Politecnico di Torino

Tesi

Laurea Magistrale in Ingegneria Informatica

Smart Contract e Mercati Energetici: Applicazione della Blockchain per la Decentralizzazione della Gestione Energetica

Candidato:
Stefano Leto

Relatore:
Prof. Danilo Bazzanella

Correlatori:
Alfredo Favenza
Alessandro Mozzato

Indice

Elenco delle figure

Elenco delle tabelle

Listings

CAPITOLO 1

Abstract

Il presente lavoro di tesi esplora l'applicazione della tecnologia blockchain nei mercati energetici, con particolare attenzione al mercato del giorno prima (MGP) e al mercato di bilanciamento aFRR (automatic Frequency Restoration Reserve). L'obiettivo è sviluppare un'infrastruttura decentralizzata per la gestione della partecipazione di aggregatori di batterie per veicoli elettrici, utilizzando gli smart contract per ottimizzare il processo di piazzamento e accettazione delle bid.
Nel contesto energetico tradizionale, gli intermediari centralizzati rappresentano punti critici di inefficienza e rischio, oltre a una mancanza di trasparenza che spesso può condurre a pratiche non competitive, costi elevati per i consumatori e limitazioni nella gestione sostenibile delle risorse. La blockchain consente invece la creazione di un sistema trasparente e sicuro, in cui l'aggregatore gestisce la registrazione e la vendita dell'energia accumulata dalle batterie dei veicoli elettrici, garantendo un compenso ai proprietari dei veicoli per l'energia fornita. Attraverso l'implementazione di smart contract, vengono automatizzati processi chiave come la registrazione delle batterie, il piazzamento delle offerte di vendita, l'accettazione delle bid da parte del TSO (Transmission System Operator) nel caso del mercato aFRR o da buyer privati per il mercato del giorno prima, e la gestione delle transazioni di pagamento.
In particolare, lo smart contract per la gestione degli aggregator permette il controllo efficiente delle batterie registrate, il contratto per la gestione del TSO regola il processo di bilanciamento dell'energia, quello

per la gestione del MGP gestisce l'asta delle offerte piazzate dall'aggregator, garantendo che l'energia richiesta venga fornita in modo sicuro e verificabile. Questa infrastruttura riduce i tempi di transazione e i costi operativi, mantenendo al contempo l'integrità del mercato energetico.
La ricerca si concentra sul ruolo della blockchain nel migliorare la fiducia tra gli attori del mercato e nell'ottimizzare la risposta a richieste energetiche su breve termine, analizzando sia i vantaggi sia le limitazioni nell'implementazione di una piattaforma blockchain nei mercati energetici decentralizzati.

CAPITOLO 2

Introduction

L'innovazione nei sistemi di gestione dell'energia è oggi una priorità per promuovere un modello di mercato più efficiente, sostenibile e sicuro. La scelta di applicare la tecnologia blockchain ai mercati energetici nasce dalla volontà di affrontare alcune delle principali sfide dei mercati tradizionali, in particolare il Mercato del Giorno Prima (MGP) e il Mercato di Bilanciamento aFRR (automatic Frequency Restoration Reserve). L'obiettivo di questa tesi è sviluppare un'infrastruttura decentralizzata capace di ottimizzare il processo di offerta e accettazione delle bid attraverso smart contract, concentrandosi sulla partecipazione degli aggregatori di batterie per veicoli elettrici.

2.1 Limiti dei mercati energetici tradizionali

I mercati energetici tradizionali, pur essendo sofisticati, soffrono di alcune limitazioni intrinseche legate alla centralizzazione delle operazioni e alla necessità di intermediari per garantire la fiducia tra le parti. Nel mercato del giorno prima, i partecipanti (come le utility e i produttori di energia) forniscono offerte e richieste per il giorno successivo,
permettendo di programmare la generazione e il consumo energetico con un giorno di anticipo. Tuttavia, le variazioni nelle condizioni meteorologiche e le fluttuazioni di domanda possono alterare questa pianificazione. In caso di squilibri tra domanda e offerta, entra in gioco il mercato aFRR,

un mercato di bilanciamento in cui i gestori di rete (Transmission System Operators, o TSO) acquistano o vendono energia in tempo quasi reale per mantenere la frequenza del sistema stabile.
Inoltre vi è una forte centralizzazione e la presenza di intermediari, che implicano costi di transazione elevati e tempi di elaborazione delle transazioni spesso lunghi. In un contesto in cui la reattività e la trasparenza sono essenziali per la stabilità della rete, tali limitazioni rappresentano un ostacolo per una gestione energetica ottimale.

2.2 Obiettivi della blockchain

In questo contesto, la blockchain rappresenta una soluzione promettente per superare i limiti del modello tradizionale, grazie alla capacità di registrare e verificare le transazioni in modo permanente, permettendo la costruzione di un sistema privo di intermediari e riducendone tempi e costi di esecuzione, aumentando di conseguenza la fiducia tra gli attori del mercato. Gli smart contract, in particolare, automatizzano funzioni chiave come la registrazione delle batterie, il piazzamento delle offerte e l'accettazione delle bid, garantendo così una gestione più reattiva e sicura delle risorse energetiche. Ciò favorisce la transizione verso un sistema più efficiente e orientato alle esigenze della nuova era energetica.

CAPITOLO 3

Background

3.1 Introduzione alla Rete Elettrica

La rete elettrica è il cuore pulsante dei sistemi energetici moderni, responsabile della trasmissione e distribuzione dell'elettricità dai punti di generazione fino ai consumatori finali. Operare e mantenere una rete elettrica affidabile richiede una stretta coordinazione tra diversi attori, ognuno con ruoli specifici per garantire la sicurezza e la stabilità del sistema. Tra i principali operatori troviamo i *Transmission System Operator* (TSO) e i *Distribution System Operator* (DSO), le cui responsabilità coprono diverse aree dell'infrastruttura energetica.
I **TSO** si occupano della gestione delle reti di trasmissione ad alta tensione, generalmente superiori ai 220 kV, assicurando il trasporto efficiente e sicuro dell'energia dai grandi impianti di produzione alle reti di distribuzione locali. In Europa, il coordinamento tra i TSO è garantito dall'organizzazione *ENTSO-E* (European Network of Transmission System Operators for Electricity), che riunisce 39 operatori di 35 Paesi, promuovendo l'integrazione delle reti elettriche nazionali in un unico mercato europeo dell'energia. Tra le principali responsabilità dei TSO troviamo:

- Garantire l'equilibrio continuo tra domanda e offerta di energia elettrica.

- Gestire i flussi transfrontalieri per ottimizzare l'uso delle risorse energetiche.
- Fornire servizi ancillari, come il mantenimento della frequenza di rete e la capacità di *black start* (riavvio della rete dopo un blackout).
- Monitorare e pianificare l'espansione delle infrastrutture per soddisfare le esigenze future.

I **DSO**, invece, gestiscono le reti di distribuzione locali e regionali, che operano a tensioni medie e basse (tipicamente da 6-50 kV e 250-400 V), trasportando l'elettricità ai consumatori finali. Con l'aumento delle risorse energetiche distribuite (DER), come veicoli elettrici, pannelli fotovoltaici e pompe di calore, i DSO stanno assumendo un ruolo sempre più critico nel garantire la flessibilità e l'integrazione delle rinnovabili nel sistema energetico. Tra i compiti principali dei DSO troviamo:

- Monitorare in tempo reale le condizioni della rete, come congestioni e carichi sui trasformatori.
- Integrare le risorse energetiche distribuite nei mercati locali, garantendo al contempo la stabilità della rete.
- Coordinare le interazioni con i TSO per fornire servizi di supporto, come il controllo della tensione e la gestione dei picchi di carico.

L'integrazione crescente delle energie rinnovabili e delle risorse distribuite sta trasformando il movimento dell'energia, che non è più unidirezionale (dalla produzione centralizzata al consumatore), ma bidirezionale e decentralizzato. Questo nuovo paradigma richiede una maggiore cooperazione tra TSO e DSO, nonché l'adozione di soluzioni digitali avanzate per migliorare la trasparenza e l'efficienza operativa.

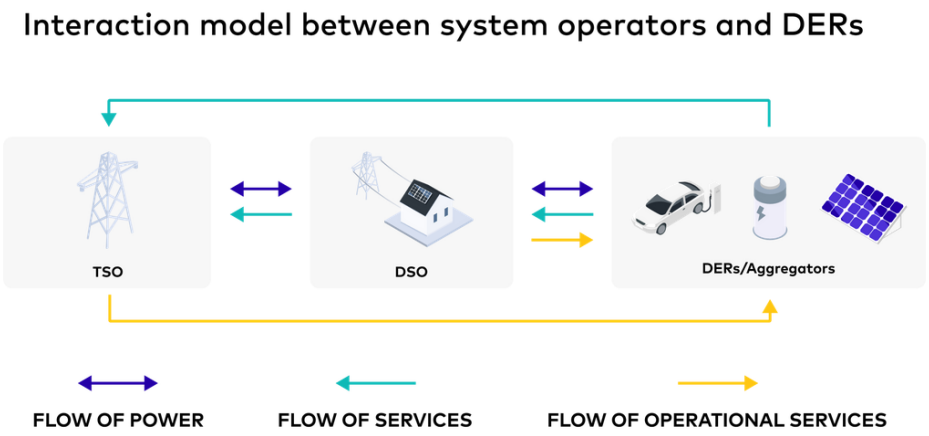


Figura 3.1: Interazione tra TSO, DSO e DER [1]

La **digitalizzazione**, infatti, è diventata un pilastro fondamentale per la gestione delle reti intelligenti (*smart grids*). Tecnologie come i sistemi di gestione delle risorse energetiche distribuite (DERMS) e piattaforme di rete intelligente consentono ai gestori di integrare segnali provenienti da diversi attori del mercato, ottimizzare l'utilizzo delle infrastrutture esistenti e rispondere in tempo reale alle fluttuazioni della domanda e dell'offerta. Un esempio rilevante è il progetto *Grid Operation 4.0* di Westnetz, che ha collegato 100 reti locali a un sistema digitale per migliorare l'utilizzo della capacità di distribuzione.
La crescente complessità dei sistemi energetici decentralizzati richiede una collaborazione continua e una rapida evoluzione delle tecnologie di monitoraggio e controllo. Questi cambiamenti, guidati dalla digitalizzazione e dall'integrazione delle energie rinnovabili, sono essenziali per garantire un futuro energetico sostenibile, resiliente e accessibile [1].

3.2 I Mercati Energetici

In questo paragrafo si introduce il funzionamento dei mercati energetici, con un focus sui mercati del giorno prima (MGP) e di bilanciamento aFRR, fornendo una panoramica su caratteristiche, obiettivi e modalità operative.
In Europa, il processo di liberalizzazione dei mercati energetici è iniziato

negli anni '90, con l'obiettivo di creare un mercato interno dell'energia più competitivo, sicuro e trasparente. Questo cambiamento ha portato alla nascita di un sistema energetico basato su un'ampia varietà di attori che gestiscono la produzione, il trading, la trasmissione e la fornitura di elettricità, all'interno di un quadro normativo ben definito. *L'obiettivo finale è garantire una fornitura energetica sicura, prezzi competitivi e servizi migliori per i consumatori* [2] attraverso un mercato che favorisca la concorrenza tra operatori, migliorando così l'efficienza complessiva del sistema.

Uno degli aspetti chiave della gestione del mercato energetico europeo è l'impossibilità di immagazzinare facilmente l'elettricità, questo rende essenziale un equilibrio continuo tra domanda e offerta. Infatti, per mantenere la stabilità del sistema elettrico, è necessario che la frequenza della rete (50 Hz nella maggior parte dei Paesi europei) sia costantemente regolata. Un disallineamento tra domanda e offerta può causare gravi problemi, come interruzioni della fornitura o addirittura blackout. Per evitare ciò, i mercati energetici a breve termine, come il mercato intraday e il mercato di bilanciamento, rivestono un ruolo fondamentale nel garantire che l'offerta soddisfi la domanda in tempo reale.

Il mercato europeo dell'energia è oggi altamente integrato grazie a meccanismi come il *Market Coupling*, che consente un'ottimizzazione dei flussi di energia elettrica tra i diversi Paesi. Questo sistema di accoppiamento dei mercati, attraverso la condivisione delle interconnessioni transfrontaliere, permette di distribuire l'elettricità in modo più efficiente, assicurando che *l'energia fluisca dove è maggiormente richiesta, contribuendo a ridurre i costi complessivi e migliorare la sicurezza dell'approvvigionamento.* [2]

Un altro strumento fondamentale per il bilanciamento della rete è rappresentato dai mercati di riserva automatica come l'aFRR (automatic Frequency Restoration Reserve). In questo contesto, i Transmission System Operators (TSO) acquistano servizi di bilanciamento dai Balancing Service Providers (BSP) per garantire la stabilità della rete. Grazie a piattaforme come PICASSO (Platform for the International Coordi-

nation of Automated Frequency Restoration and Stable System Operation), è possibile coordinare gli scambi di aFRR tra i diversi Paesi europei, armonizzando i mercati di bilanciamento a livello continentale. L'integrazione dei mercati energetici europei ha anche facilitato lo sviluppo di mercati spot, come il mercato intraday. Questo mercato consente scambi fino a pochi minuti prima della consegna dell'energia, offrendo ai partecipanti la flessibilità necessaria per bilanciare la produzione e il consumo in tempo reale. Ciò è particolarmente rilevante per l'integrazione delle fonti di energia rinnovabile, come eolico e solare, che sono intrinsecamente variabili. In questo modo, il mercato intraday aiuta a compensare le fluttuazioni nella produzione e a ridurre la necessità di riserve aggiuntive costose.
La struttura attuale dei mercati energetici europei è stata progettata non solo per migliorare l'efficienza economica, ma anche per supportare la transizione energetica verso un sistema più sostenibile. La spinta verso un maggiore utilizzo delle energie rinnovabili ha reso necessario un adattamento continuo dei meccanismi di mercato per gestire la crescente variabilità della produzione. *Il mercato elettrico europeo è considerato uno dei più efficienti al mondo* [2] grazie alla sua capacità di integrare volumi crescenti di energia rinnovabile, minimizzando al contempo gli impatti sui prezzi e migliorando la resilienza complessiva del sistema energetico.
Nonostante i successi raggiunti, rimangono sfide significative per il futuro, come la necessità di migliorare ulteriormente le interconnessioni tra i Paesi e adattare i mercati per accogliere volumi sempre maggiori di energie rinnovabili. Questo richiede continui investimenti in nuove infrastrutture e tecnologie, oltre a un costante aggiornamento del quadro normativo per garantire che il mercato resti efficiente, trasparente e accessibile a tutti i partecipanti.

3.2.1 Mercato del Giorno Prima (MGP)

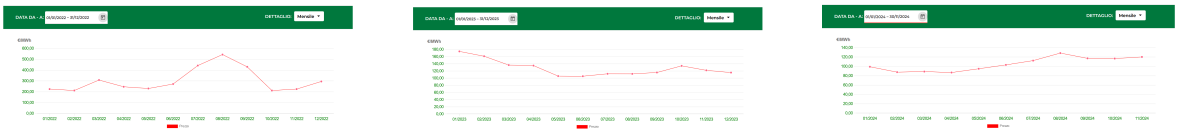
Il *Mercato del Giorno Prima (MGP)* rappresenta uno dei principali strumenti per la contrattazione di energia elettrica in Italia, nonché in molti

mercati energetici liberalizzati a livello internazionale. In questo mercato, le transazioni vengono effettuate il giorno precedente alla consegna effettiva dell'energia. Lo scopo principale del MGP è quello di assicurare un corretto approvvigionamento energetico per il giorno successivo, favorendo un efficiente equilibrio tra domanda e offerta.

Funzionamento del Mercato

Questo mercato si appoggia su un meccanismo di asta in cui i partecipanti (produttori di energia, grandi consumatori, operatori di rete e trader) presentano offerte per la vendita o l'acquisto di blocchi orari di energia per il giorno successivo. Come descritto dal GME, "gli operatori partecipano presentando offerte nelle quali indicano la quantità e il prezzo massimo/minimo al quale sono disposti ad acquistare o vendere" [3]. Le offerte vengono accettate dopo la chiusura della seduta di mercato sulla base del merito economico, ovvero partendo dalle offerte più vantaggiose fino al soddisfacimento della domanda.
Il mercato del giorno prima è un mercato ad asta e non un mercato a contrattazione continua; questo significa che tutte le offerte vengono raccolte entro un orario prestabilito e processate simultaneamente, con l'assegnazione che avviene in base alla curva di domanda e offerta. In particolare, le offerte di vendita e di acquisto sono ordinate per prezzo e vengono accettate fino a coprire la domanda totale del sistema. In presenza di limiti di trasmissione tra diverse zone, i prezzi possono variare localmente. Pertanto, tutte le offerte accettate riferite alle zone virtuali estere vengono valorizzate al prezzo marginale di equilibrio della zona a cui appartengono. [3]
Per valorizzare le offerte accettate nelle zone geografiche italiane viene utilizzato il *PUN, Prezzo Unico Nazionale*, che rappresenta una media ponderata dei prezzi delle zone, basata sulle quantità acquistate in ciascuna di esse.
Dalle immagini riportate di seguito (fig. 3.2), è possibile osservare l'andamento del PUN negli anni 2022, 2023 e 2024. Nel 2022, il prezzo dell'energia era nettamente superiore rispetto agli anni successivi, men-

tre nel 2023 e 2024 il prezzo sembra essersi stabilizzato su valori più contenuti.
Questa significativa differenza può essere spiegata da diversi fattori che hanno influenzato il mercato dell'energia elettrica. Nel 2022, il forte aumento dei prezzi dell'energia è stato principalmente causato dalle tensioni geopolitiche internazionali, come il conflitto tra Russia e Ucraina, che ha generato una forte instabilità nell'approvvigionamento di gas naturale, risorsa fondamentale per la produzione di energia in Italia e in gran parte dell'Europa. La riduzione delle forniture di gas ha comportato un'impennata dei costi del combustibile, riflettendosi direttamente sui prezzi del PUN.
Nel 2023 e nel 2024, invece, il mercato ha mostrato una maggiore stabilità, grazie a una combinazione di fattori. In primo luogo, la diversificazione delle fonti di approvvigionamento di gas naturale, l'introduzione di nuove infrastrutture di rigassificazione e l'aumento delle importazioni da Paesi diversi dalla Russia hanno ridotto la pressione sui prezzi. Inoltre, l'accelerazione della transizione energetica verso fonti rinnovabili ha consentito di ridurre la dipendenza dalle risorse fossili, contribuendo a stabilizzare i prezzi dell'energia.
Allo stesso tempo, politiche europee di regolamentazione e interventi governativi volti a contenere l'impatto dei costi energetici sui consumatori hanno contribuito a calmierare il mercato, favorendo un progressivo ritorno a prezzi più normali rispetto al picco registrato nel 2022.



(a) PUN nel 2022 (b) PUN nel 2023 (c) PUN nel 2024

Figura 3.2: Variazione del PUN negli ultimi tre anni

Ruolo dei Balancing Responsible Parties (BRP)

Nella maggior parte dei mercati liberalizzati, il commercio di energia nel MGP è gestito dai *Balancing Responsible Parties* (BRP), ossia entità

che si assumono la responsabilità di mantenere in equilibrio il proprio wallet di energia, evitando sbilanciamenti tra la produzione e il consumo. I BRP utilizzano il MGP per ottimizzare il proprio bilancio energetico quotidiano, acquistando o vendendo energia in base alle previsioni di consumo e produzione.
Ad esempio, un operatore di un grande parco eolico può delegare a un trader elettrico il ruolo di BRP per tutta l'energia generata dal parco [4]. In tal modo, il trader utilizza dati in tempo reale e previsioni meteorologiche per determinare quanta energia sarà generata e garantire che la quantità di energia venduta sia allineata con la produzione effettiva. Eventuali surplus o deficit imprevisti possono essere corretti tramite transazioni sul mercato del giorno prima.

Il System Marginal Price

Il meccanismo di determinazione del prezzo nel MGP è basato sul *system marginal price*. Questo significa che tutte le offerte accettate vengono remunerate al prezzo marginale, che è il prezzo dell'ultima offerta accettata necessaria per soddisfare la domanda totale. Questo approccio garantisce che tutti i partecipanti, sia acquirenti che venditori, ricevano un prezzo uniforme, favorendo l'efficienza economica.
Tuttavia, in presenza di congestioni tra zone di mercato, il prezzo può variare a livello locale. Quando i limiti di transito tra le zone vengono saturati, i prezzi di equilibrio si differenziano, riflettendo le condizioni specifiche di ciascuna area geografica. Questo può portare a situazioni in cui alcune zone registrano prezzi più alti a causa di vincoli infrastrutturali.

Aste e Contrattazioni Over-the-Counter (OTC)

Oltre al mercato organizzato delle aste, molte transazioni nel giorno precedente avvengono tramite contratti bilaterali Over-the-Counter (OTC). In questi casi, le parti coinvolte negoziano direttamente il prezzo e la quantità di energia, al di fuori delle piattaforme di mercato centralizzate. Questo permette una maggiore flessibilità, specialmente per i grandi consumatori industriali e i trader che desiderano coprire i propri rischi

in maniera più precisa.
Il ricorso alle contrattazioni OTC è particolarmente utile per i BRP che devono assicurare il bilanciamento del proprio wallet, eliminando eventuali surplus o deficit di energia [4]. Quando la previsione della produzione o del consumo si discosta dalle aspettative, i BRP possono ricorrere al mercato del giorno prima per correggere gli sbilanciamenti e ottimizzare i propri costi.

Criticità e Sfide del MGP

Nonostante i vantaggi, il MGP presenta alcune criticità legate alla sua natura centralizzata. La presenza di pochi intermediari che gestiscono le transazioni può portare a inefficienze e a un aumento dei costi per i partecipanti. Inoltre, le offerte e i dati di mercato non sempre sono accessibili a tutti i partecipanti in modo trasparente, limitando la competitività.
L'introduzione della tecnologia blockchain e degli smart contract potrebbe offrire una soluzione a queste problematiche, permettendo un accesso decentralizzato al mercato e garantendo una maggiore trasparenza nelle transazioni. Gli smart contract potrebbero automatizzare il processo di asta, riducendo i tempi di transazione e garantendo una liquidazione immediata dei pagamenti, migliorando così l'efficienza complessiva del mercato.

3.2.2 Mercato di Bilanciamento aFRR

Il *mercato di bilanciamento aFRR (automatic Frequency Restoration Reserve)*, noto anche come riserva secondaria, è fondamentale per mantenere stabile la frequenza della rete elettrica (vedere fig. 3.3). Per garantire che la frequenza rimanga entro specifiche soglie, i gestori dei sistemi di trasmissione (TSO) attivano servizi di bilanciamento come l'aFRR, acquistati dai fornitori di questi servizi(i Balancing Service Provider, BSP). Dopo l'armonizzazione dei mercati europei, i BSP sono ora obbligati a garantire la disponibilità della riserva entro un massimo di 5 minuti (Full

Activation Time). L'aFRR si attiva automaticamente entro 30 secondi per sostituire gradualmente la riserva primaria (FCR), e, se l'instabilità persiste oltre i 12,5 minuti, può essere supportata o sostituita dalla riserva terziaria (mFRR).
Le reti elettriche richiedono una frequenza costante di 50 Hz o 60 Hz (in alcune nazioni). Se la frequenza supera una certa soglia (+/- 0,2 Hz), può verificarsi una caduta di tensione (brownout) o persino un blackout [5]. Per prevenire questi rischi, i TSO bilanciano costantemente l'energia immessa e prelevata dalla rete, mantenendo un equilibrio dinamico tra domanda e offerta.
Tradizionalmente, la riserva secondaria viene fornita principalmente da centrali che offrono alta controllabilità e automazione, come impianti idroelettrici a pompaggio e turbine a gas. Negli ultimi anni, tuttavia, alcune risorse energetiche rinnovabili, come impianti di trattamento del biogas e cogeneratori (CHP), sono state aggregate in virtual power plants (VPP) per contribuire alla stabilità della rete e fornire servizi di aFRR.

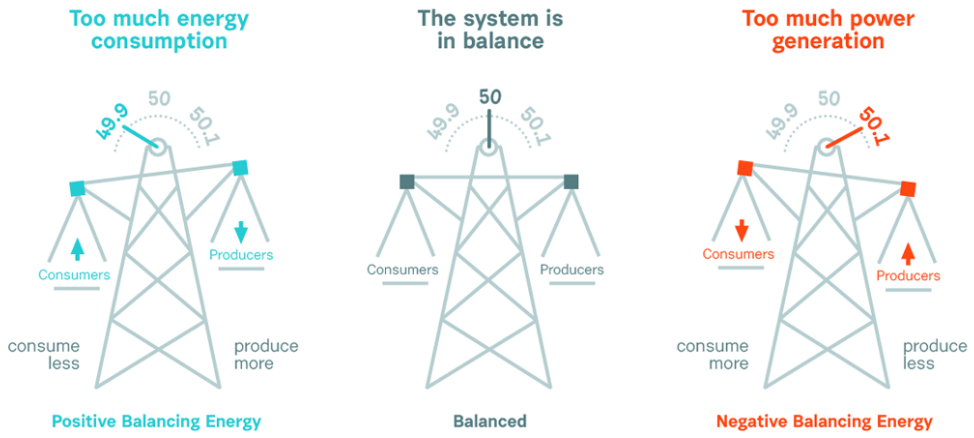


Figura 3.3: Bilanciamento nel mercato aFRR [6]

Struttura del Mercato aFRR in Europa

In Europa, la struttura del mercato aFRR è regolamentata dalle European Balancing Guidelines (EB GL), che forniscono una base normativa

comune per i Paesi partecipanti. Nell'ambito di questa armonizzazione, è stata creata la piattaforma PICASSO (Platform for the International Coordination of Automated Frequency Restoration and Stable System Operation), che consente lo scambio di servizi di aFRR tra Paesi europei, seguendo regole comuni per la stabilità della rete. In Italia l'aFRR rappresenta la seconda riserva di energia elettrica, la cui gestione del mercato è affidata al TSO Terna [7].
Terna è il principale operatore del sistema di trasmissione dell'energia elettrica in Italia e detiene la proprietà della rete di trasmissione nazionale (RTN) ad alta e altissima tensione. Con oltre 75.000 km di linee elettriche gestite, rappresenta uno dei più grandi operatori indipendenti di trasmissione elettrica in Europa [8].
Essa opera in regime di monopolio regolato dall'Autorità di Regolazione per Energia Reti e Ambiente (ARERA), seguendo le linee guida del Ministero dello Sviluppo Economico (MiSE). Questo posizionamento le consente di coordinare l'integrazione delle fonti rinnovabili nel sistema elettrico italiano, contribuendo così alla transizione energetica del Paese. La gestione del dispacciamento è uno degli aspetti più importanti delle sue operazioni: Terna è responsabile di bilanciare costantemente domanda e offerta di energia, operando 24 ore su 24 per mantenere la stabilità della frequenza di rete a 50 Hz.
Inoltre si vuole mirare a rafforzare il ruolo dell'Italia come hub energetico europeo, promuovendo la decarbonizzazione e la resilienza del sistema elettrico nazionale e prevedendo di investire oltre 21 miliardi di euro nei prossimi dieci anni per accelerare la transizione verso un sistema energetico più sostenibile, riducendo la dipendenza dalle fonti estere e migliorando la capacità di integrazione delle rinnovabili.

Meccanismo d'Asta e Remunerazione

I TSO acquisiscono le riserve di bilanciamento tramite aste, in cui stabiliscono la quantità di energia di controllo necessaria sulla base di dati storici e delle condizioni specifiche del mercato. In molti Paesi, le offerte sono ordinate in una lista di merito crescente, basata sul prezzo.

Le offerte sono accettate progressivamente fino a coprire la domanda, garantendo l'efficienza del sistema in quanto le offerte più economiche sono selezionate per prime [5].
La remunerazione dei servizi di bilanciamento forniti tramite l'aFRR si articola in due livelli: una remunerazione per la capacità, che copre i costi di mantenimento della riserva disponibile, e una remunerazione per l'attivazione dell'energia di bilanciamento effettiva. Questo sistema bifasico assicura che la riserva sia utilizzata solo quando effettivamente necessaria, evitando attivazioni non indispensabili.

Sfide e Problematiche Recenti

Recentemente, il mercato comune per la capacità aFRR nel Nord Europa ha registrato prezzi insolitamente elevati. Questo fenomeno non è il risultato di errori, ma è dovuto a una combinazione di fattori strutturali. Il primo problema è legato alla ridotta capacità di trasmissione tra alcune zone, ciò ha portato all'accettazione di offerte più costose per soddisfare la domanda dei TSO. In queste condizioni, i prezzi di clearing aumentano inevitabilmente.
Un secondo fattore riguarda l'utilizzo di offerte bloccate (*block bids*), le quali vincolano la fornitura di energia per un determinato periodo di tempo, anziché per singole ore. Le *block bids* non possono determinare il *marginal price*, introducendo regole di pricing complesse per evitare che queste offerte stabiliscano un prezzo marginale troppo elevato [9]. Di conseguenza, il prezzo di clearing non corrisponde necessariamente al prezzo delle *block bids*, ma viene calcolato per garantire che queste offerte siano redditizie durante il periodo del blocco senza influenzare eccessivamente il prezzo marginale. Tuttavia, questo può portare a segnali di prezzo meno trasparenti e distorsivi, un aspetto che i TSO stanno cercando di migliorare sia nel breve che nel lungo termine, con l'obiettivo di rendere il sistema più trasparente e rappresentare meglio i costi sottostanti.
Infine, il design delle aste prevede un limite massimo di tempo per il calcolo dei risultati, poiché i dati sono fondamentali per il calcolo giorna-

liero della capacità nella regione nordica. Quando si verificano problemi matematici complessi, l'algoritmo di clearing può restituire un risultato approssimativo. Sebbene questa soluzione garantisca che le offerte accettate e la riserva di capacità transfrontaliera rimangano ottimali, i prezzi possono risultare non intuitivi tra le zone d'asta, a causa di una semplificazione del calcolo delle congestioni.
Implementando la blockchain tutte le offerte (comprese le block bids) verrebbero registrate in un registro immutabile e trasparente. Questo consentirebbe a tutti i partecipanti di accedere alle informazioni sulle offerte e sui prezzi di clearing in tempo reale, eliminando qualsiasi ambiguità nel processo di selezione delle bid.
Inoltre le block bids, per loro natura, richiedono una gestione complessa, in quanto devono essere accettate o rifiutate per l'intero blocco di ore. Attualmente, questo processo richiede intervento umano e l'uso di algoritmi centralizzati per il clearing, che possono portare a inefficienze e ritardi. Qui potrebbero entrare in gioco gli smart contract, i quali possono gestire in modo autonomo il processo di offerta e selezione delle bid, compresa l'accettazione delle block bids in base a determinate condizioni. Questo elimina il bisogno di intermediari centralizzati, riducendo i costi operativi e i tempi di transazione.

3.3 Introduzione alla Blockchain

La *Blockchain* è una tecnologia di gestione decentralizzata delle transazioni e dei dati, inizialmente sviluppata per supportare la criptovaluta Bitcoin. Introdotta per la prima volta nel 2008, questa tecnologia ha rapidamente catturato l'attenzione globale grazie alle sue proprietà uniche: garantire sicurezza, anonimato e integrità dei dati senza l'intervento di terze parti [10]. Nei sistemi tradizionali, le transazioni finanziarie tra individui o aziende sono centralizzate e gestite da intermediari come banche o fornitori di carte di credito. Questo tipo di transazioni comporta costi aggiuntivi e affidamento su un'entità terza che controlla l'intero processo. La Blockchain è stata sviluppata per eliminare tali

intermediari, offrendo un ambiente decentralizzato in cui le transazioni avvengono direttamente tra le parti coinvolte, senza la necessità di affidarsi ad un'autorità centrale.

Struttura della Blockchain

La Blockchain funziona come un database distribuito che mantiene un elenco crescente e cronologico di blocchi di dati, confermati dai nodi partecipanti alla rete. Ogni blocco all'interno della catena è collegato al precedente attraverso un hash di riferimento, noto come *parent block*, formando così una struttura sequenziale. Il primo blocco della catena è chiamato *genesis block* e non ha un blocco padre [11]. Come si può notare nella figura 3.4 un blocco è composto da due sezioni principali: **header** e **body**. L'header contiene metadati cruciali come la versione del blocco, l'hash del blocco precedente, il root hash dell'albero di Merkle, il timestamp, nBits e il nonce. Questi elementi garantiscono che ogni blocco sia unico e collegato al precedente.
Il corpo del blocco, invece, è composto da un *transaction counter*, che indica il numero di transazioni, e dalla lista effettiva delle transazioni registrate in esso. La dimensione massima del blocco e quella di ogni transazione determinano il numero di transazioni che un singolo blocco può contenere.

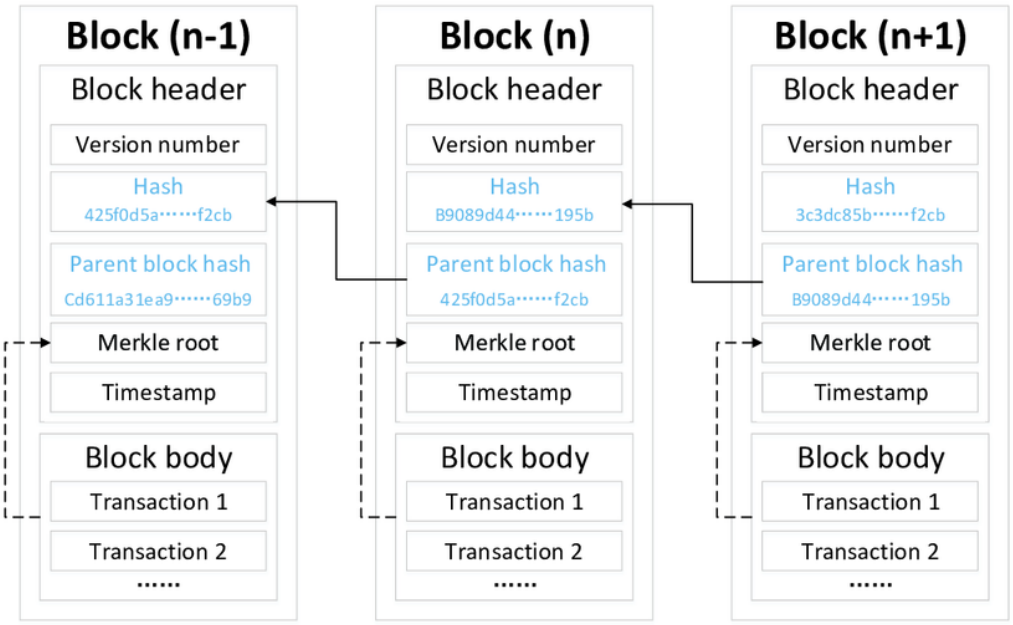


Figura 3.4: Struttura dei blocchi nella blockchain [12]

Caratteristiche della Blockchain

Una delle principali caratteristiche della Blockchain è la sua natura decentralizzata. A differenza dei sistemi tradizionali, dove una singola entità è responsabile della verifica delle transazioni, nella Blockchain *le informazioni su ogni transazione completata sono condivise e disponibili per tutti i nodi della rete*, ciò aumenta la trasparenza e riduce il rischio di frodi [10]. Inoltre, la decentralizzazione elimina il rischio di un singolo punto di fallimento, rendendo il sistema più resiliente.
Questa tecnologia utilizza la crittografia asimmetrica per validare l'autenticità delle transazioni. In questo processo, ogni partecipante alla rete possiede una coppia di chiavi, una privata e una pubblica. La chiave privata è utilizzata per firmare o criptare una transazione, mentre la chiave pubblica è visibile a tutti e permette di decriptare le transazioni [11]. Questa architettura permette agli utenti di interagire in modo sicuro senza la necessità di fidarsi di una terza parte.

Persistenza e Immutabilità Una delle caratteristiche fondamentali della Blockchain è la sua capacità di garantire la persistenza dei dati. Una

volta registrata una transazione, essa non può essere modificata senza alterare tutti i blocchi successivi nella catena, rendendo la manipolazione estremamente difficile. *Ogni blocco è legato al precedente tramite un hash, quindi qualsiasi modifica a un blocco richiederebbe la modifica di tutti i blocchi successivi, il che è computazionalmente impraticabile* [11]. Questa immutabilità rende la Blockchain una soluzione quasi a prova di manomissione, ideale per contesti in cui l'integrità dei dati è essenziale.

Anonimato e Sicurezza Un altro aspetto cruciale della Blockchain è l'anonimato che offre ai partecipanti. Gli utenti possono interagire con la rete utilizzando indirizzi generati casualmente, senza dover rivelare la propria identità. Poiché la Blockchain non richiede un'autorità centrale per monitorare o registrare le informazioni private degli utenti, il sistema garantisce un elevato livello di privacy. Tuttavia, l'anonimato può anche portare a sfide nella tracciabilità, soprattutto nei contesti di prevenzione del riciclaggio di denaro.

Auditabilità Tutte le transazioni registrate nella rete Blockchain sono validate tramite un timestamp digitale, questo rende possibile tracciare ogni operazione iterativamente. Questa capacità di audit rende la Blockchain particolarmente utile in settori che richiedono trasparenza e tracciabilità, come la gestione della supply chain e i mercati finanziari.

3.3.1 Crittografia e sicurezza dei dati

La *crittografia* è uno degli elementi chiave nelle Blockchain e garantisce molte delle sue proprietà fondamentali [10]:

- **Confidentiality (Confidenzialità)**: assicura che i dati siano disponibili solo per entità autorizzate.
- **Integrity (Integrità)**: garantisce che l'informazione possa essere modificata solo da entità autorizzate.
- **Authentication (Autenticazione)**: assicura che l'identità di un'entità della rete sia verificata.

- **Non-repudiation (Non Ripudio)**: si riferisce alla capacità delle Blockchain di dimostrare che un'azione sia stata eseguita da un utente specifico.

Per assicurare queste proprietà è necessario utilizzare la crittografia. Sebbene non sia obiettivo di questa tesi approfondire tutte le tecniche crittografiche, verranno illustrate alcune delle funzioni fondamentali applicate all'interno delle Blockchain.

Funzione di Hash

Per garantire l'*Integrità* e l'*Autenticità* dei dati, viene comunemente utilizzata la funzione di Hash.

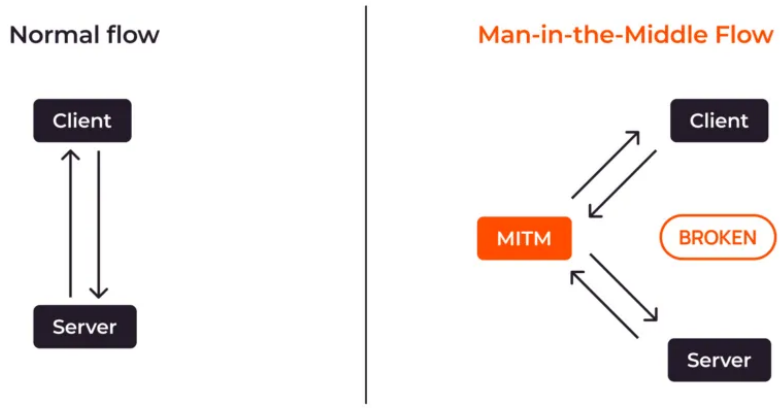


Figura 3.5: L'assenza di hash può causare un man in the middle attack [13]

Quando due entità scambiano dati, è possibile che un'intercettazione da parte di terzi comprometta la comunicazione, modificando i dati trasmessi, come illustrato in Figura 3.5. Per controllare l'integrità dei dati, è possibile utilizzare una funzione che genera una sorta di "prova" di come erano i dati al momento della trasmissione.
Quando il destinatario riceve il messaggio, può riapplicare la stessa funzione per calcolare una propria prova. Se le due prove coincidono, allora il messaggio non è stato alterato durante il trasferimento. La funzione di Hash serve esattamente a questo scopo, generando un *digest* a lunghezza

fissa dai dati in input. Se i dati cambiano anche di un solo carattere, il digest cambia completamente. Uno degli algoritmi più utilizzati è *SHA-3*, l'ultima versione della famiglia di funzioni *SHA*. All'interno della Blockchain, *SHA* viene spesso utilizzato per calcolare l'hash del blocco e per proteggere l'integrità delle transazioni all'interno di esso.

Crittografia Asimmetrica

Per garantire l'*autenticazione*, viene utilizzata la *crittografia asimmetrica* (anche detta crittografia a chiave pubblica). In questo sistema, vengono utilizzate due chiavi distinte per cifrare e decifrare i dati: una chiave privata e una chiave pubblica. La chiave privata è utilizzata per cifrare i dati, mentre la chiave pubblica viene usata per decifrarli. Le due chiavi sono collegate tramite complesse relazioni matematiche e non sono generate casualmente, bensì attraverso algoritmi matematici avanzati.

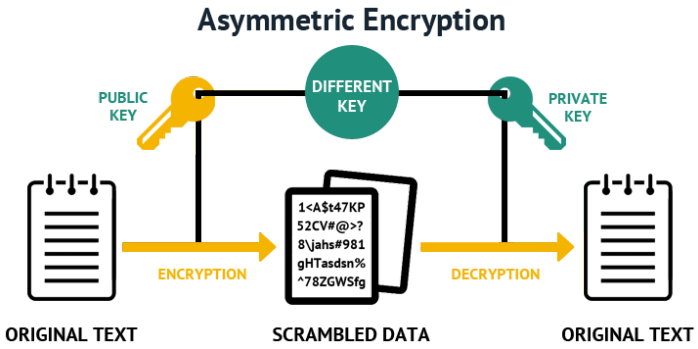


Figura 3.6: Scambio di dati con crittografia asimmetrica [14]

Come illustrato in Figura 3.6, l'entità A può inviare un testo cifrato utilizzando la chiave pubblica dell'entità B. Solo l'entità B, che possiede la chiave privata corrispondente, potrà decifrare il testo, infatti quest'ultima viene utilizzata quando un'entità desidera dimostrare la propria identità. In questo caso, l'entità A cifra il messaggio con la propria chiave privata. Se qualcuno riesce a decifrare il messaggio utilizzando la chiave pubblica di A, sarà certo che il mittente è effettivamente l'entità A stessa.

3.3.2 Vantaggi della Blockchain nei Sistemi Decentralizzati

A differenza dei sistemi centralizzati, dove ogni transazione deve essere convalidata attraverso un'autorità centrale (ad esempio, una banca centrale), la Blockchain consente transazioni peer-to-peer dirette senza la necessità di un intermediario. Questo approccio riduce i costi associati alla gestione centralizzata e allevia i colli di bottiglia delle prestazioni. Tuttavia, l'uso di meccanismi di consenso come il *Proof of Work* può comportare costi elevati in termini di energia e risorse computazionali, rendendo difficile la scalabilità della tecnologia.

3.4 Bitcoin

Bitcoin è un sistema decentralizzato che combina diverse tecnologie per creare un ecosistema di moneta digitale. In Bitcoin, le unità di valuta, chiamate *bitcoin*, vengono utilizzate per immagazzinare e trasferire valore tra i partecipanti all'interno della rete Bitcoin. Gli utenti interagiscono tra di loro tramite il protocollo Bitcoin, principalmente via Internet, sebbene possano essere utilizzate anche altre reti di trasporto. Il software open source alla base del protocollo può essere eseguito su una vasta gamma di dispositivi, rendendo la tecnologia accessibile a chiunque [15].

3.4.1 Storia di Bitcoin

Bitcoin è stato introdotto nel 2008 attraverso un *whitepaper* intitolato *"Bitcoin: A Peer-to-Peer Electronic Cash System"* pubblicato sotto lo pseudonimo di *Satoshi Nakamoto*. Il sistema è stato progettato per risolvere il problema della *double-spending*, che affliggeva le precedenti forme di valuta digitale. L'innovazione di Nakamoto è stata quella di utilizzare un algoritmo di *proof-of-work* per creare consenso decentralizzato tra i partecipanti senza la necessità di un'autorità centrale [15].
Sebbene Nakamoto abbia abbandonato lo sviluppo pubblico nel 2011, il

codice e la rete sono stati mantenuti e migliorati da una comunità di sviluppatori volontari. Il risultato è un sistema che, grazie alla trasparenza e alla sicurezza crittografica, sta rivoluzionando il modo in cui gestiamo il denaro e le transazioni digitali.

3.4.2 Caratteristiche del sistema Bitcoin

Bitcoin è un sistema *peer-to-peer* distribuito che non si affida a un server centrale o a un punto di controllo unico. Tra le principali caratteristiche troviamo:

- **Decentralizzazione**: In Bitcoin, non esiste un'entità centrale che gestisce il sistema. Le transazioni avvengono direttamente tra i partecipanti, eliminando la necessità di intermediari come banche o fornitori di carte di credito.
- **Trasparenza**: Tutte le transazioni vengono registrate in un registro pubblico chiamato *blockchain*, che è accessibile a tutti i nodi della rete. Questo garantisce che ogni transazione sia visibile e verificabile.
- **Sicurezza**: Bitcoin utilizza la crittografia asimmetrica per garantire che solo i proprietari legittimi possano accedere ai propri fondi. Ogni utente possiede una coppia di chiavi: una chiave privata per firmare le transazioni e una chiave pubblica per verificarle.

3.4.3 Come funziona Bitcoin

Gli utenti possono trasferire *bitcoin* attraverso la rete per compiere una vasta gamma di operazioni, come acquistare beni, inviare denaro o estendere crediti. Le transazioni su Bitcoin non richiedono alcuna autorità centrale, rendendo il sistema più veloce, sicuro e senza confini. Le transazioni sono immagazzinate all'interno di blocchi, collegati tra loro tramite un *hash* che fa riferimento al blocco precedente. Questa struttura a catena rende la blockchain praticamente immutabile.

3.4.4 Concetti chiave di Bitcoin

Bitcoin si basa su diversi concetti fondamentali che insieme creano un sistema sicuro e decentralizzato:

- **Blockchain** [16]: rappresenta la storia di ogni transazione Bitcoin confermata. Essa è una struttura dati rappresentata da un elenco ordinato e collegato di blocchi composti da transazioni. Questi sono collegati a ritroso, ciascun blocco ha un riferimento al blocco precedente. Ogni blocco all'interno della Blockchain è identificato con un hash, generato utilizzando l'algoritmo di hash crittografico *SHA256*. Il blocco dispone del campo **hash del blocco precedente**. La grandezza di un blocco Bitcoin è di 1MB.
- **Nodi** [15]: Nella rete Bitcoin, i nodi svolgono un ruolo cruciale nel mantenere la sicurezza e l'integrità del sistema. Esistono due tipi principali di nodi:
  - **Full node**: Un full node è un programma che verifica l'intera storia delle transazioni Bitcoin, assicurandosi che tutte le transazioni seguano le regole del protocollo. I full node possono anche
    memorizzare transazioni già convalidate e fornire dati ad altri programmi Bitcoin sulla stessa macchina o su internet. Utilizzare un full node richiede notevoli risorse computazionali, paragonabili al consumo di banda di un'ora di video in streaming per ogni giorno di transazioni Bitcoin. Tuttavia, offrono completa autonomia ai loro utenti, senza dipendere da terze parti.
  - **Lightweight client (SPV)**: Un lightweight client, noto anche come client SPV (*Simplified Payment Verification*), si connette a un full node o a un server remoto per inviare e ricevere informazioni sulle transazioni Bitcoin. Tuttavia, memorizza il portafoglio dell'utente localmente e convalida solo parzialmente

le transazioni ricevute, consentendo comunque di creare transazioni in uscita in modo indipendente. Questo approccio riduce il carico di lavoro rispetto a un full node, ma implica una maggiore dipendenza dai server remoti.

- **Mining**: Il processo di mining è il meccanismo con cui vengono create nuove unità di bitcoin e confermate le transazioni. I miner competono per risolvere complessi problemi matematici, e il primo che riesce ad aggiungere un nuovo blocco alla blockchain viene ricompensato con nuovi *bitcoin*.

- **Proof-of-Work** [15]: Bitcoin utilizza un algoritmo di *proof-of-work* per garantire che le transazioni siano valide e che la rete sia sicura. Questo sistema richiede ai miner di dedicare potenza computazionale per risolvere i problemi, rendendo estremamente difficile per un attaccante modificare i blocchi già confermati.

- **Wallet Bitcoin**: Un wallet Bitcoin è un'interfaccia che consente agli utenti di inviare e ricevere bitcoin. I Wallet possono essere:

  - **Desktop wallet**: Programmi che offrono un elevato livello di controllo e autonomia, ma che possono essere vulnerabili su sistemi operativi generali.
  - **Mobile wallet**: Applicazioni su dispositivi mobili, ottimali per l'uso quotidiano grazie alla loro semplicità.
  - **Web wallet**: Wallet accessibili tramite browser che conservano le chiavi degli utenti su server di terze parti, compromettendo la privacy.
  - **Hardware wallet**: Dispositivi che conservano le chiavi in hardware specializzato, offrendo un livello superiore di sicurezza contro attacchi.

3.4.5 Proof of Work e Double Spending

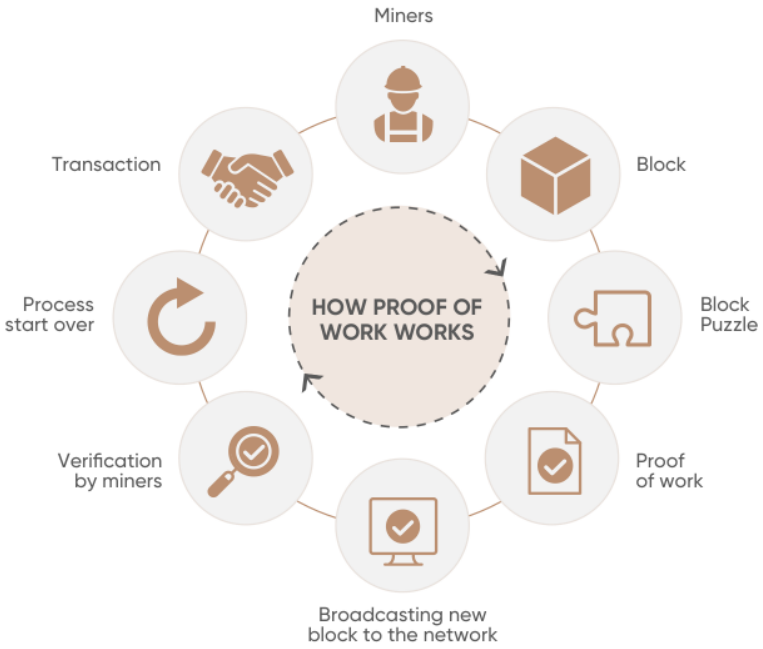


Figura 3.7: Proof of work funzionamento [17]

La *Proof of Work* (PoW) è un meccanismo fondamentale per garantire la sicurezza delle reti blockchain, introdotto da Satoshi Nakamoto nel whitepaper di Bitcoin del 2008. Il suo scopo principale è prevenire la doppia spesa (*double spending*), un problema comune nei sistemi di pagamento digitali dove lo stesso denaro potrebbe essere speso più di una volta. In un sistema tradizionale centralizzato, un'autorità centrale (come una banca) verifica ogni transazione, impedendo duplicazioni. Tuttavia, in un sistema decentralizzato come quello di Bitcoin, la PoW permette ai partecipanti della rete di raggiungere un consenso senza la necessità di un'entità centrale.
Il processo di PoW prevede che i miner, utilizzando risorse computazionali ed energia, competano per risolvere complessi puzzle crittografici. Ogni blocco contenente transazioni viene raggruppato e sottoposto a una funzione di hashing, che genera un *hash* unico. Per aggiungere un nuovo blocco alla blockchain, il miner deve trovare un hash che soddisfi speci-

fiche condizioni predefinite. Poiché ogni modifica anche minima ai dati cambia completamente l'hash, i miner devono continuamente modificare un valore chiamato *nonce* per tentare di ottenere un hash valido.
Il processo di mining non solo garantisce la sicurezza della rete, ma anche l'integrità delle transazioni: una volta che un blocco è stato aggiunto, modificarlo richiederebbe il rifacimento di tutti gli hash dei blocchi successivi, rendendo praticamente impossibile la manipolazione dei dati senza un'enorme quantità di potenza di calcolo.
Grazie a questo meccanismo, la PoW disincentiva comportamenti fraudolenti, poiché i costi di tentare di imbrogliare superano ampiamente i potenziali benefici. In sintesi, *la Proof of Work garantisce che agire onestamente sia più profittevole rispetto a cercare di compromettere la rete* [18].

3.5 Ethereum Blockchain

Ethereum è spesso descritto come "il computer mondiale" [19]. A differenza di Bitcoin, il cui obiettivo principale è essere una valuta digitale, Ethereum mira a diventare una piattaforma decentralizzata per lo sviluppo di applicazioni distribuite (*Decentralized Applications* o DApps) tramite l'uso degli smart contract. In questa sezione vengono approfondite le caratteristiche principali di Ethereum, con un focus sulla sua struttura e sul funzionamento.

3.5.1 Introduzione ad Ethereum

Ethereum è una piattaforma open-source che consente agli sviluppatori di creare applicazioni decentralizzate attraverso l'uso di smart contract. Introdotto nel 2015 da Vitalik Buterin, Ethereum ha ampliato il concetto di blockchain da un semplice sistema di transazione a una piattaforma programmabile che supporta applicazioni complesse.
Da un punto di vista tecnico, *Ethereum è una macchina a stati deterministici ma praticamente illimitata, composta da uno stato globale accessibile e da una macchina virtuale, chiamata Ethereum Virtual Machine*

(EVM), che gestisce le modifiche a tale stato [19]. Questa struttura consente di eseguire programmi chiamati smart contract, che sono archiviati sulla blockchain e possono essere eseguiti automaticamente una volta soddisfatte determinate condizioni.
Ethereum utilizza la sua blockchain per sincronizzare e registrare le modifiche di stato del sistema, con una criptovaluta nativa chiamata Ether (ETH) che funge da unità di misura per i costi di esecuzione [20].

3.5.2 Struttura di Ethereum

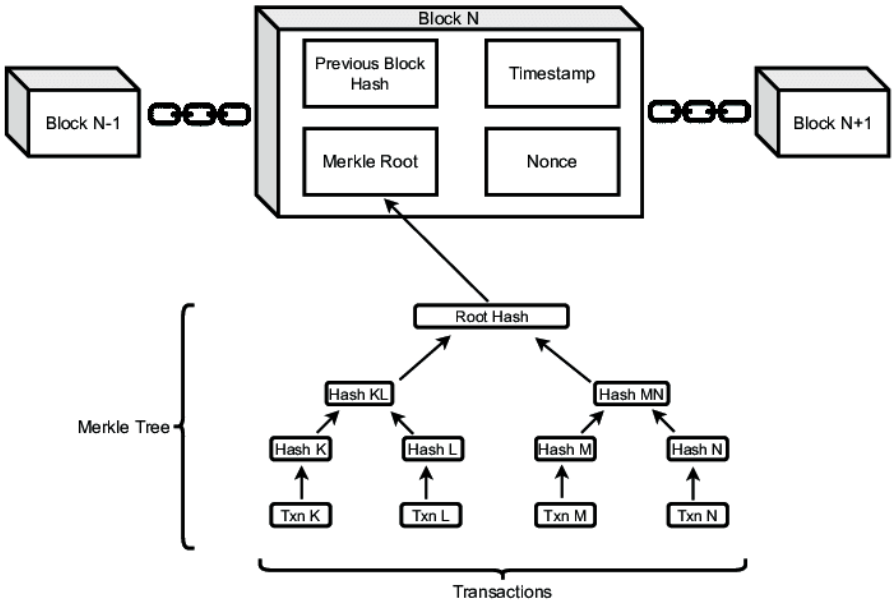


Figura 3.8: Struttura della piattaforma Ethereum [21]

La blockchain di Ethereum è una struttura complessa che tiene traccia non solo delle transazioni finanziarie, come avviene in Bitcoin, ma anche degli stati di contratti intelligenti e DApps. Ethereum utilizza una struttura dati chiamata *Merkle Patricia Tree*, che consente una gestione più efficiente delle transazioni e dello stato globale [19].
I componenti principali della piattaforma Ethereum includono:

- **Rete Peer-to-Peer**: Ethereum utilizza una rete distribuita per sincronizzare i nodi e garantire l'integrità della blockchain.

- **Macchina a Stati** (*State Machine*): L'EVM elabora le transizioni di stato eseguendo smart contract scritti in linguaggi come Solidity.
- **Algoritmo di Consenso**: Fino al 2022, Ethereum ha utilizzato un algoritmo di consenso Proof of Work (PoW). Con l'aggiornamento Ethereum 2.0, è passato a un sistema Proof of Stake (PoS) per migliorare l'efficienza energetica e la scalabilità [20].
- **Transazioni**: Le transazioni su Ethereum includono mittente, destinatario, valore e dati aggiuntivi, e possono attivare smart contract per automatizzare processi.
- **Smart Contracts**: Gli smart contract consentono di automatizzare operazioni complesse, come la gestione di token, prestiti decentralizzati e altri casi d'uso [20].

3.5.3 Dal Proof of Work al Proof of Stake

Inizialmente, Ethereum ha utilizzato il meccanismo di consenso *Proof of Work* (PoW), simile a Bitcoin, per garantire la sicurezza della rete. Tuttavia, questo approccio richiedeva un'elevata quantità di risorse computazionali ed energia. Nel 2022, con l'aggiornamento noto come Ethereum 2.0, la rete è passata al modello *Proof of Stake* (PoS) [20].
Nel modello PoS, i validatori devono mettere in staking una certa quantità di Ether per partecipare alla validazione delle transazioni. Questo sistema riduce il consumo energetico e aumenta la scalabilità della rete. I validatori selezionati ricevono le commissioni di transazione come ricompensa per il loro contributo alla sicurezza della rete, in questo modo si incoraggiano i validatori ad essere onesti, mettendo in gioco i propri Ether, punendo, con la loro perdita, coloro che si comportano in maniera disonesta.

3.5.4 Smart Contract su Ethereum

I *smart contract* rappresentano uno degli elementi chiave della piattaforma Ethereum, differenziandola dalle altre blockchain come Bitcoin.

Il termine *smart contract* fu coniato negli anni '90 dal crittografo Nick Szabo, che li definì come "un insieme di promesse, specificate in forma digitale, che includono protocolli all'interno dei quali le parti si impegnano a rispettare tali promesse" [22]. Tuttavia, nel contesto di Ethereum, essi non sono contratti legali nel senso tradizionale, ma piuttosto programmi immutabili che operano in modo deterministico all'interno della *Ethereum Virtual Machine* (EVM).

Struttura e Funzionamento

I *smart contract* su Ethereum sono semplicemente programmi che vengono eseguiti automaticamente una volta che vengono attivati da una transazione inviata da un *externally owned account* (EOA). Ogni *smart contract* è identificato da un indirizzo univoco, generato al momento della sua creazione. A differenza degli EOA, che sono controllati tramite chiavi private, i *smart contract* non possiedono chiavi private e "si controllano da soli" in base al codice programmato [22].

Caratteristiche Principali

I *smart contract* su Ethereum possiedono le seguenti caratteristiche:

- **Immutabilità**: Una volta distribuito sulla blockchain, il codice di uno *smart contract* non può essere modificato. L'unico modo per alterare la logica di un contratto è distribuirne una nuova versione.
- **Determinismo**: L'esecuzione di un *smart contract* produce sempre lo stesso risultato per tutti i nodi che la eseguono, a condizione che lo stato della blockchain e i dati della transazione siano identici.
- **Esecuzione Condizionata**: I *smart contract* non possono avviarsi da soli, ma devono essere attivati da una transazione proveniente da un EOA. Tuttavia, un contratto può chiamare altri contratti, creando una catena di esecuzioni.

Creazione e Ciclo di Vita

I *smart contract* vengono solitamente scritti in linguaggi ad alto livello come Solidity e successivamente compilati in *bytecode*, che può essere eseguito dalla EVM [22]. Il processo di distribuzione avviene tramite una transazione speciale inviata a un indirizzo di creazione contratti (0x0). Una volta distribuito, un contratto può interagire con altri contratti e accettare transazioni, ma non può essere modificato.

Eliminazione di uno Smart Contract

Sebbene un *smart contract* sia immutabile, è possibile rimuovere il codice e lo stato interno associato utilizzando l'operazione EVM chiamata **SELFDESTRUCT** (precedentemente **SUICIDE**). Questa operazione elimina il contratto dall'indirizzo, ma non rimuove la cronologia delle transazioni passate dalla blockchain [22]. L'uso di **SELFDESTRUCT** è possibile solo se il contratto è stato programmato per supportare tale funzionalità, altrimenti il contratto rimarrà sulla blockchain in modo permanente.

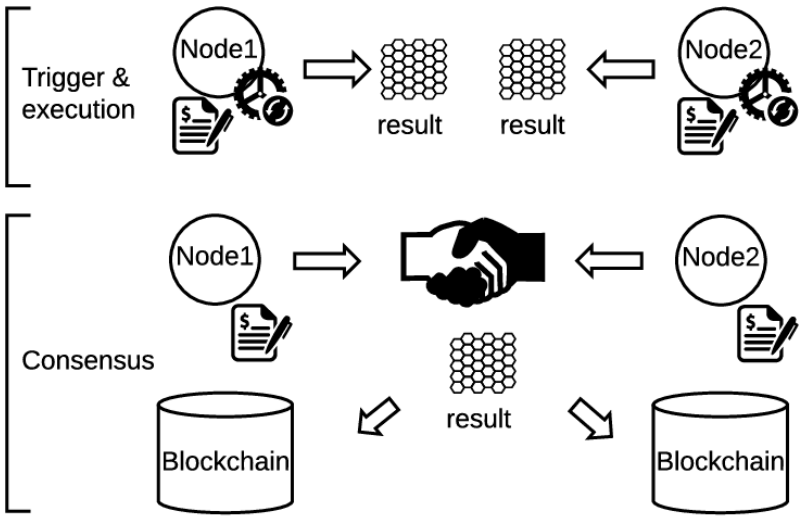


Figura 3.9: Esecuzione di uno Smart Contract su Ethereum [23]

3.5.5 Decentralized Applications (DApps)

Le *Decentralized Applications* (DApps) rappresentano un'evoluzione significativa rispetto alle applicazioni centralizzate tradizionali, in quanto sfruttano la tecnologia blockchain per offrire un ambiente decentralizzato, trasparente e resistente alla censura. Una DApp è essenzialmente un'applicazione che è prevalentemente, se non completamente, decentralizzata, utilizzando contratti intelligenti (*smart contracts*) per gestire la logica di business e una rete peer-to-peer per la comunicazione e l'archiviazione dei dati.

Il **backend** di una DApp è costituito da uno o più *smart contracts*, che gestiscono le logiche di business direttamente sulla blockchain di Ethereum. Questo approccio garantisce che la DApp sia resiliente e sempre disponibile, poiché il suo codice è distribuito su tutti i nodi della rete. A differenza delle applicazioni centralizzate che possono subire interruzioni a causa di guasti del server, una DApp non ha punti di controllo centrali e quindi non soffre di downtime. La trasparenza è garantita dal fatto che il codice degli *smart contract* è pubblico e verificabile da chiunque, ciò aumenta la fiducia degli utenti nell'applicazione. Una volta distribuito sulla rete, il codice diventa immutabile, garantendo che nessuna entità possa modificarlo o censurare l'accesso alla DApp.

L'architettura di una DApp prevede generalmente un'interfaccia **frontend** che utilizza tecnologie web standard (HTML, CSS, JavaScript) per interagire con gli utenti. L'integrazione con la blockchain avviene attraverso librerie come `web3.js`, che permettono di inviare transazioni, gestire chiavi crittografiche e firmare messaggi tramite estensioni del browser come MetaMask. Questa separazione tra backend (su blockchain) e frontend (su browser) consente agli sviluppatori di utilizzare strumenti e framework familiari per creare interfacce utente intuitive.

Tuttavia, la blockchain non è adatta a memorizzare grandi quantità di dati a causa dei costi elevati di gas e dei limiti di capacità dei blocchi. Pertanto, le DApp utilizzano spesso soluzioni di archiviazione off-chain per i dati voluminosi, come il sistema di file distribuito IPFS o Swarm.

In questo modo, è possibile mantenere solo i dati critici e sensibili sulla blockchain, mentre i dati meno importanti possono essere archiviati esternamente, riducendo così i costi di utilizzo della rete.
Le DApp offrono inoltre una forte resistenza alla censura. Finché un utente ha accesso a un nodo Ethereum (o ne esegue uno in proprio), sarà in grado di interagire con la DApp senza che nessuna autorità centrale possa bloccarne l'uso. Questa caratteristica rende le DApp particolarmente utili in contesti in cui la libertà di accesso e la trasparenza sono fondamentali [24].

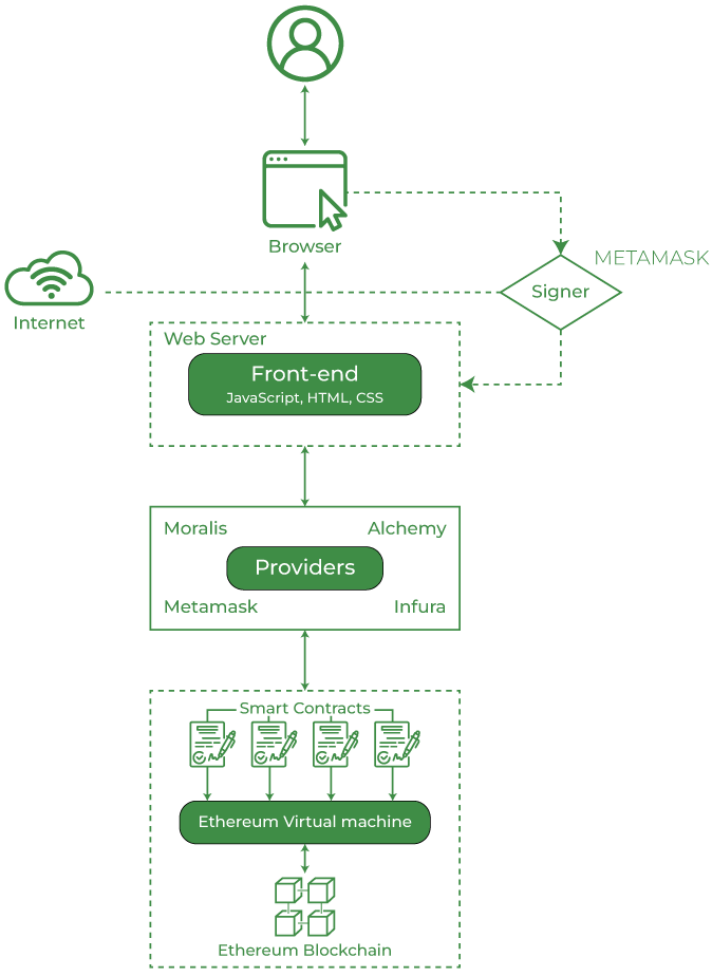


Figura 3.10: Architettura di una DApp [25]

3.6 Applicazione della Blockchain ai Mercati Energetici

In questo sottocapitolo, vengono esplorati i vantaggi dell'utilizzo della blockchain per migliorare la gestione delle offerte e dei processi di bilanciamento nel MGP e nel mercato aFRR.

3.6.1 Miglioramento della Competitività e dell' Efficienza nel MGP

Nel Mercato del Giorno Prima, la blockchain può ridurre i costi operativi eliminando la necessità di intermediari centralizzati. Attraverso la registrazione automatizzata delle offerte su blockchain, gli aggregatori di batterie possono accedere direttamente al mercato, aumentando la competizione e abbassando i prezzi per i consumatori. Inoltre, la decentralizzazione permette un accesso diretto alle risorse e garantisce una maggiore efficienza nel processo di contrattazione. Nella figura 3.11 è possibile vedere il diagramma di flusso con i vari step per sviluppare una simulazione sul mercato del giorno prima tramite l'integrazione della blockchain. Dal diagramma è possibile notare che l'utente deve solo occuparsi della registrazione delle batterie, mentre l'aggregator si occuperà dell'interazione con il mercato, controllando che ogni batteria rispetti una determinata soglia di *State of Charge (SoC)* prima di piazzare una bid che risulterebbe "inutile" nel caso dovesse vincere nel mercato ad asta perchè la batteria non potrebbe essere scaricata sotto lo 0%. Inoltre dopo la chiusura del mercato vi è un periodo breve di annuncio dei risultati dell'asta, in seguito al quale è possibile procedere con il pagamento finale da parte degli acquirenti nei confronti dell'aggregator, fornendo una commissione richiesta da quest'ultimo, e dei proprietari delle batterie le cui bid sono state selezionate durante il corso dell'asta.

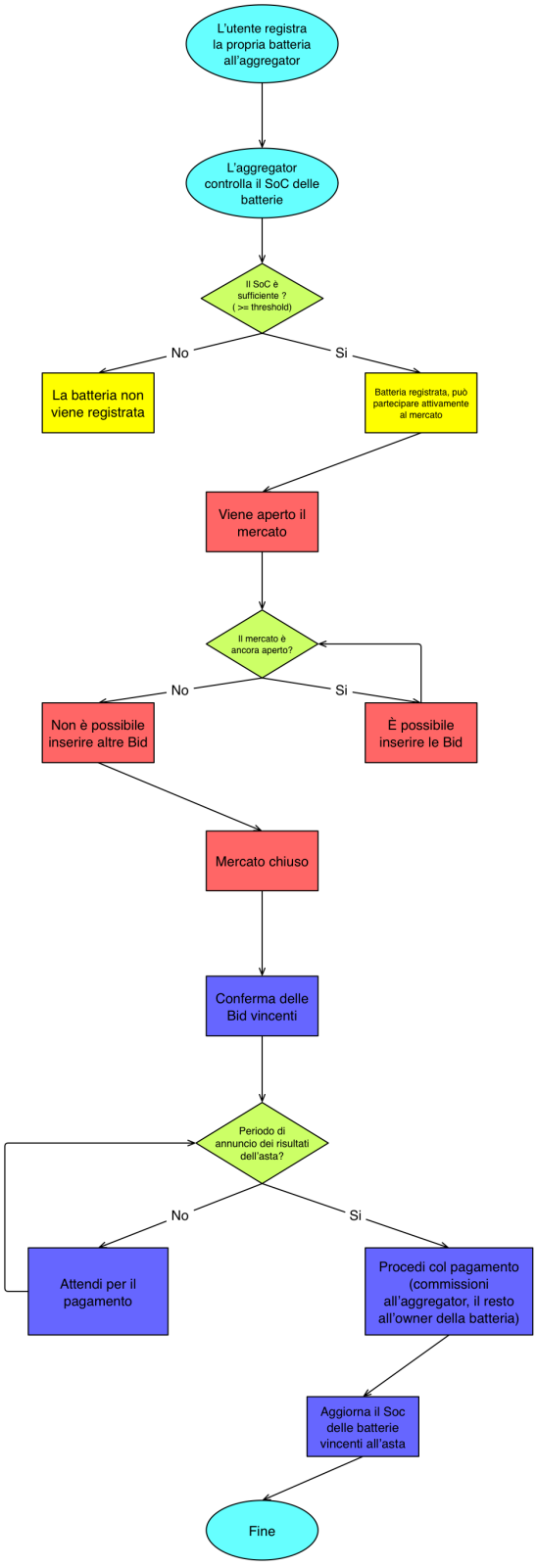


Figura 3.11: Diagramma di flusso della simulazione su MGP

3.6.2 Ottimizzazione della Risposta in Tempo Reale nel Mercato aFRR

Nel mercato aFRR, la blockchain permette una gestione decentralizzata delle risorse di riserva, facilitando il coinvolgimento di nuove risorse flessibili come le batterie dei veicoli elettrici. La trasparenza e l'automazione fornite dagli smart contract consentono al TSO di accettare e attivare le bid più rapidamente, mantenendo l'equilibrio della rete in tempo reale e garantendo la stabilità operativa. Gli smart contract possono inoltre gestire le penalità e le ricompense per incentivare comportamenti corretti, aumentando l'affidabilità complessiva del mercato di bilanciamento. Nella figura 3.12 è possibile vedere il diagramma di flusso con i vari step per sviluppare una simulazione sul mercato di bilanciamento aFRR tramite l'integrazione della blockchain. Anche qui si può osservare che l'aggregatore si occupa di gestire il piazzamento delle bid sulle batterie registrate, inoltre il controllo del SoC questa volta viene fatto con condizioni differenti in base al tipo di riserva di energia richiesta dal TSO. Si parla di riserva positiva quando il TSO richiede l'immissione di energia nella rete, di conseguenza le batterie dovranno scaricare energia, nel caso di riserva negativa invece queste ultime dovranno assorbire energia dalla rete, quindi ricaricarsi. Dopo la chiusura del mercato il TSO seleziona le bid migliori fino a soddisfare la richiesta di riserva, pagando una commissione all'aggregator e il prezzo della bid ai proprietari delle batterie vincenti.

Figura 3.12: Diagramma di flusso della simulazione su aFRR

CAPITOLO 4

Implementation Overview

In questo capitolo, si fornisce una panoramica sull'architettura del sistema decentralizzato per la gestione e la vendita di energia prodotta da batterie di veicoli elettrici nei mercati MGP e aFRR. L'infrastruttura è costruita su una serie di smart contract Solidity che regolano la registrazione delle batterie, il piazzamento delle offerte, l'accettazione e il completamento delle transazioni. Questi contratti sono stati testati e simulati usando Ganache e Hardhat per garantire un ambiente di testing e debugging funzionale. Inoltre è stata sviluppata una simulazione con interfaccia per il mercato aFRR, dove front end e back end interagiscono per mostrare tutto quello che sta accadendo durante varie sessioni di mercato, dalla registrazione delle batterie, al piazzamento delle bid e infine i pagamenti effettuati dal TSO, in modo da permettere all'utente medio di comprendere tramite una *demo* il funzionamento di un mercato dell'energia e dell'interazione che esso può avere con la blockchain.

4.1 Il Ruolo degli Aggregatori

Nel contesto dei mercati energetici decentralizzati, gli aggregatori svolgono un ruolo cruciale nell'ottimizzazione delle risorse energetiche distribuite, in particolare per quanto riguarda le batterie dei veicoli elettrici (EV). Essi agiscono come intermediari tra i proprietari dei veicoli elettrici e i mercati dell'energia, consentendo a questi ultimi di partecipare

attivamente senza doversi interfacciare direttamente con le complessità del mercato.

4.1.1 Funzione e Benefici

Gli aggregatori rappresentano gruppi di veicoli elettrici che possono fornire energia accumulata dalle batterie al mercato dell'energia. Essi raccolgono l'energia disponibile da una pluralità di veicoli e la offrono come un'unica risorsa aggregata. Questo approccio consente di superare la sfida rappresentata dalla partecipazione diretta dei singoli proprietari di veicoli, ottimizzando la gestione delle offerte e dei pagamenti sul mercato.
Attraverso l'utilizzo di smart contract, gli aggregatori possono automatizzare l'intero processo di partecipazione ai mercati MGP e aFRR. Ad esempio, possono registrare dinamicamente le batterie disponibili e inviare offerte all'interno del mercato.

4.2 Architettura degli smart contract

Il sistema è composto da tre contratti principali:

- **Aggregator.sol**: Gestisce la registrazione e il monitoraggio dello stato delle batterie.
- **TSO.sol**: Regola le operazioni di bilanciamento dell'energia nel mercato aFRR.
- **Market.sol**: Consente il piazzamento di offerte e le transazioni nel mercato del giorno prima (MGP).

4.2.1 Smart contract per la gestione dell'aggregator

Il contratto Aggregator è il punto di accesso principale per i proprietari delle batterie e gli aggregatori. Implementa funzioni fondamentali per la registrazione e la gestione delle risorse di accumulo, come descritto di seguito:

- **Registrazione delle batterie**: Tramite la funzione *registerBattery*, ogni utente può registrare la propria batteria specificando capacità e stato di carica (*State of Charge - SoC*). L'evento BatteryRegistered viene emesso per tracciare ogni registrazione avvenuta con successo.
- **Aggiornamento del SoC dopo la vendita**: La funzione *updateBatterySoCAfterSale* aggiorna il SoC delle batterie dopo ogni transazione, distinguendo tra riserve positive e negative per un calcolo accurato della capacità residua.

4.2.2 Smart contract per la gestione del TSO nel mercato aFRR

Il contratto TSO rappresenta l'operatore di sistema di trasmissione responsabile della gestione delle riserve automatiche (aFRR). Questo contratto abilita una serie di funzioni essenziali per il bilanciamento della rete, tra cui:

- **Apertura e chiusura del mercato**: Con *openMarket*, il TSO apre il mercato specificando la quantità di energia necessaria e il tipo di riserva (positiva o negativa) richiesto. La funzione *closeMarket* segna la fine della fase di offerta, impedendo l'aggiunta di ulteriori bid.
- **Piazzamento delle offerte di riserva**: Gli aggregatori piazzano offerte tramite *placeBid*, che verifica il SoC e la capacità della batteria per garantire che l'offerta sia supportata. Viene emesso l'evento BidPlaced per tracciare ogni offerta.
- **Accettazione delle offerte e pagamento finale**: Una volta chiuso il mercato, il TSO può accettare le offerte tramite *acceptBid*, che emette un pagamento per il proprietario della batteria e una commissione all'aggregatore. Il SoC viene poi aggiornato per riflettere la vendita.

4.2.3 Smart contract per la gestione del MGP

l contratto Market regola le operazioni di transazione nel mercato del giorno prima, consentendo la negoziazione anticipata dell'energia. Le funzioni principali sono:

- **Impostazione dei tempi di mercato**: La funzione *setMarketTimes* permette di definire apertura, chiusura e annuncio dei risultati, separando le fasi di piazzamento offerte, accettazione e acquisto.
- **Piazzamento delle offerte**: Gli aggregatori piazzano offerte tramite *placeBid*, che verifica il SoC e la capacità della batteria. Se la capacità è sufficiente, l'offerta viene registrata e l'evento BidPlaced emesso.
- **Accettazione e acquisto delle offerte**: I compratori possono accettare e finalizzare le offerte tramite *purchaseEnergy*. La funzione gestisce il pagamento, calcola la commissione per l'aggregatore e aggiorna il SoC della batteria. Gli eventi PaymentToBatteryOwnerRecorded e PaymentToAggregatorOwnerRecorded tracciano ogni transazione.

4.3 Test degli smart contract

Per garantire il corretto funzionamento del sistema, gli smart contract sono stati testati e simulati usando Ganache e Hardhat, strumenti di sviluppo che permettono la creazione di ambienti di test su blockchain locali.

4.3.1 Ganache

Ganache è uno strumento essenziale per lo sviluppo e il testing di applicazioni blockchain, consentendo agli sviluppatori di simulare una rete blockchain locale direttamente sul proprio computer.
Durante la simulazione, Ganache è stato utilizzato per testare scenari complessi legati alla registrazione delle batterie, al piazzamento delle

offerte nel mercato MGP e aFRR, e all'accettazione delle bid, consentendo di monitorare in tempo reale lo stato dei contratti smart dopo ogni operazione eseguita. La possibilità di vedere istantaneamente i risultati delle transazioni e di analizzare i dati relativi alle offerte ha facilitato l'ottimizzazione del codice degli smart contract.
Ganache offre due versioni: un'interfaccia grafica (UI) e una command line interface (CLI) [26]. La versione UI è stata utilizzata per questo progetto grazie alla sua interfaccia intuitiva, che ha permesso di visualizzare in tempo reale informazioni su account, bilanci, transazioni ed eventi, oltre a includere un *block explorer* integrato per esaminare i dettagli di ciascun blocco e transazione. Questa funzionalità ha semplificato notevolmente il debug, consentendo di analizzare il codice degli smart contract linea per linea, facilitando l'individuazione e la correzione dei bug.
Oltre alla versione UI, la versione CLI offre una soluzione più flessibile per gli sviluppatori che preferiscono lavorare direttamente dal terminale. Essa può essere integrata con altri strumenti di sviluppo e script per automatizzare il testing e il deployment degli smart contract, migliorando ulteriormente l'efficienza del flusso di lavoro.
Indipendentemente dalla versione utilizzata, Ganache ha dimostrato di essere uno strumento fondamentale per accelerare lo sviluppo delle applicazioni decentralizzate (DApp) in ambiente Ethereum, garantendo che gli smart contract funzionino correttamente prima del loro deployment su una rete principale.

4.3.2 Hardhat

Hardhat è uno strumento avanzato e open-source per lo sviluppo su Ethereum, che supporta la creazione, il testing e il deployment di smart contract in modo efficiente e sicuro. Grazie alle sue numerose funzionalità, tra cui un compilatore Solidity integrato, un framework di testing, strumenti di debug avanzati e un sistema di plugin, Hardhat è diventato una scelta popolare per sviluppatori che operano nel settore delle DApp.
Nel contesto di questo progetto, Hardhat è stato utilizzato come per

Ganache per testare e validare le funzionalità chiave del sistema. I test, scritti in Javascript, hanno coperto vari scenari, inclusi quelli limite, come tentativi di registrazione di batterie già registrate, piazzamento di offerte con SoC insufficiente, offerte piazzate fuori dai tempi di mercato e accettazione di bid a mercato chiuso. Grazie a Hardhat, è stato possibile identificare e risolvere bug relativi alla logica e alla sicurezza dei contratti.

Testing e Debugging avanzato con Hardhat Network

Una delle caratteristiche più utili di Hardhat è la *Hardhat Network*, una blockchain locale che consente agli sviluppatori di testare i propri smart contract in un ambiente sicuro e isolato, senza incorrere in costi di gas. Avviando la rete locale con il comando `npx hardhat node`, è possibile simulare una blockchain direttamente in locale, con una serie di account preconfigurati per il testing.
Hardhat fornisce un potente *framework di testing* che permette di scrivere test dettagliati sia in JavaScript che in Solidity. Utilizzando il comando `npx hardhat test`, è stato possibile verificare che ogni funzione del sistema operasse come previsto, riducendo al minimo il rischio di bug durante la fase di sviluppo. Inoltre, Hardhat offre un sistema di deployment efficiente, che consente di scrivere script in JavaScript per automatizzare il deployment dei contratti sulla rete locale o direttamente sulla mainnet.

Funzionalità di Debug avanzato: Stack Traces e Messaggi di Errore

Una delle funzionalità più apprezzate di Hardhat è la capacità di fornire *Solidity Stack Traces* e messaggi di errore automatici. Questi strumenti di debug consentono di identificare rapidamente la causa degli errori, offrendo una traccia dettagliata del call stack che ha portato all'errore. Quando si verifica un errore durante l'esecuzione di uno smart contract, Hardhat genera un messaggio dettagliato che include informazioni sulle funzioni e i parametri coinvolti, semplificando così l'individuazione del

problema. Questo approccio permette di risparmiare tempo prezioso durante la fase di sviluppo, eliminando la necessità di una ricerca manuale degli errori.

Sistema di Plugin e Estendibilità

Un altro aspetto che rende Hardhat particolarmente flessibile è il suo sistema di plugin, che consente agli sviluppatori di estendere le funzionalità dello strumento e integrarlo con altri servizi e strumenti di sviluppo. Ad esempio, è possibile utilizzare plugin per verificare automaticamente i contratti su Etherscan dopo il deployment, semplificando così il processo di verifica. [27]

4.4 Architettura della DApp

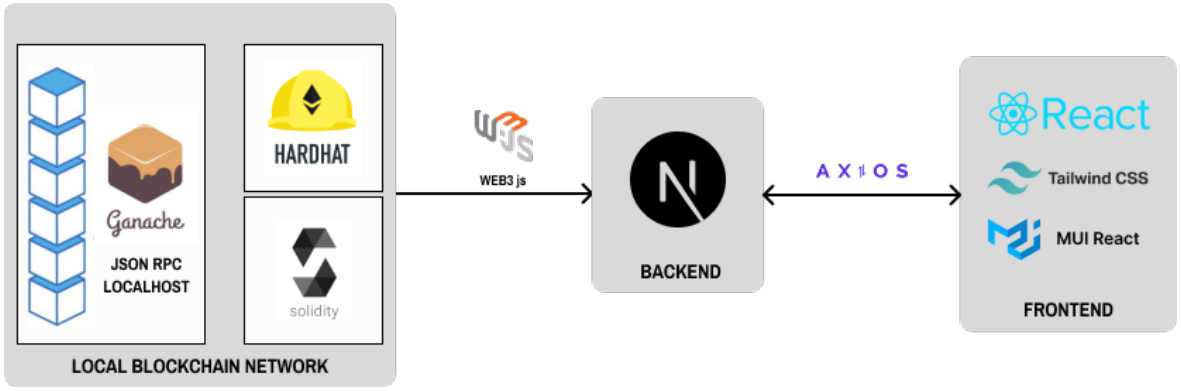


Figura 4.1: Architettura software per la simulazione aFRR

La DApp sviluppata per la simulazione del mercato aFRR è stata costruita utilizzando un'architettura moderna che consente un'interazione fluida con la blockchain Ethereum, offrendo al contempo un'interfaccia utente responsiva. L'applicazione sfrutta Next.js, React con TypeScript, Tailwind CSS, Material-UI (MUI), Axios e Web3.js, integrati con i wallet di Ganache per l'interazione con gli smart contract. In questa sezione verrà illustrata l'architettura generale della DApp, concentrandosi sia sul frontend che sul backend.

4.4.1 Struttura del Frontend con Next.js e React

Il frontend della DApp è stato sviluppato utilizzando **Next.js**, un potente framework basato su **React** che semplifica lo sviluppo di applicazioni web moderne e performanti. Next.js fornisce un'ampia gamma di funzionalità, tra cui *server-side rendering* (SSR) e *static site generation* (SSG), che ottimizzano le prestazioni della DApp e migliorano l'esperienza utente anche durante simulazioni complesse. Grazie al supporto nativo per SSR, le pagine vengono renderizzate direttamente sul server, garantendo tempi di caricamento iniziali più rapidi e migliorando l'ottimizzazione per i motori di ricerca (SEO). Allo stesso tempo, l'uso di SSG consente di pre-renderizzare le pagine durante la fase di build, generando file HTML statici che possono essere distribuiti facilmente su Content Delivery Networks (CDN), assicurando così caricamenti fulminei per l'utente finale [28].

In aggiunta, Next.js include un sistema di **client-side routing** integrato, che permette di costruire applicazioni a pagina singola (SPA) senza la necessità di configurazioni complesse. Questo sistema di routing si basa su una struttura a file: basta creare un file JavaScript nella directory `pages` per aggiungere automaticamente una nuova route, semplificando l'organizzazione del codice. Next.js gestisce inoltre il **code splitting** automatico, suddividendo il codice JavaScript in piccoli blocchi, in modo che gli utenti scarichino solo il codice necessario per la pagina corrente. Questo approccio riduce i tempi di caricamento e migliora le prestazioni complessive.

L'intero progetto è stato scritto in **TypeScript**, che introduce la tipizzazione statica al codice JavaScript, migliorandone la robustezza e il debugging. Grazie alla combinazione di React e Next.js, è stato possibile creare un'interfaccia utente modulare, scalabile e facilmente estendibile, permettendo un rapido sviluppo di nuove funzionalità.

4.4.2 Stilizzazione con Tailwind CSS e Material-UI

Per la stilizzazione della DApp, è stata adottata una combinazione di **Tailwind CSS** e **Material-UI (MUI)**. L'utilizzo di **Tailwind CSS** è stato particolarmente efficace grazie al suo approccio *utility-first*, che consente di creare interfacce personalizzate senza la necessità di scrivere codice CSS personalizzato. Esso promuove un processo di sviluppo più efficiente, consentendo di utilizzare classi utility direttamente all'interno del codice HTML. Questo approccio riduce la complessità del codice, eliminando la necessità di nomi di classi complessi e di file CSS di grandi dimensioni, rendendo il codice più leggero e facilmente modificabile.
Uno dei principali vantaggi di questo framework è la semplificazione nella creazione di design responsivi grazie alle classi utility predefinite che supportano il *mobile-first design*. Ciò consente di realizzare layout mobile-friendly, garantendo un'ottima esperienza utente su dispositivi di varie dimensioni. Inoltre, Tailwind offre un controllo granulare sullo stile, permettendo una personalizzazione precisa e rapida dei componenti, il che accelera il processo di prototipazione [29].
Per garantire un'interfaccia moderna e professionale, è stato integrato anche **Material-UI (MUI)**, che fornisce componenti predefiniti di alta qualità, facilmente adattabili e personalizzabili tramite le classi utility di Tailwind.
MUI è una libreria di componenti open source basata sui principi di UI/UX di *Material Design* sviluppati da Google [30], pensata per creare interfacce utente coerenti e intuitive. Oltre al suo core **Material UI**, il sistema MUI include soluzioni aggiuntive come:

- **Joy UI**: Un'alternativa basata sulle linee guida di *Joy Design*.
- **Base UI**: Segue un approccio *headless*, offrendo componenti privi di stili predefiniti, lasciando maggiore controllo sulla personalizzazione tramite CSS.
- **MUI X**: Una raccolta di componenti avanzati per applicazioni complesse e ricche di dati, come griglie, selettori di data e grafici.

La combinazione di Tailwind e MUI ha permesso di mantenere coerenza stilistica, flessibilità e modularità, migliorando al contempo la velocità di sviluppo dell'interfaccia utente.

4.4.3 Integrazione con la Blockchain tramite Web3.js e Ganache

La comunicazione con gli smart contract è gestita tramite **Web3.js**, una libreria che funge da ponte tra la *blockchain* locale e il frontend / backend.
Web3.js utilizza il protocollo *JSON-RPC* (*JavaScript Object Notation Remote Procedure Call*) per comunicare con la blockchain. Esso è un metodo che consente di inviare richieste a un singolo nodo Ethereum e funziona in modo analogo a una comunicazione tra un server web e un'API JSON, facilitando l'interazione tra il frontend della DApp e i contratti intelligenti. Grazie a Web3.js, è possibile *validare i contratti distribuiti sulla blockchain locale supportata da Ganache* [31].
Ganache viene utilizzato per creare un ambiente di testing che simula una rete Ethereum locale, fornendo account preconfigurati con Ether virtuale, consentendo così di testare le funzionalità della DApp senza incorrere in costi di gas reali. I wallet generati da Ganache permettono agli utenti di connettersi alla blockchain locale, effettuare transazioni e piazzare offerte.

4.4.4 Comunicazione Frontend-Backend con Axios

Per facilitare la comunicazione tra il *frontend* e il *backend*, è stata utilizzata **Axios**, una libreria JavaScript che semplifica il processo di invio di richieste HTTP sia lato client che lato server. Grazie alla sua API intuitiva, Axios rende semplice effettuare operazioni comuni come richieste *GET*, *POST* e *DELETE*, permettendo di interagire con gli *smart contract* e ottenere informazioni in tempo reale sullo stato del mercato aFRR.
Una delle caratteristiche distintive di Axios è la sua capacità di gestire

automaticamente il parsing dei dati JSON, semplificando l'interazione con le API e rendendo il codice più leggibile e manutenibile. Inoltre, supporta funzionalità avanzate come *request / response interceptors* per gestire al meglio le risposte del server, nonché la cancellazione delle richieste e il controllo degli errori, garantendo una user experience fluida. [32]
Le chiamate API tramite Axios sono state utilizzate nella DApp per eseguire diverse operazioni cruciali:

- Ottenere il *State of Charge* (SoC) delle batterie registrate.
- Piazzare nuove offerte sul mercato aFRR.
- Recuperare i dettagli sui pagamenti effettuati dal TSO ai partecipanti.

4.4.5 Backend e API per la Gestione delle Funzionalità

Il **backend** è stato sviluppato per gestire le interazioni con gli smart contract e le funzionalità legate al mercato aFRR. È stato realizzato utilizzando Next.js, con endpoint API che facilitano la comunicazione tra il frontend e la blockchain. Il backend si occupa di:

- Inizializzare e connettersi alla blockchain locale tramite Web3.js.
- Interagire con i contratti per la registrazione delle batterie, il piazzamento delle offerte e la gestione dei pagamenti.
- Gestire la logica di business, come il controllo delle tempistiche di mercato con l'apertura e la chiusura di quest'ultimo.

Le API fornite dal backend sono invocate dal frontend tramite Axios, permettendo di mantenere il codice organizzato e modulare.

4.4.6 Workflow della Simulazione del Mercato aFRR

La DApp guida l'utente attraverso il workflow completo della simulazione del mercato aFRR, suddiviso in tre fasi principali:

1. **Registrazione delle batterie**: L'utente registra contemporaneamente tutte le batterie disponibili per la simulazione tramite un apposito pulsante che invia i dati agli smart contract attraverso le API gestite dal backend.

2. **Simulazione e Piazzamento delle offerte**: Una volta registrate le batterie, si può accedere al mercato e, tramite delle notifiche, si può assistere al piazzamento automatico di varie offerte.

3. **Accettazione e Pagamento finale**: Alla chiusura della sessione, l'utente, impersonando il TSO, può accettare le offerte e i pagamenti vengono gestiti automaticamente grazie allo smart contract TSO.sol.

CAPITOLO 5

Implementation Details

5.1 Smart contract nel dettaglio

5.1.1 Smart Contract Aggregator

Il contratto `Aggregator.sol` è stato sviluppato per gestire le batterie degli utenti all'interno della piattaforma di scambio energetico, considerando sia la partecipazione al mercato del giorno prima sia al mercato aFRR. Il suo obiettivo principale è consentire ai proprietari di batterie di registrare i propri dispositivi, monitorare lo stato di carica (*State of Charge* - SoC) e aggiornare tale stato in base all'energia venduta o acquistata durante le sessioni di mercato. Di seguito viene fornita una spiegazione dettagliata delle strutture dati e delle funzioni principali implementate.

Struttura Dati e Variabili di Stato

Il cuore del contratto è rappresentato dalla struttura `Battery`, che contiene i seguenti campi:

- **capacity**: indica la capacità massima della batteria in kWh.
- **SoC (State of Charge)**: rappresenta lo stato di carica attuale della batteria, espresso in percentuale.
- **isRegistered**: flag booleano che indica se la batteria è stata registrata correttamente nel sistema.

Inoltre, il contratto utilizza le seguenti variabili di stato:

- **batteries**: una mappatura che associa ogni indirizzo di proprietario della batteria a un'istanza della struttura `Battery`. Questa mappa funge da database per tutte le batterie registrate.
- **owner**: indirizzo del proprietario del contratto, tipicamente l'admin dell'aggregatore.
- **commissionRate**: tasso di commissione (in percentuale) applicato dall'aggregatore sulle transazioni effettuate nel mercato.

Eventi

Il contratto definisce un evento:

- **BatteryRegistered**: evento emesso ogni volta che una nuova batteria viene registrata con successo, in modo da rimanere tracciabile sulla blockchain

Funzioni Principali

Di seguito vengono descritte le principali funzioni implementate nel contratto:

Funzione `registerBattery()` Questa funzione consente agli utenti di registrare la propria batteria nel sistema. L'utente deve specificare la capacità e lo stato di carica iniziale della batteria.
Prima di registrare una batteria, la funzione verifica che l'indirizzo chiamante non abbia già registrato una batteria. Se la registrazione ha successo, i dati vengono salvati nella mappa `batteries` e viene emesso l'evento `BatteryRegistered`.

Funzione `updateBatterySoCAfterSale()` Questa funzione permette di aggiornare lo stato di carica (*State of Charge*) di una batteria dopo una transazione di vendita o acquisto di energia.

Essa prende in input l'indirizzo del proprietario della batteria, la quantità di energia venduta/acquistata e un flag che indica se si tratta di una riserva positiva o negativa. In base al tipo di richiesta del mercato possiamo avere:

- **Riserva Positiva:** le batterie devono scaricare, di conseguenza vi deve essere una diminuzione di *State of Charge* attuale della batteria. Per il calcolo del nuovo SoC è stata utilizzata questa formula matematica:

$$\text{newSoC} = \text{SoC}_{\text{batteria}} - \left(\frac{\text{energia venduta}}{\text{capacità della batteria}} \times 100 \right)$$

- **Riserva Negativa:** le batterie devono assorbire energia dalla rete, quindi il SoC attuale aumenta.

$$\text{newSoC} = \text{SoC}_{\text{batteria}} + \left(\frac{\text{energia venduta}}{\text{capacità della batteria}} \times 100 \right)$$

La funzione viene richiamata nei contratti di gestione dei due mercati durante la fase di pagamento delle bid. Inoltre il calcolo del nuovo SoC viene effettuato considerando il caso ideale, altrimenti avremmo dovuto moltiplicare il tutto per un parametro di efficenza che tiene conto della perdite di energia di una batteria durante il processo di carica / scarica.

5.1.2 Smart Contract Market (MGP) & Test

Lo smart contract `Market.sol` è stato sviluppato per gestire il funzionamento del mercato del giorno prima (Day-Ahead Market). Questo contratto consente agli aggregatori di piazzare le bid sulle batterie elettriche a disposizione e agli acquirenti di accettare tali offerte, con la gestione automatica dei pagamenti e delle commissioni.

Strutture Dati

Il contratto utilizza diverse strutture dati per gestire il flusso del mercato:

- Struttura `Bid`: rappresenta un'offerta che include informazioni come l'indirizzo del proponente (`bidder`), il proprietario della batteria (`batteryOwner`), la quantità di energia offerta (`amount`) in kWh, il prezzo per kWh (`price`), uno stato di selezione (`isSelected`), e l'indirizzo dell'acquirente che ha accettato l'offerta (`acceptedBy`).
- Mapping `bids`: una mappatura che collega un ID univoco a ciascuna offerta (`bidId`) per facilitare il tracciamento delle offerte piazzate.
- Mapping `aggregators`: mappa l'indirizzo del proprietario della batteria all'indirizzo dell'aggregatore responsabile, garantendo che ogni offerta sia associata al corretto aggregatore (caso in cui abbiamo più aggregatori).

Funzionalità Principali

Di seguito vengono descritte le principali funzioni implementate nel contratto `Market.sol`.

- Gestione dei Tempi di Mercato: La funzione `setMarketTimes` permette all'amministratore del mercato di definire i tempi di apertura (`marketOpenTime`), chiusura (`marketCloseTime`), e annuncio dei risultati (`resultsAnnouncementTime`). Ciò consente di gestire le diverse fasi del mercato in modo automatizzato.
- Registrazione degli Aggregatori: La funzione `setAggregator` associa i proprietari delle batterie agli indirizzi degli aggregatori. Questo consente di stabilire una relazione tra le batterie e gli aggregatori che gestiscono le offerte sul mercato.
- Piazzamento delle Offerte: Durante il periodo di apertura del mercato, gli aggregatori possono utilizzare la funzione `placeBid` per piazzare un'offerta. La funzione verifica che la batteria abbia un livello di carica (*State of Charge*, SoC) sufficiente. Le offerte vengono registrate nella struttura dati `bids`.

- Accettazione delle Offerte: Una volta chiuso il mercato, è possibile accettare le offerte tramite la funzione `acceptBid`. Questa funzione cambia lo stato `isSelected` in *true* e associa l'indirizzo dell'acquirente che ha accettato l'offerta.
- Acquisto dell'Energia: Durante la fase di annuncio dei risultati, dallo smart contract viene chiamata la funzione `purchaseEnergy` per confermare l'acquisto delle offerte selezionate. I pagamenti vengono gestiti automaticamente:
  - Viene calcolata una commissione da versare all'aggregatore. La parte restante del pagamento viene trasferita al proprietario della batteria.
  - Viene aggiornato il SoC della batteria dopo l'acquisto, utilizzando la funzione `updateBatterySoCAfterSale` dello smart contract `Aggregator.sol`.

Test in Javascript

Per garantire la correttezza del contratto `Market.sol`, sono stati sviluppati diversi test utilizzando il framework `Hardhat`. I test verificano che le varie funzioni del contratto operino come previsto, simulando scenari reali.

Registrazione batterie Prima di piazzare le offerte, è necessario registrare le batterie. Di seguito uno snippet che verifica la corretta registrazione delle batterie da parte dei proprietari tramite il contratto `Aggregator.sol`:

```
1  await aggregatorContract1.connect(owner1Signer)
      .registerBattery(100, 80);
2  await aggregatorContract2.connect(owner4Signer)
      .registerBattery(120, 70);
3  const battery1 = await
      aggregatorContract1.batteries(owner1);
4  const battery3 = await
      aggregatorContract2.batteries(owner3);
```

```
5     assert.equal(battery1.capacity, 100, "Capacità della
          batteria 1 non corretta");
6     assert.equal(battery1.SoC, 80, "SoC della batteria 1
          non corretto");
7     assert.equal(battery4.capacity, 120, "Capacità della
          batteria 4 non corretta");
8     assert.equal(battery4.SoC, 70, "SoC della batteria 4
          non corretto");
```

Listing 5.1: Registrazione batterie (MGP)

I test sopra verificano che le batterie vengano registrate correttamente con la capacità e il SoC specificati.

Piazzamento delle bid I test assicurano che le offerte vengano piazzate correttamente durante il periodo di apertura del mercato da parte degli aggregatori:

```
1     await
          market.connect(aggregator1Signer).placeBid(owner1,
          50, 10);
2     await
          market.connect(aggregator2Signer).placeBid(owner4,
          60, 8);
3
4     const bid0 = await market.bids(0);
5     assert.equal(bid0.amount, 50, "Quantità dell'offerta
          con id 0 errata");
6     assert.equal(bid0.price, 10, "Prezzo dell'offerta con
          id 0 errato");
7     assert.equal(bid3.amount, 60, "Quantità dell'offerta
          con id 0 errata");
8     assert.equal(bid3.price, 8, "Prezzo dell'offerta con
          id 0 errato");
```

Listing 5.2: Piazzamento delle bid (MGP)

Questo test conferma che le offerte vengono registrate con i valori corretti per quantità e prezzo.

Accettazione delle bid Dopo la chiusura del mercato, gli acquirenti possono accettare le offerte:

```
1     await market.connect(buyerSigner).acceptBid(0);
2     await market.connect(buyerSigner).acceptBid(3);
3     const bid0 = await market.bids(0);
4     const bid3 = await market.bids(3);
5     assert.equal(bid0.isSelected, true, "Offerta 0 non
         accettata correttamente");
6     assert.equal(bid3.isSelected, true, "Offerta 3 non
         accettata correttamente");
```

Listing 5.3: Accettazione delle bid (MGP)

Il test verifica che l'offerta venga accettata e che lo stato di `isSelected` venga aggiornato.

Pagamento finale Durante la fase di annuncio dei risultati, vengono finalizzati gli acquisti:

```
1     const price0 = bid0.price;
2     const price3 = bid3.price;
3     const tx0 = await market.purchaseEnergy(0, { value:
         price0 });
4     const tx3 = await market.purchaseEnergy(3, { value:
         price3 });
5
6     const battery1 = await
         aggregatorContract1.batteries(owner1);
7     const battery4 = await
         aggregatorContract2.batteries(owner4);
8     assert.equal(battery1.SoC.toNumber(), 30, "SoC della
         batteria 1 non aggiornato correttamente");
9     assert.equal(battery4.SoC.toNumber(), 22, "SoC della
         batteria 4 non aggiornato correttamente");
10
11    const events0 = receipt0.events;
12    assert(Array.isArray(events0), "events0 should be an
         array");
13    assert(
14      events0.some(
15        (e) =>
16          e.event === "PaymentToAggregatorOwnerRecorded" &&
17          e.args.bidId.toNumber() === 0 &&
18          e.args.aggregator === aggregator1
19      ),
```

```
20        "Payment to aggregatorOwner1 not recorded"
21      );
22      assert(
23        events0.some(
24          (e) =>
25            e.event === "PaymentToBatteryOwnerRecorded" &&
26            e.args.bidId.toNumber() === 0 &&
27            e.args.batteryOwner === owner1
28        ),
29        "Payment to batteryOwner1 not recorded"
30      );
```

Listing 5.4: Pagamento finale (MGP)

Il test verifica che il pagamento venga gestito correttamente tramite gli events emessi dallo smart contract e che il SoC della batteria venga aggiornato dopo l'acquisto.

5.1.3 Smart Contract TSO

Il contratto `TSO` (*Transmission System Operator*) rappresenta uno degli elementi chiave della DApp sviluppata per simulare il mercato dell'*automatic Frequency Restoration Reserve* (aFRR). Questo smart contract gestisce il processo di piazzamento delle offerte da parte degli aggregatori di batterie, selezione delle offerte migliori da parte del TSO, e distribuzione dei pagamenti ai proprietari delle batterie partecipanti. Di seguito, vengono illustrate le strutture dati e le principali funzioni che lo compongono.

Struttura Dati

Il contratto utilizza la struttura `Bid` per rappresentare l'offerta piazzata dall'aggregator su una determinata batteria. Essa include:

- `batteryOwner`: l'indirizzo del proprietario della batteria.
- `amount`: la quantità di energia offerta, espressa in kWh.
- `price`: il prezzo dell'energia offerta, espresso in *wei* per kWh.
- `isSelected`: un flag che indica se l'offerta è stata accettata.

Le offerte sono memorizzate in una mappa (`mapping`) con una chiave numerica che rappresenta l'`ID` dell'offerta. Inoltre, il contratto mantiene variabili di stato per gestire l'apertura e la chiusura del mercato, la richiesta di energia che il TSO deve soddisfare e il totale di energia selezionata dalle offerte.

Funzionalità Principali

Il contratto `TSO` fornisce una serie di funzioni per gestire l'intero ciclo di vita del mercato aFRR. Di seguito, sono descritte le funzionalità principali:

Apertura del Mercato La funzione `openMarket()` permette l'apertura del mercato, specificando la quantità di energia richiesta (`requiredEnergy`) e il tipo di riserva (`isPositiveReserve`). Quando il mercato è aperto, i partecipanti possono iniziare a piazzare le loro offerte. L'evento `MarketOpened` viene emesso per notificare l'apertura del mercato.

Piazzamento delle Offerte Gli aggregatori di batterie possono piazzare le loro offerte utilizzando la funzione `placeBid()`. Ogni offerta include:

- La quantità di energia in kWh.
- Il prezzo per MWh, che viene convertito in *wei* per kWh.

Prima di piazzare un'offerta, il contratto verifica che:

- La batteria abbia sufficiente *State of Charge* (SoC) per soddisfare l'importo offerto, nel caso di riserva positiva.
- La batteria non sia completamente carica, nel caso di riserva negativa.

Le offerte vengono memorizzate nella mappa `bids` e l'evento `BidPlaced` viene emesso per confermare l'offerta.

Chiusura del Mercato La funzione `closeMarket()` consente di chiudere il mercato, impedendo ulteriori piazzamenti di offerte. Una volta chiuso, il TSO può procedere con la selezione delle offerte migliori.

Accettazione delle Offerte Il TSO può accettare un'offerta utilizzando la funzione `acceptBid()`. Durante questa fase:

- Viene verificato che l'energia totale selezionata non ecceda il fabbisogno energetico (`requiredEnergy`).
- L'offerta selezionata viene marcata come accettata.
- Il *State of Charge* della batteria viene aggiornato tramite il contratto `Aggregator`.
- Viene emesso l'evento `BidSelected` per notificare l'accettazione dell'offerta.

Gestione dei Pagamenti Una volta accettata un'offerta, il TSO esegue i pagamenti utilizzando la funzione `processPayment()`. La logica di pagamento include:

- Calcolo della commissione per l'aggregatore basato sul `commissionRate` definito nel contratto `Aggregator`.
- Trasferimento del pagamento netto al proprietario della batteria.
- Trasferimento della commissione all'aggregatore.

Gli eventi `PaymentToBatteryOwnerRecorded` e `PaymentToAggregatorOwnerRecorded` vengono emessi per registrare i dettagli dei pagamenti.

Logica di Business Il contratto `TSO` incorpora diverse verifiche per garantire la correttezza del mercato:

- Utilizzo di `modifier` come `onlyTsoAdmin` per limitare l'accesso ad alcune funzioni critiche al solo amministratore del TSO.
- Verifica che il mercato sia aperto (`onlyWhenMarketOpen`) prima di consentire il piazzamento delle offerte.
- Controllo sulle condizioni energetiche delle batterie che partecipano e sulla correttezza dei pagamenti.

5.2 Analisi Back-End aFRR

5.2.1 Inizializzazione Web3 e Connessione Ganache

La prima parte del file `web3.ts` riguarda la configurazione e la connessione a Ganache tramite l'URL fornito nel file `.env`. In questo modo, la DApp è in grado di interagire con una blockchain locale per testare le funzionalità senza costi di gas reali.

```
1 import Web3 from "web3";
2 import dotenv from "dotenv";
3
4 dotenv.config();
5
6 const ganacheUrl = process.env.GANACHE_URL;
7 const web3 = new Web3(new
     Web3.providers.HttpProvider(ganacheUrl));
```

Listing 5.5: Connessione a Ganache e configurazione di Web3

Il codice in 5.5 inizializza la libreria `Web3` utilizzando l'URL del nodo Ganache specificato nel file `.env`. Questo permette alla DApp di connettersi al nodo Ethereum locale per inviare transazioni e interagire con gli smart contract.

Gestione degli Account tramite Chiavi Private

La sezione successiva si occupa del caricamento degli account utilizzando le chiavi private di Ganache, le quali vanno memorizzate dall'utente nel file `.env`. Questo consente di interagire con la blockchain senza dover utilizzare un wallet esterno.

```
1 const accountsData = [
2   { name: "Aggregator Admin", account:
       process.env.AGGREGATOR_ADMIN_ACCOUNT, privateKey:
       process.env.AGGREGATOR_ADMIN_PRIVATE_KEY },
3   { name: "TSO Admin", account:
       process.env.TSO_ADMIN_ACCOUNT, privateKey:
       process.env.TSO_ADMIN_PRIVATE_KEY },
4 ];
5
```

```
accountsData.forEach((acc) => {
  if (acc.privateKey) {
    web3.eth.accounts.wallet.add(acc.privateKey);
  }
});
```

Listing 5.6: Sblocco degli account con chiavi private

Il codice in 5.6 aggiunge gli account all'istanza `web3.eth.accounts.wallet` utilizzando le chiavi private, permettendo così l'esecuzione di transazioni programmate direttamente dalla DApp senza interazione manuale.

Creazione delle Istanze degli Smart Contract

La parte successiva del file riguarda l'istanza degli smart contract `Aggregator` e `TSO` che sono stati precedentemente distribuiti sulla blockchain locale. Gli indirizzi dei contratti vengono recuperati dalle variabili di ambiente.

```
import AggregatorContract from
   "./contracts/Aggregator.json";
import TSOContract from "./contracts/TSO.json";

const aggregatorContractAddress =
   process.env.AGGREGATOR_CONTRACT_ADDRESS;
const tsoContractAddress =
   process.env.TSO_CONTRACT_ADDRESS;

const aggregatorContract = new web3.eth.Contract(
  AggregatorContract.abi,
  aggregatorContractAddress
);

const tsoContract = new web3.eth.Contract(
  TSOContract.abi,
  tsoContractAddress
);
```

Listing 5.7: Inizializzazione degli smart contract

Il codice in 5.7 crea due istanze di contratti utilizzando gli indirizzi e gli `ABI` degli smart contract. Questo permette alla DApp di chiamare le funzioni definite negli smart contract direttamente dal backend.

Funzione per Ottenere gli Account

Una funzione aggiuntiva `getAccounts` è stata implementata per restituire una lista di account caricati, utile per le API.

```
export const getAccounts = () => {
  const accounts: { [key: string]: { address: string |
     undefined; privateKey: string | undefined } } = {};

  accountsData.forEach((acc) => {
    accounts[acc.name] = { address: acc.account,
       privateKey: acc.privateKey };
  });

  return accounts;
};
```

Listing 5.8: Funzione per ottenere gli account

La funzione in 5.8 restituisce un oggetto contenente gli account e le rispettive chiavi private, facilitando l'accesso da altre parti del codice.

5.2.2 API per interagire con gli Smart Contract

Di seguito, vengono riportate le principali API sviluppate:

API per la registrazione delle batterie

L'endpoint registerBattery.ts consente agli utenti di registrare le batterie nel contratto Aggregator. Questo processo coinvolge la verifica dell'indirizzo del proprietario e l'interazione con la funzione registerBattery dello smart contract.

```
const tx = await
   aggregatorContract.methods.registerBattery(capacity,
   SoC).send({ from: address, gas: "3000000"
   }).on("receipt", async (tx) => { console.log(tx); });
```

Listing 5.9: Funzione di registrazione delle batterie

Descrizione: La funzione invia una transazione alla blockchain per registrare una batteria con la capacità (*capacity*) e lo stato di carica

(SoC) forniti. Se la transazione ha successo, viene generato un hash di transazione che viene utilizzato per confermare l'operazione.

Gestione dei dati: Una volta che la batteria è registrata con successo, i dettagli di quest'ultima vengono salvati nel file registeredBatteries.json per tenerne traccia localmente.

```
registeredBatteries.push(newBattery);
fs.writeFileSync( filePath,
   JSON.stringify(registeredBatteries, null, 2), "utf-8" );
```

Listing 5.10: Scrittura dei dati della batteria registrata

API per il piazzamento delle offerte

L'endpoint placeBid.ts permette agli utenti di piazzare offerte per la vendita di energia tramite il contratto TSO. Gli utenti specificano l'indirizzo del proprietario della batteria, la quantità di energia offerta (*amountInKWh*), e il prezzo per MWh (*pricePerMWh*).

```
const tx = await
   tsoContract.methods.placeBid(batteryOwner, amountInKWh,
   pricePerMWh).send({ from: aggregatorAdminAccount
   }).on("receipt", (tx) => { console.log(tx); });
```

Listing 5.11: Piazzamento di un'offerta

Descrizione: La funzione invia una transazione per piazzare un'offerta. Viene calcolato il prezzo totale convertendo il prezzo da MWh a kWh, e la nuova offerta viene salvata nel file placedBids.json:

```
placedBids.push(newBid);
fs.writeFileSync( bidsFilePath, JSON.stringify(placedBids,
   null, 2), "utf-8" );
```

Listing 5.12: Salvataggio dell'offerta piazzata

Incremento del bidId: Per ogni nuova offerta, viene generato un ID univoco incrementale utilizzando lastBidId.json:

```
let newBidId = JSON.parse(fs.readFileSync(lastBidIdPath,
    "utf-8")).lastBidId + 1;
fs.writeFileSync( lastBidIdPath, JSON.stringify({
    lastBidId: newBidId }, null, 2), "utf-8" );
```

Listing 5.13: Aggiornamento del bidId

Questo bidId serve per poter riprendere il counting delle bid piazzate dopo la fine di una sessione e l'inizio di una nuova dall'ultimo valore memorizzato, in modo da rimanere sincronizzato col mapping *bids* nello smart contract TSO.

API per l'accettazione delle offerte

L'endpoint acceptBid.ts permette al TSO di accettare le offerte piazzate dai proprietari delle batterie. Quando un'offerta viene accettata, viene effettuato un pagamento al proprietario della batteria e viene aggiornato il SoC tramite il contratto Aggregator.

```
const tx = await
    tsoContract.methods.acceptBid(bidId).send({ from:
    tsoAdminAccount, value: totalPrice, gas: "3000000"
    }).on("receipt", async (tx) => { console.log(tx); });
```

Listing 5.14: Accettazione di un'offerta

Descrizione: La funzione invia una transazione per accettare un'offerta specificata dal parametro *bidId*. Se la transazione ha successo, l'offerta accettata viene salvata in acceptedBids.json:

```
acceptedBids.push(acceptedBid);
fs.writeFileSync( acceptedBidsPath,
    JSON.stringify(acceptedBids, null, 2), "utf-8" );
```

Listing 5.15: Salvataggio dell'offerta accettata

Aggiornamento dello stato di carica (SoC): Dopo che un'offerta è stata accettata, viene aggiornato il SoC della batteria tramite il metodo updateBatterySoCAfterSale dello smart contract Aggregator.

5.3 Interfaccia

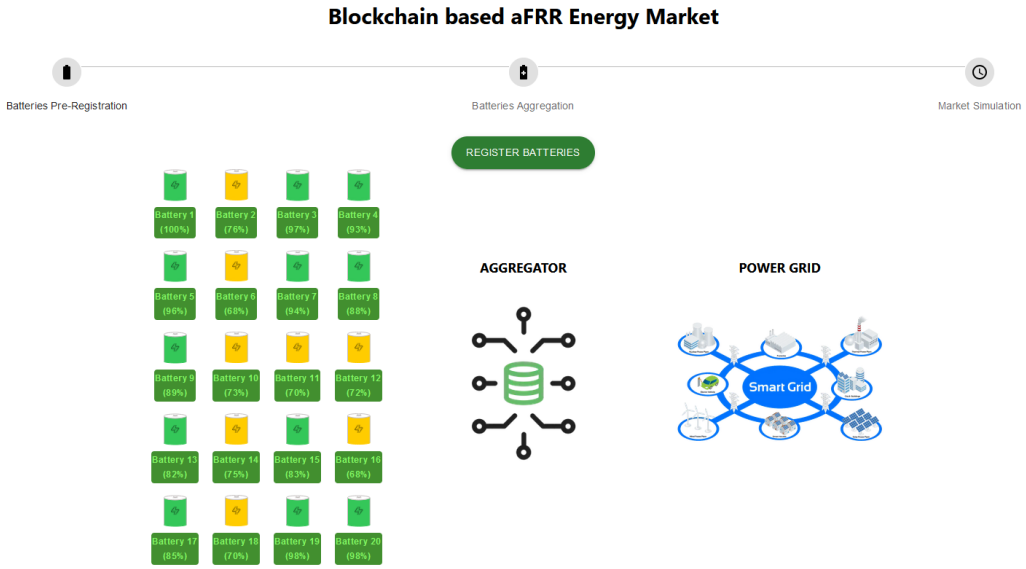


Figura 5.1: Prima pagina della simulazione

Nel momento in cui viene fatto runnare il programma si apre la prima pagina della simulazione (vedi fig. 5.1), dove vengono mostrati gli attori principali del mercato aFRR: le batterie sulla sinistra (verde: SoC >= 80, giallo: 60 <= SoC < 80, arancione: 30 <= SoC < 60, rosso: 0 < SoC < 30, senza colore: SoC = 0), l'aggregatore al centro e la Power Grid sulla destra. Nel momento in cui viene premuto il pulsante *Register Batteries* lo stepper si aggiorna su *Battery Aggregation* e si finisce nella pagina in fig. 5.2.

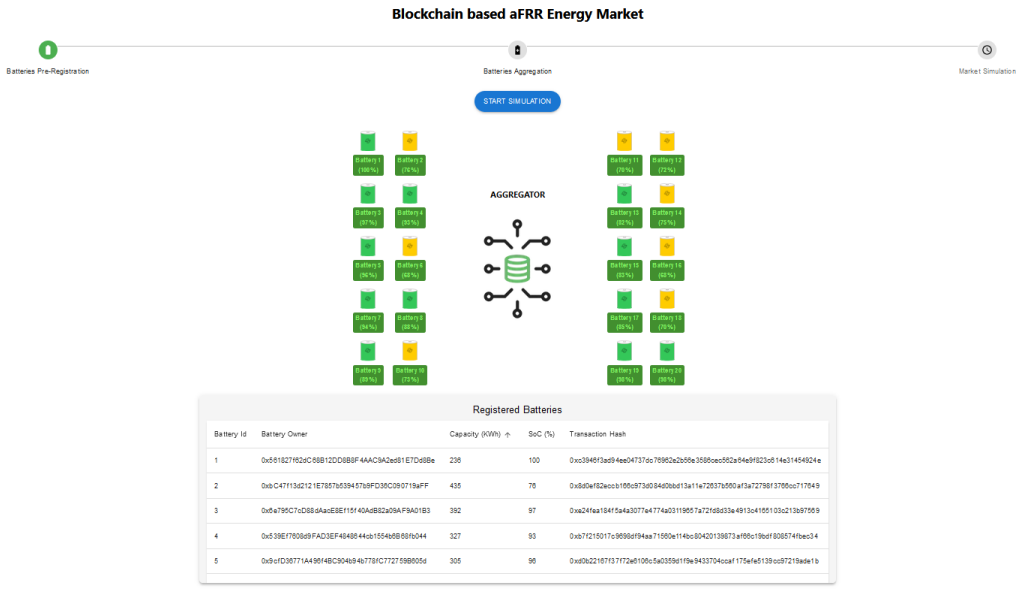


Figura 5.2: Seconda pagina della simulazione: Aggregazione delle batterie

Qui le batterie vengono posizionate vicino l'aggregatore per indicare che sono state tutte registrate, infatti nella parte inferiore della pagina si può notare la presenza di una tabella che raccoglie i dati di ogni singola batteria che è appena stata registrata. Essa è costituita da 4 campi:

- **Battery Owner**: indirizzo del proprietario della batteria.
- **Capacity**: capacità della batteria in KWh.
- **SoC**: stato di carica iniziale della batteria.
- **Transaction Hash**: identificativo della transazione sulla blockchain.

Inserendo l'hash su ganache è possibile vedere i dettagli della transazione, come ad esempio il costo in termini di gas.
Il prossimo step è l'avvio della sessione di mercato, quindi cliccando su *Start Simulation* si caricherà la pagina in fig. 5.3.

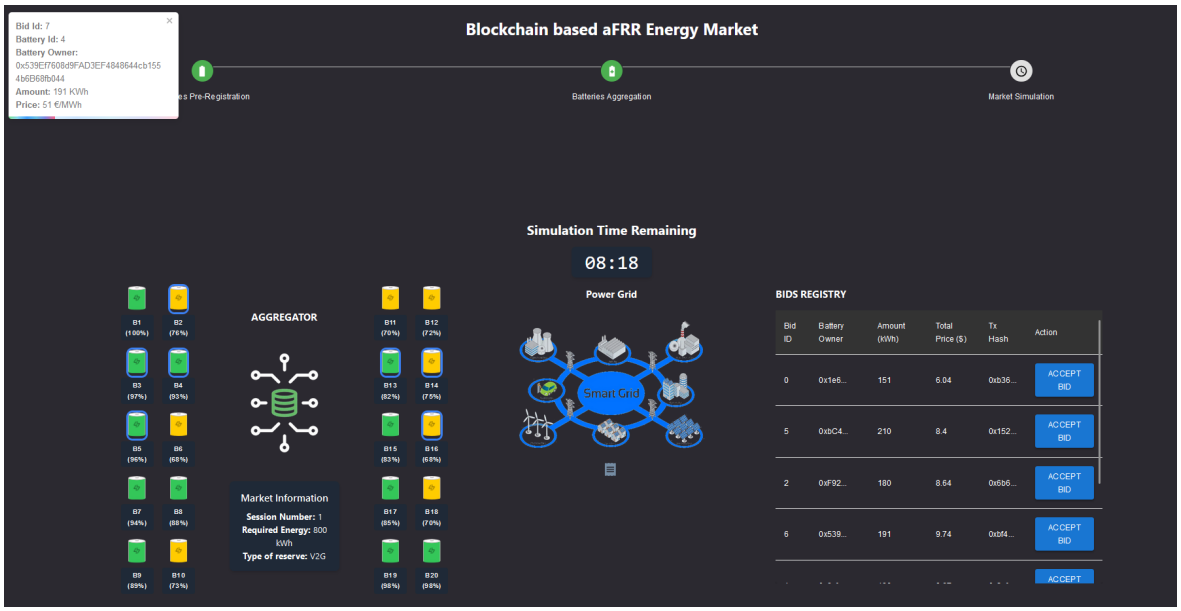


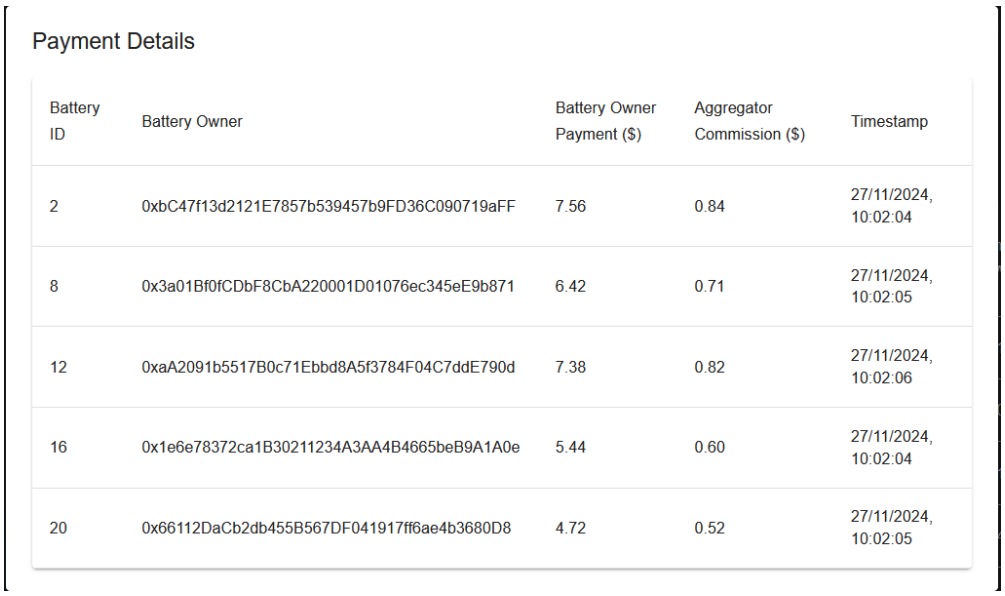
Figura 5.3: Terza pagina della simulazione: Apertura del mercato

In questa pagina rivediamo tutti gli attori della simulazione, compresa la Power Grid con il TSO che si occupa dell'apertura del mercato con la richiesta energetica da soddisfare specificata sotto l'aggregatore e il tipo di riserva che viene indicato come **V2G** nel caso di riserva positiva perchè la batteria sta cedendo energia alla rete, mentre nel caso di riserva negativa si parla di **G2V** perchè la batteria acquisisce energia dalla rete caricandosi. Inoltre vi è la fase di accettazione delle bid.
Nel momento in cui si avvia la sessione il timer parte da 15:00 e durante questi minuti (a velocità 10x per far durare la simulazione meno di 15 minuti effettivi) vengono piazzate automaticamente le bid, infatti in alto a sinistra è possibile notare una notifica che segnala questa azione da parte dell'aggregatore, inoltre la batteria su cui è stata piazzata la bid acquisisce un contorno di colore blu.
Quando il timer scade, viene chiuso il mercato e il TSO può accettare le offerte migliori cliccando il pulsante *Accept Bid* accanto a ciascuna bid nella tabella. Quest'ultima è ordinata per miglior rapporto *amount/price* ed è costituita da 5 campi (esclusa l'action):

- **Bid Id**: l'Id della bid piazzata. Esso è sequenziale.
- **Battery Owner**: indirizzo del proprietario della batteria.

- **Amount**: energia offerta in KWh.
- **Total price**: prezzo di vendita dell'energia.
- **Transaction Hash**: identificativo della transazione sulla blockchain.

Dopo aver accettato le offerte necessarie per soddisfare la richiesta energetica, cliccando sull'icona della ricevuta presente sotto il TSO si può vedere la tabella in fig. 5.4.



Payment Details

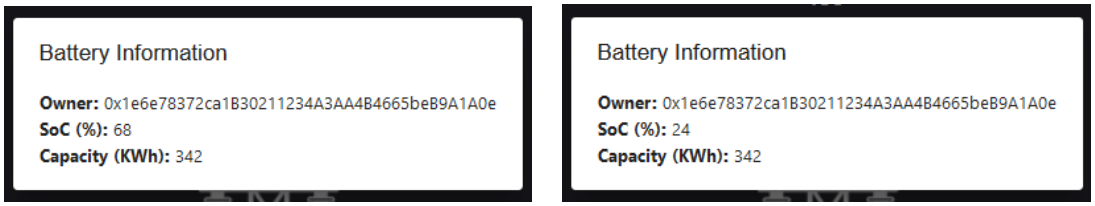
Battery ID	Battery Owner	Battery Owner Payment ($)	Aggregator Commission ($)	Timestamp
2	0xbC47f13d2121E7857b539457b9FD36C090719aFF	7.56	0.84	27/11/2024, 10:02:04
8	0x3a01Bf0fCDbF8CbA220001D01076ec345eE9b871	6.42	0.71	27/11/2024, 10:02:05
12	0xaA2091b5517B0c71Ebbd8A5f3784F04C7ddE790d	7.38	0.82	27/11/2024, 10:02:06
16	0x1e6e78372ca1B30211234A3AA4B4665beB9A1A0e	5.44	0.60	27/11/2024, 10:02:04
20	0x66112DaCb2db455B567DF041917ff6ae4b3680D8	4.72	0.52	27/11/2024, 10:02:05

Figura 5.4: Tabella dei pagamenti

Oltre l'indirizzo del proprietario della batteria abbiamo il Battery Id, il timestamp che indica data e orario in cui è stato effettuato il pagamento e due campi che indicano il pagamento finale:

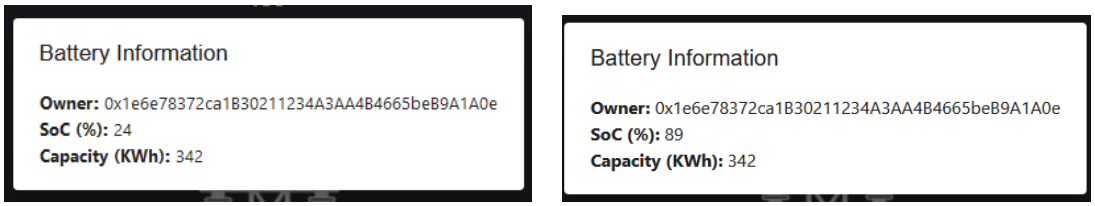
- **Battery Owner Payment**: il pagamento finale al proprietario della batteria.
- **Aggregator Commission**: la commissione dovuta all'aggregatore.

Inoltre, cliccando su ogni batteria è possibile vederne le informazioni. Quando viene accettata una bid, la batteria corrispondente subisce l'aggiornamento in tempo reale del SoC:



(a) *Soc iniziale* (b) *SoC aggiornato (1a sessione)*

Figura 5.5: Esempio di aggiornamento del SoC (Riserva positiva)



(a) *Nuovo Soc (1a sessione)* (b) *SoC aggiornato (2a sessione)*

Figura 5.6: Esempio di aggiornamento del SoC (Riserva negativa)

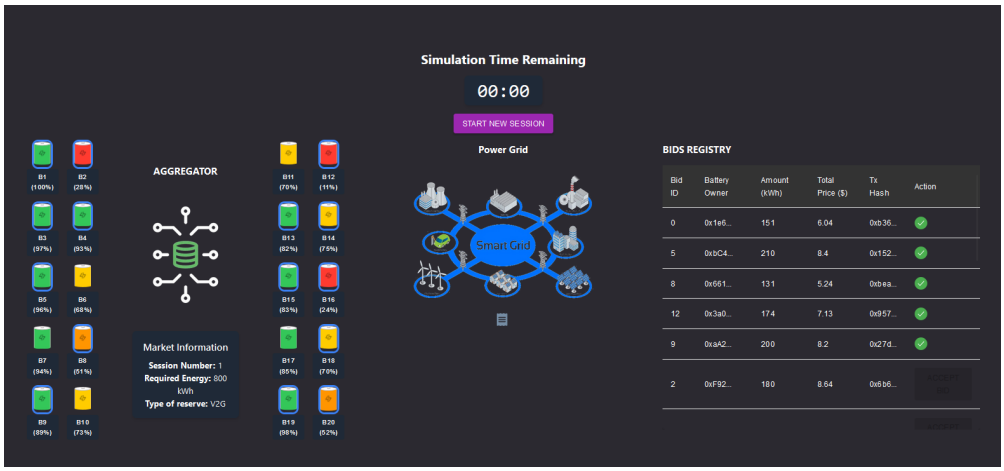


Figura 5.7: Esempio di aggiornamento visivo del Soc tramite il colore delle batterie

Per avvisare l'utente del raggiungimento del fabbisogno energetico viene mostrato questo banner nella parte alta della pagina:

Figura 5.8: Conferma del raggiungimento del fabbisogno energetico

Nel momento in cui si raggiunge questa condizione è possibile cominciare una nuova sessione di mercato, quindi il timer riparte da capo, le tabelle vengono azzerate e le batterie si aggiornano col nuovo colore in base al nuovo valore di SoC. Nelle seguenti tabelle vengono mostrati i dati principali delle prime due sessioni (una con riserva positiva e l'altra negativa):

Parametro	**Valore Simulato**	**Unità**
Numero di batterie registrate	20	-
Numero di bid piazzate	15	-
Richiesta energetica	800	kWh
Volume totale offerto	2843	kWh
Volume totale selezionato	866	kWh
Prezzo medio (Bid selezionate)	7,00	$

Tabella 5.1: Dati simulazione con riserva positiva

Parametro	**Valore Simulato**	**Unità**
Numero di batterie registrate	20	-
Numero di bid piazzate	15	-
Richiesta energetica	1200	kWh
Volume totale offerto	2618	kWh
Volume totale selezionato	1331	kWh
Prezzo medio (Bid selezionate)	19,21	$

Tabella 5.2: Dati simulazione con riserva negativa

La simulazione va avanti per 4 sessioni, al termine della quarta compare il pulsante *See Summary Graphs* che apre un modal dove vengono mostrati i grafici in fig. 5.9.

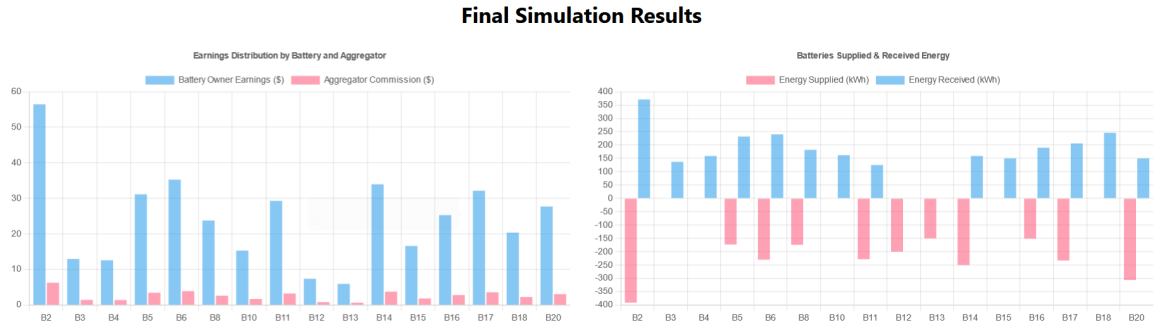


Figura 5.9: Il grafico a sinistra mostra i guadagni finali, quello a destra l'energia consegnata / ricevuta dalle batterie

CAPITOLO 6

Conclusions and Future Developments

In questa tesi è stato sviluppato un sistema decentralizzato basato su tecnologia blockchain per la gestione e la vendita di energia tramite batterie di veicoli elettrici, soffermandosi soprattutto sul mercato aFRR. L'obiettivo principale del progetto era quello di esplorare come l'uso di smart contracts e tecnologie decentralizzate potesse migliorare l'efficienza e la trasparenza delle operazioni di mercato, garantendo al contempo la sicurezza delle transazioni e la riduzione dei costi legati agli intermediari tradizionali.
La piattaforma sviluppata è stata costruita utilizzando **Next.js**, **React**, **Solidity** e **Ganache**, con una particolare attenzione all'interazione tra frontend e smart contracts tramite **Web3.js** e l'uso di API. Il sistema è stato testato in un ambiente simulato, replicando scenari di registrazione delle batterie, piazzamento delle offerte, accettazione delle bid e gestione dei pagamenti da parte del TSO (Transmission System Operator).

Risultati Raggiunti

L'implementazione della DApp ha dimostrato che:

- L'uso di smart contracts basati su Ethereum può automatizzare in modo sicuro il processo di gestione delle offerte nel mercato aFRR, riducendo il rischio di errori umani e aumentando l'efficienza operativa.

- L'architettura decentralizzata consente una maggiore trasparenza, poiché tutte le transazioni e le operazioni di mercato vengono registrate sulla blockchain, rendendo i dati accessibili e immutabili.
- La simulazione ha confermato che il sistema è in grado di gestire scenari complessi di mercato, come la gestione dinamica del *State of Charge* (SoC) delle batterie e l'accettazione automatica delle offerte in base ai requisiti energetici del TSO.

Sfide Affrontate

Nonostante i risultati positivi ottenuti, il progetto ha presentato alcune sfide significative:

- La gestione dei costi di gas per le transazioni on-chain è un aspetto critico che può essere ottimizzato, specialmente in un contesto di mercato dove le transazioni sono frequenti e devono essere effettuate in tempo reale.
- La comunicazione tra frontend, backend e blockchain ha richiesto un'attenta gestione delle connessioni e dei timeout, specialmente durante il deployment e il testing in ambiente simulato.

Futuri sviluppi

Nonostante i risultati positivi ottenuti dalle simulazioni su mercato aFRR e del giorno prima (soprattutto il primo) tramite smart contracts su Ethereum, ci sono numerose opportunità di miglioramento che potrebbero rendere il sistema ancora più efficiente, scalabile e sicuro. In futuro, una delle soluzioni più promettenti potrebbe essere l'adozione del *Lightning Network* per ottimizzare i pagamenti. Il Lightning Network è stato proposto come soluzione al problema di scalabilità delle blockchain pubbliche, come Bitcoin, permettendo di gestire le transazioni off-chain. In questo modo, è possibile effettuare microtransazioni rapide tra due parti senza sovraccaricare la blockchain principale [33]. L'idea è quella di aprire canali di pagamento tra gli utenti, in cui le transazioni vengono

registrate solo al momento della chiusura del canale, riducendo così i costi e i tempi di conferma delle transazioni. Questa tecnologia potrebbe essere adattata per i pagamenti verso gli aggregatori e i proprietari di batterie, migliorando significativamente l'efficienza della piattaforma.
Parallelamente, l'utilizzo di **GoQuorum** rappresenta un'altra direzione interessante per migliorare la scalabilità e la privacy del sistema. GoQuorum, basato sul client *go-ethereum* (geth), è una variante ottimizzata per le esigenze delle imprese, offrendo funzionalità avanzate come transazioni e contratti privati. Grazie alla separazione tra stati pubblici e privati, GoQuorum garantisce che solo le parti autorizzate possano accedere ai dati sensibili [34]. Questo approccio sarebbe particolarmente utile nel contesto del mercato aFRR, dove è fondamentale proteggere la riservatezza delle offerte e delle transazioni energetiche. Inoltre, GoQuorum supporta meccanismi di consenso più performanti, come il *Raft Consensus* e *Istanbul BFT*, che consentono di raggiungere la finalità delle transazioni in tempi ridotti rispetto alla Proof of Stake utilizzata su Ethereum pubblico.
Un'ulteriore evoluzione del sistema potrebbe includere l'integrazione di algoritmi di *machine learning* per migliorare la previsione della domanda energetica e ottimizzare il piazzamento delle offerte. Utilizzando modelli predittivi basati su dati storici, il sistema potrebbe anticipare i trend di consumo, suggerendo automaticamente strategie ottimali per la partecipazione al mercato aFRR. Questo approccio permetterebbe di aumentare l'efficienza operativa e di migliorare la gestione delle risorse energetiche.
Infine, una possibile migrazione su soluzioni **Layer 2** come *Polygon* potrebbe rappresentare un ulteriore passo avanti nella scalabilità del sistema. Le reti Layer 2 consentono di gestire le transazioni fuori dalla blockchain principale, riducendo i costi di gas e aumentando il throughput delle transazioni. Questo approccio sarebbe ideale per gestire una grande quantità di offerte nel mercato aFRR senza compromettere le prestazioni della rete.
L'integrazione di queste tecnologie avanzate non solo migliorerebbe la scalabilità e l'efficienza della piattaforma, ma aprirebbe anche la stra-

da a nuove opportunità per l'automazione e la decentralizzazione nel settore energetico. Attraverso l'adozione di soluzioni innovative come il Lightning Network, GoQuorum e le reti Layer 2, è possibile creare un'infrastruttura energetica più resiliente e sostenibile, in grado di affrontare le sfide future del mercato dell'energia.

Bibliografia

[1] GridX. 2024. *What is a Grid Operator*. URL: https://www.gridx.ai/knowledge/what-is-a-grid-operator (cit. pp. iii and 7)

[2] Epexspot. *Basi del mercato dell'energia in Europa*. URL: https://www.epexspot.com/en/basicspowermarket (cit. pp. 8 and 9)

[3] Gestore Mercati Energetici. *MPE Mercato a Pronti*. URL: https://www.mercatoelettrico.org/it-it/Home/Mercati/Mercato-Elettrico/MPE-Mercato-a-pronti (cit. p. 10)

[4] Next Kraftwerke. *What is Day-Ahead Trading of Electricity?* URL: https://www.next-kraftwerke.com/knowledge/day-ahead-trading-electricity (cit. pp. 12 and 13)

[5] Next Kraftwerke. *What is aFRR (automatic frequency restoration reserve) and how does it work?* URL: https://www.next-kraftwerke.com/knowledge/afrr (cit. pp. 14 and 16)

[6] Nano Energies. *Manual Frequency Restoration Reserve (mFRR)* URL: https://nanoenergies.eu/knowledge-base/manual-frequency-restoration-reserve-mfrr (cit. pp. iii and 14)

[7] Flexcity. *Discover the aFRR service in Italy*. URL: https://www.flexcity.energy/en/afrr-in-italy (cit. p. 15)

[8] Terna. *Introducing Terna*. URL: https://www.terna.it/en/about-us/introducing-terna (cit. p. 15)

[9] Nordic Balancing Model. 2024. *High clearing prices in the Nordic aFRR Capacity Market*. URL: https://nordicbalancingmodel.net

/high-clearing-prices-in-the-nordic-afrr-capacity-market/ (cit. p. 16)

[10] Yli-Huumo J, Ko D, Choi S, Park S, Smolander K (2016) «Where Is Current Research on Blockchain Technology? — A Systematic Review». PLoS ONE 11(10): e0163477. doi:10.1371/journal.pone.0163477 (cit. pp. 17, 19, and 20)

[11] Ahmed Afif Monrat, Olov Schelén e Karl Andersson. «A survey of blockchain from the perspectives of applications, challenges, and opportunities». In: Ieee Access 7 (2019), pp. 117134–117151. (cit. pp. 18, 19, and 20)

[12] Chen, Ran and Wu, Xiaoming and Liu, Xiangzhi (2023) «RSETP: A Reliable Security Education and Training Platform Based on the Alliance Blockchain». Electronics vol.12 n.6: e0163477. doi:10.3390/electronics12061427 (cit. pp. iii and 19)

[13] GCore. 2023. *What Is a Man-in-the-Middle (MITM) Attack? — How to Prevent a MITM Attack*. URL: https://gcore.com/learning/man-in-the-middle-attack/ (cit. pp. iii and 21)

[14] Herond Academy. 2024. *Public Key vs Private Key: Key Differences Explained*. URL: https://blog.herond.org/public-key-vs-private-key-key-differences-explained/ (cit. pp. iii and 22)

[15] *Mastering Bitcoin*. 2023. URL: https://github.com/bitcoinbook/bitcoinbook/blob/develop/ch01_intro.adoc (cit. pp. 23, 25, and 26)

[16] *Mastering Bitcoin*. 2023. URL: https://github.com/bitcoinbook/bitcoinbook/blob/develop/ch11_blockchain.adoc (cit. p. 25)

[17] Comidor. 2022. *Blockchain Technology — Definition, Architecture and Fundamentals* URL: https://www.comidor.com/knowledge-base/blockchain-technology-knowledge-base/blockchain-fundamentals/ (cit. pp. iii and 27)

[18] Binance Academy. 2018. *Cos'è la Proof of Work?* URL: https://academy.binance.com/it/articles/proof-of-work-explained (cit. p. 28)

[19] Andreas M Antonopoulos e Gavin Wood. *Mastering Ethereum.* 2018. URL: https://github.com/ethereumbook/ethereumbook/blob/develop/01what-is.asciidoc (cit. pp. 28 and 29)

[20] Bitpanda Academy. *How does Ethereum work?* URL: https://www.bitpanda.com/academy/en/lessons/what-is-ethereum/ (cit. pp. 29 and 30)

[21] Kushwaha, Satpal Joshi, Sandeep Singh, Dilbag Kaur, Manjit Lee, Heung-No. (2022). «Systematic Review of Security Vulnerabilities in Ethereum Blockchain Smart Contract.» IEEE Access. PP. 1-1. 10.1109/ACCESS.2021.3140091. (cit. pp. iii and 29)

[22] Andreas M Antonopoulos e Gavin Wood. *Mastering Ethereum.* 2018. URL: https://github.com/ethereumbook/ethereumbook/blob/develop/07smart-contracts-solidity.asciidoc (cit. pp. 31 and 32)

[23] Diallo, Nour Shi, Weidong Xu, Lei Gao, Zhimin Chen, Lin Lu, Yang Shah, Nolan Carranco, Larry Le, Ton-Chanh Surez, Abraham Turner, Glenn. (2018). «eGov-DAO: a Better Government using Blockchain based Decentralized Autonomous Organization.» 166-171. 10.1109/ICEDEG.2018.8372356. (cit. pp. iii and 32)

[24] Andreas M Antonopoulos e Gavin Wood. 2018. *Mastering Ethereum.* URL: https://github.com/ethereumbook/ethereumbook/blob/develop/12dapps.asciidoc (cit. p. 34)

[25] Vlad Vovk. 2023. *What are DApps and how do they work? Learn how DApps development can change the world.* URL: https://klona.ua/en/blog/web-app-development/what-are-dapps-and-how-do-they-work-learn-how-dapps-development-can-change-the-world (cit. pp. iii and 34)

[26] Onkar Singh. 2023. *How to use Ganache for blockchain project development.* URL: https://cointelegraph.com/news/how-to-use-ganache-for-blockchain-project-development (cit. p. 43)

[27] Shardeum Content Team. 2023. *What Is Hardhat – A Comprehensive Guide.* URL: https://shardeum.org/blog/hardhat/ (cit. p. 45)

[28] Ahmed Abu Bakr. 2023. *A Comprehensive Guide to Next.js.* URL: https://medium.com/@ahmed.nums345/a-comprehensive-guide-to-next-js-5f3b03b49def (cit. p. 46)

[29] GeeksforGeeks2024. *Introduction to Tailwind CSS.* URL: https://www.geeksforgeeks.org/introduction-to-tailwind-css/ (cit. p. 47)

[30] Nefe Emadamerho-Atori. 2024. *MUI adoption guide: Overview, examples, and alternatives.* URL: https://blog.logrocket.com/mui-adoption-guide/ (cit. p. 47)

[31] Dhanvardini R, Martina PA, Vijay R, Amirtharajan R, Padmapriya Pravinkumar. «Development and Integration of dApp with blockchain smart contract Truffle Framework for user interactive applications». In: 2023 International Conference on Computer Communication and Informatics (ICCCI). (cit. p. 48)

[32] Ushna Ijaz. 2023. *A brief introduction to Axios.* URL: https://rapidapi.com/guides/what-is-axios (cit. p. 49)

[33] Ahmed Afif Monrat, Olov Schelén e Karl Andersson. «LNSC: A Security Model for Electric Vehicle and Charging Pile Management Based on Blockchain Ecosystem». In: Ieee Access 6 (2018), pp. 13565 - 13574. (cit. p. 74)

[34] Haven1. 2023. *Understanding GoQuorum.* URL: https://docs.haven1.org/learn/understanding-goquorum (cit. p. 75)