UNIVERSITÀ DI PISA

Corso di Laurea in Informatica Umanistica

RELAZIONE

Utilizzo della blockchain per la gestione di una piattaforma di e-commerce per il sociale

Candidato: *Matteo Tito*

Relatori: *Andrea Marchetti*

Clara Bacciu

Anno Accademico 2021-2022

Indice generale

1. Introduzione

A partire dal suo primo sviluppo nel 2009, la tecnologia *blockchain* non ha mai smesso di attirare l'interesse di sviluppatori, aziende e governi. Sebbene sia nata principalmente per la gestione di transazioni in ambito finanziario, le sue prime applicazioni hanno permesso di ripensare al modo in cui i dati vengono gestiti, ossia eliminando la necessità di autorità centrali e restituendo il controllo agli utenti.

Difatti, dal 2014 sono nate le prime riviste, come *Ledger Publishes First Volume of Peer-Reviewed Blockchain Research - CoinDesk*, dedicate alla ricerca e all'innovazione della *blockchain*, le quali hanno permesso di diffondere informazioni riguardo nuovi metodi d'impiego di questa tecnologia, definendola come "Blockchain 2.0". Dalla cultura, alla politica, dall'economia all'ambito sociale, numerosi sono stati i campi in cui la *blockchain* ha potuto offrire un contributo sostanziale al progresso.

Seguendo la scia degli obiettivi avanzati negli ultimi anni, all'interno di questo lavoro viene proposta l'applicazione della tecnologia *blockchain* all'ambito sociale, mediante la realizzazione di una piattaforma decentralizzata dedicata al progetto *Banca del Noi*.

Quest'ultimo mira alla creazione di un sistema di scambio, all'interno di una comunità, basato sull'utilizzo di una moneta complementare, al fine di permettere a chiunque di poter contribuire con il proprio lavoro al benessere della società.

L'obiettivo di questo lavoro, dunque, è quello di proporre una soluzione in grado di garantire un sistema di scambio trasparente e sicuro, mediante l'utilizzo di una nuova tecnologia. Il lavoro, in particolare, si è incentrato sulla ricerca e l'utilizzo di una rete *blockchain* in linea con le esigenze richieste da *Banca del Noi*, inoltre, si è soffermato sullo sviluppo della piattaforma ad esso dedicata, sulla base delle funzionalità richieste. Ciò ha permesso di incontrare e analizzare, problematiche di natura strutturale, etica e realizzativa, riguardanti il tema rappresentato dalla famiglia delle tecnologie *Distributed Ledger*.

2. Stato dell'arte

2.1. Sistemi di domanda e offerta

2.1.1. Platform economy

Negli ultimi anni lo sviluppo tecnologico ha inciso sui sistemi di domanda e offerta, ossia sugli strumenti e sulle metodologie utilizzate per la compravendita di beni e servizi. L'affermazione della *platform economy*, ovvero l'attività economica e sociale basata sulle piattaforme digitali, ha portato ad un notevole cambiamento all'interno del mercato[1]. Le piattaforme, infatti, sono diventate strumenti di disintermediazione sempre più consolidati, capaci di interagire direttamente con il cliente, in modo da moltiplicare le opportunità di fidelizzarlo.

La tipologia di piattaforma più diffusa è di transazione. Il successo di tale modello dipende principalmente dall'inserimento di ordinamenti reputazionali (basati su *feedback* e *review*), volti allo sviluppo di sistemi fiduciari, in relazione alle transazioni effettuate all'interno del mercato della piattaforma. Un esempio di piattaforma di transazione è rappresentato dal *marketplace*, in italiano "luogo di mercato", ossia una piattaforma dove è possibile effettuare scambi commerciali di beni e servizi (es. Amazon.it).

Ad oggi, le piattaforme di transazione si occupano anche del mercato d'impiego, avendo come obiettivo l'agevolazione dell'incontro tra domande ed offerte di lavoro.

[1] https://ebitemp.it/wp-content/uploads/2019/01/Adapt_completo-2.pdf
Rapporto su l' *Evoluzione del mercato dell'incontro tra domanda e offerta di lavoro all'epoca della disintermediazione e dell'uso delle piattaforme tecnologiche,* a cura di E. Dagnini e S. Spattini, Dicembre 2017.

2.1.2. Gig economy

Le nuove piattaforme digitali che si occupano del mercato del lavoro, come per le piattaforme commerciali di beni, sfruttano il modello di transazioni, applicandolo all'incontro tra domande ed offerte di lavoro. Ciò ha dato luogo al fenomeno della *gig economy*, ossia un mercato del lavoro basato su posizioni lavorative tipicamente su richiesta, indipendenti e a breve termine.

All'interno della *gig economy*, sulla base del compenso ceduto, è possibile individuare tre categorie di piattaforme:

- Basate sulla moneta: laddove lo scambio avviene tramite moneta reale o complementare (es. *Fiverr*);
- Basate sul tempo: laddove lo scambio avviene utilizzando come unità di misura il tempo impiegato (es. *Time Republik*);
- Senza compenso: laddove non è integrato alcun metodo di compenso (es. *MyANPAL*).

La gestione delle transazioni da parte di ognuna delle diverse tipologie di piattaforme rappresenta un elemento comune e fondamentale per il corretto funzionamento di queste. Secondo tali esigenze, infatti, numerose sono le piattaforme digitali che hanno deciso di utilizzare una nuova tecnologia, la *blockchain*, al fine di sfruttarne i vantaggi.

2.2.Blockchain

2.2.1. Definizione

La tecnologia *blockchain* fa parte dell'insieme di tecnologie *Distributed Ledger*, ossia "registro distribuito", che permettono che un insieme di dati digitali sia replicato, condiviso e accessibile a tutti gli utenti senza necessità di un'autorità centrale. La *blockchain*, in italiano "catena di blocchi", è dunque definita come un registro digitale aperto e distribuito, in cui i dati, tipicamente noti come "transazioni"[2], sono memorizzati e raggruppati in forma sicura, verificabile e permanente, all'interno di blocchi. Una volta registrati, infatti, i dati presenti in un blocco non possono essere alterati, a meno che non vengano modificati tutti i blocchi successivi. L'aggiunta di un blocco è possibile solo attraverso un sistema di validazione, cioè un protocollo condiviso che prevede il raggiungimento di un consenso tra tutti gli utenti.

La *blockchain*, dunque, è un registro in continua crescita, in cui i blocchi sono interconnessi in ordine cronologico, attraverso la registrazione di una marca temporale e protetti mediante l'utilizzo della crittografia.

2.2.2. Descrizione dei blocchi

I blocchi, come precedentemente accennato, rappresentano un insieme di transazioni, connessi tra di loro e ordinati cronologicamente (vedi fig. 1). Ogni blocco contiene un codice *hash*, ossia una chiave crittografica utilizzata per la mappatura dei dati, con la quale fa riferimento al blocco precedente. Poiché l'*hash* viene calcolato sulla base dei dati del blocco, questi ne evita l'alterazione, in quanto un qualsiasi cambiamento nel blocco, invaliderebbe tutti i blocchi successivi, poiché i rispettivi *hash* cambierebbero di conseguenza (vedi fig. 2 e 3).

[2] Termine mantenuto per motivi storici, poiché la prima blockchain è stata utilizzata per la criptovaluta Bitcoin, e le transazioni rappresentavano uno scambio di valore.

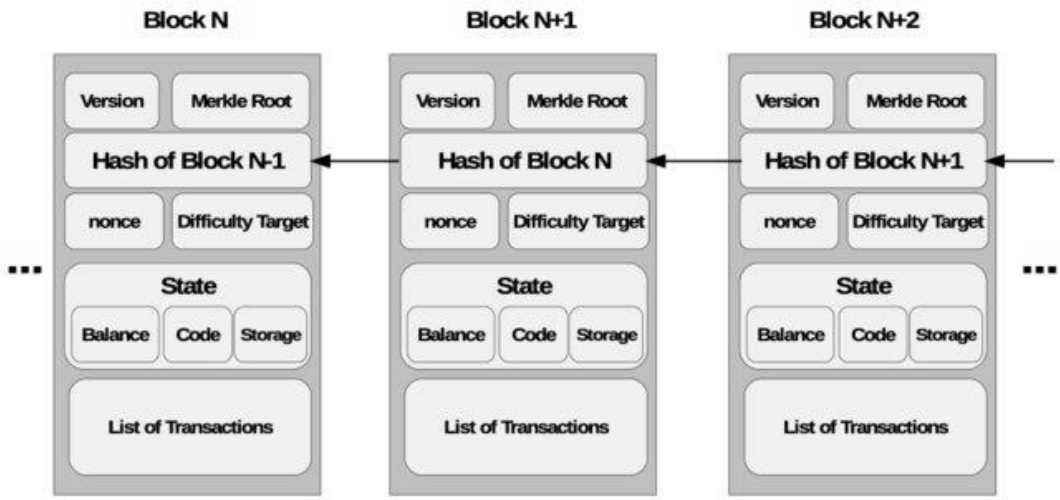

Figura 1. Esemplificazione struttura dei blocchi.

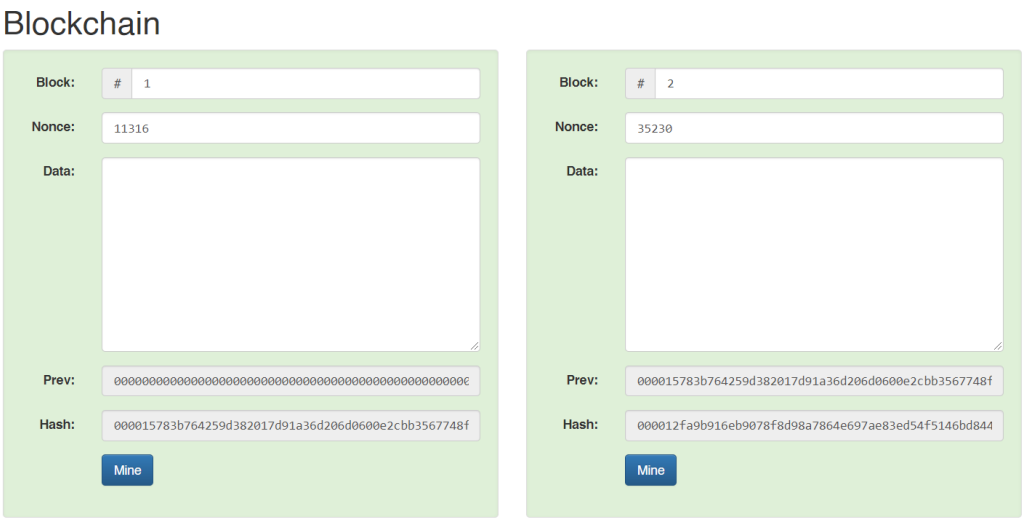

Figura 2. Esempio di hash blocchi validi.

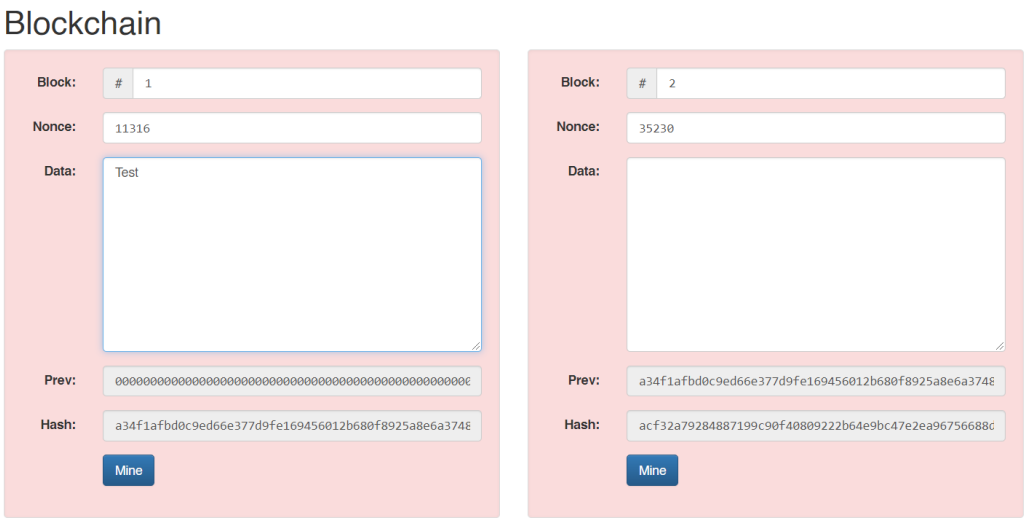

Figura 3. Esempio hash blocchi invalidi.

Per preservare l'ordine cronologico delle transazioni, i blocchi sono ordinati in modo rigoroso, difatti tutti i partecipanti alla rete concordano, attraverso un algoritmo di consenso, sul numero e sulla cronologia esatta dei blocchi. La creazione, invece, di un nuovo blocco viene gestita dal processo denominato *mining*.

Gli elementi costitutivi di un blocco sono simili nelle diverse blockchain. In particolare, in *Ethereum* si trovano i seguenti:

- *Timestamp*: ora di *mining* del blocco;
- Numero del blocco: lunghezza della *blockchian* rappresentata in numero di blocchi;
- mixHash: identificatore univoco del blocco;
- Hash padre: identificatore univoco del blocco precedente;
- Elenco transazioni: insieme delle transazioni del blocco;
- Stato sistema;
- Nonce: *hash* utilizzato per provare l'ottenimento del consenso.

2.2.3. Algoritmi di consenso

Al fine di poter concordare sullo stato corrente all'interno della rete *blockchain*, vengono utilizzati gli algoritmi di consenso. Questi, denominati anche protocolli o meccanismi di consenso, permettono ai sistemi distribuiti di interagire e garantire sicurezza. Ad oggi ne esistono diverse tipologie, tuttavia i più comuni sono gli algoritmi: *Proof of Work* e *Proof of Stake*.

Il primo algoritmo, utilizzato anche da *Ethereum*, si basa sulla presenza di *miner*, nodi della rete, i quali competono per la creazione di nuovi blocchi in cui inserire le transazioni elaborate. La competizione riguarda la risoluzione di un algoritmo matematico, mediante potenza computazionale, che produce il collegamento crittografico tra il blocco concorrente e il blocco che lo ha preceduto. Il vincitore condivide il nuovo blocco con il resto della rete e guadagna un compenso in criptovaluta. La sicurezza del protocollo è garantita dal fatto che la potenza di elaborazione necessaria a frodare l'intera catena supera il profitto ottenibile.

L'algoritmo *Proof of Stake*, invece, si basa sulla presenza di un insieme di validatori, i quali dimostrano il possesso di un determinato ammontare di criptovaluta, denominato *stake*. Sulla base di ciò, viene scelto soltanto un validatore, mediante estrazione casuale, il quale può, di conseguenza, creare nuovi blocchi, condividerli e guadagnare ricompense. La sicurezza del protocollo è garantita dal possesso di criptovaluta e dalla casualità di scelta del validatore. Inoltre, in caso di comportamento malevolo, viene eseguito un taglio dello *stake*.

2.2.4. Mining

Il *mining* è il processo che consente la creazione di un blocco di transazioni all'interno di una rete *blockchain*, affiancato dall'algoritmo di consenso.

Esso parte dalla richiesta di un utente di effettuare una transazione, mediante un nodo nella rete. Dopo aver ricevuto tale richiesta, ogni nodo della rete la aggiunge ad un elenco di richieste di transazioni non ancora inserite in un blocco. Successivamente, queste richieste vengono raggruppate da un *miner* o validatore, all'interno di un potenziale nuovo blocco, in modo da massimizzare le commissioni sulle transazioni che verranno guadagnate. A questo punto, esso verifica la validità di ogni transazione e inizia il processo di produzione del "certificato di legittimità" per il potenziale blocco, sulla base del protocollo di consenso in vigore. Infine, il *miner* o il validatore trasmette il blocco completato all'interno della rete. A questo punto, gli altri nodi vengono a conoscenza del nuovo blocco, ne verificano il certificato ed eseguono tutte le transazioni sul blocco. Infine, al termine di questa operazione, i nodi aggiungono il blocco alla coda della *blockchain*, aggiornandone lo stato.

L'intero processo di *mining* di una transazione avviene una sola volta, mentre per quanto riguarda la transazione, essa viene eseguita e verificata da tutti i nodi della rete.

2.2.5. Decentramento

Il termine decentramento, all'interno della definizione di rete *blockchain*, indica l'utilizzo di un *network* distribuito che consente la ripartizione e la replicazione dei dati all'interno di una rete. Ogni nodo della rete, infatti, possiede una copia della *blockchain*, in questo modo la sicurezza e la qualità del database è garantita grazie alla sua molteplice riproduzione (vedi fig. 4). In questo scenario, tutti i nodi sono allo stesso livello, difatti, svolgono le medesime operazioni, quali: validazione delle transazioni, costruzione dei blocchi e verifica della *blockchain*. Al livello di sicurezza, infine, è importante notare come tale soluzione permetta di evitare lo SPOF, *Single Point Of Failure*, ossia la possibile cessazione di un servizio da parte del sistema, dovuta al malfunzionamento hardware o software di una parte di esso. Ciò, dunque, consente di diminuire il livello di vulnerabilità del sistema.

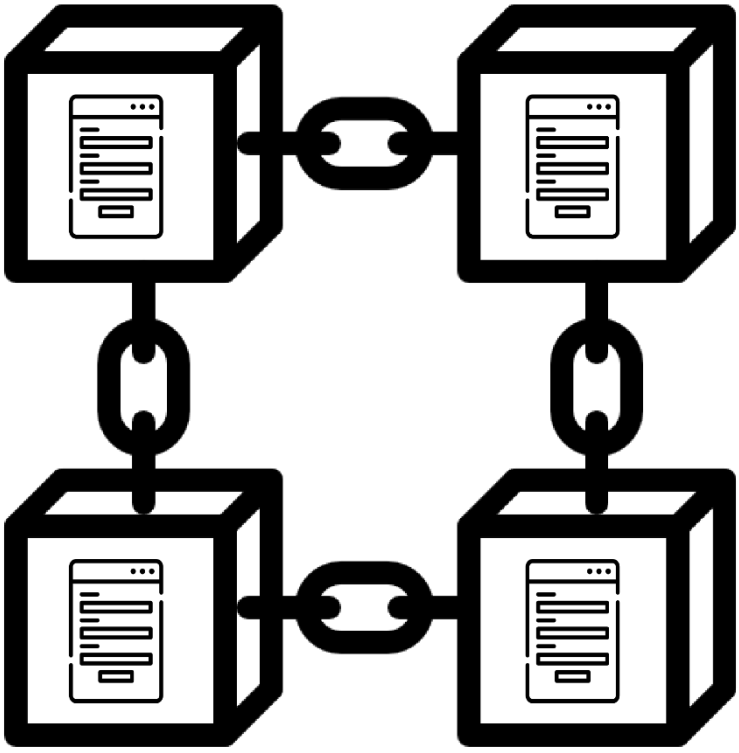

Figura 4. Rappresentazione grafica nodi blockchain.

2.2.6. Tipologie

Le reti *blockchain* si suddividono in due maggiori tipologie: pubbliche e private.

Le *blockchain* pubbliche, chiamate anche *permissionless*, sono reti che permettono a qualunque utente di poterne far parte, senza alcuna restrizione. Tali utenti, infatti, hanno accesso ai dati, in lettura e scrittura, e possono esaminare il funzionamento della rete, in maniera libera. Il mantenimento economico di questa tipologia di rete dipende dal sistema in esso integrato, generalmente basato sulla riscossione delle commissioni per ogni transazione, nel momento della creazione di un nuovo blocco (operazione chiamata *mining*). Il premio riscosso è spesso rappresentato da una *criptovaluta*, o parte di essa, definita, nel caso del protocollo *Proof of Work*, come la rappresentazione digitale del valore corrispondente alla potenza di elaborazione spesa per convalidare una transazione.

Le *blockchain* private o *permissioned*, invece, sono caratterizzate dal controllo degli accessi alla rete tramite autorizzazioni. In questo scenario vi sono:

- le *Consortium Blockchain*: reti dove i permessi sono gestiti da un insieme di nodi selezionati e ritenuti attendibili;
- le *Fully Private Blockchain*: reti dove i permessi sono gestiti in modo centralizzato.

In entrambi gli scenari sopra elencati, l'accesso alle transazioni è privato e, inoltre, il mantenimento economico della rete dipende da chi supporta il progetto. A differenza delle *blockchain* pubbliche, nelle *blockchain* private non viene previsto l'utilizzo di *criptovalute*.

2.2.7. Applicazioni

La tecnologia *blockchain* è stata pensata come una tecnologia di partenza, un elemento di sviluppo utile per realizzare nuovi strumenti e applicazioni. Essa può essere integrata in diverse aree, anche al di fuori dell'attività finanziaria. Ad oggi, uno dei maggiori impieghi di questa tecnologia è associato alle *criptovalute* (es. Bitcoin, Ethereum, Cardano, ecc.), tuttavia, ulteriori ambiti applicativi riguardano:

- Sviluppo di NFT, *Non fungible token*, cioè gettoni o simboli utilizzati per garantire l'autenticità e la proprietà di un bene fisico o digitale;
- Archiviazione di dati e gestione delle transazioni;
- Notarizzazione dei documenti per la certificazione dei trasporti pubblici e privati;
- Sistemi di pagamento per facilitare il *crowdfunding;*
- Sistemi assicurativi peer-to-peer;
- Sistemi di votazione decentralizzata.

Tale elenco rappresenta solo una esemplificazione dei molteplici impieghi e possibilità di cui la tecnologia *blockchain* dispone[3]. Il motivo di tale successo e diffusione è legato principalmente ai numerosi vantaggi che questa tecnologia offre, in particolare, riguardo la realizzazione di registri rapidi, economici e sicuri. Per queste ragioni, infatti, numerosi sono i sistemi di compravendita di beni e servizi che utilizzano questa tecnologia, in quanto essi possono contare su affidabilità, immutabilità e verificabilità delle transazioni.

[3] Ulteriori approfondimenti all'interno dell'analisi *Come la tecnologia blockchain può cambiarci la vita*, a cura di P. Boucher, Febbraio 2017.

2.2.8. Limiti

Nonostante la tecnologia *blockchain* presenti vantaggi e venga applicata in numerosi campi, ad oggi sono ancora presenti diversi limiti legati al suo utilizzo. Tra questi, i principali riguardano:

- Scalabilità: ossia la capacità di gestire un numero sempre maggiore di transazioni senza che le prestazioni della rete ne risentano. In particolare, il basso *throughput*, presente nelle *blockchain* pubbliche come *Ethereum* e *Bitcoin*, consente di effettuare dalle 7 alle 30 transazioni al secondo, rappresentando un limite considerevole all'espansione del sistema, specialmente se paragonato ai canonici sistemi di pagamento che, invece, possono elaborare fino a 50.000 transazioni al secondo. Una possibile soluzione, tuttavia, è rappresentata dall'uso di *blockchain* private, capaci di processare fino a 3500 transazioni al secondo.
- Latenza: ossia l'intervallo di tempo necessario a processare una transazione. Nelle reti che utilizzano il protocollo *Proof of Work*, come *Bitcoin*, è necessario attendere dai 10 ai 30 minuti di tempo affinché una transazione venga elaborata, il che rappresenta, un ulteriore limite alla scalabilità del sistema. Una soluzione proposta fa riferimento all' implementazione di *Lighting Network*, un protocollo di secondo livello che si sovrappone alla rete *blockchain* per consentire transazioni più veloci.
- Consumo di risorse: il processo di *minig* comporta una notevole necessità di capacità di calcolo e, di conseguenza di energia elettrica, in particolar modo nelle reti che utilizzano il protocollo *Proof of Work*, in cui il problema computazionale da risolvere è estremamente complesso. Una possibile soluzione è rappresentata dall'utilizzo di protocolli alternativi, come il *Proof of Stake*, i quali necessitano di un volume energetico nettamente inferiore.

I limiti precedentemente illustrati, portano alla luce la necessità di un'attenta valutazione, a priori, dell'uso di questa tecnologia, volta alla corretta predisposizione del sistema in funzione delle caratteristiche ricercate.

2.2.9. Principi etici

La *blockchain* è una tecnologia che ha la capacità di ridefinire il concetto di valore e di fiducia, sia in un contesto sociale che in quello economico. Come precedentemente illustrato, numerosi sono i campi di applicazione di questa tecnologia, i limiti e i vantaggi che essa offre, tuttavia, altrettanti sono le preoccupazioni ed i dubbi legati alla sua implementazione.

La natura decentralizzata della *blockchain*, infatti, sembra presumere un sistema di autoregolamentazione che, in linea di principio, opererebbe in parallelo ai classici strumenti giuridici, tuttavia, in assenza di una corretta protezione istituzionale, essa potrebbe portare allo sviluppo di sistemi oligarchici. Di conseguenza, ciò renderebbe impossibile l'esecuzione delle canoniche azioni legali laddove vi si presenta una violazione del diritto.

Un'ulteriore preoccupazione riguarda il possibile scoppio della bolla speculativa rappresentata dall'alto numero di investitori in criptovalute e NFT. Considerati via via come beni innovativi, l'aumento della loro domanda è cresciuto notevolmente nel corso degli anni, portando ad un conseguente aumento di prezzo. Tuttavia, non è stata ancora definita l'effettiva utilità di questi beni, e ciò potrebbe portare gli investitori ad effettuare un cambio di direzione, favorendo così lo scoppio della bolla e causando un tracollo economico.

In conclusione, i vantaggi che questa tecnologia offre, per quanto riscontrabili, risultano comunque opinabili e non assoluti. Ad oggi, dunque, l'utilizzo della *blockchain* e dei suoi strumenti, come avviene per ogni nuova tecnologia, presenta ancora dei rischi, i quali necessitano dell'intervento di una normativa e di competenze adeguate. Ciò, tuttavia, non toglie la possibilità di sperimentazione ed innovazione consapevole.

2.3.Inclusione sociale

2.3.1. Definizione

L’inclusione sociale indica la condizione in cui tutti gli individui vivono in stato di equità e pari opportunità, indipendentemente dalla presenza di elementi limitanti. In tal senso, l’inclusione sociale si rivolge a tutti gli individui senza che vi siano differenze definite da categorie o criteri, poiché la differenza è concepita come elemento personale di ciascuno di essi. Ulteriore obiettivo riguarda l’eliminazione di ogni forma di discriminazione, spingendo il sistema culturale e sociale verso il cambiamento, inteso come una partecipazione attiva e completa di tutti. Essa, infine, aspira alla costruzione di contesti inclusivi, capaci di abbattere ogni forma di barriera e non ammette la concezione di abilismo.

2.3.2. Strumenti

L’inclusione sociale, all’interno del panorama normativo italiano, è un fenomeno legato principalmente a due elementi limitanti: povertà e disabilità.

Il tema della povertà viene trattato dall’ordinamento come un fenomeno complesso, connesso alle opportunità e, dunque, alla possibilità di partecipare pienamente alla vita sociale del paese. Le politiche nazionali per l'inclusione sociale, difatti, si differenziano in un insieme di iniziative specifiche sia per ambito di intervento, sia per tipologia di strumento.

Nello specifico, alcune delle politiche nazionali si caratterizzano come l'insieme delle misure volte a sostenere i redditi dei cittadini, con particolare riguardo agli interventi di inclusione attiva, finalizzati all’inserimento o reinserimento sociale.

Tra le misure di sostegno al reddito, a partire dal 6 marzo 2019, vi è il Reddito di cittadinanza, di cui i cittadini possono fare richiesta seguendo un percorso personalizzato di inserimento lavorativo e di inclusione sociale.

Per quanto riguarda il tema della disabilità in relazione all'inclusione sociale, invece, importante riferimento normativo è rappresentato dalla legge 68 pubblicata nel Supplemento Ordinario n. 57/L alla Gazzetta Ufficiale il 23 marzo 1999. Tale legge, con riferimento al diritto al lavoro, promuove l'inserimento e l'integrazione delle persone affette da disabilità attraverso servizi di sostegno e di collocamento mirato in base alle singole esigenze. Con l'aggiornamento della legge, risalente al Decreto legislativo 14 settembre 2015, n. 15, essa viene applicata a:

- Persone affette da limitazioni fisiche, psichiche, sensoriali ed intellettive;
- Persone invalide con grado superiore al 33%;
- Persone affette da gravi disabilità sensoriali;
- Persone invalide di guerra.

Questo decreto, inoltre, sancisce il dovere dei datori di lavoro di preservare il posto di lavoro per coloro i quali hanno acquisito una disabilità durante il periodo lavorativo.

Secondo quanto emerge dal sito ufficiale del Parlamento europeo[4]:

" Le statistiche ufficiali sulla disabilità sono scarse e per lo più non disaggregate per tipo di disabilità, razza/origine etnica, orientamento sessuale, ecc., e i dati sul tipo di occupazione cui hanno accesso le persone con disabilità sono pressoché inesistenti, tuttavia sappiamo che solo il 50,6 % delle persone con disabilità ha un lavoro (il 48,3 % delle donne e il 53,3 % degli uomini), rispetto al 74,8 % delle persone senza disabilità [Statistiche dell'Unione europea sul reddito e sulle condizioni di vita nell'UE (EU-SILC 2017). In base alle più recenti statistiche disponibili, soltanto il 20,7 % delle donne con disabilità ha un'occupazione a tempo pieno, rispetto al 28,6 % degli uomini con disabilità (Indice sull'uguaglianza di genere 2019). Tuttavia, queste cifre non indicano quante persone sono occupate nel mercato del lavoro aperto ed escludono le persone con disabilità che vivono in istituti di cura e che hanno molte meno probabilità di avere un lavoro o di essere incluse nella comunità in qualsiasi modo possibile. (K. Langensiepen, 2021) "

[4] https://www.europarl.europa.eu/doceo/document/A-9-2021-0014_IT.html#_ftn2
Relazione su l' *Applicazione della direttiva 2000/78/CE del Consiglio che stabilisce un quadro generale per la parità di trattamento in materia di occupazione e di condizioni di lavoro alla luce della UNCRPD.*

Per tali ragioni, la presenza degli strumenti normativi sopra elencati, non risulta sufficiente a ricoprire la totalità dei casi di soggetti socialmente esclusi, né a garantire una totale inclusione sociale. Un esempio è rappresentato dai cosiddetti NEET, *Neither in Employment or in Education or Training*, persone, generalmente di fascia compresa tra i 16 e i 35 anni, non impiegate nell'istruzione o nella formazione, né nella ricerca o lo svolgimento di un impiego.

Ciò, dunque, conduce alla necessità di dover integrare nuove metodologie di inclusione, volte a proteggere e garantire pari opportunità a coloro ancora oggi socialmente esclusi.

2.3.3. Monete complementari

Le monete complementari sono uno strumento di scambio che si affianca alle valute ufficiali, senza sostituirle. Esse possono essere utilizzate per ottenere beni e servizi, solitamente all'interno di una comunità o un territorio circoscritto. Poiché non vengono emesse dallo Stato, il loro utilizzo viene garantito da coloro i quali le accettano come valuta di scambio, e che, dunque, vi ripongono fiducia. Per tali ragioni, il legame che vi è tra la moneta complementare e la comunità è un legame concreto e simbolico. Ciò rende le monete complementari anche delle monete solidali, in quanto esse rafforzano il legame sociale tra i membri di una stessa comunità attraverso un sistema di scambio.

Ad oggi nel mondo esistono diversi sistemi di valuta complementare: alcuni si basano sul tempo, attribuendo un valore alle ore di lavoro dei partecipanti al circuito, altri funzionano come sistemi di credito reciproco, altri ancora sono coperti da un riferimento esterno, sia esso un bene o un servizio. Tuttavia, indipendentemente dal tipo di sistema, la loro caratteristica principale è la capacità di proporre un sistema di scambio alternativo alla valuta reale, necessario nei casi di mancata disponibilità economica.

Secondo quanto dimostrato da realtà quali *Arcipelago SCEC*[5] e *SardexPay*[6], l'utilizzo delle monete complementari ha contribuito alla crescita umana ed economica di specifici contesti. Ciò ha permesso di pensare a queste come uno strumento utile all'inclusione sociale. In particolare, utilizzandole come meri mezzi di scambio e, dunque, eliminando la possibilità di accumulazione e pressione economica, è possibile ottenere un sistema di scambio equo e solidale, in cui viene garantito pari accesso e pari opportunità a tutti i membri della comunità che ne fa uso. Uno dei progetti che nasce secondo tale pensiero è il progetto *Banca del Noi*.

[5] Arcipelago SCEC sito ufficiale: http://www.arcipelagoscec.net/come-funziona-lo-scec/
[6] SartexPay sito ufficiale: https://www.sardexpay.net/

3. Progetto Banca del Noi

3.1.Finalità

Il progetto *Banca del noi* nasce al fine di realizzare un sistema di scambio di beni e servizi messi a disposizione all'interno di una comunità. La comunità *Banca del Noi* è una realtà associativa che si sviluppa prioritariamente in modalità virtuale. L'adesione avviene mediante una procedura di iscrizione che prevede la registrazione con dati accertati, utilizzando la *user-account-activation,* e il versamento di una quota associativa monetaria minima. Pertanto, tale sistema si basa sull'utilizzo di una moneta complementare in un mercato virtuale circoscritto, che funziona in affiancamento al mercato della moneta reale. Tuttavia, rispetto a quest'ultimo, il mercato di *Banca del Noi* propone un nuovo modello di società, maggiormente inclusivo, capace di permettere a tutti di contribuire al benessere della società. Il progetto, infatti, mira a dare un'opportunità, in primo luogo, alle persone che vivono una situazione di fragilità, circostanza che colpisce parte della popolazione, in particolare le categorie di persone con disabilità definitiva, temporanea o contingente, ed ha l'obiettivo di fornir loro stabilità e sostentamento.

La forma di economia proposta da *Banca del Noi* è ispirata ai sistemi di scambio di prestazioni basati sullo scambio reciproco di tempo, utilizzando, però, una moneta complementare come unità per compiere le transazioni. Questa prende il nome di *ScaTTO*, acronimo di "scambia con tatto", e non ha alcun corrispettivo in valuta reale, dunque può essere ceduta in cambio di beni e servizi esclusivamente all'interno della piattaforma. Obiettivo concreto della *Banca del Noi* è, quindi, la creazione di un ciclo virtuoso capace di far circolare talenti e *ScaTTO*.

La *Banca del Noi*, inoltre, utilizza alcune regole semplici ed efficaci per prevenire qualunque forma di prevaricazione, accumulo e controllo economico. Attraverso di esse, il progetto auspica la creazione di uno strumento capace di restituire dignità a quella parte della società che, per motivi strutturali o contingenti, non riesce a inserirsi, o reinserirsi, all'interno del mercato tradizionale.

3.2.Utenti

Le categorie di utenti previsti all'interno di *Banca del Noi* sono:

- Non registrato: ha la possibilità di visualizzazione alcune informazioni generali della piattaforma, tra cui quelle riguardanti il suo funzionamento;
- Registrato: ha la possibilità di navigare all'interno della piattaforma, visualizzare beni e servizi offerti e, inoltre, ricevere notizie riguardo eventuali aggiornamenti;
- Associato: ha accesso a tutte le funzionalità della piattaforma, tra cui la possibilità di poter effettuare scambi o di offrire un nuovo bene o servizio alla comunità, contattare o aggiungere *feedback* agli utenti.

Risulta importante sottolineare che la piattaforma vincola tutti gli utenti alla sottoscrizione di un codice di comportamento in fase di iscrizione. Inoltre, parte della quota associativa viene investita in una assicurazione RC, al fine di coprire eventuali danni e garantire un corretto uso della piattaforma.

3.3.Beni e servizi

Essendo beni e servizi elementi fondamentali all'interno della piattaforma, poiché merci di scambio, occorre effettuare delle precisazioni al riguardo.

Viene definito bene, infatti, qualunque elemento che presenta il requisito di materialità, perciò risulta cedibile o noleggiabile (es. astuccio, piatto, computer, ecc.). Viene definito, invece, come servizio, qualunque tipologia di prestazione che può essere svolta in presenza o da remoto (es. baby-sitting, consulenza medica, ecc.).

Ogni bene o servizio è associato a un codice di categoria. Ogni categoria presenta un intervallo di valore prestabilito entro cui l'offerente può scegliere il valore del bene o servizio offerto; tale intervallo di valori viene assegnato secondo l'importanza sociale e non in base al prestigio sociale.

3.4.Sviluppi

Tra i possibili sviluppi futuri, il progetto *Banca del Noi* prevede, in prima istanza, la definizione di una certificazione interna per beni e servizi, utile come forma di garanzia della qualità di ciò che viene offerto all'interno della piattaforma e, in particolare per quanto riguarda i servizi, valida anche come certificazione spendibile nel CV.

Un'ulteriore previsione riguarda l'apertura di sedi fisiche situate in punti strategici, volte a consentire una fruizione non solo virtuale dei servizi offerti, puntando progressivamente alla copertura dell'intero territorio nazionale.

Infine, per garantire trasparenza e affidabilità agli scambi all'interno della piattaforma, il progetto prevede l'utilizzo della tecnologia blockchain, elemento centrale del prototipo da me realizzato, il cui sviluppo è trattato nei capitoli successivi.

4. Tecnologie utilizzate

Il presente capitolo mira ad illustrare, e descrivere brevemente, l'insieme delle tecnologie utilizzate per la realizzazione del prototipo del progetto *Banca del Noi*. L'ordine di presentazione delle tecnologie proposte ricalca quello di utilizzo, nonché la loro importanza all'interno del processo di sviluppo.

4.1.Node.js

Node.js è la prima tecnologia utilizzata per la realizzazione del prototipo per il progetto *Banca del Noi*, poiché rappresenta un prerequisito fondamentale per il funzionamento dell'intera piattaforma. Node.js è, infatti, un *runtime system open source*, ossia un *software* che fornisce servizi necessari all'esecuzione di programmi, pur non facendo parte nativamente del sistema operativo. Tale *software* ha un'architettura orientata agli eventi, dove l'esecuzione delle istruzioni dipende dal verificarsi di determinati metodi, ed è utilizzato per l'esecuzione di codice Javascript lato server. Importante elemento presente all'interno di Node.js è il gestore di pacchetti denominato npm, *Node Package Manager*, formato da un *client* accessibile da linea di comando e un *database* online di pacchetti[7]. L'utilizzo di quest'ultimo, in particolar modo, è stato utile per l'installazione e l'esecuzione di alcune tecnologie illustrate di seguito, necessarie per lo sviluppo dell'intera piattaforma.

4.2.Ethereum

Prima di poter illustrare la prossima tecnologia utilizzata, è necessario descrivere cosa è *Ethereum*. *Ethereum* è una piattaforma *blockchain* che permette la creazione e la pubblicazione di contratti intelligenti, *smart contract*, creati attraverso linguaggio di programmazione. Gli *smart contract*, sono protocolli informatici con cui è possibile controllare l'esecuzione di clausole all'interno della rete. Per poter essere eseguiti, è necessario il pagamento dell'utilizzo della potenza computazionale della *blockchain*, tramite una criptovaluta, in questo caso *Ether*

[7] Per maggiori approfondimenti consultare la documentazione al sito: https://nodejs.dev/learn

(ETH). Diversamente da quanto avviene all'interno di altre reti, dunque, la possibilità di esecuzione di contratti intelligenti permette il controllo e lo sviluppo di nuove operazioni possibili all'interno della rete. La *blockchain* di *Ethereum*, inoltre, viene definita come una macchina a stati, *state machine*, rispetto alla canonica definizione di registro distribuito.

Il motivo risiede nella capacità, da parte di *Ethereum*, di memorizzare una grande struttura dati, contenete non solo tutti i conti ed i saldi delle transazioni, ma anche uno stato della macchina che può mutare da blocco a blocco, insieme a specifiche regole di esecuzione. Tale cambiamento di stato è regolamentato dalla EVM, *Ethereum Virtual Machine*[8].

Ethereum, infine, è un progetto *open source*, perciò ogni sviluppatore può contribuire ed utilizzare il codice sorgente. Da qui, molti sono stati i progetti sviluppati basandosi su questa tecnologia, tra questi vi è *Hyperledger Besu*.

4.2.1. Hyperledger Besu

Hyperledger Besu è un *client* basato su Ethereum, *open source*, scritto in linguaggio Java che consente l'esecuzione di un nodo sulla rete. Esso implementa la specifica EEA, *Enterprise Ethereum Alliance*, la quale, fondata nel 2017, è stata stabilita per garantire lo sviluppo di interfacce comuni tra i vari progetti *open* e *closed source* all'interno di *Ethereum*, inoltre, essa svincola gli utenti da obblighi verso il fornitore. Oltre tale specifica, *Hyperledger Besu* presenta una serie di caratteristiche[9] peculiari, di cui di seguito viene riportato un elenco:

- Algoritmi di consenso: la presenza di diverse tipologie di algoritmi coinvolti nella convalida delle transazioni, nella validazione dei blocchi e nella produzione dei blocchi (es. *Proof of Work, Proof of Stake* ecc.).
- Archiviazione: implementazione di un database di chiave-valore per mantenere i dati della catena localmente. Tali dati sono suddivisi in:

[8] Per maggiori approfondimenti consultare la documentazione al sito: https://ethereum.org/it/developers/docs/evm/
[9] Per maggiori approfondimenti consultare la documentazione al sito: https://wiki.hyperledger.org/display/BESU/Hyperledger+Besu

    - *Blockchain*: composti da: intestazioni di blocco, utilizzate per verificare crittograficamente lo stato della blockchain, i corpi di blocco, che contengono l'elenco delle transazioni ordinate comprese in ciascun blocco, e le ricevute di transazione, che contengono metadati relativi all'esecuzione della transazione, inclusi i registri delle transazioni.
    - Stato: ogni intestazione di blocco fa riferimento a uno stato globale, ossia una mappatura dagli indirizzi degli account. Gli account di proprietà esterna contengono un saldo *Ether*, mentre gli account *smart contract* contengono anche codice eseguibile e spazio di archiviazione.

- Monitoraggio: possibilità di monitorare le prestazioni del nodo e della rete attraverso appositi servizi.
- Privacy: le transazioni vengono mantenute private, nessuno può accedere al contenuto della transazione, al mittente o all'elenco delle parti partecipanti. Tale sistema è implementato attraverso un gestore di transazioni denominato *Private Transaction Manager*.
- Autorizzazione: possibilità di creare una rete autorizzata, o privata, in cui la partecipazione di ogni nodo dipende dalla sua abilitazione e/o dall'autorizzazione dell'account sulla rete da parte dei suoi gestori.

Le caratteristiche sopra elencate, insieme con la qualità di un codice open source e la possibilità di accedere ad una documentazione dettagliata, hanno rappresento i principali elementi di confronto con servizi alternativi.

A seguito di un'attenta analisi, pur ammettendo la presenza di qualità uniche all'interno dei diversi servizi, *Hyperledger Besu* è risultato maggiormente in linea con gli obiettivi e le specifiche proposte da *Banca del Noi*.

4.3. Truffle

Truffle è uno strumento, con codice *open source*, che racchiude diverse risorse. Queste, sono utili per l'interazione con tutte le *blockchain* che utilizzano *Ethereum Virtual Machine*, poiché permettono di eseguire operazioni sugli *smart contract*. Alcune operazioni[10] riguardano:

- Compilazione dei contratti: avviene mediante apposito comando e genera gli artefatti, ossia file scritti in JSON che riflettono quanto definito nei contratti, necessari al corretto funzionamento interno di *Truffle*.
- Test dei contratti: avviene attraverso l'utilizzo di un framework di test incluso e consente di effettuare due tipologie di test:
    - In *JavaScript* o *TypeScript*, per testare il contratto dall'esterno;
    - In *Solidity*, per testare il contratto in scenari avanzati.
- Distribuzione dei contratti: avviene mediante specifico comando, con la possibilità di specificare su quale rete far avvenire la distribuzione. Questa operazione dipende dalla corretta impostazione degli artefatti e delle migrazioni. Le migrazioni sono file JavaScript responsabili della gestione temporanea delle attività di distribuzione e vengono scritti in modo che le esigenze di distribuzione possano variare.

Oltre all'esecuzione delle operazioni sopra elencate, *Truffle* mette a disposizione due *console* interattive, utili per interagire direttamente con i contratti intelligenti. La prima *console*, denominata *Truffle Console*, rappresenta una console di base capace di connettersi ad ogni tipologia di client *Ethereum*. Essa viene utilizzata principalmente nei casi in cui sorge la necessità di connettersi ad un indirizzo specifico, o di migrare il proprio progetto da una rete di test ad una ufficiale. La seconda *console* invece, denominata *Truffle Develop*, rappresenta una *console* più complessa, che genera una blockchain di sviluppo e viene utilizzata principalmente nei casi in cui non vi è la necessita di inserire indirizzi specifici e di gestire un client blockchain separato. Il motivo principale del suo utilizzo nella realizzazione del progetto *Banca del Noi* risiede negli strumenti che esso offre, utili a facilitare le operazioni di test e sviluppo di contratti intelligenti.

[10] Per maggiori approfondimenti consultare la documentazione al sito: https://trufflesuite.com/docs/truffle/

4.4.Web3

Web3 è una libreria *JavaScript* che consente di interagire con la *blockchain* di *Ethereum*. In *Ethereum*, infatti, i nodi forniscono interfacce di basso livello per consentire agli utenti di inviare transazioni. Queste possono essere trasmesse tramite interfacce di tipo JSON RPC, dove JSON RPC rappresenta un formato di codifica testuale che consente ai processi in esecuzione di ricevere dati[11].

Web3, in tal senso, consente di comunicare con le interfacce JSON RPC di *Ethereum* attraverso il linguaggio *JavaScript*, il che lo rende direttamente utilizzabile nella tecnologia *web*, poiché supportato in modo nativo in tutti i *browser web*. Esso, inoltre, è anche comunemente utilizzato lato *server*, nelle applicazioni *Node.js*.

Web3 può essere utilizzato per connettersi alla rete *Ethereum* tramite qualsiasi nodo *Ethereum* che consente l'accesso tramite HTTP, *HyperText Transfer Protocol*, e può essere un nodo locale, un nodo ospitato da un provider o un *gateway* pubblico, che gestisce punti di accesso *Ethereum* gratuiti.

Per poter utilizzare *Web3*, è necessario ottenere una copia della libreria. Il metodo più comune per ottenerla è tramite il gestore di pacchetti npm. La connessione al nodo *Ethereum* avviene specificando un *provider*, mentre l'interazione con lo *smart contract* distribuito si verifica specificando l'indirizzo del contratto e l'ABI, *Application Binary Interface*, ovvero una descrizione dell'interfaccia pubblica del contratto, sotto forma di oggetto JSON. L'interazione di *Web3.js* è asincrona.

L'utilizzo di questa libreria, all'interno del prototipo del progetto *Banca del Noi*, è risultato fondamentale ai fini dello sviluppo della piattaforma *web based* e della comunicazione tra questa e la *blockchain*.

[11] Per maggiori approfondimenti consultare la documentazione al sito: https://www.json.org/json-it.html

4.5.XAMPP

La necessità di un database centrale, nata per nascondere determinate informazioni altrimenti pubbliche all'interno della *blockchain*, ha portato all'utilizzo di XAMPP.

XAMPP è un software libero, caratterizzato da un approccio incentrato sulla facilità di utilizzo, con il quale è possibile realizzare un *application server*, ovvero una tipologia di *server* che fornisce l'infrastruttura e le funzionalità logiche di un *server* ma in un contesto distribuito, utili per lo sviluppo *web* dinamico. I principali componenti di questo software sono: *Apache HTTP Server* e il database *MariaDB*.

Il primo componente, *Apache HTTP Server* o più comunemente *Apache*, è un *server web*, ossia un *software* che permette di gestire le richieste di trasferimento di pagine *web* di un *client*, tramite protocollo HTTP o HTTPS, utilizzando porte specifiche.

Il database *MariaDB*, invece, è un DBMS o *database server*, ossia un software progettato per consentire la creazione, manipolazione e l'interrogazione di un *database*, ospitato su un'architettura hardware dedicata.

Oltre queste due componenti principali, XAMPP, implementa diversi strumenti utili e necessari per l'utilizzo dei linguaggi di programmazione PHP e Perl.

Alla luce di quanto descritto, l'impiego di XAMPP, all'interno del processo di realizzazione del prototipo di *Banca del Noi*, è risultato necessario al fine della creazione, manipolazione e interrogazione di un database locale, gestito tramite richieste HTTP generate dalla piattaforma web.

4.6.jQuery

jQuery è una libreria *JavaScript* per applicazioni *web*, *open source*. Le sue funzionalità permettono: la selezione e manipolazione degli elementi presenti nel DOM, *Document Object Model*, di una pagina *web*, la gestione degli eventi, nonché la semplificazione dell'uso di AJAX, *Asynchronous JavaScript and XML*.

Per ottenere l'accesso agli elementi, la libreria utilizza i selettori, ossia strumenti che sfruttano la medesima sintassi del CSS, *Cascading Style Sheet*, ovvero:

- ID di un elemento;
- Classe, con possibilità di selezione di uno o più elementi appartenenti ad essa;
- Parole chiave per selezione gerarchica (ad es. `.sibling`, `.prev`, ecc.);
- Pseudo classi (ad es. `:first`, `:last`, ecc.);
- Attributi o contenuti (ad es. `:contain`, `:has`, `[type="text"]`, ecc.)

Inoltre, *jQuery* propone numerosi metodi e funzioni utili alla navigazione del DOM, tra cui creazione, rimozione e modifica di elementi alla pagina, comprendendone il controllo dello stile[12]. Per quanto riguarda la gestione degli eventi, invece, la libreria riconosce gli oggetti di tipo *event* e provvede a modificare le loro proprietà rendendoli uniformi e, dunque, semplificandone la progettazione.

Ai fini della realizzazione del prototipo di *Banca del Noi*, ciò che rende interessante *jQuery* è la gestione delle chiamate asincrone AJAX, eseguite tramite funzioni specifiche ed utili allo scambio di dati in *background* fra *web browser* e *server*, consentendo l'aggiornamento dinamico di una pagina *web* senza esplicito ricaricamento da parte dell'utente. Tali funzioni, dunque, hanno permesso l'interazione dinamica con il *database* locale realizzato tramite lo strumento, precedentemente descritto, XAMPP.

[12] Per maggiori approfondimenti consultare la documentazione al sito: https://api.jquery.com/

4.7.Crypto.js

Cripto.js è una raccolta di algoritmi crittografici implementati in linguaggio *JavaScript*, utilizzati per garantire la confidenzialità e la riservatezza all'interno di un sistema di comunicazione o di scambio di dati. Gli algoritmi presenti in *Crypto.js* si suddividono in due categorie: *hashing* e cifratura.

La prima categoria di algoritmi permette di mappare una stringa di qualsiasi dimensione, in modo da ottenere una stringa di *bit* di dimensione fissa. Tale meccanismo è irreversibile e, inoltre, non è possibile creare un *input* che corrisponda ad un *output* specifico. Per queste ragioni, i casi in cui questa categoria viene utilizzata riguardano principalmente la verifica dell'integrità dei *file* e l'autenticazione all'interno di applicazioni e protocolli di sicurezza.

La seconda categoria di algoritmi, invece, permette di trasformare una stringa di qualsiasi dimensione, in una stringa di *bit* a dimensione fissa, mediante l'utilizzo di una chiave. Quest'ultima è fondamentale all'interno del processo, in quanto decide della lunghezza della stringa in *output*, e consente di risalire alla stringa originaria, mediante procedura inversa. Ad oggi, secondo quanto emanato dal GDPR[13], *Regolamento generale sulla protezione dei dati*, i casi di utilizzo di questa categoria si estendono a qualsiasi tipologia di dati circolanti sul *web*.

In entrambe le categorie presenti all'interno di *Crypto.js*, gli algoritmi in esse racchiuse sono organizzati in base ad un livello di sicurezza crescente[14].

L'utilizzo di questa tecnologia, all'interno del prototipo del progetto *Banca del Noi,* è risultato fondamentale ai fini dell'archiviazione di dati sensibili all'interno del *database* centralizzato. Ciò, infatti, è stato utile a garantire sicurezza e riservatezza ai dati archiviati.

[13] Testo integrale del regolamento del Parlamento europeo al sito: https://eur-lex.europa.eu/legal-content/IT/TXT/HTML/?uri=CELEX:32016R0679

[14] Per maggiori approfondimenti consultare la documentazione al sito: https://cryptojs.gitbook.io/docs/

4.8.Figma

Figma è uno strumento di progettazione di interfacce utente con generazione di codice, che viene impiegato per la realizzazione di *mockup* ad alta fedeltà, aventi capacità interattive. Esso è basato principalmente sul *web*, al fine di garantire una compatibilità con qualsiasi sistema operativo, tuttavia, è possibile utilizzarne anche una versione *desktop*, disponibile esclusivamente per i sistemi *Windows* e *Mac*. Una delle caratteristiche degne di nota di *Figma* riguarda la gestione dei progetti. Essi, infatti, vengono salvati all'interno di uno spazio personale *online* con la possibilità condividerli con altri utenti. Qualora un progetto venisse condiviso, esso può essere modificato, in tempo reale, da tutti gli utenti.

Oltre agli strumenti tipici presenti all'interno di qualunque *software* di progettazione, *Figma* presenta alcune caratteristiche distintive, tra cui:

- Creazione di librerie: definito il gruppo di utenti che ha accesso al progetto, è possibile creare componenti riutilizzabili condivisi all'interno del team;
- Vector Networks: costruzione di forme complesse, dette reti vettoriali, ottenute collegando più segmenti di un nodo in un vettore;
- Commenti: inclusione di commenti al progetto posizionati direttamente sulla tela di sviluppo;
- Generatore di codice: dato un progetto è possibile ottenerne il relativo codice nei formati CSS, SVG, iOS o Android.

L'elenco sopra illustrato rappresenta solo un'esemplificazione delle diverse caratteristiche uniche che distinguono *Figma* da altri strumenti di progettazione[15]. L'utilizzo di questo strumento, all'interno della fase di progettazione grafica del prototipo di *Banca del Noi*, è risultato particolarmente efficacie ai fini di una presentazione più completa ed interattiva con il committente e i responsabili del progetto.

[15] Per maggiori approfondimenti consultare la documentazione al sito: https://help.figma.com/hc/en-us

4.9. GitHub

GitHub è un servizio di *hosting* per progetti *software*, il cui nome deriva dall'implementazione del servizio *Git*, con il quale è possibile creare sistemi di archiviazione di progetti, suddivisi per versione, denominati *repository*.

Ciò che, tuttavia, rende *Git* particolarmente vantaggioso, è che si tratta di un sistema di controllo distribuito, in cui laddove vengono apportate modifiche ad un progetto, l'intero *repository* viene clonato sul sistema, favorendone la condivisione tra più utenti. Questi ultimi, oltre alla possibilità di modifica, hanno a disposizione diversi strumenti per poter interagire con lo sviluppatore, come un sistema di *issue tracking*, *pull request* e commenti, utili a migliorare, e/o ampliare, il codice e a favorire il lavoro condiviso.

L'utilizzo di GitHub può avvenire secondo diverse modalità. Per ciò che concerne il prototipo di *Banca del Noi*, l'utilizzo di questa tecnologia ha riguardato la condivisione del codice sorgente all'interno della piattaforma *web* proprietaria, sotto forma di *repository* privato, e alla modifica di quest'ultimo tramite l'estensione *GitHub Repositories*, presente all'interno dell'editor *Visual Studio Code*.

Il motivo di tale scelta risiede, prevalentemente, nella volontà comune, del sottoscritto e della committenza, di mantenere riservate le informazioni e il materiale prodotto, al fine di salvaguardare la proprietà intellettuale delle singole parti coinvolte.

5. Realizzazione del prototipo

All'interno del seguente capitolo verrà illustrata l'intera procedura che ha portato alla realizzazione del prototipo per il progetto *Banca del Noi*.

5.1. Installazione di Node.js e Truffle

La fase iniziale della realizzazione del prototipo ha avuto come obiettivo l'installazione e la configurazione delle tecnologie necessarie a predisporre il sistema a sviluppare il progetto. La prima di queste è stata *Node.js.* Per poterne avviare la procedura di installazione è stato necessario collegarsi al sito ufficiale del *software* e scaricare l'ultima versione disponibile del sistema operativo in uso. Trattandosi di un sistema *Windows*, all'interno di tale procedura, è stato necessario selezionare l'opzione riguardante l'installazione di strumenti essenziali (vedi fig.5), tra cui *Chocolately*. Tale strumento, infatti, rappresenta un gestore di pacchetti da riga di comando specifico per il sistema *Windows*, utile per semplificare i processi di installazione e distribuzione di *software*.

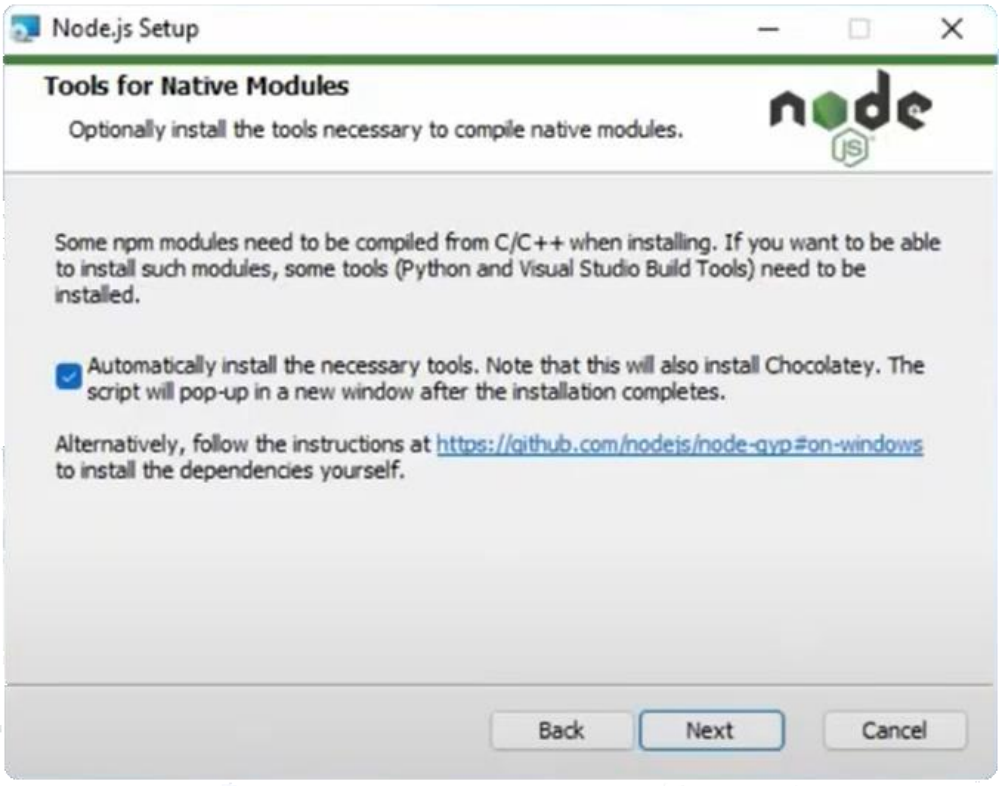

Figura 5. Opzione di installazione di strumenti essenziali.

Al termine della procedura di installazione, ne è stata verificata la corretta esecuzione tramite il comando

```
node -v
```

all'interno del terminale, il quale produce come risultato il numero di versione presente nel sistema (vedi fig. 6).

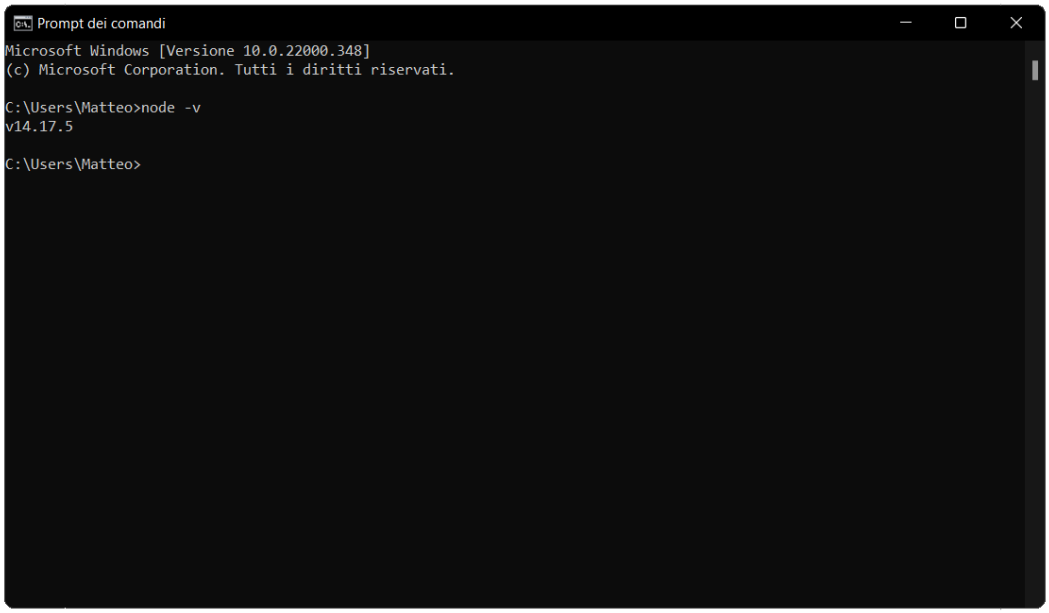

Figura 6. Esecuzione comando: `node -v`.

Terminata e verificata la prima installazione, è stato possibile procedere con la seconda tecnologia, ossia il client *Ethereum Hyperledger Besu*. Per poterla ottenere sul sistema operativo *Windows* è stato necessario utilizzare la piattaforma *GitHub*. Dal *repository* ufficiale di *Hyperledger*, infatti, è stato possibile scaricare il *file binary* del client. Una volta ottenuto, per poterlo eseguire, è stato necessario spostare l'intera cartella all'interno del percorso

C:\Program Files

e, successivamente, inserire il *path* della cartella *binary* all'interno delle variabili d'ambiente del sistema (vedi fig. 7).

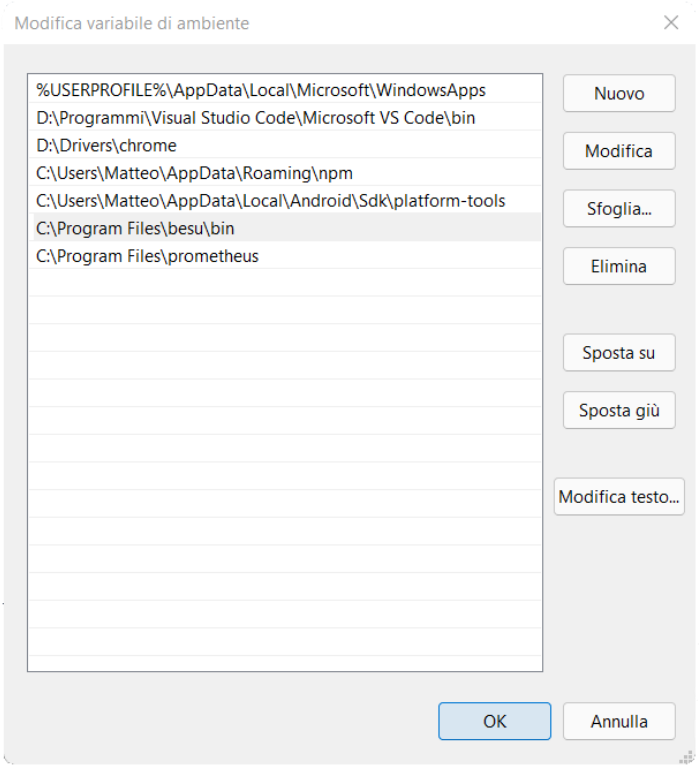

Figura 7. Path cartella binary di Hyperledger Besu tra le variabili d'ambiente.

La verifica della corretta installazione di questa tecnologia è stata svolta eseguendo il comando

```
besu --help
```

il quale produce come risultato una lista di comandi eseguibili sul client installato (vedi fig.8).

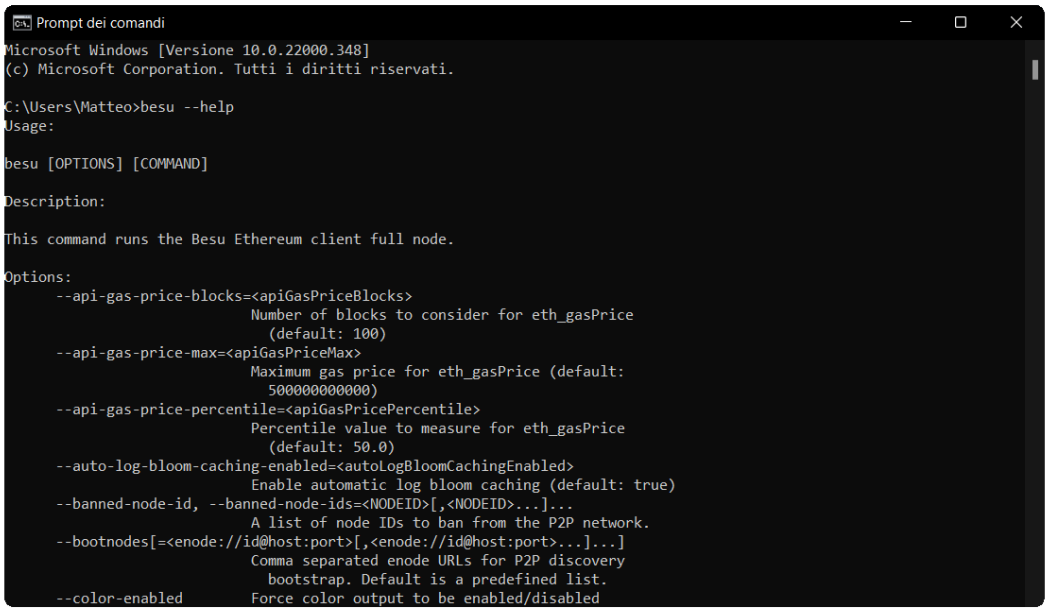

Figura 8. Esecuzione comando: `besu --help`.

L'ultima tecnologia utile a predisporre il sistema alla realizzazione del prototipo è stata *Truffle*. Per poterla ottenere, è stato utilizzato il gestore di pacchetti *npm*, incluso all'interno di *Node.js*, ossia mediante l'esecuzione all'interno del terminale del comando

```
npm install -g truffle
```

Con la conclusione di tale esecuzione si chiude il processo di predisposizione del sistema, permettendo così di procedere alla configurazione e all'inizializzazione del prototipo per *Banca del Noi*.

5.2.Inizializzazione del progetto

Il primo passo di questa fase ha riguardato l'esecuzione del comando

```
truffle init
```

il quale genera, all'interno di una *directory* specifica, una struttura composta da:

- contracts/: *directory* per i contratti intelligenti;
- migrations/: *directory* per i *file* di distribuzione;
- test/: *directory* per i *file* utili a testare applicazioni e contratti;
- truffle-config.js: *file* di configurazione di *Truffle*.

Questi rappresentano i componenti di un progetto *Truffle*, utili per lo sviluppo e l'interazione di contratti intelligenti. A questo punto, prima di poter proseguire, è risultato necessario decidere il tipo di nodo e, dunque, la rete *blockchain* su cui sviluppare il prototipo. Poiché trattasi di una versione non definitiva, si è scelto di utilizzare una rete di test locale, eseguibile mediante il comando

```
besu --network=dev --miner-enabled --miner-
coinbase=0xfe3b557e8fb62b89f4916b721be55ceb828dbd73
--rpc-http-cors-origins="all" --host-allowlist="*" -
-rpc-ws-enabled --rpc-http-enabled --data-
path=/tmp/tmpDatdir
```

presente all'interno della documentazione ufficiale di *Hyperledger Besu*. Ogni componente del comando rappresenta una coppia chiave-valore che definisce una particolare specifica della rete, ossia:

- `--network = "dev"`: tipo di rete;
- `--miner = "enabled"`: presenza di un nodo miner;
- `--miner-coinbase = "0xfe3b.."`: indirizzo del miner;
- `--rpc-http-cors-origins="all"`: condivisione delle risorse;
- `--host-allowlist="*"`: accesso all'API JSON;
- `--rpc-ws-enabled= "true"`: attiavzione *Web Socket API*;
- `--rpc-http-enabled = "true"`: connessioni remote;
- `--data-path= "/tmp/tmpDatdir"`: *path* memorizzazione dati;

Proseguendo nel processo di inizializzazione del progetto, al fine di consentire l'esecuzione dei componenti di *Truffle* all'interno della *blockchain* istanziata da *Besu*, è stato necessario modificare la struttura di partenza del progetto. La prima modifica consiste nell'esecuzione, all'interno della *directory* del progetto, del comando

```
npm init -y
```

esso, ha generato un nuovo pacchetto *npm*, sottoforma di *file* JSON precompilato, al quale è stato aggiunto lo *script* di esecuzione della rete locale di test, rendendolo eseguibile mediante parola chiave. Di seguito viene illustrato il codice del *file*, denominato package.json.

```
{    "name": "test",
     "version": "1.0.0",
     "description": "",
     "main": "truffle-config.js",
     "directories": {
          "test": "test" },
     "scripts": {
          "besu": "besu --network=dev --miner-
          enabled --miner-
          coinbase=0xFE3B557E8Fb62b89F4916B721be55cE
          b828dBd73 --rpc-http-cors-origins=* --
          host-allowlist=* --rpc-ws-enabled --rpc-
          http-enabled --data-path=/tmp/tmpDatdir"},
     "keywords": [],
     "author": "",
     "license": "ISC",
     "dependencies": {
          "@truffle/hdwallet-provider": "^1.5.0",
          "web3": "^1.6.0",
          "web3js-quorum": "^21.7.0-rc1"}
}
```

La seconda, ed ultima, modifica al progetto ha riguardato il *file* truffle-config.js, un *file JavaScript* con il quale è possibile creare una configurazione specifica di *Truffle*. La modifica del *file*, infatti, ha permesso di configurare *Truffle* in modo da connettersi al nodo *Besu* precedentemente definito. In particolare, è stato specificato un nuovo *provider*, avente, come indirizzo sulla rete, l'*url* del nodo *Besu* e, come *array* di chiavi private, gli indirizzi delle chiavi di test predefinite, presenti all'interno della documentazione ufficiale[16]. Di seguito viene illustrato il codice del *file*.

```
const PrivateKeyProvider =
require('@truffle/hdwallet-provider');


const privateKeys =
['0x8f2a55949038a9610f50fb23b5883af3b4ecb3c3bb792cbc
efbd1542c692be63',
'0xc87509a1c067bbde78beb793e6fa76530b6382a4c0241e5e4
a9ec0a0f44dc0d3',
'0xae6ae8e5ccbfb04590405997ee2d52d2b330726137b875053
c36d94e974d162f',
'0x0fe7cc53f6ed1f80973247165afc5e1fe0c681e3b33b8713a
1eb38169215e62a'];


const privateKeyprovider = new
PrivateKeyProvider(privateKeys,
'http://127.0.0.1:8545', 0, 3);


module.exports = {
networks: {
  besu: {
    network_id: "2018",
    provider: privateKeyprovider
      }
}
```

[16] In riferimento alla documentazione Hyperledger Besu presente al sito: https://besu.hyperledger.org/en/stable/Reference/Accounts-for-Testing/

La verifica della corretta esecuzione delle modifiche sopra illustrate è avvenuta, mediante l'utilizzo, all'interno della *directory* del progetto, del comando

```
npm run besu
```

il quale ha prodotto come risultato l'esecuzione dello *script* definito all'interno del *file* package.json e, di conseguenza, l'avvio del nodo *Besu* sulla rete locale (vedi fig. 9).

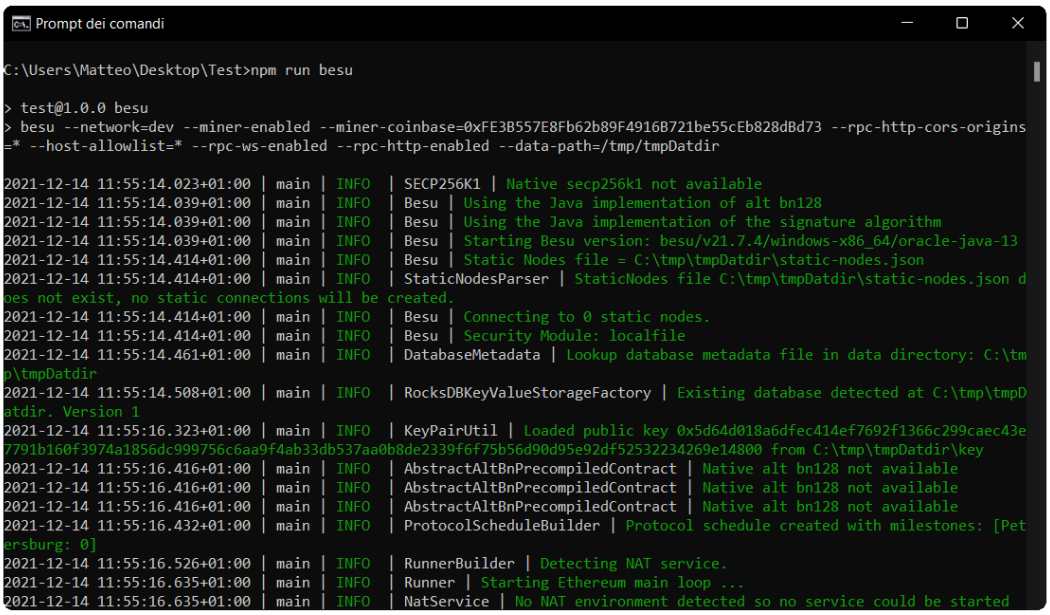

Figura 9. Esecuzione comando: `npm run besu`.

Mantenendo in esecuzione il nodo, all'interno di una nuova istanza del terminale, è stata poi verificata la corretta comunicazione con *Truffle*, attraverso l'utilizzo del comando

```
truffle console –network besu
```

il quale ha generato una nuova istanza della *console* di *Truffle* sulla rete locale (vedi fig. 10).

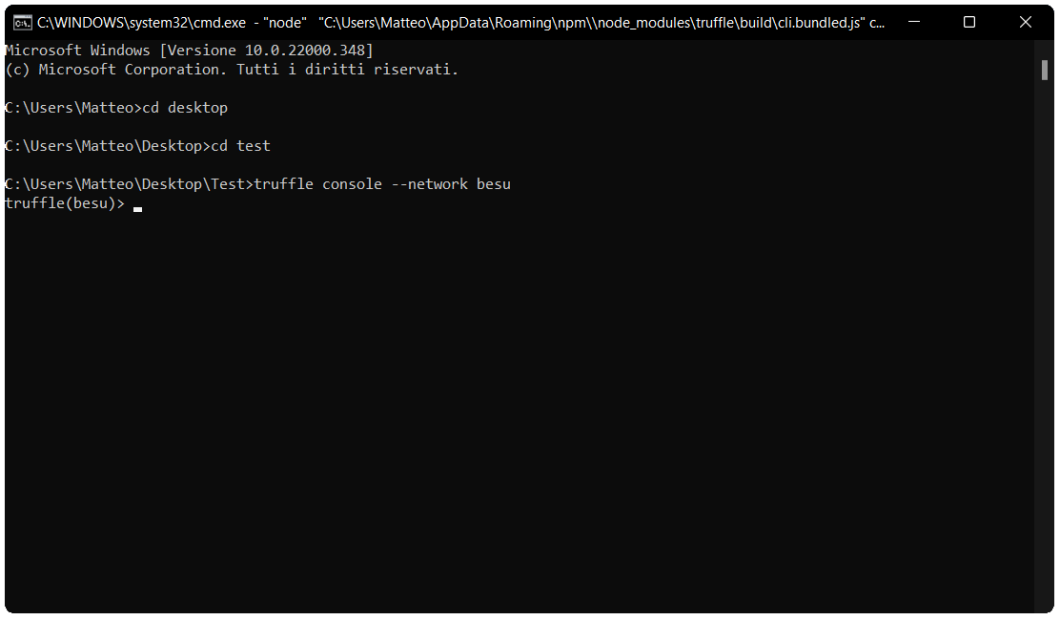

Figura 10. Esecuzione comando : `truffle console --network besu`.

All’interno di quest’ultima, infine, è stato poi utilizzato il comando

```
accounts
```

il quale ha mostrato la lista di indirizzi presenti sulla rete *Besu*, ossia i proprietari delle chiavi private inserite all’interno del *file* truffle-config.js, riscontrabili all’interno della documentazione ufficiale (vedi fig. 11).

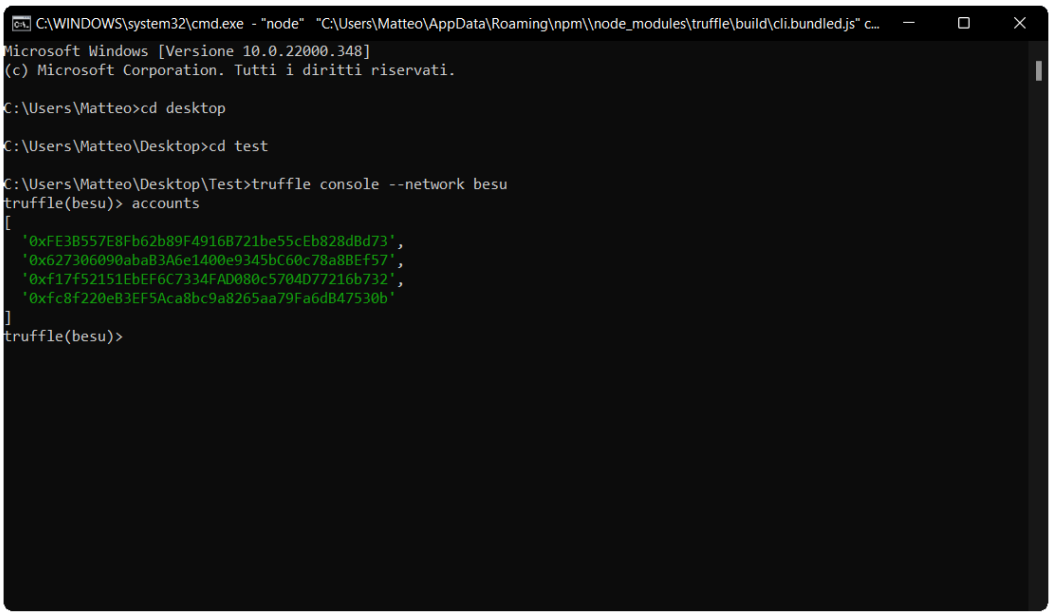

Figura 11. Esecuzione comando: `accounts`.

La conclusione del processo di verifica delle modifiche apportate, permette di terminare la fase riguardante l’inizializzazione del progetto. Ciò conduce, dunque, alla fase successiva della realizzazione del prototipo, ossia lo sviluppo del contratto intelligente.

5.3.Sviluppo del contratto intelligente

La fase di sviluppo del contratto intelligente per il prototipo di *Banca del Noi*, si è incentrata principalmente sulla scrittura e sulla verifica del codice del contratto. Quest'ultimo è stato scritto in *Solidity*, un linguaggio orientato agli oggetti, simile a *Javascript*, ma pensato per interagire con la *blockchain* di *Ethereum*. Al suo interno sono state inserite le funzioni e i dati da memorizzare sulla rete, in linea con gli obiettivi e le politiche di *Banca del Noi*. Per quanto riguarda i dati da memorizzare, infatti, all'interno del contratto sono state predisposte due principali strutture, l'una riguardante l'insieme delle informazioni di un account (es. nome, tipo, bilancio, ecc.), e l'altra quelle di un bene o servizio (es. nome, tipo, prezzo, ecc.). Successivamente, sono state utilizzate delle tabelle *hash*, con cui è stato possibile assegnare le strutture dati a specifiche chiavi, in modo da mantenere una corrispondenza univoca tra chiave e valore e permetterne la ricerca immediata. Di seguito vengono presentati due esempi dal codice.

```
struct Account {
      string name;
      string class;
      uint balance;
      address add;
      bool exists;
  }

mapping (string => Account) public nameToAcc;
```

Per quanto riguarda, invece, le funzioni presenti nel contratto, ne sono state predisposte due diverse tipologie. L'una eseguibile da qualsiasi utente associato, e l'altra eseguibile soltanto dal proprietario del contratto o da una specifica categoria di utenti. Ciò è stato possibile mediante l'utilizzo di un modificatore, un elemento che modifica il comportamento di una funzione. Di seguito viene presentato un esempio dal codice.

```
modifier onlySpecial {
      require(msg.sender == owner ||
      keccak256(abi.encodePacked(ipToAcc[msg.sender].
      class))==keccak256(abi.encodePacked("admin")));
      _;
  }
```

Le funzioni, quindi, eseguibili da tutti gli utenti riguardano:

- Creazione di un account;
- Creazione di un bene o servizio;
- Eliminazione di un bene o servizio;
- Scambio di bene o servizio con *ScaTTO*.

Di queste, inoltre, fanno parte anche le funzioni *getter*, ossia funzioni utilizzate per recuperare il valore di una variabile dall'esterno. Di seguito vengono presentati due esempi dal codice.

```
function createAccount(string memory _name, address
_address) public {
      require(!ipToAcc [_address].exists &&
      !nameToAcc[_name].exists, "Indirizzo già
      esistente.");
      uint i= accounts.push(Account(_name, "normal",
      0, _address, true)) -1;
      ipToAcc [_address]= accounts[i];
      nameToAcc[_name]= accounts[i];
      ipToItem [_address]= items;
      emit newAccount(_name, 0);
 }
```

```
function checkLen() public view returns (uint) {
      return accounts.length;
}
```

Le funzioni, invece, eseguibili solo dal proprietario del contratto o da una specifica categoria di utenti riguardano:

- Modifica del tipo di un account;
- Invio diretto di *ScaTTO*;
- Generazione di *ScaTTO*;
- Eliminazione di un account.

Di seguito viene presentato un esempio dal codice.

```
function setClassAccount(string memory _name, string
memory _class) public onlySpecial {
      require(nameToAcc[_name].exists, "Nome
      inesistente");
      nameToAcc[_name].class=_class;
      ipToAcc[nameToAcc[_name].add].class=_class;
      emit typeChanged(_name, _class);
 }
```

Conclusa la scrittura del codice, il contratto è stato salvato all'interno della *directory* contracts/ del progetto. Prima di procedere con la verifica del codice, è stato necessario creare un *file* di migrazione del contratto e salvarlo all'interno della *directory* migrations/ del progetto. Questo, rappresenta un *file Javascript* utile per la distribuzione del contratto all'interno della rete. Di seguito ne viene presentato il codice.

```
const BDN = artifacts.require("BDN");

module.exports = function(deployer) {
      deployer.deploy(BDN);
};
```

Creato e salvato il *file* di migrazione, successivamente, è stato distribuito all'interno della rete locale, mediante il comando

```
truffle migrate --reset network besu
```

Poiché la distribuzione del contratto è di per sé una transazione, ciò ha prodotto come risultato sia l'indirizzo del contratto distribuito che le informazioni della transazione stessa (vedi fig. 12). Inoltre, *Truffle*, automaticamente, ha creato all'interno del progetto una *directory* build/, contenente l'artefatto del contratto.

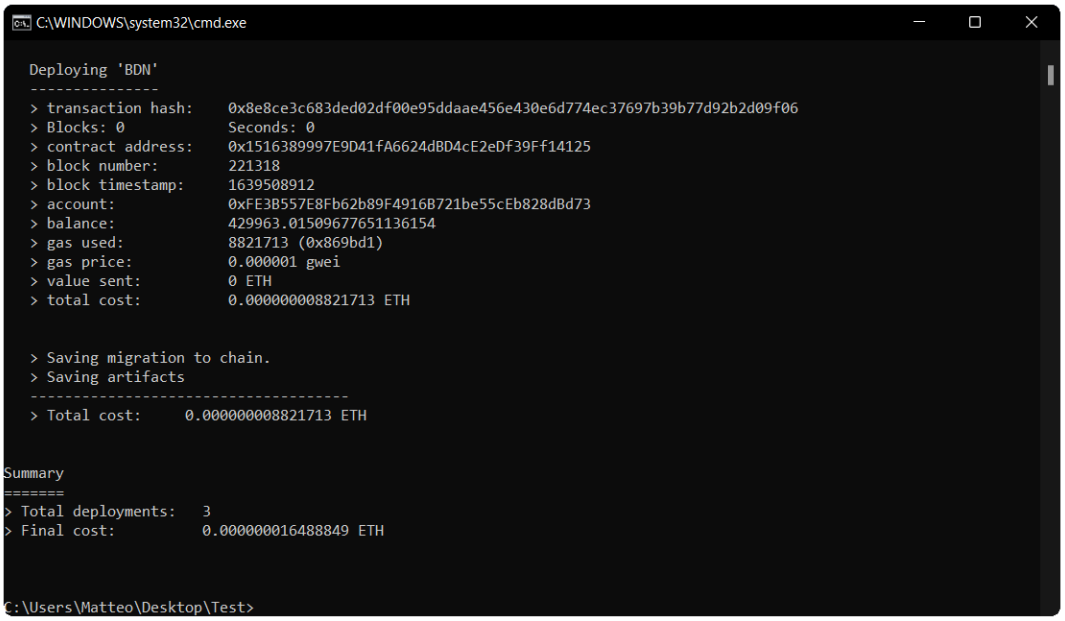

Figura 12. Risultato distribuzione contratto.

Conclusa la distribuzione del contratto, è stato possibile procedere alla verifica del codice, mediante la *console* di *Truffle*. All'interno di essa, infatti, è stato possibile interagire con il contratto e verificarne il corretto funzionamento interrogando e richiamando le funzioni in esso definite. La corretta esecuzione di quest'ultime ha permesso di concludere la fase di sviluppo del contratto e di procedere con l'implementazione di *Web3*.

5.4.Implementazione con Web3

Prima di poter procedere con l'implementazione vera e propria con *Web3* è stato necessario inserire la libreria all'interno del progetto. Per fare ciò, è stato utilizzato il comando

```
npm install web3
```

Come precedentemente descritto, *Web3* è una libreria che consente di interagire con la *blockchain* di *Ethereum* tramite linguaggio *JavaScript*, il che la rende direttamente utilizzabile all'interno della tecnologia *web*. Per tali ragioni, l'implementazione con *Web3* si è svolta di pari passo con lo sviluppo dello scheletro della piattaforma *web*, ossia una pagina HTML contenente soltanto gli elementi necessari all'interazione. Di seguito viene presentato un esempio dal codice.

```
<div id="creaccount">
        <h1> Crea un Account</h1>
        <label for="name">Nome:</label><br />
        <input type="text" id="name"><br />
        <label for="address">Indirizzo:</label><br />
        <input type="text" id="address"><br />
        <br />
        <button id="creaAccount">Crea</button>
        <hr>
</div>
```

L'implementazione vera e propria con *Web3* invece, comprendente la gestione delle funzioni di interazione, è stata definita all'interno di un *file JavaScript*, connesso alla pagina *web*.

All’interno di tale *file*, infatti, *Web3* è stato utilizzato, in un primo momento, per stabilire la connessione con il contratto sulla rete *blockchain* locale. Ciò è stato possibile mediante la definizione di parametri quali:

- Indirizzo della rete;
- Indirizzo del contratto distribuito sulla rete;
- ABI del contratto.

Di seguito viene presentato un esempio dal codice.

```
var myContract;

var web3js = new Web3('http://127.0.0.1:8545');

var contractAddress =
'0x1516389997E9D41fA6624dBD4cE2eDf39Ff14125';

var myABI = […];

myContract = new web3js.eth.Contract(myABI,
contractAddress);
```

Successivamente, è stato possibile utilizzare questi parametri per sviluppare funzioni capaci di interagire con il contratto. Quest’ultime, infatti, sono state realizzate mediante l’utilizzo di chiamate asincrone a metodi specifici del contratto, permettendone così l’interazione. Di seguito viene presentato un esempio dal codice.

```
function eliminaAccount() {
   var address =
   document.getElementById("address").value;


   var output = document.getElementById("output");


   info(address).then(function (result) {
      if (result.exists) {

            transazioneRaw(myContract.methods.deleteAc
            count(address).encodeABI());
            return output.innerHTML = "Account
            eliminato";
      }
      else {
            return output.innerHTML = "Account
            inesistente";
      };
  });
}
```

Come si evince dall'esempio sopra illustrato, poiché i metodi enunciati consentono sia l'accesso che la modifica di parametri, è stato necessario utilizzare due tipologie di chiamate differenti.

La prima, riguardante l'accesso ai parametri del contratto, è stata realizzata mediante l'utilizzo del metodo

```
.call()
```

Di seguito viene presentato un esempio dal codice.

```
myContract.methods.ipToAcc(id).call();
```

La seconda invece, poiché riguardante la modifica di parametri del contratto, è stata realizzata mediante l'utilizzo di una funzione apposita, in grado di permettere l'invio di transazioni da parte di un account della rete. All'interno della funzione è stato necessario specificare nuovi parametri, ossia:

- Prezzo della commissione;
- Limite del valore della commissione;
- Tipologia di dati trasmessi;
- Chiave privata dell'account;

Di seguito viene presentato un esempio dal codice.

```
function transazioneRaw(en) {
     var encoded = en;
     var tx = {
          to: contractAddress,
          gasPrice: web3js.utils.toWei('1', 'gwei'),
          gasLimit: 1000000,
          data: encoded
     };
     web3js.eth.accounts.signTransaction(tx,
     key).then(
          signed => {
          web3js.eth.sendSignedTransaction(
          signed.rawTransaction).on('receipt',
          console.log) }
     );
}


transazioneRaw(
     myContract.methods.deleteItem(ide).encodeABI()
);
```

Terminata l'implementazione con *Web3*, con annesso lo sviluppo dello scheletro della piattaforma *web* (vedi fig.13*)*, è sorta la necessità di utilizzare un *database* centralizzato, al fine di garantire maggiore controllo e sicurezza a determinate tipologie di dati. Ciò ha portato all'utilizzo di XAMPP.

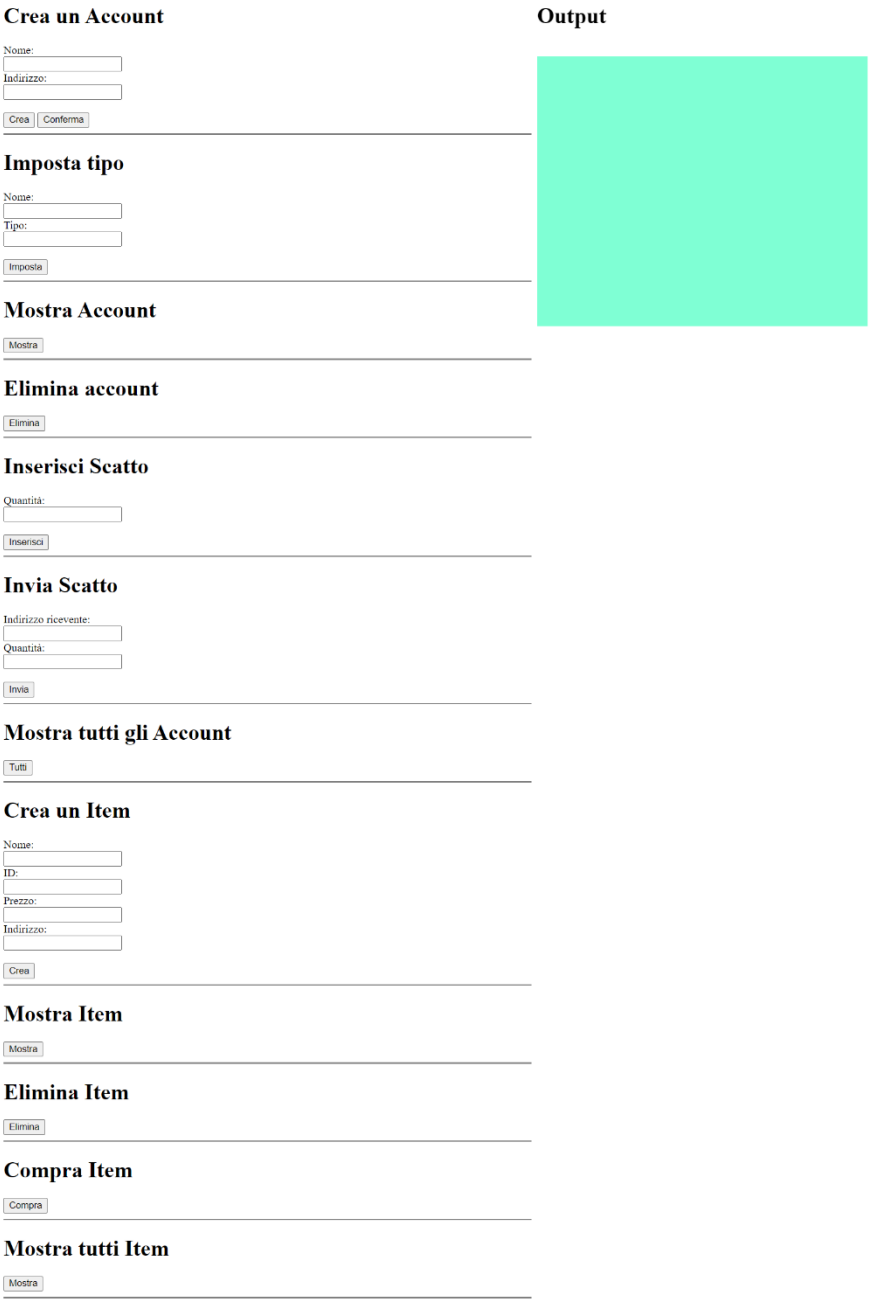

Figura 13. Scheletro della piattaforma web.

5.5. Utilizzo di XAMPP

L'esigenza di un *database* centralizzato nasce dalla volontà di garantire maggiore controllo e sicurezza alle tipologie di dati utili nei sistemi di autenticazione all'interno della piattaforma. Ciò ha portato all'utilizzo di XAMPP, un *software* che permette di creare, manipolare e interrogare un *database* senza la necessità di un *hardware* dedicato. A seguito della sua installazione, tramite sito ufficiale[17], è stato possibile accedere alle sue funzionalità, gestite mediante appositi pulsanti all'interno del pannello di controllo del *software* (vedi fig.14).

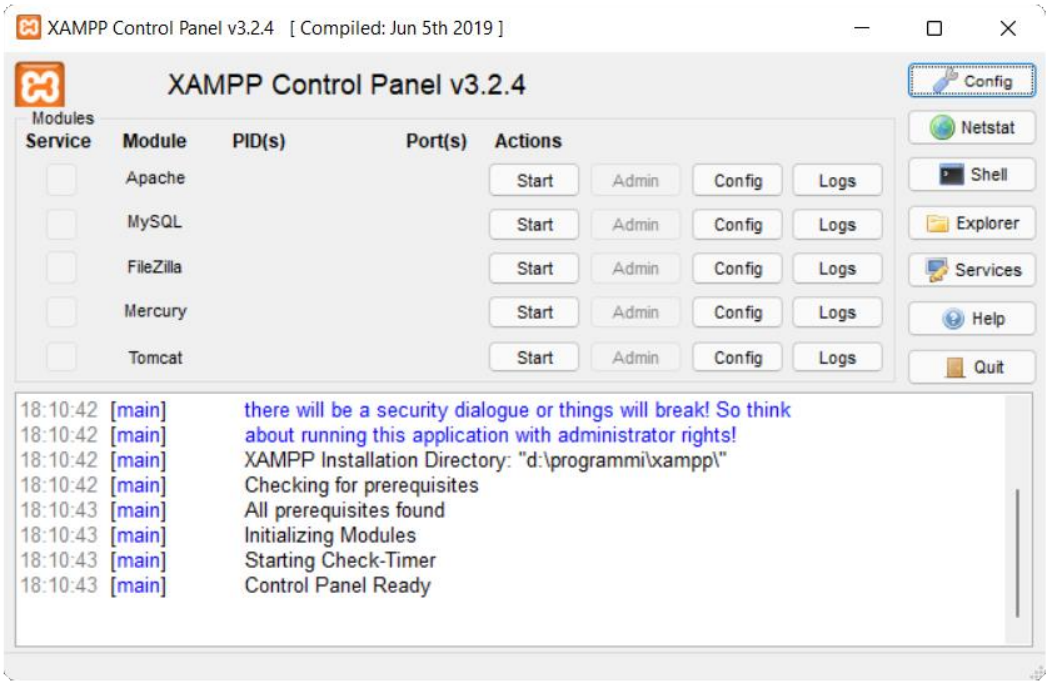

Figura 14. Pannello di controllo XAMPP

Avviando i servizi *Apache* e *MySQL* dal pannello sopra illustrato, è stato possibile accedere a *phpMyAdmin*, un *software* incluso all'interno di XAMPP che consente di eseguire operazioni sul *database MySQL*, mediante interfaccia *web* (vedi fig.15). All'interno di quest'ultimo, dunque, è stato possibile creare il *database* per il prototipo della piattaforma *Banca del Noi*.

[17] In riferimento al sito ufficiale di XAMPP, presente al link: https://www.apachefriends.org/it/index.html

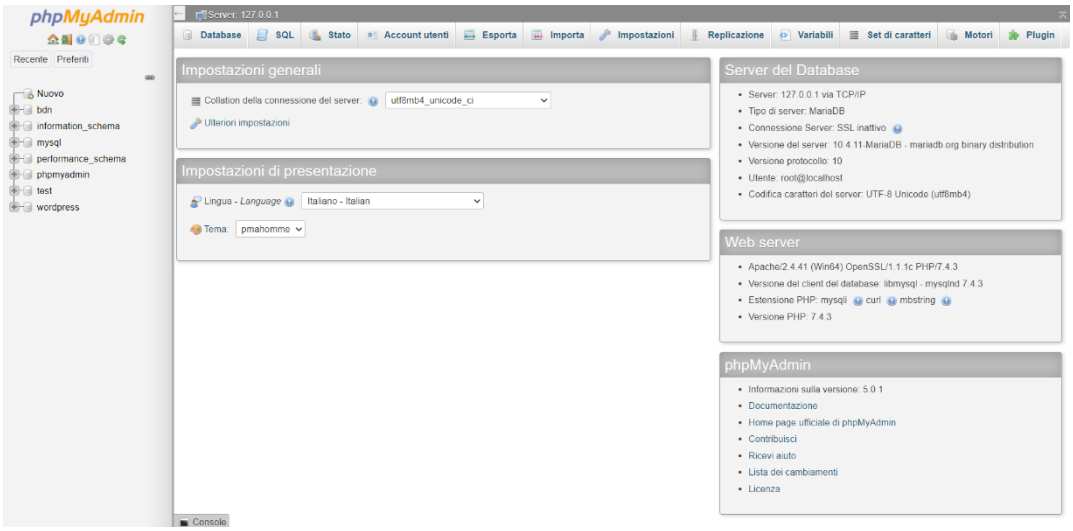

Figura 15. Interfaccia web phpMyAdmin

Il processo di creazione del *database* si è svolto tenendo conto della tipologia di dati da ospitare. Poiché essi riguardano principalmente informazioni sensibili, legate agli *account* della piattaforma, ossia: nome, *password*, pin e chiave privata dell'*account*, è stata predisposta un'unica tabella, la cui struttura viene illustrata di seguito (vedi fig.16).

	#	Nome	Tipo	Codifica caratteri	Attributi	Null	Predefinito	Commenti	Extra	Azione
☐	1	ID	int(11)			No	*Nessuno*		AUTO_INCREMENT	Modifica Elimina Più
☐	2	nome	varchar(20)	utf8_general_ci		No	*Nessuno*			Modifica Elimina Più
☐	3	password	varchar(64)	utf8_general_ci		No	*Nessuno*			Modifica Elimina Più
☐	4	pin	varchar(64)	utf8_general_ci		No	*Nessuno*			Modifica Elimina Più
☐	5	chiave	varchar(256)	utf8_general_ci		No	*Nessuno*			Modifica Elimina Più

Figura 16. Struttura tabella database accounts.

Completata la creazione del *database*, prima di procedere con lo sviluppo dei sistemi di autenticazione, è stato necessario predisporre il codice utile al popolamento del *database*, rappresentato dal processo di registrazione alla piattaforma. Poiché l'architettura di XAMPP consente l'interazione da *browser* con il *database* esclusivamente mediante l'utilizzo del *server Apache*, tale processo è stato realizzato impiegando il linguaggio PHP e la libreria *jQuery*.

L’utilizzo del linguaggio PHP si è incentrato nello sviluppo del codice utile ad effettuare la connessione e l’interrogazione del *database*. In particolare, quest’ultima, ha riguardato l’utilizzo di una *query* per l’inserimento dei *record*. Di seguito vengono riportati due esempi dal codice.

```
<?php
     $database = "bdn";
     $username = "root";
     $password = "";
     $servername = "localhost";
     $conn = mysqli_connect($servername, $username,
     $password, $database) or die ("Errore
     connessione ".mysqli_connect_error($conn));
?>
```

```
$sqlR = "INSERT INTO accounts (ID, nome, password,
pin, chiave) VALUES
('null','$userR','$passwordR','$pin','$chiave')";
```

Il passaggio di parametri da inserire all’interno del *database*, invece, è stato gestito mediante l’utilizzo della libreria *jQuery*. All’interno di un *file JavaScript* collegato ad una pagina HTML dedicata, infatti, è stata realizzata una funzione che, mediante la creazione di un nuovo *account* sulla *blockchain*, ne trasmette i parametri tramite URL. Di seguito viene presentato un esempio dal codice.

```
$.getJSON("api/regist.php", { user: name, password:
password, pin: CryptoJS.SHA256(pin).toString(), key:
key });
```

Come si evince dal codice sopra illustrato, per motivi di sicurezza legati alla trasmissione e archiviazione dei parametri, essi, fatta eccezione per il nome dell'*account*, sono stati sottoposti a criptazione. Per poter effettuare tale operazione, è stata utilizzata la libreria *Crypto.js*. Inoltre, poiché all'interno della piattaforma è necessario l'utilizzo della chiave privata nella sua forma originale, sono state predisposte due funzioni specifiche per la criptazione e decriptazione mediante parola chiave. Di seguito viene presentato un esempio dal codice.

```
function Encrypt(x, y) {
      var encJson =
      CryptoJS.AES.encrypt(JSON.stringify(x),
      y).toString();

      var encData =
      CryptoJS.enc.Base64.stringify(CryptoJS.enc.Utf8
      .parse(encJson));

      return encData;
}
```

```
function Decrypt(x, y) {
      var decData =
      CryptoJS.enc.Base64.parse(x).toString(CryptoJS.
      enc.Utf8);

      var bytes = CryptoJS.AES.decrypt(decData,
      y).toString(CryptoJS.enc.Utf8);

      return JSON.parse(bytes);
  }
```

Terminato lo sviluppo del sistema di registrazione, è stato possibile procedere con la realizzazione del codice relativo ai processi di autenticazione, quali: *login* e firma delle transazioni. Il primo, seguendo la linea del sistema di registrazione, è stato realizzato utilizzando la medesima procedura. In particolare, è stato creato un *file* PHP per l'interrogazione del *database*, nel quale, tuttavia, è stata inserita una *query* di ricerca per le credenziali di accesso. Di seguito viene presentato un esempio dal codice.

```
$sql = "SELECT nome, pin, chiave FROM accounts WHERE
nome= '$user' AND password= '$password'";
```

Il passaggio di parametri, come per la registrazione, è avvenuto mediante l'utilizzo di *jQuery*, tuttavia in questo caso, esso è stato utile anche alla gestione dei parametri prodotti dal risultato della *query*. All'interno del codice, infatti, essi sono stati utilizzati per permettere l'operazione di *login* e, di conseguenza, per permetterne l'utilizzo all'interno della piattaforma vera e propria. Di seguito viene presentato un esempio dal codice.

```
$.getJSON("api/login.php", { user: user, password:
password }, verifica);


function verifica(data) {

      …

      return window.location = 'redirect.html?n=' +
      data['nome'] + "&p=" + data['pin'] + "&k=" +
      data['chiave'];

      …


}
```

Per quanto riguarda lo sviluppo del processo di firma delle transazioni, esso si è incentrato sulla realizzazione di una funzione in grado di generare un numero randomico di cinque cifre. Questo numero viene generato ed assegnato in fase di registrazione ad ogni utente, inoltre, esso viene inserito all'interno del *database* ed utilizzato per le funzioni di criptazione e decriptazione della chiave privata. Di seguito viene presentato un esempio dal codice.

```
pin= getRandomInt(10000,99999);

key = Encrypt(chiave, pin);

function getRandomInt(min, max) {
     var number;
     min = Math.ceil(min);
     max = Math.floor(max);
     number = Math.floor(Math.random() * (max - min)
     + min);
     return number.toString();
  }
```

In questo modo, è stato possibile sviluppare un meccanismo di firma delle transazioni sicuro, legato alla decriptazione della chiave privata. Ciò ha permesso di non ricorrere a *software* alternativi di firma, i quali avrebbero aumentato la complessità d'uso della piattaforma. Attraverso questo sistema, invece, viene richiesto all'utente la sola memorizzazione di un codice numerico, mantenendo così un livello di complessità minimo.

Il termine dello sviluppo di quest'ultimo processo, sancisce la fine del codice necessario all'implementazione delle funzionalità della piattaforma. Ciò porta alla fase conclusiva della realizzazione del prototipo, ossia lo studio e l'applicazione del relativo design.

5.6. Studio e applicazione del design

L'ultima fase, relativa alla realizzazione del prototipo di *Banca del Noi*, è rappresentata dallo studio progettuale e metodologico del *design* della piattaforma, comprendendone la relativa applicazione. Ciò, permette di scandire questa fase in tre diversi stadi, ossia: analisi, *concept* e sviluppo.

Il primo stadio, si è incentrato sullo studio approfondito degli obiettivi e finalità del progetto, con i quali è stato possibile condurre un'analisi comparativa. Tale analisi, è stata svolta sfruttando la raccolta di alcune piattaforme analoghe per obiettivi e contenuti[18], al fine di valutarne i rispettivi risultati d'uso. Al termine dell'analisi, è stato possibile definire le linee guida e la struttura logica dei contenuti presenti all'interno della piattaforma, utili per il raggiungimento di obiettivi comunicativi, grafici e legati all'usabilità della piattaforma stessa.

La definizione di quest'ultimi ha permesso di procedere con la realizzazione del *concept* della piattaforma. Per *concept*, si intende un prototipo grafico che, in questo caso specifico, è stato realizzato mediante il *software Figma*. Poiché la struttura logica dei contenuti ha previsto la suddivisione delle funzionalità della piattaforma in diverse sezioni, il primo elemento del *concept* ad esser stato disegnato è stato il sistema di navigazione. Esso, rappresentato da un *menu* contestuale posto sulla sinistra, è stato progettato in modo da essere sempre accessibile, simulando il comportamento di un pannello di controllo. All'interno del menu contestuale, inoltre, sono state inserite le informazioni base riguardo l'utente che ne sta facendo uso e un pulsante adibito al *logout* (vedi fig. 15 e 16). Conseguentemente al sistema di navigazione, il secondo elemento ad esser stato disegnato è stato il contenuto delle varie sezioni. Posto a sinistra del menù contestuale, esso presenta elementi diversi in base alla sezione di riferimento. In particolare, la sezione denominata *Dashboard*, mostra le informazioni riguardanti: il numero di utenti registrati, il numero di beni e servizi disponibili e la percentuale di *ScaTTO* utilizzato dagli utenti (vedi fig. 19 e 20). Per quanto riguarda le sezioni successive, invece, esse presentano rispettivamente gli elementi strutturali utili al conseguimento delle funzioni in esse descritte (vedi da fig. 21 a 28).

[18] In riferimento alla sezione 2.1.2 del presente elaborato.

Con riferimento agli obiettivi di inclusione del progetto *Banca del Noi*, il registro di stile utilizzato è stato strettamente informale, in modo da infondere all'utente sensazioni di accoglienza e di conforto. Per queste ragioni, è stato adoperato un *font* tondeggiante e senza grazie, denominato *Poppins*, appartenente a *Google Fonts*[19] (vedi fig. 17).

Figura 17. Testo d'esempio del font Poppins.

La palette di colori scelta, invece, ha incluso i colori rosso e blu (vedi fig. 18), utilizzati sotto forma di gradienti per evidenziare elementi specifici.

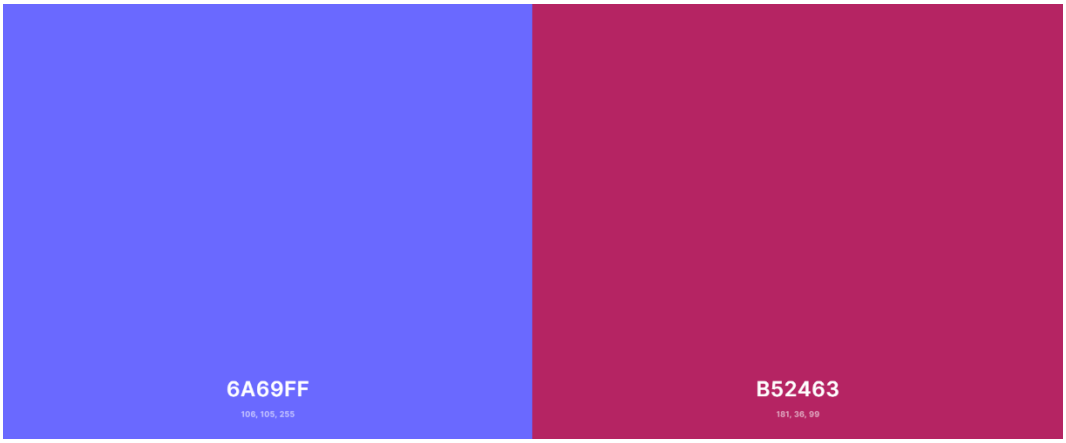

Figura 18. Colori palette.

[19] Per maggiori approfondimenti consultare il sito ufficiale: https://fonts.google.com/specimen/Poppins#standard-styles

Terminata la realizzazione del *concept*, l'ultimo stadio ha riguardato il suo sviluppo. Trattandosi di una piattaforma web, ciò è stato eseguito mediante la modifica dello scheletro HTML, precedentemente generato, e l'utilizzo del linguaggio CSS. Di seguito viene presentato un esempio dal codice.

```
<div id="profilesection">
      <img id="profile-pic" src="src/profile.jpg"
      alt="profile-pic">
       <div id="profile-text">
           <header id="profile-name">Utente</header>
           <p id="profile-type">Tipo</p>
        </div>
</div>
```

```
#profilesection {
      display: flex;
      align-items: center;
      padding-left: 15%;
      padding-top: 30%;
      padding-bottom: 15%;
}
```

All'interno di questa fase, inoltre, sono stati gestiti i comportamenti dinamici dei contenuti, mediante codice *JavaScript* dedicato. Questi completano l'identità della piattaforma e, inoltre, supportano la corretta presentazione dei contenuti. Tra questi, in particolare, vi è il pulsante dedicato al cambio tema, con il quale l'utente può passare da una modalità chiara ad una scura. Tale comportamento, è stato implementato al fine di migliorare la leggibilità dei testi, anche in caso di scarsa luminosità dello schermo o di presenza di particolari disabilità cognitive degli utenti. Di seguito viene presentato un esempio dal codice.

```
function changeTheme() {
      if (!theme) {
          theme = true;

          var feather =
          document.getElementsByClassName("feather")
          ;
          for (i = 0; i < feather.length - 1; i++) {
              feather[i].style.stroke = "white";
          }

      }
      else {
          theme = false;

          var feather =
          document.getElementsByClassName("feather")
          ;
          for (i = 0; i < feather.length - 1; i++) {
              feather[i].style.stroke = "#222222";
          }
      }

  }
```

La scelta dei colori all'interno delle due modalità ha seguito le linee guida dettate dal *Material design*[20].

[20] Per maggiori approfondimenti consultare il sito: https://material.io/design/color/dark-theme.html

La fine dei processi di sviluppo del *concept* e gestione degli elementi interattivi sancisce la conclusione del processo di realizzazione del prototipo di *Banca del Noi*. Di seguito ne vengono presentati i risultati.

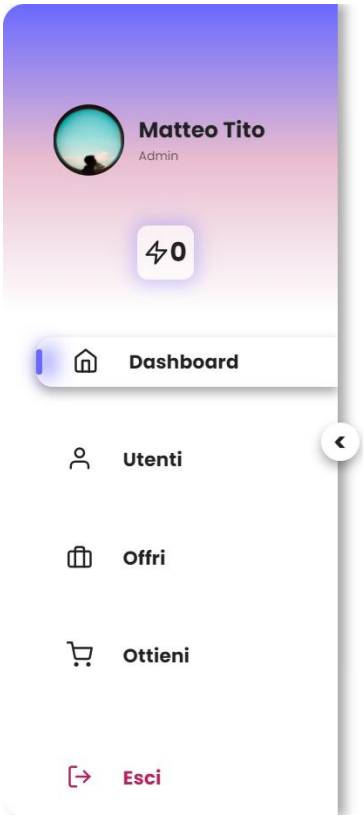

Figura 19. Menu contestuale tema chiaro

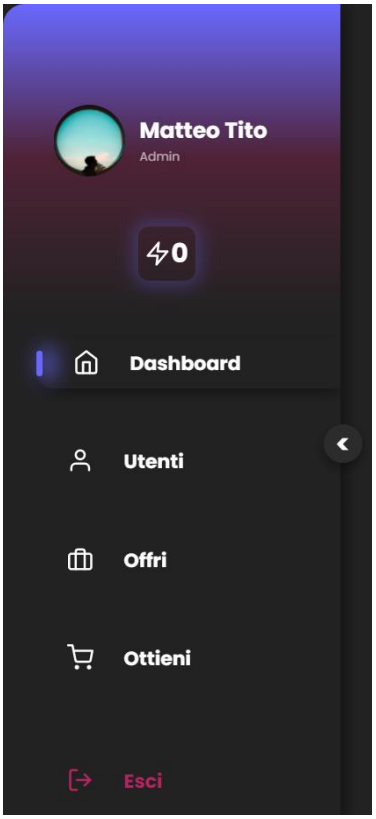

Figura 20. Menu contestuale tema scuro

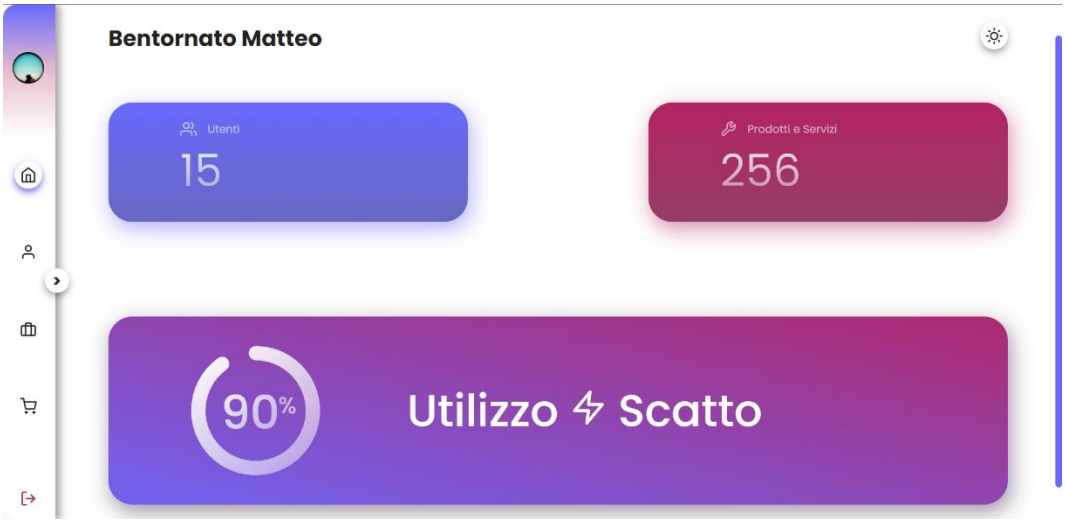

Figura 21. Sezione Dashboard tema chiaro

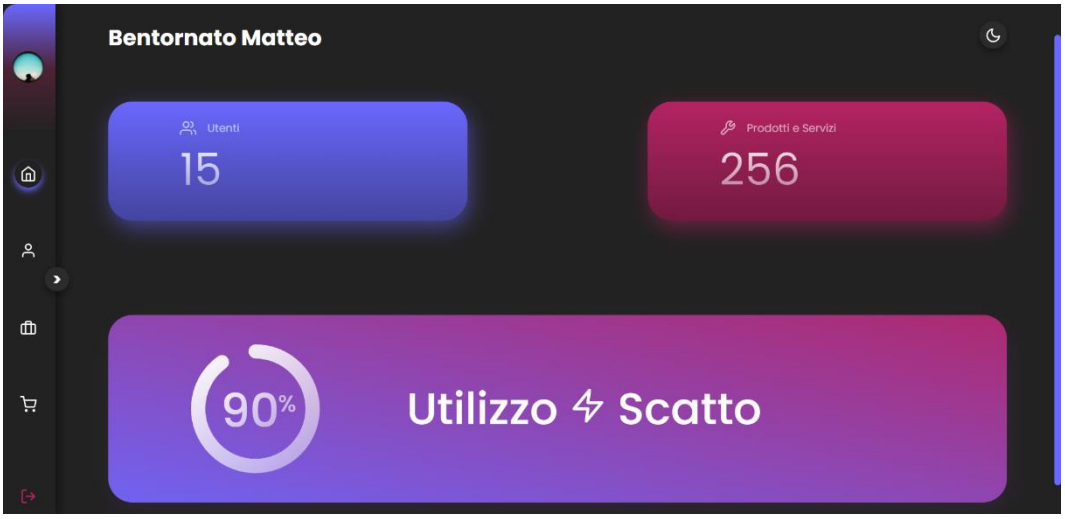

Figura 22. Sezione Dashboard tema scuro

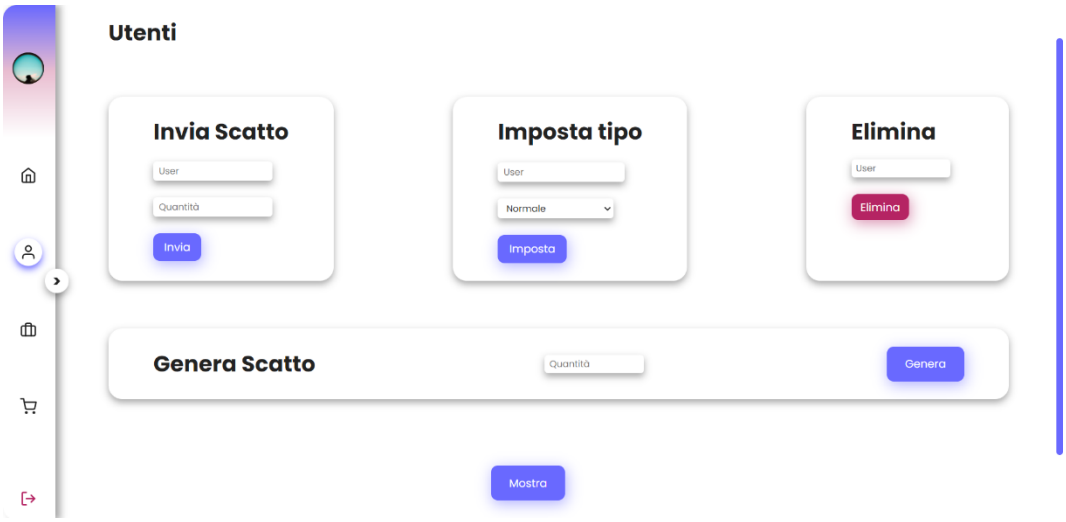

Figura 23. Sezione Utenti tema chiaro

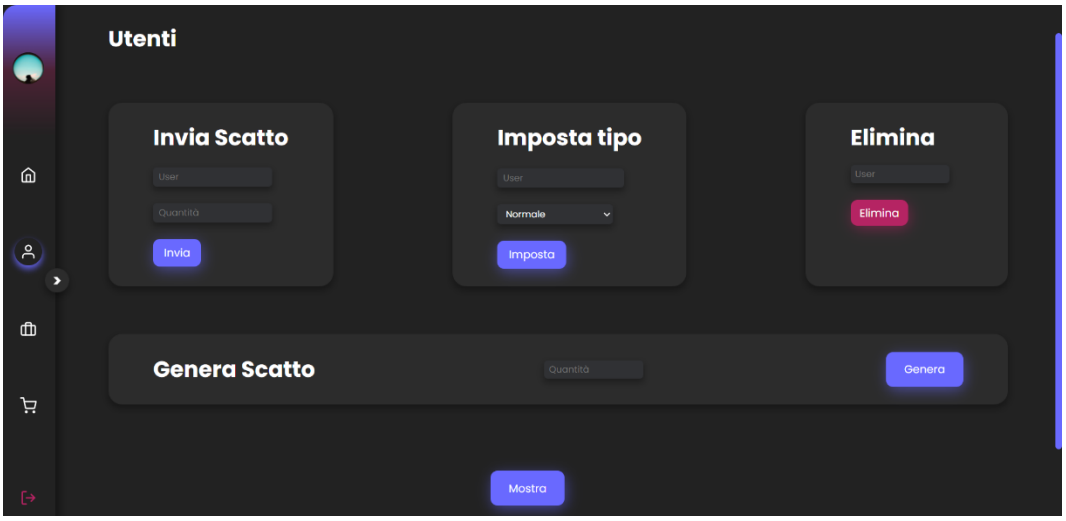

Figura 24. Sezione Utenti tema scuro

Figura 25. Sezioni Offri tema chiaro

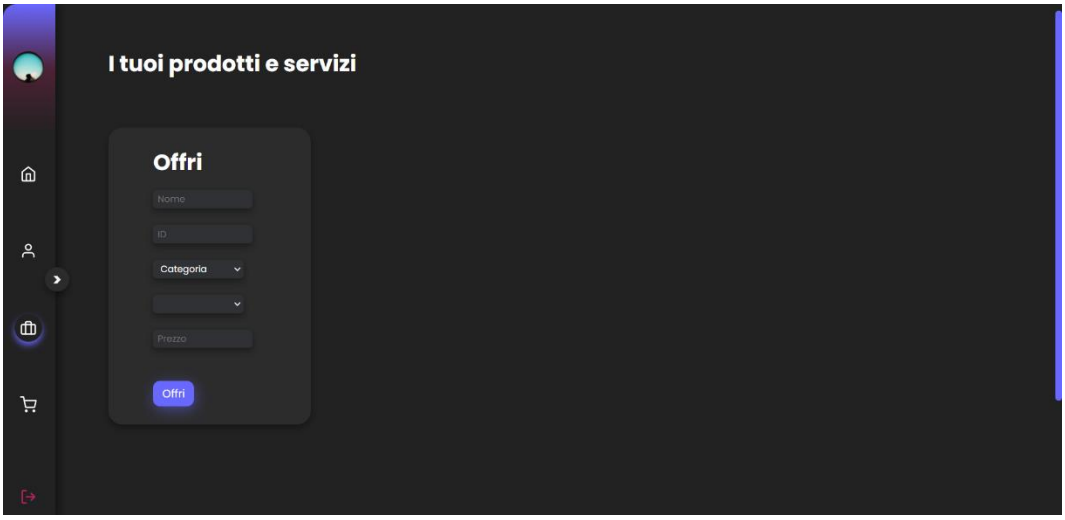

Figura 26. Sezione Offri tema scuro

Figura 27. Sezione Ottieni tema chiaro

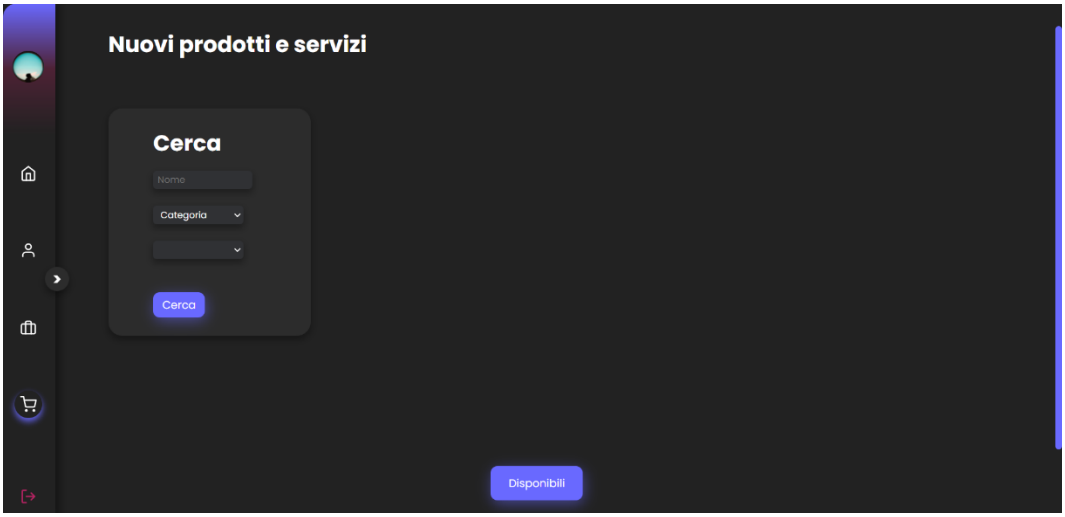

Figura 28. Sezione Ottieni tema scuro

6. Conclusioni

Questo lavoro ha cercato di rispondere all'esigenza di sviluppo di una piattaforma decentralizzata per la gestione di transazioni. A tal fine, a seguito di un'attenta ricerca e alla risoluzione di problematiche di natura etica e strutturale, si è proceduto con la scelta e l'implementazione delle tecnologie necessarie al suo sviluppo, proseguendo poi con l'effettiva realizzazione della piattaforma, in linea con gli obiettivi dichiarati dal progetto *Banca del Noi*.

Il lavoro ha dimostrato come l'utilizzo di una piattaforma decentralizzata abbia migliorato notevolmente la gestione delle transazioni, favorendo un sistema di scambio trasparente e sicuro, in cui gli utenti hanno il controllo sui loro dati. Questo risultato è coerente con quanto espresso all'interno dell'elaborato, in particolare in merito all'utilizzo della tecnologia *blockchain* e ai vantaggi che essa offre.

Il motivo di tale successo è legato all'utilizzo di *Hyperledger Besu*, un *client* basato su *Ethereum* progettato sia per reti autorizzate che pubbliche. Ciò ha permesso di realizzare una rete privata efficiente, in cui è stato possibile distribuire e testare contratti intelligenti. Lo sviluppo di quest'ultimi, infine, ha permesso di inserire i vincoli necessari per la verifica e la corretta negoziazione del sistema di scambio, rappresentato dalla piattaforma.

Risulta importate precisare che il seguente lavoro si è incentrato esclusivamente sullo sviluppo di una specifica tipologia di piattaforma, in uno stato ancora prototipale, tenendo presenti gli obiettivi e le finalità del committente. Le scelte progettuali, pertanto, sono da considerarsi specifiche e non applicabili al caso generale.

Possibili sviluppi futuri potrebbero riguardare lavori di *testing* di nuovi protocolli per la rete *blockchain*, utili per determinare eventuali differenze legate alle prestazioni e ai consumi energetici dell'intero sistema. Ulteriore lavoro potrebbe riguardare il *test* sul campo del prototipo, la raccolta del *feedback* degli utenti e il conseguente adattamento della piattaforma.

Bibliografia

- Alan G. Konheim. 2010. *Hashing in Computer Science: Fifty Years of Slicing and Dicing*. Wiley-Interscience.

- Asadullah, Ahmad, Faik, Isam, e Kankanhalli, Atreyi. 2018. *Digital Platforms: A Review and Future Directions*. In: *PACIS 2018 Proceedings*. pp. 248.

- Daina R. Bouquin. *GitHub.* 2015. In: *Journal of the Medical Library Association.* 3,103. pp. 166-167.

- Emanuele Dagnino, Silvia Spattini. Dicembre 2017. *Evoluzione del mercato dell'incontro tra domanda e offerta di lavoro all'epoca della disintermediazione e dell'uso delle piattaforme tecnologiche.* Edizioni Booklab. https://ebitemp.it/wp-content/uploads/2019/01/Adapt_completo-2.pdf

- Gavin Wood. *A secure decentralised generalised transaction ledger*. 2014. In: *Ethereum Project Yellow Paper*, 151.

- Giovanni Federle, Carla Stefani, 2012. *Gli strumenti del grafico*. Roma, CLITT.

- Jacopo Moschini, Francesco Bruschi. 2021. *Il futuro della blockchain*. Assolombarda.

- Letizia Bertazzon, Maria Pia Favaretto, Davide Girardi, Mavi Lodoli, Paolo Tommasin, Giulia Carfagnini. 2015. Rapporto di ricerca: *Educazione ed inclusione sociale: modelli, esperienze e nuove vie per la IeFP*. Centro Nazionale Opere Salesiane, Formazione Aggiornamento Professionale. https://www.cnos-fap.it/sites/default/files/pubblicazioni/Educazione_e_Inclusione_Sociale.pdf

- Maria Eugenia Pesce, Riccardo Milano, Federico Zoppei. Dossier: *Monete Complementari*. DES di Padova. http://www.denicolaonline.org/public/download/file/materiale%20didattico/economia/dossier%20monete%20complementari%20051109-1.pdf

- Matthieu Montalban, Vincent Frigant, Bernard Jullien. Luglio 2019. *Platform economy as a new form of capitalism: a Régulationist research program*. In: *Cambridge Journal of Economics*. Volume 43, numero 4, pp. 805 - 824.

- Michael Juntao Yuan. 2019. *Sviluppare applicazioni Blockchain: Guida per creare sistemi decentralizzati su reti distribuite.* Feltrinelli Editore.

- Penny Williams, Paula McDonald e Robyn Mayes. 2021. *Recruitment in the gig economy: attraction and selection on digital platforms*. In: *The International Journal of Human Resource Management.* Volume 32, pp. 4136 – 4162

- Philip Boucher, Susana Nascimento, Mihalis Kritikos. Febbraio 2017. Analisi approfondita: *Come la blockchain può cambiarci la vita*. Servizio Ricerca del Parlamento europeo, Unità Prospettiva scientifica. https://www.europarl.europa.eu/RegData/etudes/IDAN/2017/581948/EPRS_IDA(2017)581948_IT.pdf

- Praitheeshan, P., Pan, L. Doss, R. *Private and Trustworthy Distributed Lending Model Using Hyperledger Besu*. 2021. In: *SN Computer Science*. Volume 2, pp. 115.

- Regolamento (UE) 2016/679 del Parlamento europeo e del Consiglio, 27 aprile 2016. *Regolamento generale sulla protezione dei dati*.

- Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. 2008.

Sitografia

- *Arcipelago SCEC*

  http://www.arcipelagoscec.net/come-funziona-lo-scec/ (visitato il 21 dicembre 2021)

- *Cambridge Centre for Alternative Finance*

  https://ccaf.io/cbeci/index/comparisons (visitato il 21 dicembre 2021)

- *Chocolatey*

  https://chocolatey.org/ (visitato il 21 dicembre 2021)

- *CryptoJS - Documentation*

  https://cryptojs.gitbook.io/docs/ (visitato il 21 dicembre 2021)

- *Ethereum - Documentation*

  https://ethereum.org/it/developers/docs/evm/ (visitato il 21 dicembre 2021)

- *Figma*

  https://help.figma.com/hc/en-us (visitato il 21 dicembre 2021)

- *GitHub*

  https://github.com/ (visitato il 21 dicembre 2021)

- *Google Fonts - Emotive considerations for choosing typefaces*

  https://fonts.google.com/knowledge/choosing_type/emotive_considerations_for_choosing_typefaces (visitato il 21 dicembre 2021)

- *Hyperledger Besu - Documentation*

  https://besu.hyperledger.org/en/stable/ (visitato il 21 dicembre 2021)

- *Hyperledger Besu - Wiki*

  https://wiki.hyperledger.org/display/BESU/Hyperledger+Besu (visitato il 21 dicembre 2021)

- *jQuery - Documentation*

https://api.jquery.com/ (visitato il 21 dicembre 2021)

- *JSON*

  https://www.json.org/json-it.html (visitato il 21 dicembre 2021)

- *Material Design - Documentation*

  https://material.io/design/color/dark-theme.html (visitato il 21 dicembre 2021)

- *Ministero del Lavoro e delle Politiche Sociali*

  https://www.lavoro.gov.it/temi-e-priorita/poverta-ed-esclusione-sociale/Pagine/orientamento.aspx (visitato il 21 dicembre 2021)

- *Node.js - Documentation*

  https://nodejs.dev/learn (visitato il 21 dicembre 2021)

- *SartexPay*

  https://www.sardexpay.net/ (visitato il 21 dicembre 2021)

- *Solidity - Documentation*

  https://docs.soliditylang.org/en/v0.8.10/ (visitato il 21 dicembre 2021)

- *Truffle - Documentation*

  https://trufflesuite.com/docs/truffle/ (visitato il 21 dicembre 2021)

- *Web3 - Documentation*

  https://web3js.readthedocs.io/en/v1.2.6/getting-started.html (visitato il 21 dicembre 2021)

- *W3Schools - Ajax*

  https://www.w3schools.com/xml/ajax_intro.asp (visitato il 21 dicembre 2021)

- *XAMPP*

  https://www.apachefriends.org/it/index.html (visitato il 21 dicembre 2021)