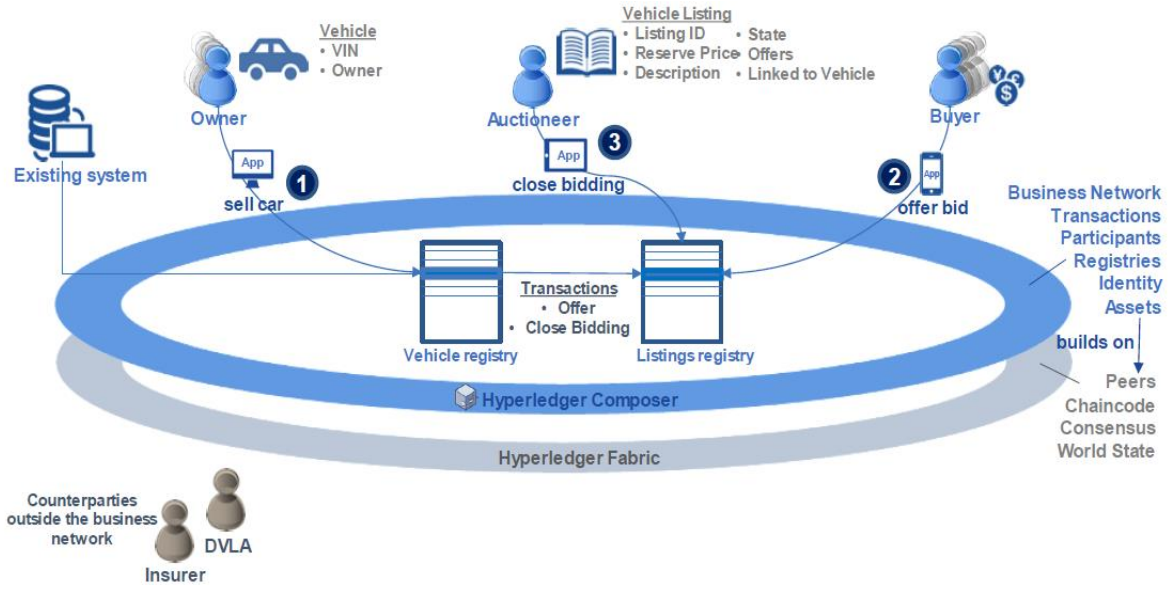


Figura 14 - Schema della business network che sarà modellata.

Usando *Hyperledger Composer Playground* modelleremo queste figure e testeremo la logica che rende l'asta funzionante.

In particolare, lavoreremo con la versione online di *Composer Playground*, ospitata sulla piattaforma di cloud-computing *Bluemix* di IBM [21].

Attraverso una semplice interfaccia web, potremo modellare e simulare l'intera rete blockchain all'interno del browser. Questo strumento, fornisce un ambiente protetto per definire, testare ed esplorare in modo rapido una business network. Sarà infatti possibile definire *Assets*, *Participants* e *Transactions*, implementare gli scripts che eseguono le transazioni e testare la rete popolando i registri ed invocando le transazioni.

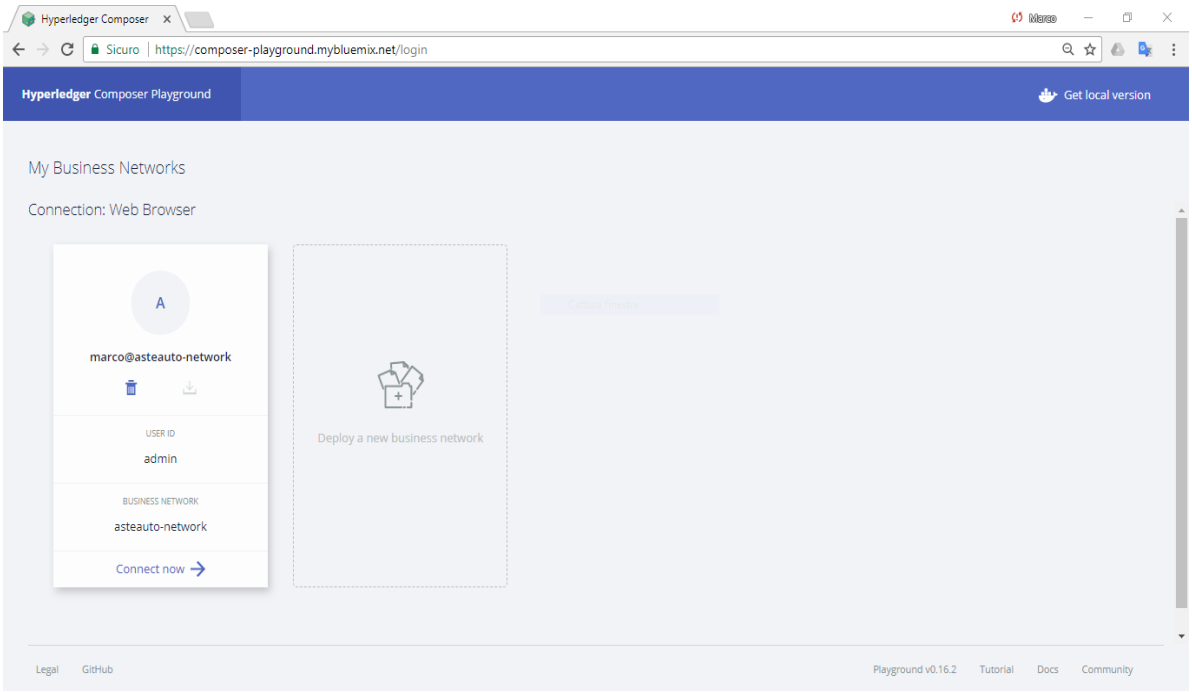


Figura 15 - Schermata iniziale di Hyperledger Composer Playground online.

5.2 Modellazione della rete

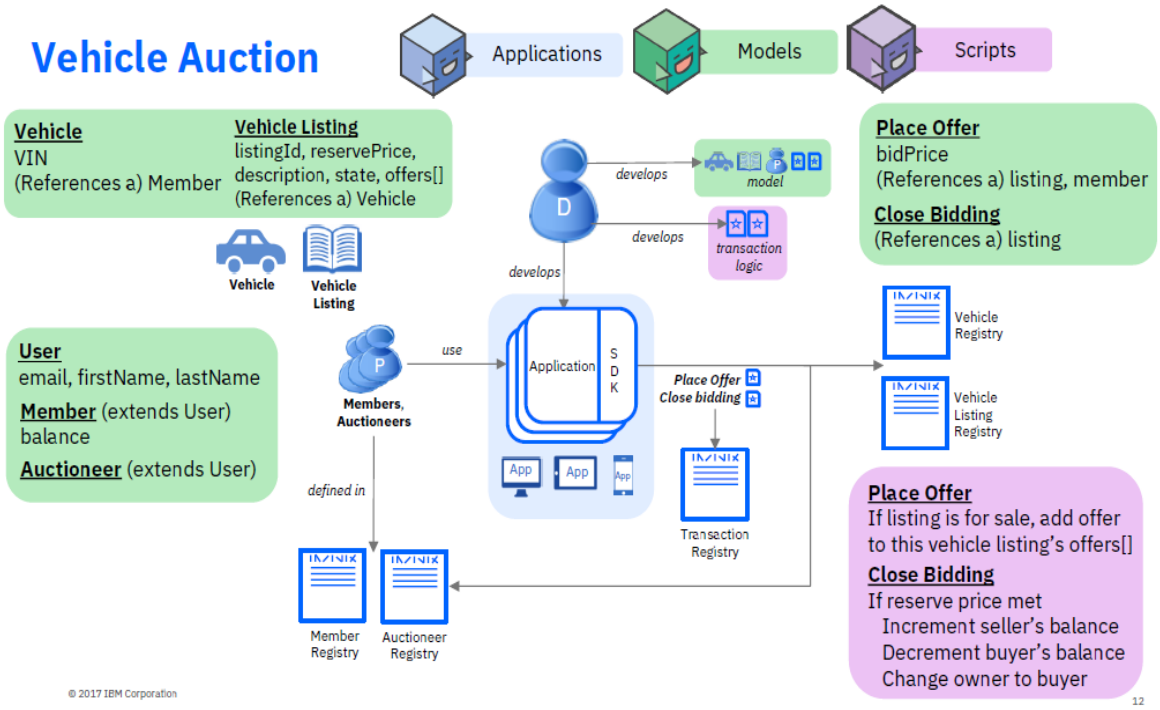
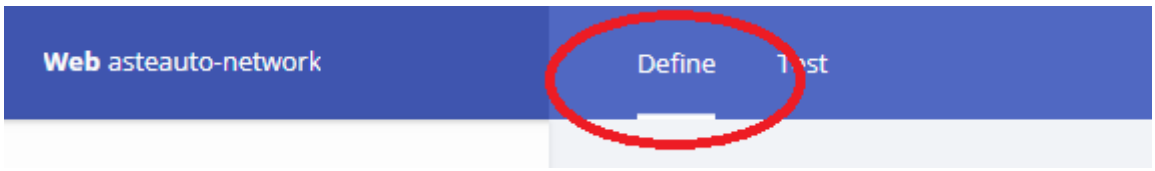


Figura 16 - Schema della rete che sarà modellata definendo le varie classi.

Per modellare la rete, occorre spostarsi nella scheda "*Define*" di Composer Playground. Sulla sinistra saranno elencati i diversi file che permettono di definire modelli, scripts e regole che modelleranno la rete secondo le nostre esigenze.



All'interno del file *"models/auction.cto"* andremo a definire le varie classi di tipo *Participants*, *Assets* o *Transactions*.

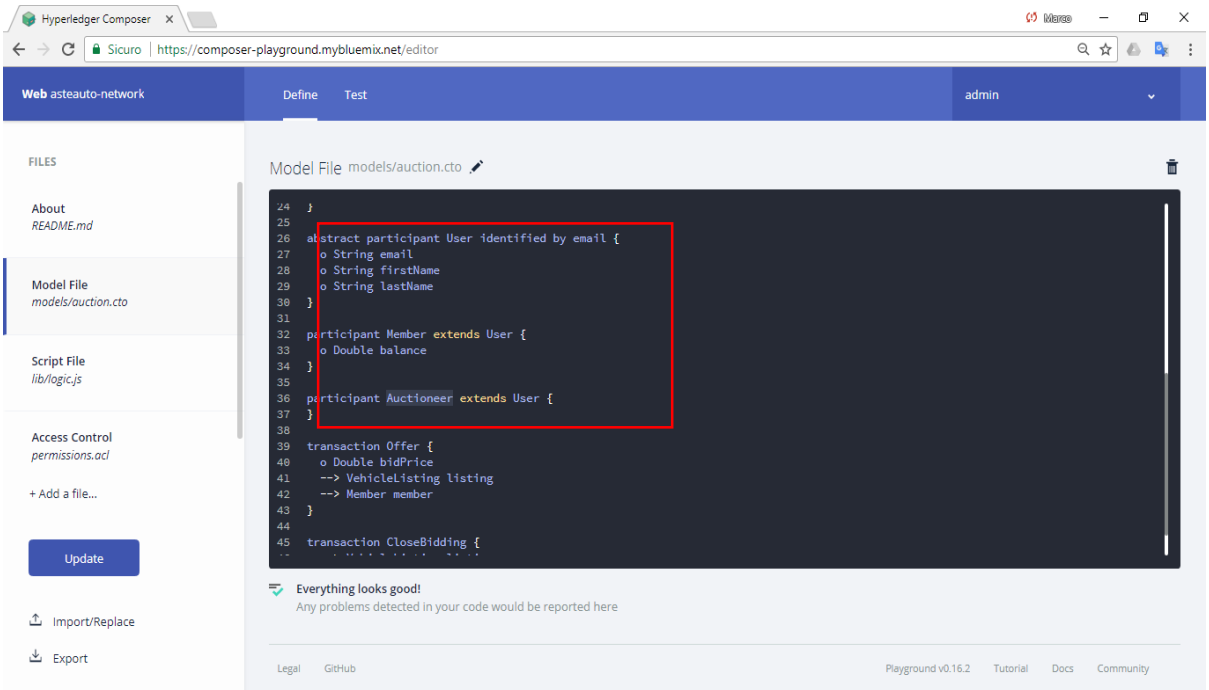
Viene utilizzato il linguaggio CTO, linguaggio di modellazione object-oriented di Composer. [23]

5.2.1 Classi di tipo *Participants*

Definiamo le classi *Member* e *Auctioneer* che estendono la classe *User*, tutte di tipo *Participants*.

Infatti i partecipanti alla rete potranno essere membri (gli utenti che vendono e acquistano auto) o banditori.

La classe *User* è definita astratta, quindi non sarà possibile creare istanze di essa. Le classi *Members* e *Auctioneer* estendono (extends) questa classe ed ereditano i suoi attributi. Ereditano anche l'identificatore, rappresentato dall'attributo email.

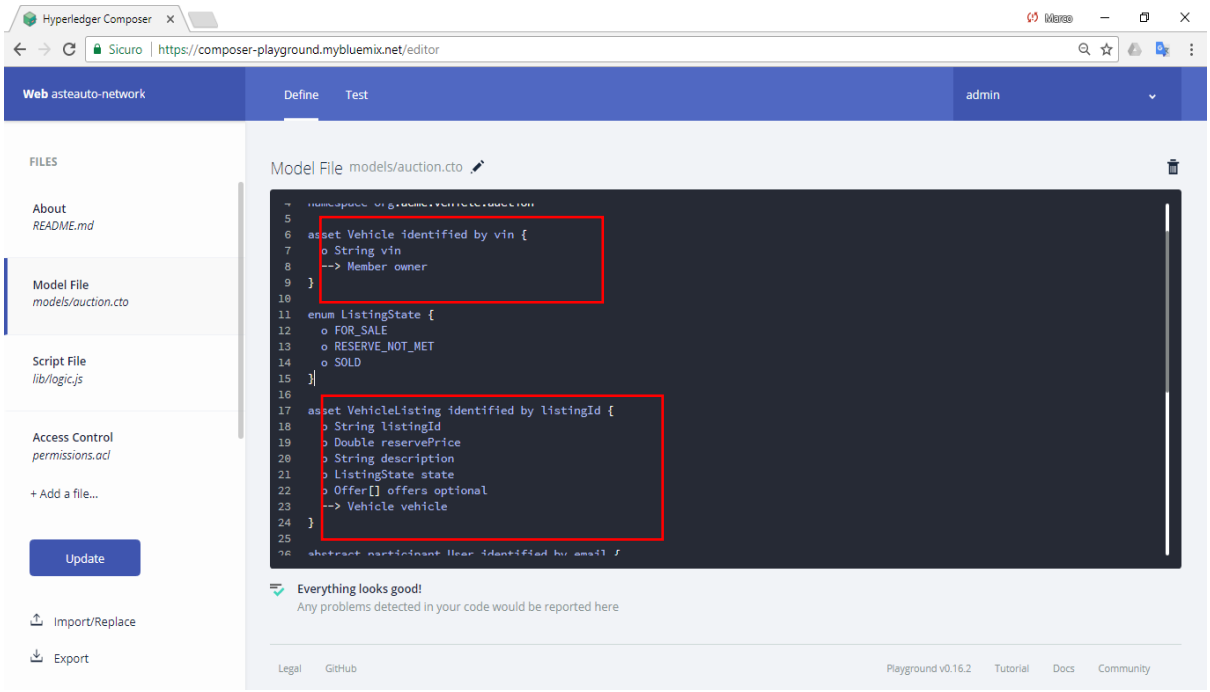


5.2.2 Classi di tipo *Assets*

Le classi *Vehicle* e *VehicleListing* sono di tipo *Assets* e rappresentano i beni oggetto delle transazioni nella rete.

La classe *Vehicle* (Veicolo) è identificata dall'attributo "vin" (Vehicle Identification Number) e contiene un riferimento (indicato con -->) al proprietario ("owner"), un'istanza della classe *Member* di tipo *Participants*.

La classe *VehicleListing* (Annuncio di auto) è identificata dall'attributo "listingId" e contiene un riferimento al veicolo oggetto dell'annuncio.

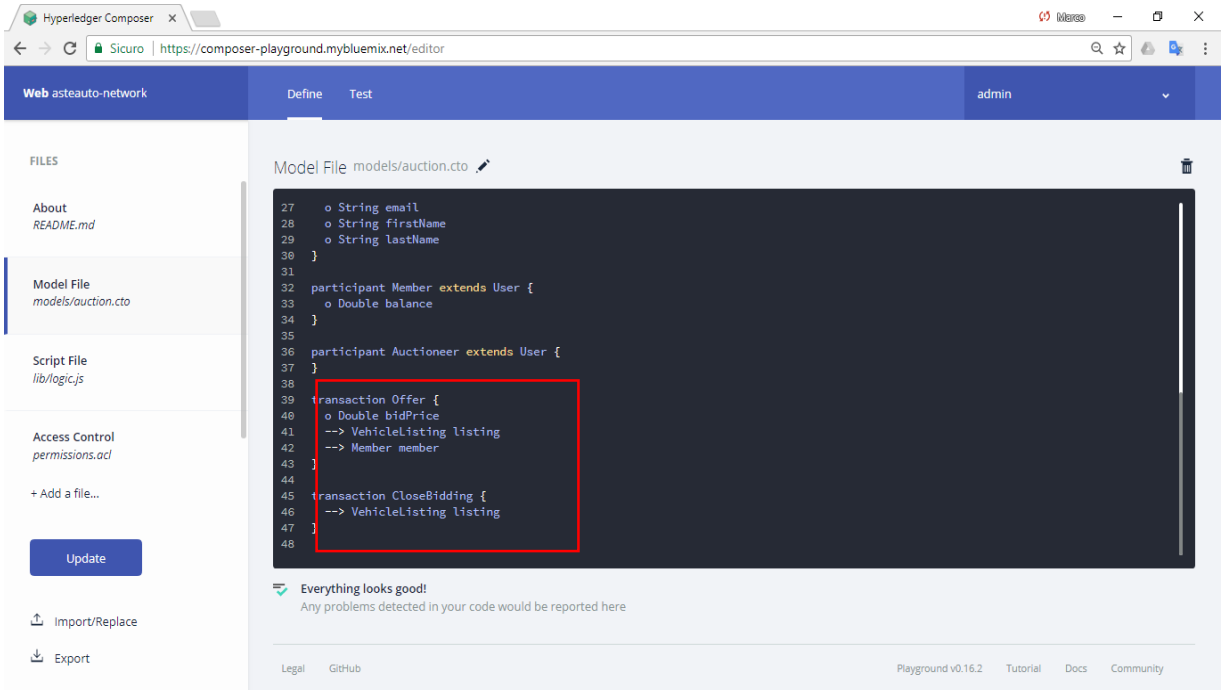


5.2.3 Classi di tipo *Transactions*

Le classi *Offer* e *CloseBidding* di tipo *Transactions* forniscono una descrizione delle transazioni che saranno eseguite nella blockchain.

La classe *Offer* (offerte) contiene l'importo dell'offerta ed i riferimenti ad un annuncio ed al membro che ha piazzato l'offerta.

La classe *CloseBidding* (chiudi le offerte) invece, contiene solamente un riferimento ad un annuncio.



Le transazioni saranno implementate in un altro file (*Transaction Processor file*).

5.2.4 Implementazione degli *Scripts*

Il file *"lib/logic.js"* è chiamato *Transaction Processor file*.

In esso vengono implementate, utilizzando il linguaggio JavaScript, le funzioni che saranno invocate all'esecuzione delle transazioni nella rete blockchain.

Attraverso il commento "@param", la funzione viene collegata alla transazione che la invocherà.

In questo esempio vengono definite due funzioni: *makeOffer* e *closeBidding*.

La funzione *makeOffer* è chiamata quando viene invocata la transazione *Offer*. La logica controlla che l'annuncio a cui si riferisce l'offerta è nello stato "FOR_SALE" (in vendita). In caso positivo, aggiunge l'offerta all'annuncio ed aggiorna il registro dell'asset *VehicleListing*.

Script File lib/logic.js

```
84
85  /**
86   * Make an Offer for a VehicleListing
87   * @param {org.acme.vehicle.auction.Offer} offer - the offer
88   * @transaction
89   */
90  function makeOffer(offer) {
91      var listing = offer.listing;
92      if (listing.state !== 'FOR_SALE') {
93          throw new Error('Listing is not FOR SALE');
94      }
95      if (listing.offers == null) {
96          listing.offers = [];
97      }
98      listing.offers.push(offer);
99      return getAssetRegistry('org.acme.vehicle.auction.VehicleListing')
100         .then(function(vehicleListingRegistry) {
101             // save the vehicle listing
102             return vehicleListingRegistry.update(listing);
103         });
104 }
105
```

La funzione *closeBidding* è chiamata quando viene invocata la transazione *CloseBidding*. La logica controlla che l'annuncio che sarà chiuso è nello stato "FOR_SALE" ed ordina le offerte in base all'importo offerto. Se il prezzo di riserva è stato raggiunto, trasferisce la proprietà del veicolo associato all'annuncio al miglior offerente. I soldi vengono trasferiti dal conto dell'acquirente a quello del venditore e successivamente vengono aggiornati tutti gli assets nei rispettivi registri.

Script File lib/logic.js

```
/**
 * Close the bidding for a vehicle listing and choose the
 * highest bid that is over the asking price
 * @param {org.acme.vehicle.auction.CloseBidding} closeBidding - the closeBidding transaction
 * @transaction
 */
function closeBidding(closeBidding) {
    var listing = closeBidding.listing;
    if (listing.state !== 'FOR_SALE') {
        throw new Error('Listing is not FOR SALE');
    }
    // by default we mark the listing as RESERVE_NOT_MET
    listing.state = 'RESERVE_NOT_MET';
    var highestOffer = null;
    var buyer = null;
    var seller = null;
    if (listing.offers && listing.offers.length > 0) {
        // sort the bids by bidPrice
        listing.offers.sort(function(a, b) {
            return (b.bidPrice - a.bidPrice);
        });
        highestOffer = listing.offers[0];
        if (highestOffer.bidPrice >= listing.reservePrice) {
            // mark the listing as SOLD
            listing.state = 'SOLD';
            buyer = highestOffer.member;
            seller = listing.vehicle.owner;
            // update the balance of the seller
            console.log('#### seller balance before: ' + seller.balance);
            seller.balance += highestOffer.bidPrice;
            console.log('#### seller balance after: ' + seller.balance);
            // update the balance of the buyer
            console.log('#### buyer balance before: ' + buyer.balance);
            buyer.balance -= highestOffer.bidPrice;
            console.log('#### buyer balance after: ' + buyer.balance);
            // transfer the vehicle to the buyer
            listing.vehicle.owner = buyer;
            // clear the offers
            listing.offers = null;
        }
    }
    return getAssetRegistry('org.acme.vehicle.auction.Vehicle')
        .then(function(vehicleRegistry) {
            // save the vehicle
            if (highestOffer) {
                return vehicleRegistry.update(listing.vehicle);
            } else {
                return true;
            }
        })
        .then(function() {
            return getAssetRegistry('org.acme.vehicle.auction.VehicleListing')
        })
        .then(function(vehicleListingRegistry) {
            // save the vehicle listing
            return vehicleListingRegistry.update(listing);
        })
        .then(function() {
            return getParticipantRegistry('org.acme.vehicle.auction.Member')
        })
        .then(function(userRegistry) {
            // save the buyer
            if (listing.state == 'SOLD') {
                return userRegistry.updateAll([buyer, seller]);
            } else {
                return true;
            }
        });
}
```

5.2.5 Controllo degli accessi

Il file *"permissions.acl"* è l'*Access Control List* (ACL) e definisce le regole che permettono o negano agli utenti l'accesso a certi aspetti della blockchain.

ACL File permissions.acl

```
/**
 * Access Control List for the auction network.
 */
rule Auctioneer {
    description: "Allow the auctioneer full access"
    participant: "org.acme.vehicle.auction.Auctioneer"
    operation: ALL
    resource: "org.acme.vehicle.auction.*"
    action: ALLOW
}

rule Member {
    description: "Allow the member read access"
    participant: "org.acme.vehicle.auction.Member"
    operation: READ
    resource: "org.acme.vehicle.auction.*"
    action: ALLOW
}

rule VehicleOwner {
    description: "Allow the owner of a vehicle total access"
    participant(m): "org.acme.vehicle.auction.Member"
```

5.3 Inserimento di membri e risorse

Per inserire membri e risorse, occorre spostarsi nella scheda "*Test*" di Composer Playground.

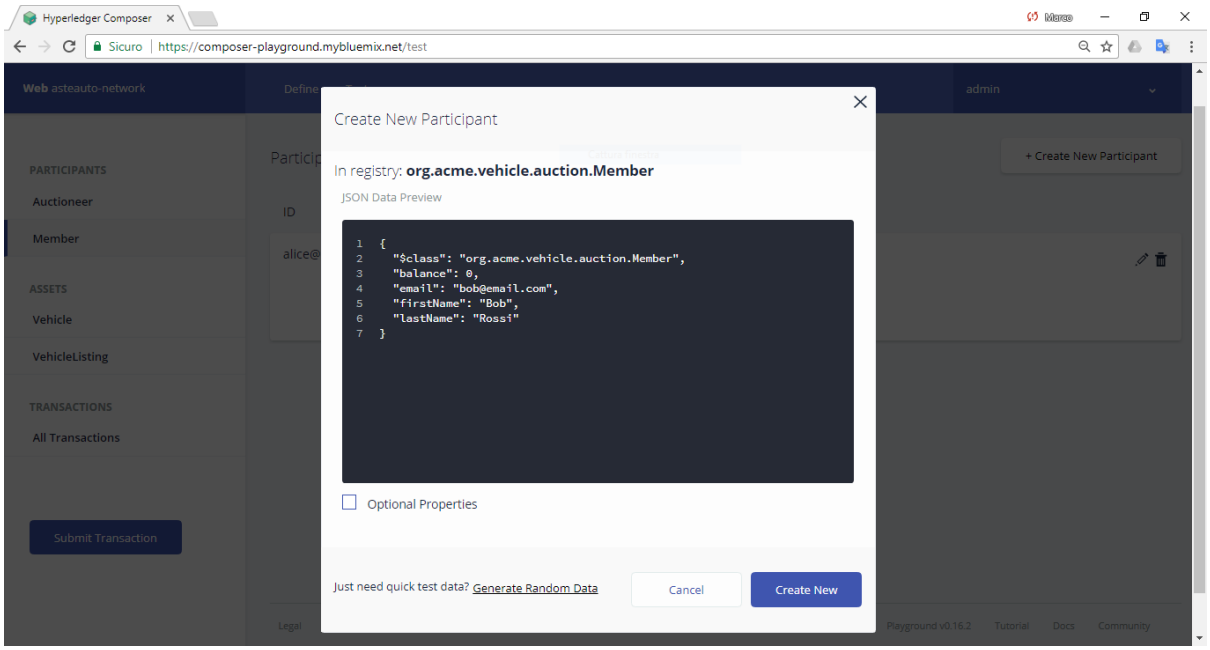
Sulla sinistra di questa schermata, è possibile vedere le varie classi (definite precedentemente nel file *"models/auction.cto"*) suddivise in base al tipo a cui appartengono.

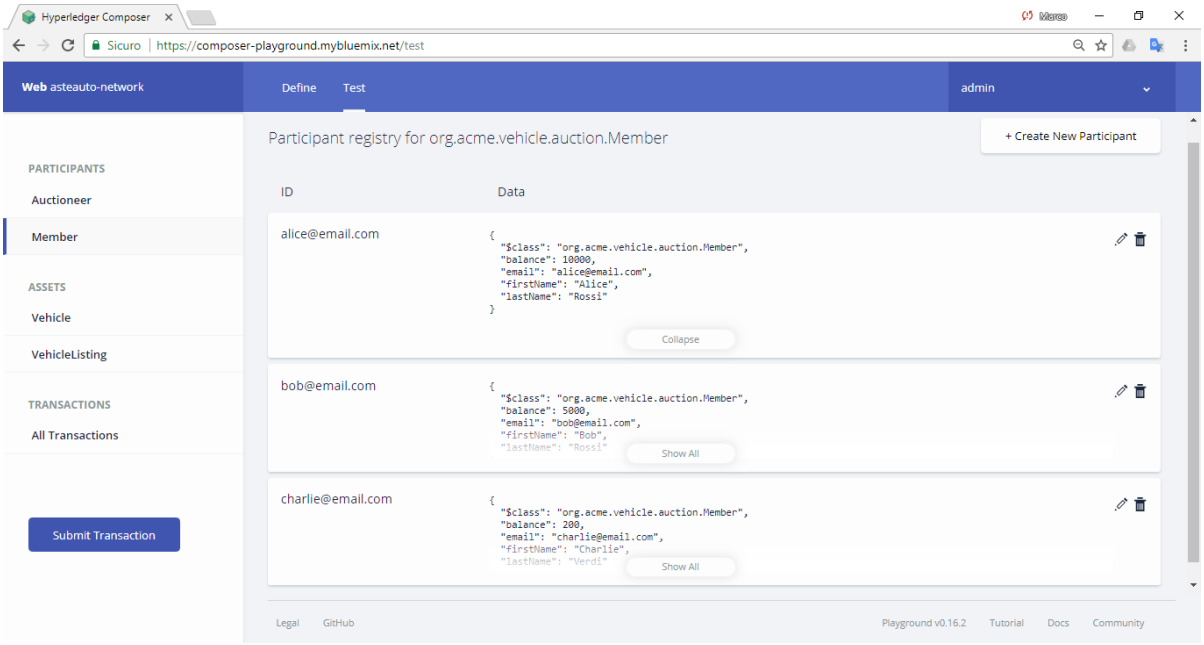
Cliccando su una classe, viene mostrato il registro contenente le varie istanze di quella classe. Per creare una nuova istanza occorre premere il pulsante "*Create New Participants*" ed inserire i valori della struttura dati JSON.

Verranno istanziati tre membri della rete:

- Alice Bianchi (alice@email.com) con un importo iniziale di 10000
- Bob Rossi (bob@email.com) con un importo iniziale di 5000
- Charlie Verdi (charlie@email.com) con un importo iniziale di 200

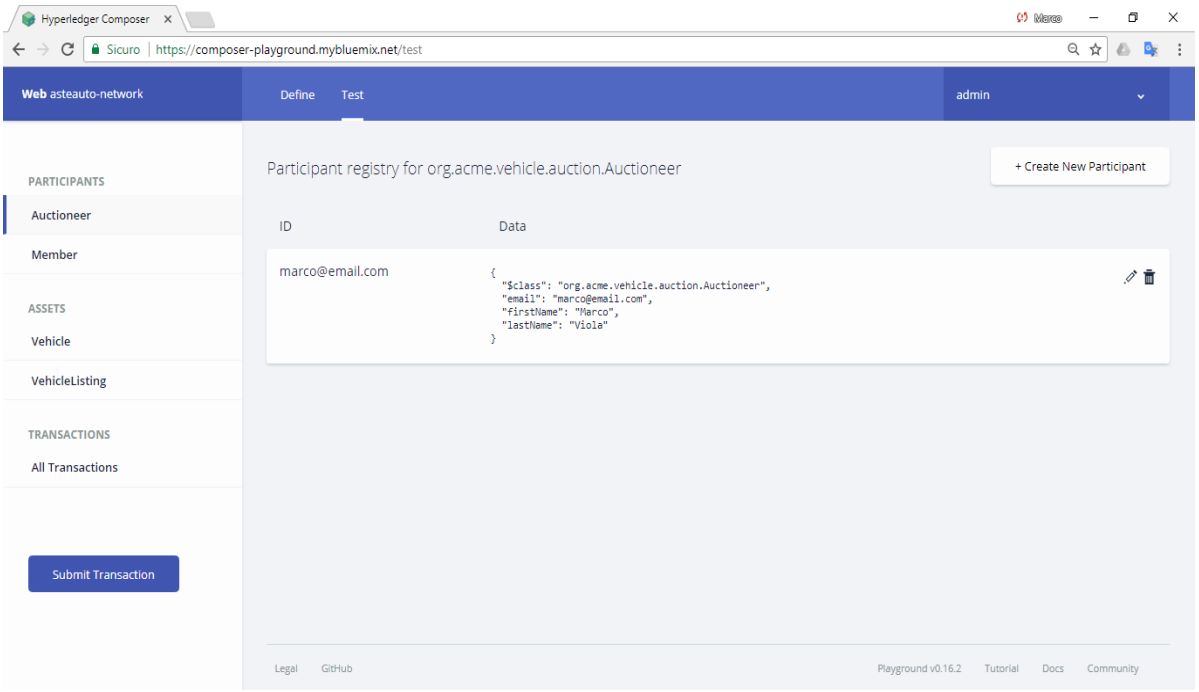
Alice e Bob piazzeranno delle offerte, Charlie invece è l'attuale possessore dell'automobile che sarà venduta all'asta.





Viene anche istanziato un banditore, che potrà avere il ruolo di supervisione, ma comunque non è indispensabile per il funzionamento della rete.

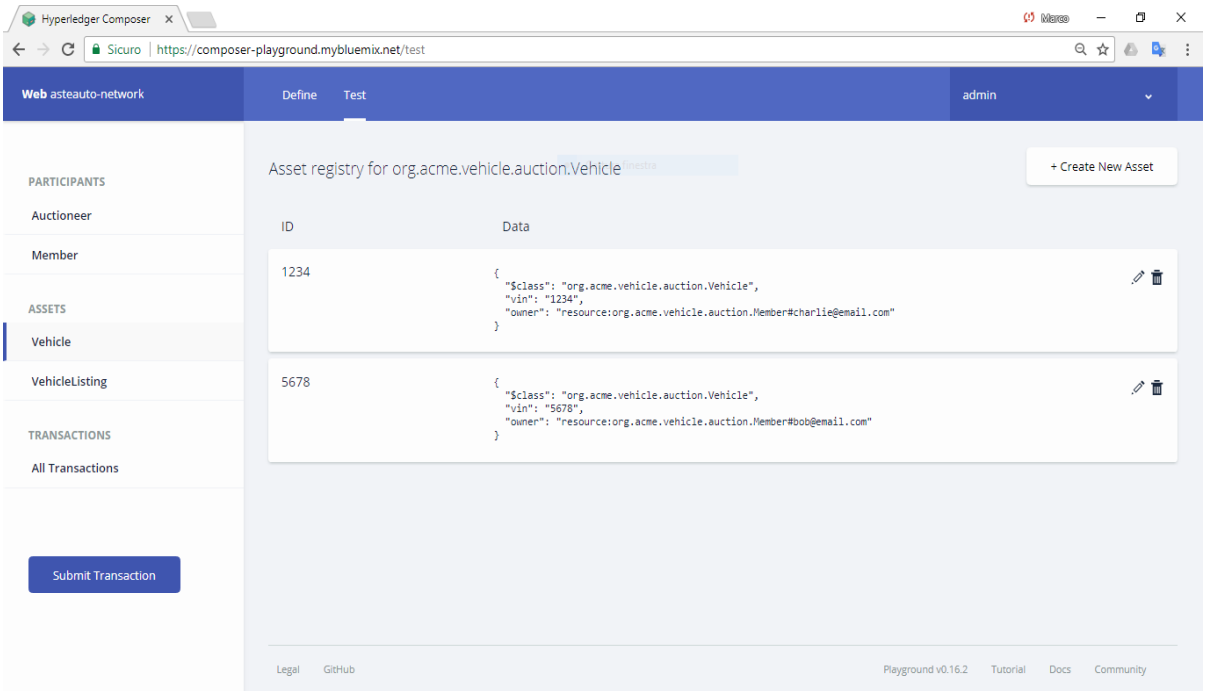
– Marco Viola (marco@email.com)



Vengono ora istanziati i beni.

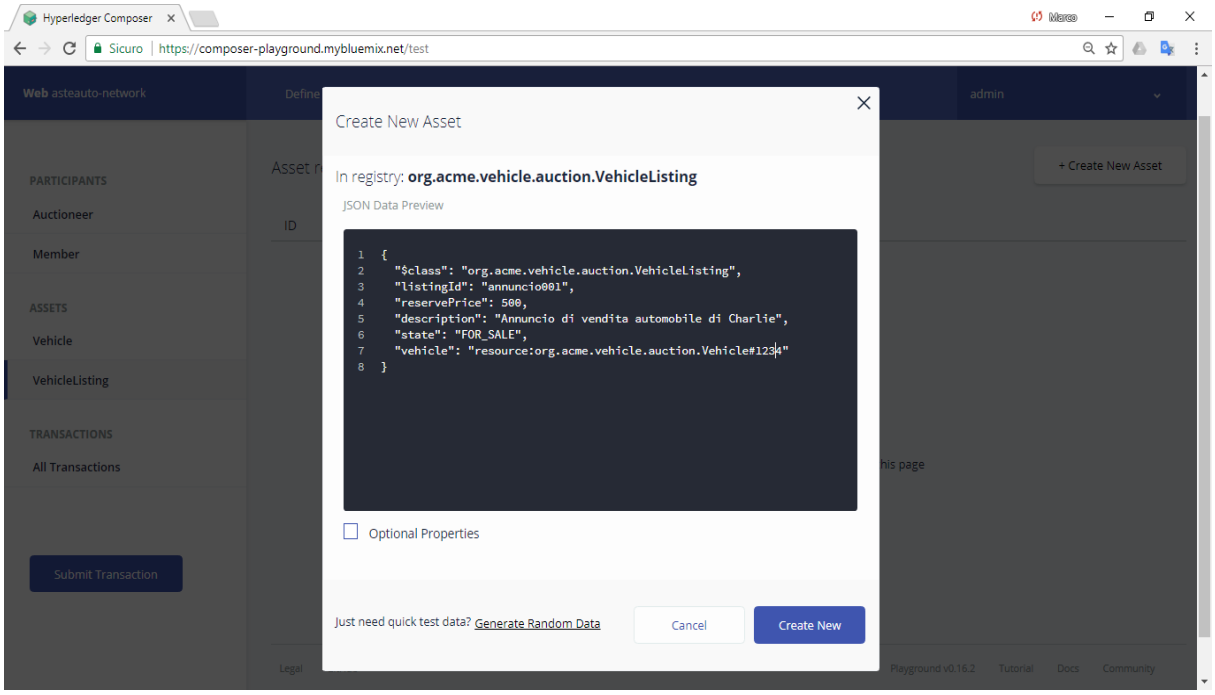
Inseriamo due automobili. La prima, identificata dal numero 1234, è l'automobile di proprietà di Charlie, che sarà oggetto dell'asta.

La seconda invece, con identificativo 5678, è di proprietà di Bob e non sarà venduta.



È possibile notare come un veicolo viene associato ad un membro (identificato dalla email) attraverso l'attributo "*owner*".

Inseriamo un nuovo annuncio di vendita creando un'istanza della classe *VehicleListing*. Questa inserzione dovrà contenere, tra le altre informazioni, un riferimento al veicolo oggetto della vendita, un prezzo di riserva e lo stato dell'annuncio (FOR_SALE).



Tutti i membri e le risorse sono stati inseriti all'interno della rete blockchain ed è possibile procedere con il test della rete.

5.4 Trasferimenti di risorse

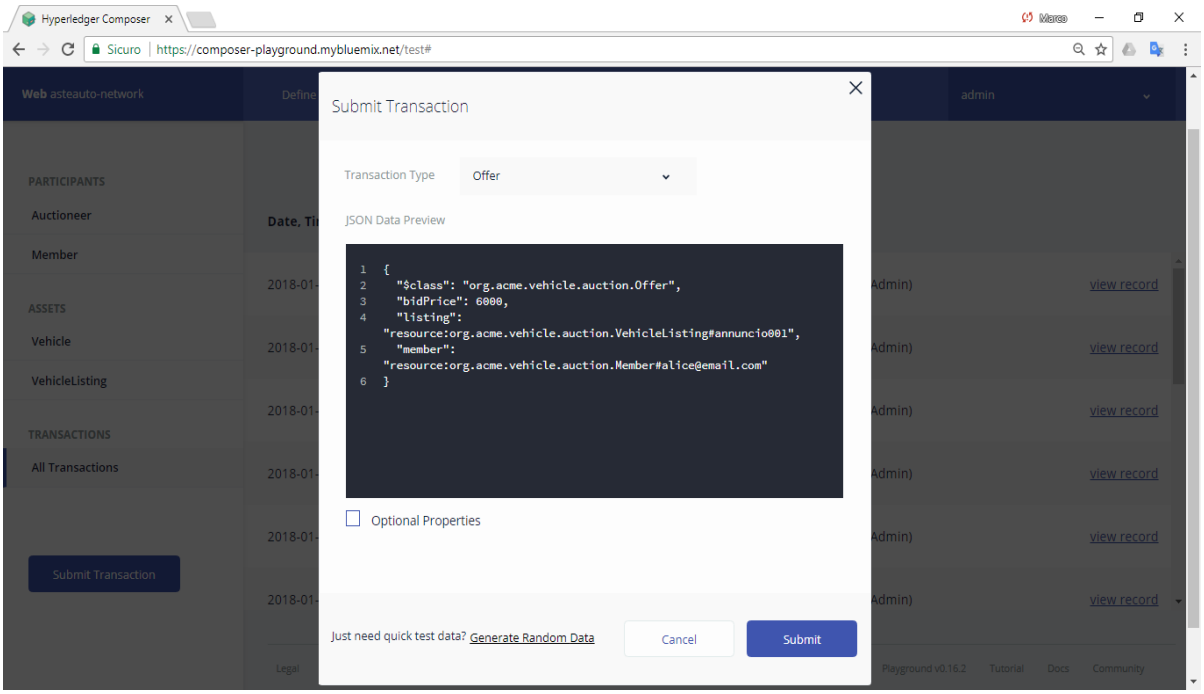
Le risorse possono essere trasferite mediante l'esecuzione di transazioni.

Per creare una nuova transazione occorre premere il pulsante "*Submit Transaction*", che si trova in basso a sinistra nella scheda "*Test*".

Dopo aver premuto tale pulsante, occorre selezionare il tipo di transazione da inserire, tra quelle definite all'interno del file *"models/auction.cto"*.

La prima transazione è di tipo "*Offer*", ossia un'offerta da parte di un membro interessato all'automobile in vendita nell'annuncio.

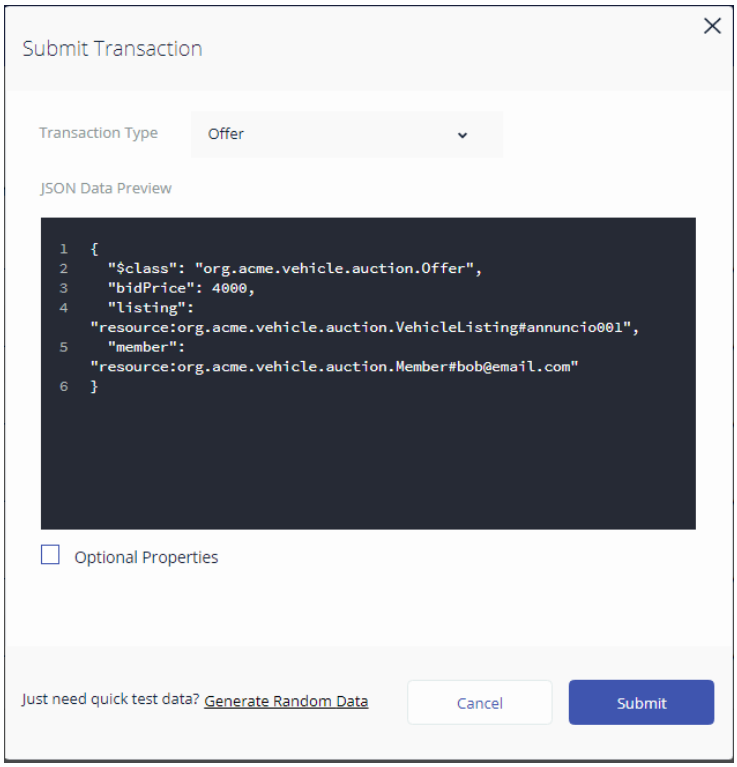
Una transazione di tipo offerta contiene l'importo, il riferimento al membro che esegue l'offerta ed il riferimento all'annuncio.



In questo caso il membro Alice ha offerto 6000 sull'unico annuncio presente nella rete.

È importante notare che la transazione non può essere modificata una volta che è stata inviata. Questa è una delle caratteristiche chiave della blockchain.

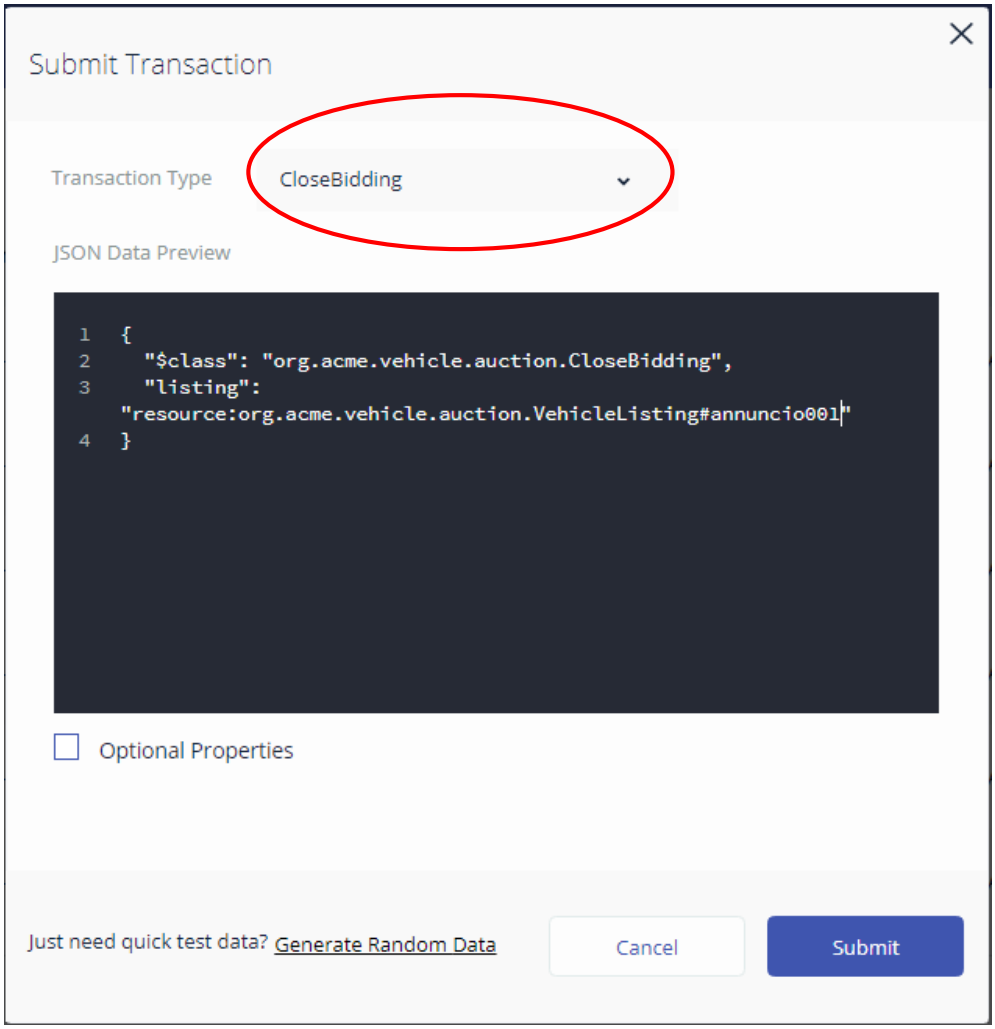
Anche Bob fa un'offerta, ma con un importo più basso, ovvero 4000.



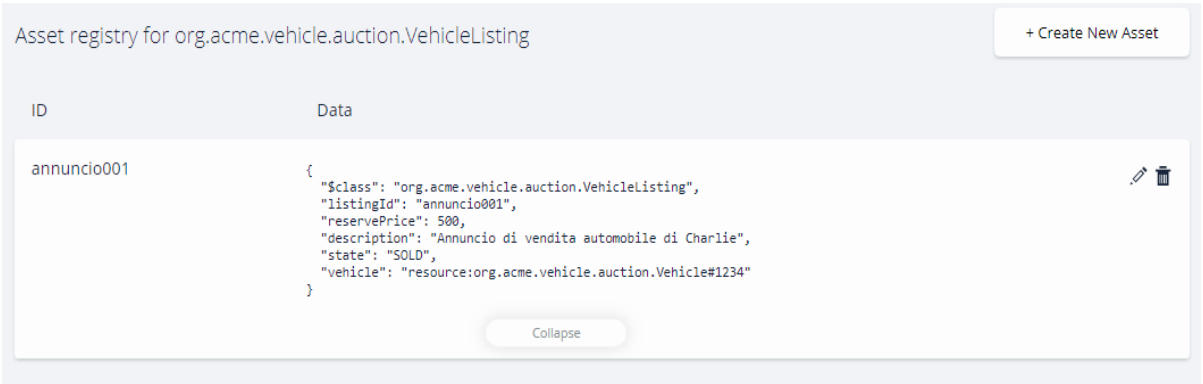
Ora vogliamo chiudere le offerte sull'annuncio.

La terza transazione che inseriremo sarà di tipo *CloseBidding*, ovvero sarà la transazione che chiuderà l'asta ed avvierà una sequenza di operazioni specificate dagli scripts del file *"lib/logic.js"*. Queste operazioni consistono nell'ordinare le offerte per determinare la maggiore, controllare che superi il prezzo di riserva, trasferire la proprietà del veicolo al membro che ha fatto l'offerta migliore e cambiare lo stato dell'annuncio su "SOLD". Inoltre verrà trasferito l'importo dal bilancio dell'acquirente a quello del venditore.

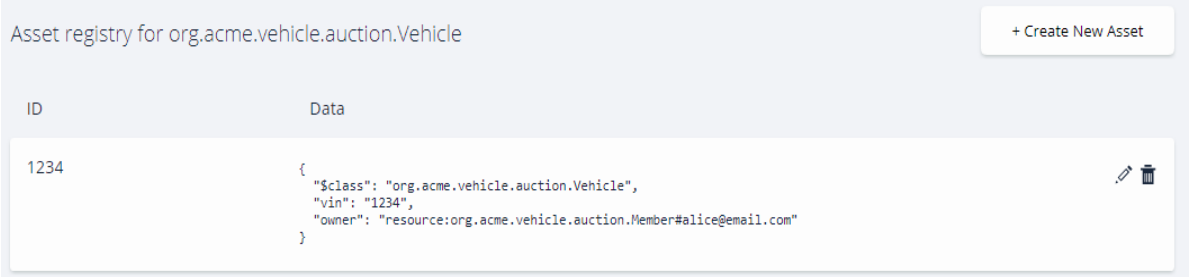
Il tutto avviene automaticamente nella rete nel momento in cui viene inserita la transazione, senza l'intervento di un utente o di terze parti.



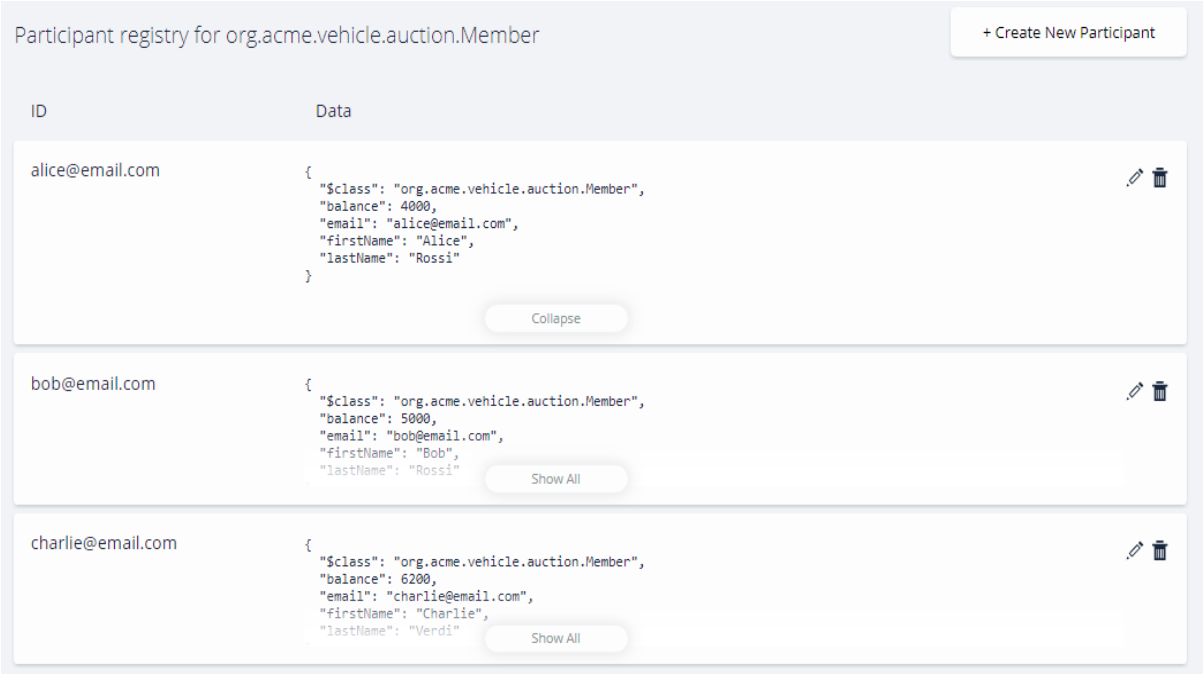
Verifichiamo se tutte le azioni sono state eseguite correttamente.



L'annuncio è stato chiuso, lo stato è cambiato in "SOLD", venduto. Non sarà più possibile inserire offerte riferite a questo annuncio.



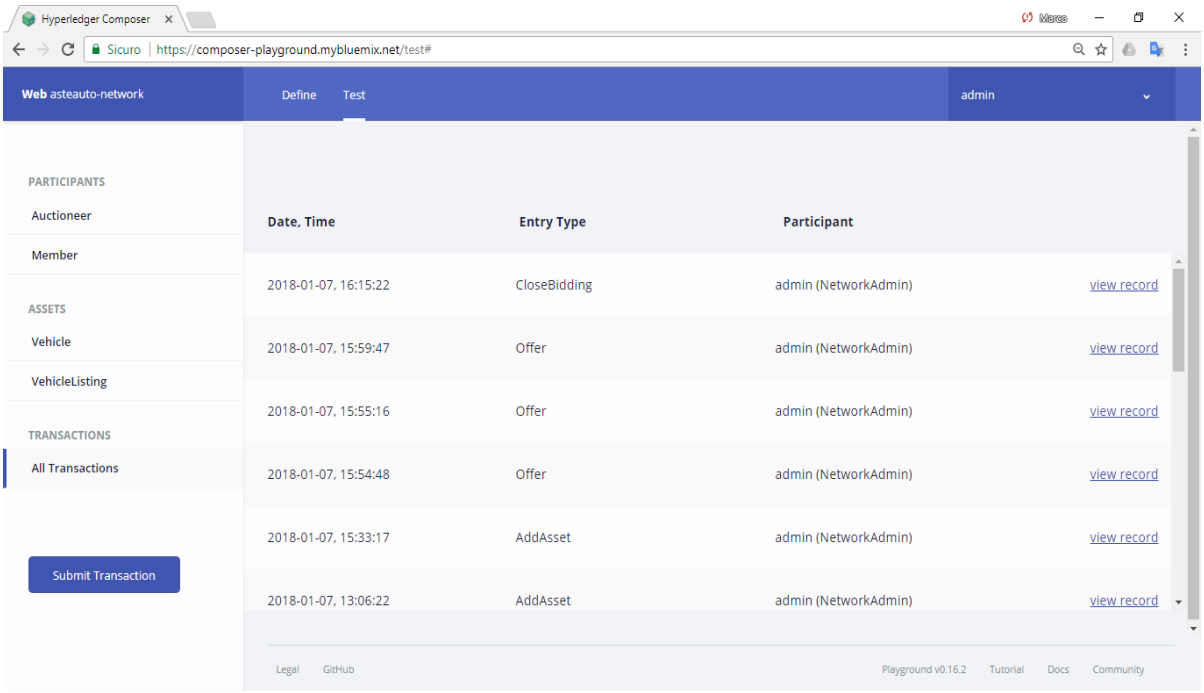
Il proprietario dell'automobile oggetto dell'asta è Alice, pertanto è avvenuto il passaggio di proprietà.



Il saldo di Alice è stato modificato a 4000 (all'inizio era 10000) poiché i 6000 offerti per l'automobile sono stati trasferiti nel saldo del venditore Charlie.

Bob ha partecipato all'asta facendo un'offerta, ma non avendo vinto il suo saldo è rimasto invariato.

Per visionare tutte le transazioni eseguite, comprese quelle generate nella creazione dei membri e dei beni, occorre aprire il registro *"All Transactions"*. Esso rappresenta l'*Historian Registry* di Hyperledger Composer, ossia il registro storico che tiene traccia di tutto ciò che avviene nella rete.



6 Commenti Conclusivi

In questo elaborato è stato presentato un esempio di come in poco tempo sia possibile modellare una business network attraverso lo strumento *Hyperledger Composer*, in particolare usando l'interfaccia grafica web *Composer Playground*.

Questi strumenti sono parte del progetto open source *Hyperledger* della Linux Foundation e permettono la realizzazione di soluzioni blockchain di tipo privato e *permissioned*, particolarmente utili in ambito business.

Una blockchain privata può essere creata da una singola azienda o da un insieme di esse e permette di trasferire qualsiasi bene materiale, non materiale o digitale, non solo valute virtuali (come nel caso di Bitcoin).

Essendo la rete privata, i membri vengono invitati a partecipare e pertanto sono conosciuti. Possono essere gestite le autorizzazioni ed è possibile assegnare ai partecipanti i diritti d'accesso solamente a certi dati. Le transazioni avvengono rispettando degli Smart Contracts i quali, una volta accettati dalle parti nel momento in cui decidono di partecipare alla rete, garantiscono l'imparzialità delle operazioni, riducendo le possibili contestazioni ed aumentando la fiducia reciproca tra le parti.

Il consenso in una blockchain *permissioned* non è ottenuto attraverso il *mining* ma attraverso un processo chiamato "approvazione selettiva". Ad alcuni membri viene affidato il compito di verificare le transazioni, più o meno la stessa cosa che avviene nelle transazioni tradizionali. Questo è diverso da Bitcoin e da tutte le altre reti di tipo pubblico e *Permissionless*, dove per soddisfare la proof-of-work l'intera rete deve lavorare per verificare le transazioni, generando problemi di tipo energetico e di scalabilità.

Se paragoniamo le blockchain pubbliche e *Permissionless* (come Bitcoin ed Ethereum) ad Internet, le blockchain private e *permissioned* (come *Hyperledger Fabric*) possono essere paragonate ad una rete Intranet, dove solo i nodi autorizzati hanno accesso.

7 Bibliografia / Sitografia

Capitolo 2:

1. https://bitcoin.org/bitcoin.pdf
2. http://bitcoin.org
3. http://en.bitcoin.it/wiki/
4. https://en.bitcoin.it/wiki/Hardware_wallet
5. https://en.wikipedia.org/wiki/Cryptography
6. http://www.hashcash.org/
7. https://it.wikipedia.org/wiki/Funzione_crittografica_di_hash
8. https://en.wikipedia.org/wiki/SHA-2
9. https://it.wikipedia.org/wiki/Nonce

Capitolo 3:

10. https://it.wikipedia.org/wiki/Blockchain
11. http://www.blockchain4innovation.it
12. https://www.ibm.com/blockchain/
13. http://hyperledger-fabric.readthedocs.io/en/release/blockchain.html
14. https://medium.com/blockchainspace/3-comparison-of-bitcoin-ethereum-and-hyperledger-fabric-cd48810e590c
15. https://developer.ibm.com/blockchain/
16. https://www.ibm.com/developerworks/cloud/library/cl-ibm-blockchain-101-quick-start-guide-for-developers-bluemix-trs/index.html

Capitolo 4:

17. https://www.hyperledger.org/
18. https://www.hyperledger.org/projects/fabric
19. https://www.hyperledger.org/projects/composer
20. https://hyperledger.github.io/composer/

Capitolo 5:

21. https://composer-playground.mybluemix.net/
22. https://ibm.ent.box.com/v/BlockFiles
23. https://hyperledger.github.io/composer/reference/cto_language.html