

UNIVERSITÀ DEGLI STUDI DI MILANO

FACOLTÀ DI SCIENZE E TECNOLOGIE

DIPARTIMENTO DI INFORMATICA

GIOVANNI DEGLI ANTONI



Laurea triennale in
Informatica per la Comunicazione Digitale

DASHBOARD DI ANALYTICS A SUPPORTO DI
APPLICAZIONI BASATE SU WEB3

Relatore: Prof. Matteo Zignani

Correlatore: Dott.ssa Alessia Galdeman

Tesi di Laurea di:
Cristian Marras
Matr. 909914

ANNO ACCADEMICO 2023 / 2024

Indice

Indice	i
1 Introduzione	1
2 Contesto	4
2.1 Web3	4
2.1.1 Blockchain	6
2.1.2 NFT	8
3 Tecnologie utilizzate	14
3.1 Python	14
3.1.1 Librerie utilizzate	15
3.2 Streamlit	18
3.2.1 Run your app	18
3.2.2 Architettura Streamlit	19
3.2.3 L'app chrome	20
3.2.4 Caching	22
3.2.5 Session state	23
3.2.6 Forms	24
3.2.7 Fragments	25
3.2.8 Widget	25
3.3 Open VPN connect	27
3.3.1 Come funziona	28
3.3.2 Tunnel SSH	28

4	Implementazione	29
4.1	Dashboard	29
4.1.1	Menu	29
4.1.2	Pagina principale	30
4.1.3	Pagina del wallet	33
4.2	Problematiche risolte	38
4.2.1	Facilitare la navigazione	38
4.2.2	Auto-completamento	39
4.2.3	Creazione grafici e ricampionamento dinamico	40
5	Conclusioni e sviluppi futuri	43
	Bibliografia	46

Capitolo 1

Introduzione

La tesi si sviluppa nel contesto del Web3 e, in particolare, delle blockchain e degli NFT. Queste tecnologie sono nate con il Web3, che si basa sul concetto di decentralizzazione. Grazie a questa caratteristica, ogni operazione all'interno della rete viene validata da tutti gli altri nodi, eliminando la necessità di un'autorità centrale che possa modificare le informazioni. Inoltre, ogni nodo della rete conserva una copia delle operazioni effettuate, rendendo il Web3 più resistente agli attacchi e garantendo a ciascun utente il controllo e la proprietà dei propri beni digitali, senza dover dipendere da enti centralizzati come le grandi aziende tecnologiche, ad esempio Google.

L'innovazione principale introdotta dal Web3 è la blockchain, un registro distribuito e immutabile delle transazioni avvenute all'interno della rete, che possono riguardare NFT o altri asset digitali. In questa tesi ci si concentrerà sugli NFT, ossia certificati digitali di proprietà che conferiscono unicità a determinati oggetti digitali, come opere d'arte, oggetti da collezione o elementi di giochi digitali.

Gli NFT possono essere scambiati su diverse blockchain, tra cui Ethereum, una delle più utilizzate. Di conseguenza, il loro valore è espresso nella criptovaluta della blockchain di riferimento. Spesso gli NFT sono organizzati in collezioni, alcune delle quali sono diventate particolarmente popolari e verranno analizzate nel corso della tesi.

Questa ricerca si concentra sulla realizzazione di una dashboard di analytics

per l'analisi dei dati provenienti dal Web3, con l'obiettivo di fornire uno strumento efficace per la visualizzazione e l'interpretazione delle informazioni relative alle collezioni di NFT, ai wallet e alle transazioni sulla blockchain. La webapp sviluppata utilizza Python, un linguaggio moderno e versatile, insieme a Streamlit, un framework intuitivo per la creazione di applicazioni web. I dati vengono raccolti e processati attraverso API specifiche, con l'intento di offrire agli utenti un'interfaccia chiara e interattiva per esplorare le metriche più rilevanti.

Durante lo sviluppo del progetto, sono state affrontate diverse sfide, in particolare quelle legate all'usabilità e all'efficienza della visualizzazione dei dati. Uno degli aspetti più importanti ha riguardato l'ottimizzazione della navigazione della webapp, per garantire un'esperienza utente fluida e intuitiva. È stata inoltre implementata una funzione di autocompletamento nella barra di ricerca delle collezioni NFT, per semplificare l'accesso alle informazioni. Infine, per migliorare la rappresentazione dei dati nel tempo, è stato introdotto un sistema di ricampionamento dinamico, che consente di adattare la granularità temporale dei grafici in base alle esigenze dell'utente.

Grazie a queste soluzioni, la dashboard sviluppata rappresenta un supporto efficace per l'analisi del mercato NFT, rendendo i dati più accessibili e facilmente interpretabili, con un impatto positivo sulla comprensione delle dinamiche del Web3.

In particolare la tesi è strutturata nel seguente modo:

Nel capitolo 2 verrà presentato il contesto teorico della ricerca, analizzando il Web3 e i suoi principi fondamentali, il funzionamento della blockchain e il ruolo degli NFT. Saranno inoltre approfondite le collezioni di NFT più rilevanti e il mercato degli asset digitali, con un'analisi delle dinamiche economiche e delle piattaforme più utilizzate per la compravendita.

Nel capitolo 3 verranno descritte le tecnologie utilizzate per la realizzazione della dashboard di analytics. In particolare, verranno approfonditi Python e Streamlit per la creazione dell'interfaccia web, e OpenVPN Connect per la gestione della connessione ai dati remoti.

Nel capitolo 4 sarà trattata l'implementazione della dashboard, illustrando le

scelte progettuali e le problematiche affrontate durante lo sviluppo. Verranno descritte le soluzioni adottate per ottimizzare la visualizzazione dei dati, migliorare l'usabilità e garantire un'esperienza utente fluida, includendo funzionalità come il ricampionamento dinamico dei dati e l'autocompletamento nella barra di ricerca.

Infine, nel capitolo 5, verranno tratte le conclusioni della ricerca, analizzando i risultati ottenuti e valutando possibili sviluppi futuri della dashboard, con un focus sulle opportunità di miglioramento e sulle potenziali applicazioni nel settore degli NFT e della blockchain.

Capitolo 2

Contesto

Questo capitolo fornirà una panoramica del contesto tecnologico intorno al mondo Web3.

2.1 Web3

Il termine *Web3*, coniato nel 2014 da Gavin Wood, co-fondatore di Ethereum e sviluppatore di Polkadot, si riferisce a un insieme di innovazioni digitali e applicazioni su internet basate sulla tecnologia blockchain. È importante non confondere il *Web3* con il concetto di *Web3.0*, che rappresenta l'evoluzione del Web1.0, caratterizzato da pagine web statiche e un'interazione passiva da parte dell'utente, e del Web2.0, in cui gli utenti sono anche creatori di contenuti in un internet dinamico. Il Web3.0, noto anche come web semantico, propone un modello in cui le pagine e i contenuti sono connessi tra loro attraverso l'uso di parole chiave, con l'intelligenza artificiale come elemento centrale.

I principi fondamentali del *Web3* includono la decentralizzazione e la proprietà dei contenuti. In questo nuovo paradigma, si aprono opportunità senza precedenti per i creatori di contenuti e gli artisti, poiché ogni creazione online diventa di proprietà dell'autore, che ha la libertà di decidere se e come monetizzarla. A differenza del Web2, in cui piattaforme centralizzate come Google, Facebook e Instagram impongono regole e limitazioni sulla pubblicazione e condivisione dei contenuti,

il *Web3* offre una maggiore libertà agli utenti. Tra le principali innovazioni che caratterizzano il Web3 ci sono:

- **La blockchain**, che costituisce la base del Web3, fornisce un sistema decentralizzato e immutabile per registrare transazioni e dati.
- **Gli smart contract**, che sono programmi auto-eseguibili che utilizzano la tecnologia blockchain per registrare e eseguire transazioni di criptovalute e altri beni digitali senza bisogno di intermediari centralizzati. Questi contratti digitali definiscono una serie di condizioni che, una volta soddisfatte, si eseguono automaticamente, garantendo che gli accordi tra le parti siano rispettati in modo sicuro e trasparente. Poiché gli smart contract sono registrati permanentemente sulla blockchain, offrono decentralizzazione, immutabilità e verifica pubblica, eliminando la necessità di fiducia tra le parti coinvolte e prevenendo frodi o manipolazioni.
- **I wallet**, che permettono di custodire criptovalute, eseguire transazioni e interagire con la blockchain, identificando univocamente l'utente. Ogni wallet è basato su una coppia di chiavi crittografiche: una chiave pubblica, che funge da identificativo (simile all'IBAN di un conto bancario), e una chiave privata, utilizzata per firmare digitalmente le transazioni. La chiave privata fornisce una prova matematica che la transazione è stata generata dal legittimo proprietario del wallet. Nel contesto del *Web3*, il wallet rappresenta l'identità digitale dell'utente all'interno della rete, permettendo di riconoscerlo e autorizzarlo nelle interazioni sulla blockchain.
- **Le criptovalute**, monete digitali basate sulla tecnologia blockchain, utilizzate come mezzo di scambio e riserva di valore. Le più comuni, come Bitcoin (BTC), Ethereum (ETH) e Binance Coin (BNB) consentono transazioni decentralizzate, rapide e sicure senza la necessità di intermediari tradizionali come le banche.
- **I token**, che sono strumenti digitali utilizzati per rappresentare beni, diritti o servizi all'interno di una blockchain. I token permettono transazioni rapide

tra venditori e acquirenti senza intermediari, con un notevole risparmio di tempo e risorse rispetto agli strumenti finanziari tradizionali. Inoltre, alcuni token sono frazionabili, cioè possono essere suddivisi in parti più piccole, consentendo la proprietà condivisa da più utenti. I token si suddividono in fungibili e non fungibili, questi ultimi, noti come NFT, saranno trattati in dettaglio successivamente.

- **Le dApp (applicazioni decentralizzate)**, che sono software basati su smart contract che funzionano senza un controllo centralizzato. Queste applicazioni sono generalmente open source e conferiscono agli utenti il pieno controllo delle proprie attività digitali.
- **Le DAO (Organizzazioni Autonome Decentralizzate)**, che permettono ai partecipanti di una piattaforma di collaborare in modo equo nella creazione e distribuzione dei contenuti, grazie a una governance condivisa.

In generale, questi strumenti offrono agli utenti maggiore autonomia e controllo nella gestione dei propri beni digitali, promuovendo un ecosistema basato sulla fiducia, la trasparenza e l'interazione diretta tra i partecipanti [1, 2].

2.1.1 Blockchain

La tecnologia blockchain è un meccanismo di database distribuito che permette la condivisione trasparente di informazioni garantendo immutabilità dei dati memorizzati nella stessa. Una blockchain archivia i dati in blocchi collegati tra loro in una catena. I dati sono cronologicamente coerenti perché non è possibile eliminare o modificare la catena senza il consenso dei membri della rete. Di conseguenza, è possibile utilizzare la tecnologia blockchain per creare un libro mastro inalterabile (**distribute ledger**) o immutabile per tracciare gli ordini, i pagamenti, gli account e altre transazioni. Il sistema dispone di meccanismi integrati che impediscono l'inserimento di transazioni non autorizzate e creano coerenza nella visualizzazione condivisa di tali transazioni.

Solitamente, i database tradizionali presentano diverse sfide nella registrazione di transazioni finanziarie. Si consideri ad esempio la vendita di una proprietà.

Una volta che il denaro viene scambiato, il possesso della proprietà viene trasferito all'acquirente. Individualmente, sia l'acquirente che il venditore possono registrare le transazioni monetarie, ma nessuna fonte è affidabile. Il venditore potrebbe dichiarare di non aver ricevuto il denaro, sebbene lo abbia ricevuto, e l'acquirente può a sua volta sostenere di aver versato il denaro, anche se non lo ha fatto. Per evitare eventuali questioni legali, una terza parte fidata deve supervisionare e convalidare le transazioni. La presenza di questa autorità centrale non solo complica la transazione ma crea un unico punto di vulnerabilità (**single point of failure**). Se il database centrale fosse compromesso, entrambe le parti subirebbero danni.

La blockchain limita simili problematiche creando un sistema decentralizzato e a prova di manomissione per registrare le transazioni. Nello scenario di transazione immobiliare, la blockchain crea un libro mastro per l'acquirente e uno per il venditore. Tutte le transazioni devono essere approvate da entrambe le parti e sono automaticamente aggiornate in entrambi i libri mastri in tempo reale. Qualsiasi danno alle transazioni storiche danneggerà l'intero libro mastro. Queste caratteristiche della tecnologia blockchain hanno spinto al suo utilizzo in diversi settori, inclusa la creazione di valuta digitale [3]. Le caratteristiche principali delle blockchain sono le seguenti:

- **Maggiore fiducia:** Con la blockchain, in quanto afferente a una rete di soli membri, si ha la certezza di ricevere dati accurati in maniera tempestiva e che i record riservati della blockchain siano condivisi solo con i membri della rete a cui si è garantito l'accesso.
- **Maggiore sicurezza:** Il consenso sull'accuratezza dei dati è richiesto da tutti i membri della rete e tutte le transazioni convalidate sono immutabili perché vengono registrate in modo permanente. Le transazioni non possono essere eliminate in nessun caso, nemmeno da un amministratore di sistema.
- **Maggiore efficienza:** Grazie a un libro mastro distribuito condiviso tra i partecipanti della rete, si eliminano le lunghe procedure di riconciliazione dei record. Inoltre, per accelerare le transazioni, è possibile registrare sulla

blockchain un insieme di regole integrate in uno smart contract, eseguito automaticamente [4].

2.1.2 NFT

Gli NFT, acronimo di Non-Fungible Token, sono certificati digitali unici basati sulla tecnologia blockchain. In italiano, il termine può essere tradotto come "gettoni non fungibili", indicando il loro carattere insostituibile e non replicabile. Gli NFT vengono utilizzati per attestare la proprietà di un oggetto digitale o fisico, come opere d'arte, brani musicali, video e altri asset, garantendone autenticità e unicità. Questi token rappresentano una delle innovazioni più significative nel campo della finanza decentralizzata e offrono un nuovo paradigma nella gestione della proprietà digitale. La loro validità e integrità sono garantite da contratti intelligenti, protocolli automatizzati che regolano i diritti di proprietà e le transazioni in modo sicuro e trasparente. **Acquistare un NFT non significa ottenere il possesso dell'opera ma piuttosto il diritto a dimostrarne l'autenticità e la provenienza.** Questo diritto è certificato attraverso uno smart contract registrato in maniera immutabile sulla blockchain, rendendo ogni NFT univoco e impossibile da falsificare. Negli ultimi anni, gli NFT hanno rivoluzionato diversi settori, dall'arte digitale al collezionismo, aprendo nuove possibilità per creatori e investitori. La loro diffusione continua a crescere, alimentando il dibattito sulle potenzialità e le implicazioni di questa tecnologia nel panorama economico e culturale globale [5].

Mercato NFT

Un marketplace NFT è una piattaforma digitale dedicata alla compravendita di NFT. Queste piattaforme consentono agli utenti di depositare, esporre e vendere i propri NFT. Alcuni marketplace di NFT consentono anche di coniare NFT sulla piattaforma stessa. A fronte del pagamento di una commissione, il marketplace di NFT gestisce le transazioni di NFT da venditore ad acquirente. Ogni marketplace ha il proprio sistema di funzionamento. I tipi di NFT disponibili su un sito, l'importo delle commissioni, le opzioni di pagamento, le blockchain e i token supportati e altre condizioni variano in base al marketplace che si utilizza [6].

Tra le piattaforme più famose e utilizzate troviamo:

- **OpenSea:** lanciato nel 2017 negli Stati Uniti, è uno dei marketplace di NFT più grandi e rinomati a livello globale. La piattaforma ospita una vasta gamma di NFT, tra cui opere d'arte, musica, fotografie, asset per il gaming, carte collezionabili e mondi virtuali. Le transazioni su OpenSea avvengono principalmente tramite criptovalute come Ethereum, Solana, Polygon, Avalanche e BNB, sebbene siano supportate anche altre valute digitali. Tuttavia, non è possibile acquistare NFT utilizzando valute fiat come euro o dollari. OpenSea applica il 2,5% di commissione a ogni transazione. È previsto anche il pagamento delle gas fee, ossia commissioni pagate dagli utenti per compensare il dispendio di energia necessario per elaborare e convalidare le transazioni sulla blockchain di Ethereum.
- **Rarible:** permette agli utenti di acquistare e vendere opere d'arte, oggetti da collezione, risorse per videogiochi e altri NFT. Le transazioni sulla piattaforma possono essere effettuate utilizzando criptovalute come Ethereum, Polygon, Tezos e Immutable X, ma sono previste delle commissioni. La piattaforma applica una commissione fissa del 2,5% su ogni transazione, oltre a eventuali gas fee. Un aspetto distintivo di Rarible è il suo approccio decentralizzato alla governance: anziché avere un team centrale che prende le decisioni, Rarible coinvolge direttamente la sua comunità attraverso il token nativo RARI. I titolari di RARI possono partecipare al processo decisionale, votando su questioni aziendali, come modifiche alle politiche interne. Nel 2021, Rarible ha annunciato una collaborazione con Adobe per migliorare la verifica e la protezione dei metadati dei contenuti digitali, inclusi gli NFT.
- **Binance:** uno dei principali exchange di criptovalute, ha lanciato il proprio marketplace di NFT, Binance NFT, nel 2021, entrando così nel mercato degli NFT insieme ad altri attori come Crypto.com con la sua piattaforma dedicata. Il catalogo di Binance NFT Marketplace è simile a quello delle principali piattaforme concorrenti, offrendo una vasta selezione di opere d'arte, oggetti da gioco e collezionabili. Un grande punto di forza di Binance

NFT è rappresentato dalle commissioni molto basse: la piattaforma applica una commissione di negoziazione dell'1%. Inoltre, è una piattaforma facile da usare, costruita con una tecnologia e un layout simili a quelli dell'exchange, che la rende particolarmente accessibile per chi è già abituato a Binance. Inoltre, grazie alla sua posizione di leader nel settore delle criptovalute, Binance possiede la propria blockchain, la BNB Chain, che le conferisce un ulteriore vantaggio competitivo.

- **Magic Eden:** è un marketplace NFT leader, fondato nel 2021, che ha rapidamente guadagnato popolarità come piattaforma principale per la compravendita di NFT sulla blockchain di Solana. Con oltre il 90% della quota di mercato per le vendite secondarie di NFT su Solana, Magic Eden ha dimostrato una crescita esponenziale, raggiungendo una valutazione di 1,6 miliardi di dollari nel 2023 dopo un round di finanziamento di successo. Magic Eden ha superato i concorrenti come OpenSea in termini di volume di scambi su Solana, catturando tra il 97% e il 99% del volume giornaliero per gli NFT basati su questa blockchain. Non ci sono costi per l'elenco degli NFT e la commissione uniforme per le transazioni è del 2%. La sua crescita è stata alimentata da un forte focus sulla sicurezza e sull'esperienza dell'utente, rendendola una scelta preferita sia per i collezionisti che per i creatori [7] [8].

Il recente calo dei prezzi delle criptovalute ha avuto un impatto significativo sul mercato degli NFT, rispecchiando l'andamento generale del settore. Sebbene negli ultimi mesi gli NFT avessero mostrato segnali di ripresa, il loro slancio si è indebolito dall'inizio dell'anno. A febbraio 2025, il volume complessivo delle transazioni di NFT è sceso a 498 milioni di dollari, registrando una contrazione del 50% rispetto al mese precedente, mentre il numero totale di vendite è diminuito del 16%. Questo andamento conferma la stretta correlazione tra il valore delle criptovalute e quello degli NFT, evidenziando come il sentiment del mercato influenzi direttamente le dinamiche di scambio [9].

Collezioni NFT

Le collezioni NFT sono insiemi di più NFT che generalmente condividono un tema o una caratteristica comune, che possono essere creato da uno stesso artista o appartenenti a una stessa serie. Le collezioni di NFT possono essere limitate o aperte. Le collezioni limitate avranno un numero predeterminato di NFT, creando rarità e valore per i collezionisti, mentre le collezioni aperte possono permettere la creazione di NFT illimitata entro una certa finestra temporale [10].

Ad oggi alcune delle collezioni più famose sono le seguenti:

- **CryptoPunk**: icona della criptoarte, hanno trasformato la percezione della proprietà nel regno digitale. Lanciata da Larva Labs nel 2017, questa raccolta di 10.000 caratteri unici come NFT sulla blockchain di Ethereum ha raggiunto oggi un valore di mercato di 1.300 miliardi di dollari. Ogni CryptoPunk nella collezione crittografica è stato generato algoritmicamente con caratteristiche uniche, garantendone l'unicità e l'autenticità. La loro influenza ha ispirato artisti e sviluppatori a esplorare l'arte generativa e gli NFT, oltre a promuovere un dialogo sull'autenticità e il valore dell'arte digitale. Secondo dati della piattaforma, il prezzo minimo di vendita di CryptoPunks è 37,23 ETH (circa \$ 130.300), corrispondente ad un aumento del 33% negli ultimi 3 mesi.



Figura 1: CryptoPunk #5822: uno degli NFT più costosi mai venduti. [11]

- **Pudgy Penguins:** introdotto nel 2021 sulla blockchain di Ethereum, ha guadagnato popolarità nel 2024 grazie al lancio del suo token nativo, PENGU. Questa collezione di 8.888 NFT raffiguranti pinguini si distingue per il suo design accattivante e per l'attenzione alla creazione di una comunità solida. Recentemente, Pudgy Penguins ha superato il Bored Ape Yacht Club (BAYC), raggiungendo una capitalizzazione di mercato di \$702,3 milioni, con un prezzo minimo per NFT pari a \$22,77 ETH (circa \$79,600). L'introduzione del token PENGU ha contribuito a questa crescita, suscitando un rinnovato interesse tra investitori e collezionisti. Attualmente, PENGU è classificato come la 56^{esima} criptovaluta per capitalizzazione di mercato, con un prezzo unitario di \$0,036.



Figura 2: Pudgy penguin #8123 [12]

- **Bored Ape Yacht Club:** è una delle collezioni NFT più iconiche, dominando il mercato sin dal suo lancio nel 2021. Composta da 10.000 scimmie uniche, ciascuna caratterizzata da tratti distintivi e stili vari, BAYC si è affermata come un vero e proprio status symbol nel mondo delle criptovalute. Possedere un BAYC non significa solo acquisire un'opera d'arte digitale, ma anche ottenere l'accesso a eventi esclusivi e privilegi riservati ai membri della sua attiva comunità. Tuttavia, in seguito alla crescente competizione con Pudgy Penguins, la capitalizzazione di BAYC è scesa a 540,5 milioni

di dollari, posizionandosi attualmente al terzo posto tra le collezioni NFT più capitalizzate. Il prezzo minimo di vendita per un BAYC è pari a 15,84 ETH (circa \$55.400), registrando un aumento superiore al 50% negli ultimi 90 giorni [13].

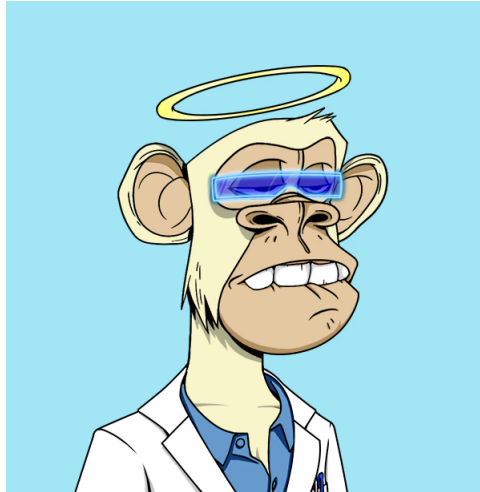


Figura 3: Bored Ape #2121 [14]

Capitolo 3

Tecnologie utilizzate

3.1 Python

Python è un linguaggio di programmazione open source di alto livello progettato da Guido van Rossum. Rilasciato nel 1991, la filosofia della progettazione di questo linguaggio di programmazione mira alla leggibilità del codice, che è il motivo per cui è uno dei linguaggi più facili da imparare.

Python è un linguaggio orientato agli oggetti, riflessivo, funzionale, procedurale e strutturale. Il supporto a diversi paradigmi di programmazione permette ai programmatori di scrivere codice logico e chiaro per progetti di ogni scala.

Al momento, ci sono tre versioni principali di Python, di cui Python 3.13 è la più recente. Supporta anche una vasta gamma di librerie per semplificare la scrittura del codice [15].

Python è un linguaggio pseudocompilato: un interprete si occupa di analizzare il codice sorgente (semplici file testuali con estensione .py) e, se sintatticamente corretto, di eseguirlo. In Python, non esiste una fase di compilazione separata (come avviene in C, per esempio) che generi un file eseguibile partendo dal sorgente.

L'esser pseudointerpretato rende Python un linguaggio portabile. Una volta scritto un sorgente, esso può essere interpretato ed eseguito sulla gran parte delle piattaforme attualmente utilizzate, siano esse di casa Apple (Mac) che PC (Microsoft Windows e GNU/Linux). Semplicemente, basta la presenza della versione

corretta dell'interprete.

Infine, Python è free software: non solo il download dell'interprete per la propria piattaforma, così come l'uso di Python nelle proprie applicazioni, è completamente gratuito; ma oltre a questo Python può essere liberamente modificato e così ridistribuito, secondo le regole di una licenza pienamente open-source.

Queste caratteristiche hanno fatto di Python il protagonista di un enorme diffusione in tutto il mondo, e anche in Italia, negli ultimi anni. Questo perché garantisce lo sviluppo rapido (e divertente) di applicazioni di qualsiasi complessità in tutti i contesti: dal desktop al web, passando dallo sviluppo di videogiochi e dallo scripting di sistema [16].

3.1.1 Librerie utilizzate

Requests

La libreria Requests di Python è uno strumento fondamentale per la gestione delle richieste HTTP in modo semplice e intuitivo. Questa libreria, scritta per "esseri umani" [17], astrae la complessità associata alla creazione e gestione delle richieste, rendendo l'interazione con le API e i servizi web molto più accessibile. Esempio d'uso:

```
def get_collection():  
    url = f"{BASE_URL}/nft/collections"  
    headers = {"accept": "application/json"}  
    response = httpx.get(url, headers=headers, timeout=10)  
    return handle_request_error(response) or response.json()  
()
```

Chiaramente questa e tutte le altre funzioni per chiamate API devono essere sottoposte a un controllo degli errori, che gestisco con la funzione `handle_request_error()`.

```
def handle_request_error(response):  
    try:  
        response.raise_for_status()  
    except httpx.HTTPStatusError as e:
```

```
        logging.error(f"Errore HTTP-{e.response.status_code}:\n-{e.response.text}")
    except httpx.RequestError as e:
        logging.error(f"Errore di connessione: -{e}")
    except Exception as e:
        logging.error(f"Errore generico: -{e}")
    return None
```

Datetime

La gestione delle date e dell'orario rappresenta un elemento cruciale in molte applicazioni, in particolare quando si tratta di adattare i formati temporali alle esigenze degli utenti. Nel mio progetto, è stato spesso necessario convertire date provenienti da formati internazionali al formato italiano. A tal fine, ho utilizzato la funzione `datetime.fromisoformat`, che permette di interpretare stringhe in formato ISO 8601 e convertirle in oggetti `datetime`, rendendo così le informazioni temporali più comprensibili e fruibili per l'utente italiano [18].

Per facilitare questa conversione, ho implementato la seguente funzione, che trasforma una data in formato ISO in una stringa formattata secondo lo standard italiano:

```
def format_date(iso_date):
    dt = datetime.fromisoformat(iso_date)
    return dt.strftime("%d/%m/%Y-%H:%M:%S")
```

Pandas

Pandas è una libreria open-source per il linguaggio di programmazione Python, utilizzata principalmente per la manipolazione e l'analisi dei dati. Essa fornisce strutture dati flessibili e potenti, come il `DataFrame`, che permettono di lavorare facilmente con dati eterogenei e di grandi dimensioni. Grazie alla sua capacità di gestire tabelle, dati temporali, e operazioni complesse di aggregazione e filtraggio, pandas è uno strumento essenziale in ambito scientifico, statistico e ingegneristico.

Per la gestione e visualizzazione dei dati nel mio progetto, ho utilizzato `pandas` e in particolare, ho sfruttato la funzione `pd.DataFrame` per la creazione di tabelle e `pd.to_datetime` per convertire le date in un formato coerente e facilmente gestibile. Successivamente, le tabelle così ottenute sono state trasformate in grafici interattivi mediante l'uso della libreria **Plotly**, che mi ha permesso di creare visualizzazioni dinamiche e ben strutturate, adatte per una migliore analisi dei dati [19].

Plotly

Plotly è una libreria open-source per la visualizzazione dei dati, che consente di creare grafici interattivi di alta qualità in Python. Plotly supporta diversi tipi di grafici, tra cui grafici a dispersione, lineari, a barre, a torta, e altri. La libreria è particolarmente utile per la creazione di dashboard interattive e grafici complessi che richiedono un'esperienza utente fluida e dinamica.

Nel mio progetto, ho utilizzato Plotly per trasformare i dati elaborati tramite `pandas` in grafici interattivi, consentendo una visualizzazione intuitiva delle informazioni. In particolare, per generare un grafico temporale, ho utilizzato il codice seguente:

```
fig = px.line(df, x=df.index, y='value', title="Transazioni  
- wallet")  
st.plotly_chart(fig)
```

Questo esempio crea un grafico lineare che mostra le transazioni del wallet nel tempo, utilizzando i dati contenuti nel DataFrame `df`. La funzione `px.line` di Plotly consente di creare il grafico a partire dalle colonne `x` e `y`, mentre la funzione `st.plotly_chart` di Streamlit rende il grafico interattivo e visibile nella webapp. I grafici realizzati con Plotly permettono di esplorare i dati in modo più approfondito, migliorando l'analisi e la comprensione delle informazioni [20].

3.2 Streamlit

Streamlit è una libreria open-source per Python che aiuta a creare applicazioni web per condividere risultati analitici, costruire esperienze interattive complesse e migliorare nuovi modelli di apprendimento automatico. Inoltre, sviluppare e distribuire applicazioni con Streamlit è estremamente rapido e flessibile, riducendo spesso il tempo di sviluppo da giorni a ore.

Streamlit dispone di metodi integrati e convenienti per tutto, dall'acquisizione di input utente come testo e date alla visualizzazione di grafici interattivi utilizzando le librerie Python più popolari e potenti [21].

3.2.1 Run your app

Esistono diversi modi di fare il run della propria applicazione Streamlit. Prima si crea il proprio file Python e poi si fa il run. Vediamo ora alcuni metodi, in base al caso d'uso.

Streamlit run

Una volta creato il proprio script, per esempio `main_app.py`, il modo più semplice per fare il run è:

```
streamlit run your_script.py
```

Una volta fatto ciò, un server locale Streamlit verrà avviato e l'app verrà aperta in una nuova tab del browser predefinito. È inoltre possibile passare argomenti o URL al proprio script nel seguente modo:

```
streamlit run your_script.py [-- script args]
```

```
streamlit run https://raw.githubusercontent.com/  
streamlit/demo-uber/master/streamlit_app.py
```

Esegui Streamlit come modulo Python

Un altro modo per eseguire Streamlit è avviarlo come modulo Python. Questo è utile quando si configura un IDE come PyCharm per lavorare con Streamlit.

```
python -m streamlit run your_script.py
```

[22]

3.2.2 Architettura Streamlit

Le applicazioni Streamlit seguono un'architettura client-server. Il backend Python dell'applicazione funge da server, mentre il frontend, visualizzato tramite un browser, opera come client. Durante lo sviluppo locale, lo stesso computer esegue sia il server che il client. Nel caso in cui un altro utente acceda all'applicazione attraverso una rete locale o globale, server e client vengono eseguiti su macchine distinte. Quando si desidera condividere o distribuire l'applicazione, è fondamentale comprendere questa architettura client-server al fine di evitare errori comuni [22].

Backend

Quando si esegue il comando `streamlit run your_app.py`, il computer utilizza Python per avviare un server Streamlit. Il server Streamlit gira sulla macchina dove l'app è stata inizializzata con `streamlit run`. La macchina che esegue il server Streamlit è anche chiamata *host*.

Frontend

Quando un utente visualizza l'applicazione tramite un browser, il dispositivo in uso funge da client Streamlit. Se l'applicazione viene visualizzata dallo stesso computer su cui è in esecuzione o in fase di sviluppo, server e client operano sulla medesima macchina. Al contrario, quando l'applicazione viene visualizzata attraverso una rete locale o su Internet, il client viene eseguito su una macchina distinta rispetto al server.

3.2.3 L'app chrome

L'app Streamlit dispone di alcuni widget nell'angolo in alto a destra che aiutano durante lo sviluppo. Questi widget sono utili anche agli utenti mentre utilizzano l'app. Questa parte dell'interfaccia è comunemente chiamata "chrome dell'app". Il chrome include un'area di stato, una barra degli strumenti e il menu dell'app.

Il menu dell'app è configurabile. Per impostazione predefinita, è possibile accedere alle opzioni per sviluppatori dal menu dell'app quando questa viene visualizzata localmente o su Streamlit Community Cloud, con un account che dispone di accesso amministrativo. Durante la visualizzazione dell'app, è sufficiente cliccare sull'icona nell'angolo in alto a destra per accedere al menu [22].

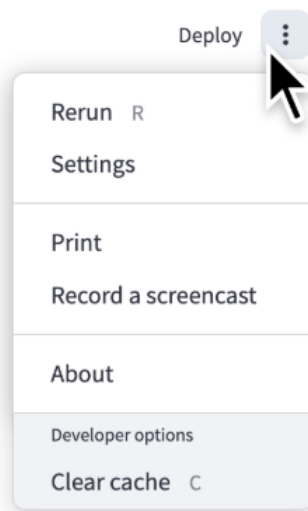


Figura 4: Menu opzioni Streamlit

Rerun

È possibile avviare manualmente una nuova esecuzione dell'applicazione selezionando l'opzione "Rerun" dal menu dell'app. Questa operazione non comporta il reset della sessione: gli stati dei widget e i valori memorizzati in `st.session_state` verranno mantenuti. In alternativa è possibile riavviare l'esecuzione premendo

il tasto "R" sulla tastiera, a condizione che il focus non sia attualmente su un elemento di input.

Settings

Con l'opzione "Settings" è possibile controllare l'aspetto dell'applicazione durante la sua esecuzione. Se l'app viene visualizzata localmente, è possibile configurare come l'app risponda alle modifiche nel codice sorgente.

Print

Cliccando su "Print" si apre la finestra di dialogo per la stampa. Questa opzione utilizza la funzione di stampa in PDF integrata nel browser.

Record a screencast

È possibile registrare facilmente lo schermo direttamente dall'applicazione. La registrazione dello schermo è supportata nelle ultime versioni di Chrome, Edge e Firefox. Bisogna assicurarsi che il browser sia aggiornato per garantire la compatibilità. A seconda delle impostazioni correnti, potrebbe essere necessario concedere al browser il permesso di registrare lo schermo. È inoltre possibile condividere la registrazione.

About

È possibile verificare quale versione di Streamlit è in esecuzione tramite l'opzione "About". Gli sviluppatori hanno anche la possibilità di personalizzare il messaggio visualizzato in questa sezione utilizzando `st.set_page_config`.

Developer options - Clear cache

È possibile resettare la cache dell'applicazione cliccando su "Clear cache" nel menu dell'app o premendo "C" sulla tastiera, a condizione che il focus non sia su un elemento di input. Questo rimuoverà tutte le voci memorizzate nella cache per `@st.cache_data` e `@st.cache_resource`.

3.2.4 Caching

Streamlit esegue lo script dall'inizio alla fine ad ogni interazione dell'utente o modifica del codice. Questo modello di esecuzione rende lo sviluppo molto semplice, ma presenta due sfide principali:

- Le funzioni a lunga durata vengono eseguite ripetutamente, rallentando l'applicazione.
- Gli oggetti vengono ricreati continuamente, rendendo difficile la gestione dello stato dell'applicazione tra le nuove esecuzioni o sessioni.

Tuttavia, Streamlit permette di affrontare entrambe le problematiche grazie al suo meccanismo di caching integrato. Il caching memorizza i risultati delle chiamate a funzioni lente, in modo che vengano eseguite solo una volta. Questo rende l'applicazione molto più veloce e aiuta a mantenere gli oggetti tra le esecuzioni. I valori memorizzati nella cache sono disponibili per tutti gli utenti dell'applicazione. Se è necessario salvare i risultati che dovrebbero essere accessibili solo all'interno di una sessione, viene consigliato di utilizzare invece Session State.

Per memorizzare nella cache una funzione in Streamlit, è necessario decorarla con uno dei due decoratori (`st.cache_data` o `st.cache_resource`):

```
@st.cache_data
def long_running_function(param1, param2):
    return ...
```

In questo esempio, decorando `long_running_function` con `@st.cache_data`, si istruisce Streamlit a controllare due cose ogni volta che la funzione viene chiamata:

- I valori dei parametri di ingresso (in questo caso, `param1` e `param2`).
- Il codice all'interno della funzione.

Se questa è la prima volta che Streamlit vede questi valori di parametro e il codice della funzione, esegue la funzione e memorizza il valore restituito dalla funzione nella cache. La prossima volta che la funzione viene chiamata con gli stessi

parametri e codice, Streamlit salta l'esecuzione della funzione e restituisce invece il valore memorizzato nella cache. Durante lo sviluppo, la cache viene aggiornata automaticamente ogni volta che il codice della funzione cambia, garantendo che le modifiche più recenti siano riflesse nella cache [22].

Come detto in precedenza, vi sono due diversi decorator di cache.

Cache data

`st.cache_data` è il metodo raccomandato per memorizzare nella cache i calcoli che restituiscono dati: caricare un DataFrame da un file CSV, trasformare un array NumPy, fare una richiesta a un'API o qualsiasi altra funzione che restituisce un oggetto di dati serializzabile (str, int, float, DataFrame, array, list, ...). Crea una nuova copia dei dati ad ogni chiamata della funzione, rendendolo sicuro contro le mutazioni e le condizioni di gara.

Cache resource

`st.cache_resource` è il metodo raccomandato per memorizzare nella cache risorse globali come modelli di machine learning o connessioni a database – oggetti non serializzabili che non si desidera caricare più volte. Utilizzandolo, è possibile condividere queste risorse tra tutte le esecuzioni e sessioni di un'app senza copiarle o duplicarle.

3.2.5 Session state

Il Session State è un meccanismo che permette di conservare variabili e dati tra i diversi eventi di interazione con l'utente. Streamlit ricarica completamente la pagina ogni volta che un utente interagisce con un widget (come un pulsante o una selezione), e i dati associati a questi eventi verrebbero normalmente persi, mentre utilizzando Session State si risolve questo problema permettendo di mantenere i dati durante la durata di una sessione utente.

In Streamlit, ogni utente ha il proprio stato di sessione isolato, quindi le modifiche effettuate da un utente non influenzeranno le sessioni degli altri utenti. Questo

è utile quando si sviluppano applicazioni web interattive in cui gli utenti possono eseguire azioni indipendenti, ma è necessario mantenere traccia delle informazioni durante la loro sessione.

Il Session State in Streamlit è implementato attraverso la variabile `st.session_state`. Al suo interno è possibile memorizzare qualsiasi tipo di dato (ad esempio, stringhe, numeri, liste, dizionari) e accedervi in qualsiasi parte dell'applicazione, senza perdere i dati tra i ricaricamenti della pagina [22].

Un esempio di utilizzo del Session State è il seguente:

```
# Impostare una variabile nel Session State
if 'counter' not in st.session_state:
    st.session_state.counter = 0

# Visualizzare il valore attuale della variabile
st.write(f"Contatore: {st.session_state.counter}")

# Incrementare il contatore quando un pulsante viene
premutato
if st.button('Incrementa'):
    st.session_state.counter += 1
```

3.2.6 Forms

I form in Streamlit sono strumenti essenziali per gestire l'input degli utenti in modo organizzato e intuitivo. Essi consentono di raggruppare vari widget di input, come caselle di testo, slider e pulsanti, all'interno di un'unica interfaccia, facilitando l'interazione con l'applicazione. Un form in Streamlit è un contenitore che raggruppa elementi interattivi e include un pulsante di invio ("Submit"). Quando l'utente preme il pulsante di invio, tutti i valori dei widget all'interno del form vengono inviati in un'unica operazione. Questo approccio riduce il numero di esecuzioni del codice, migliorando l'efficienza e l'esperienza utente [22].

3.2.7 Fragments

I "rerun" sono una parte centrale di ogni applicazione Streamlit. Quando gli utenti interagiscono con i widget, lo script viene eseguito nuovamente dall'inizio alla fine, aggiornando l'interfaccia dell'applicazione. Streamlit offre diverse funzionalità per facilitare lo sviluppo dell'applicazione all'interno di questo modello di esecuzione. Nelle ultime versioni Streamlit ha introdotto i frammenti, che consentono di eseguire nuovamente una porzione del codice anziché l'intero script. Man mano che l'applicazione cresce in dimensioni e complessità, questi "rerun" di frammenti contribuiscono a rendere l'applicazione più efficiente e performante. I frammenti offrono un controllo più preciso e di facile comprensione sul flusso di esecuzione dell'applicazione [22].

Questa funzionalità è implementabile tramite il relativo decorator:

```
@st.fragment
def fragment_function():
    if st.button("Saluta!"):
        st.write("Ciao!")

fragment_function()
```

3.2.8 Widget

I widget (come `st.button`, `st.selectbox` e `st.text_input`) sono al centro delle applicazioni Streamlit. Essi rappresentano gli elementi interattivi di Streamlit che trasferiscono le informazioni dagli utenti al codice Python.

Streamlit offre una varietà di widget che possono essere utilizzati per raccogliere input dagli utenti. I widget principali sono:

- **Button:** Permette all'utente di eseguire azioni cliccando su un pulsante.

```
if st.button("Cliccami"):
    st.write("Hai cliccato il pulsante!")
```

- **Checkbox:** Permette all'utente di selezionare o deselezionare un'opzione.

```
if st.checkbox("Accetta i termini"):
    st.write("Hai accettato i termini.")
```

- **Slider:** Permette all'utente di selezionare un valore da un intervallo.

```
valore = st.slider("Seleziona un valore", 0, 100)
st.write("Hai selezionato:", valore)
```

- **Selectbox:** Fornisce un menu a discesa per la selezione tra diverse opzioni.

```
scelta = st.selectbox("Scegli un'opzione", ["A", "B"])
st.write("Hai scelto:", scelta)
```

- **Multiselect:** Permette la selezione di più opzioni da un elenco.

```
scelte_multiple = st.multiselect("Scegli opzioni",
    ["A", "B", "C"])
st.write("Hai scelto:", scelte_multiple)
```

Streamlit consente anche una gestione efficace del layout attraverso l'uso della **Sidebar** e delle **Colonne**. La sidebar è utile per inserire widget che controllano le impostazioni dell'applicazione, mentre le colonne permettono di organizzare il contenuto in modo più strutturato.

Sidebar

La sidebar può essere utilizzata per raggruppare i widget in un'area separata, migliorando l'esperienza utente.

```
with st.sidebar:
    st.header("Impostazioni")
    valore_slider = st.slider("Seleziona un valore", 0,
        100)
```

Colonne

Le colonne possono essere create per affiancare diversi elementi dell'interfaccia utente, facilitando la visualizzazione simultanea di informazioni correlate.

```
col1 , col2 = st.columns(2)

with col1:
    st.header("Colonna Sinistra")
    st.button("Pulsante A")

with col2:
    st.header("Colonna Destra")
    st.button("Pulsante B")
```

I widget di Streamlit costituiscono un elemento essenziale per la creazione di applicazioni web interattive, grazie alla loro capacità di semplificare l'interazione con i dati e migliorare l'esperienza utente. La varietà di widget offerti e la loro intuitiva implementazione permettono agli sviluppatori di progettare interfacce funzionali in tempi ridotti, favorendo un'interazione dinamica con i dati. Inoltre, l'integrazione di funzionalità avanzate come la gestione del layout e il caching contribuiscono a ottimizzare le prestazioni delle applicazioni, consolidando Streamlit come una soluzione ideale per lo sviluppo di applicazioni basate su dati utilizzando Python [22].

3.3 Open VPN connect

OpenVPN Connect è un software client VPN (Virtual Private Network) che permette di stabilire connessioni sicure a reti private attraverso Internet. La principale funzione di OpenVPN Connect è quella di creare un tunnel criptato tra il client (come il computer dell'utente) e il server VPN, garantendo così la privacy e la sicurezza dei dati trasmessi. Questo tipo di connessione è essenziale quando si ha

la necessità di accedere a risorse private o protette da una rete locale, come nel caso di un server universitario o aziendale [23].

3.3.1 Come funziona

OpenVPN Connect utilizza il protocollo OpenVPN, che è uno dei più diffusi e sicuri per la creazione di connessioni VPN. La connessione sicura è garantita attraverso l'uso di crittografia SSL/TLS, che protegge i dati in transito da potenziali intercettazioni o manomissioni. Quando un utente si connette a una rete VPN utilizzando OpenVPN Connect, viene stabilito un tunnel virtuale che può essere utilizzato per inviare e ricevere dati in modo sicuro.

3.3.2 Tunnel SSH

OpenVPN Connect può essere utilizzato per creare un tunnel SSH (Secure Shell) verso un server remoto, come nel caso di un server dell'università. SSH è un protocollo di rete sicuro che permette di eseguire operazioni da remoto su un server, garantendo la protezione dei dati e l'autenticazione dell'utente.

Per connettersi a un server universitario tramite tunnel SSH, è necessario prima stabilire una connessione sicura tramite OpenVPN Connect. Una volta che il tunnel VPN è attivo, è possibile utilizzare il protocollo SSH per accedere al server in modo sicuro, come se si fosse fisicamente presenti sulla rete locale dell'università. Questo approccio è particolarmente utile per l'accesso remoto a server che richiedono una connessione sicura e autenticata, evitando esposizioni dirette alla rete pubblica.

Capitolo 4

Implementazione

In questo capitolo esaminerò nel dettaglio il funzionamento del progetto, quindi la dashboard, nelle sue componenti e modalità di navigazione. Verranno inoltre trattate alcune problematiche riscontrate e come sono state risolte.

4.1 Dashboard

La dashboard ha l'obiettivo di visualizzare nella maniera più diretta e semplice possibile una serie di informazioni e statistiche riguardo alcuni oggetti del Web3 come Collezioni, Wallet e NFT.

4.1.1 Menu

La dashboard è suddivisa in due pagine principali, accedibili tramite la barra di navigazione di Streamlit presente a sinistra della pagina, una pagina principale, da cui iniziare la navigazione, e una dedicata esclusivamente al wallet che verrà eventualmente selezionato nella prima pagina.

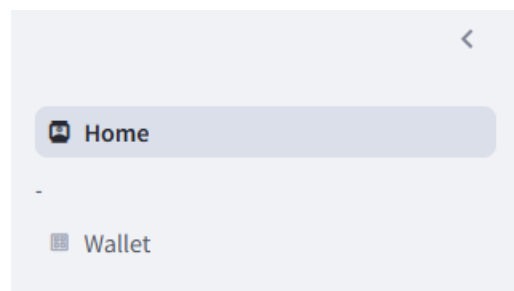


Figura 5: Menu dashboard

4.1.2 Pagina principale

Come prima cosa viene mostrata una barra di ricerca per cercare una collezione. Essa presenterà una funzione di auto-completamento per facilitare la ricerca della collezione desiderata. (L'auto-completamento verrà approfondito nella sezione 4.2.2).

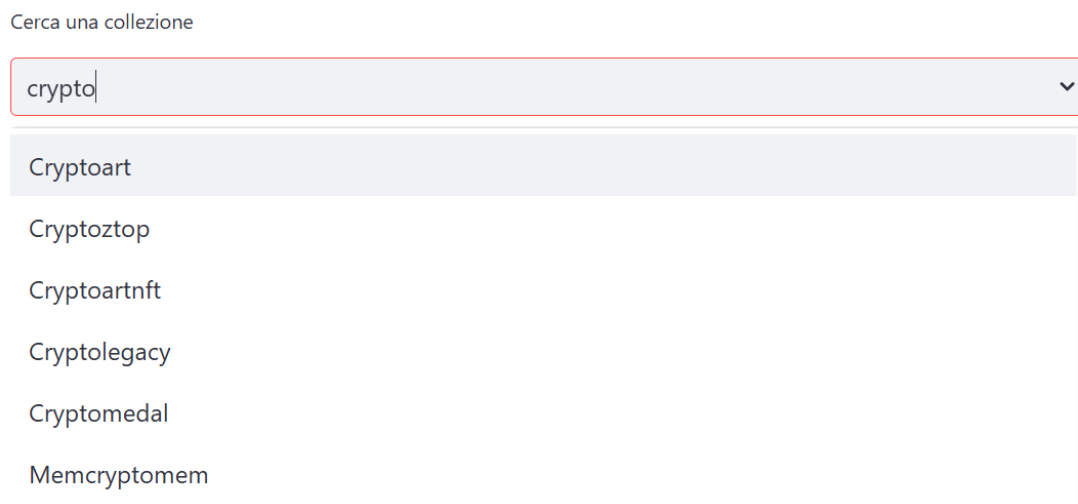


Figura 6: Barra di ricerca delle collezioni

Una volta selezionata la collezione desiderata, verranno caricate tutte le informazioni associate, a partire dal summary, che riassume i dati più rilevanti della collezione.

Successivamente, viene visualizzato un grafico che rappresenta l'andamento temporale delle transazioni della collezione. Il grafico include una funzionalità



Figura 7: Riepilogo della collezione

di ricampionamento dinamico (4.2.3), che consente di modificare la frequenza di aggregazione dei dati tra valori orari, giornalieri, settimanali o mensili. Questo processo avviene in modo efficiente, aggiornando esclusivamente il grafico senza ricaricare l'intera pagina.

Transazioni



Figura 8: Transazioni della collezione

Dopo un `st.divider()`, utile dal punto di vista del design per indicare visivamente un cambio di contenuto o argomento tramite una linea divisoria, si accede alla sezione dedicata agli NFT, che include un riepilogo delle informazioni principali e un grafico delle relative transazioni.

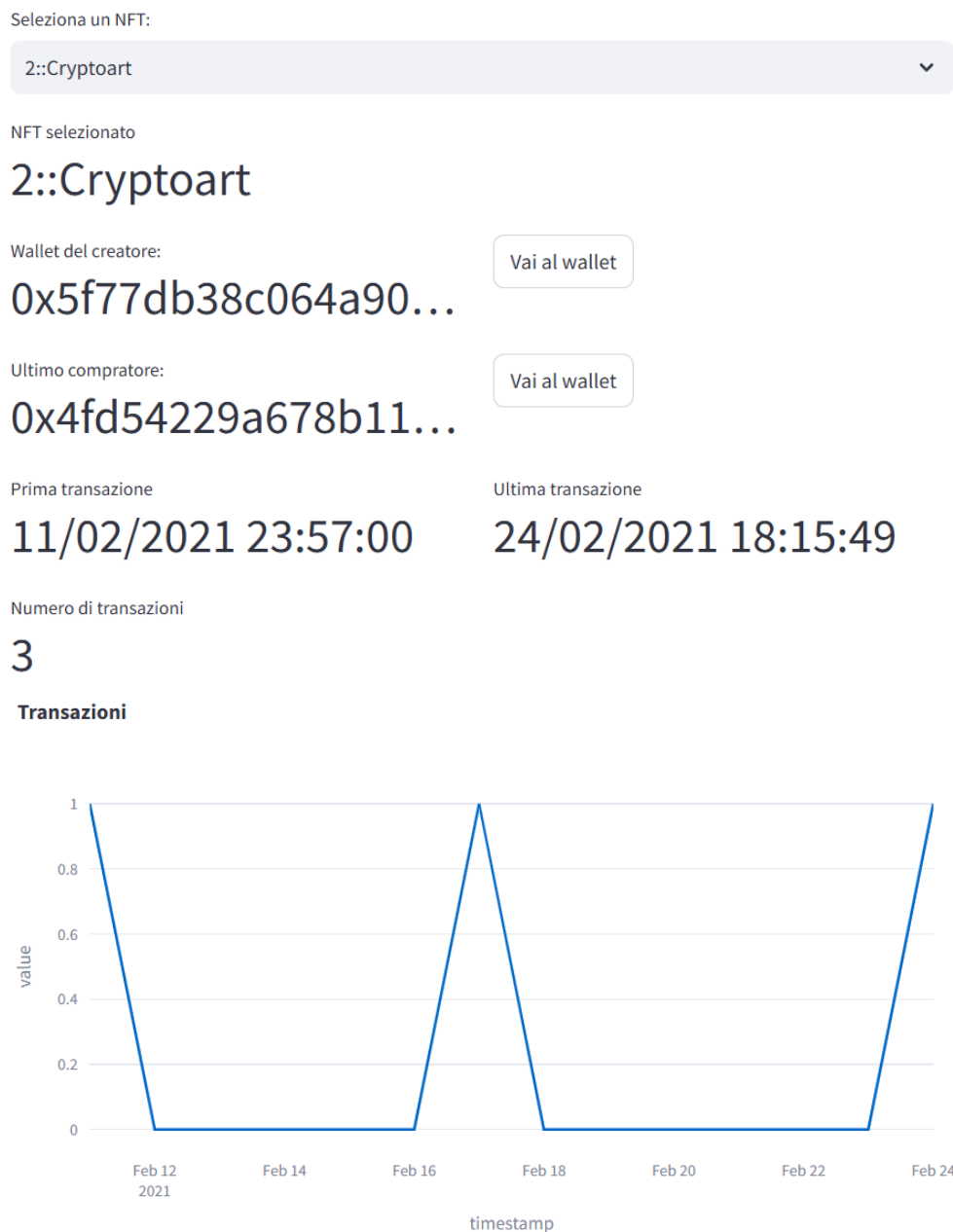


Figura 9: Riepilogo e transazioni degli NFT

Accanto all'ID del wallet del creatore e a quello dell'ultimo acquirente è presente un pulsante che consente all'utente di accedere alla sezione della dashboard dedicata al wallet selezionato.

4.1.3 Pagina del wallet

La pagina dedicata al wallet si apre con un menu di navigazione orizzontale, progettato per migliorare l'usabilità e rendere la consultazione più agevole. Questo approccio evita che la pagina diventi eccessivamente lunga e difficile da navigare, oltre a permettere il caricamento progressivo dei contenuti, riducendo il rischio di artefatti visivi indesiderati. Questa soluzione verrà approfondita successivamente (4.2.1).



Dettagli del wallet Acquisti Vendite NFT Posseduti + guadagni

Figura 10: Menu navigazione pagina wallet

Nel caso in cui il wallet selezionato non presenti transazioni, l'utente verrà notificato.

Il wallet

0xa4aafa650d30d3cefa2a13313be7ca35aab74267

non ha transazioni

Figura 11: Messaggio in assenza di transazioni

In caso contrario verranno mostrate le informazioni relative alla pagina selezionata.

Dettagli del wallet

La pagina principale relativa al wallet mostra le informazioni principali e le sue transazioni.



Dettagli del wallet

Wallet ID:

0x007880443b595eb375ab6b6566ad9a5263...

Primo acquisto

28/06/2018 16:05:01

Ultima vendita

12/04/2021 16:55:06

Transazioni d'acquisto totali

123

Acquisti totali

\$25665.51

Transazioni di vendita totali

96

Vendite totali

\$33933.76

Seleziona la frequenza di campionamento:

settimanale



Transazioni del wallet

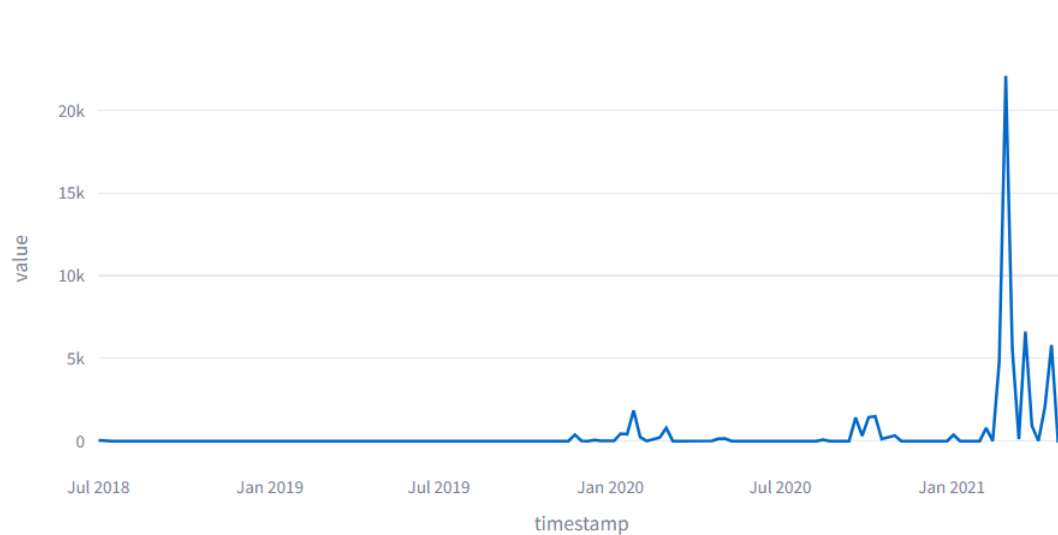


Figura 12: Dettagli e transazioni del wallet

Acquisti

Sezione dedicata agli acquisti, dove viene mostrato l'elenco degli NFT acquistati, con il relativo riepilogo e il grafico delle transazioni.

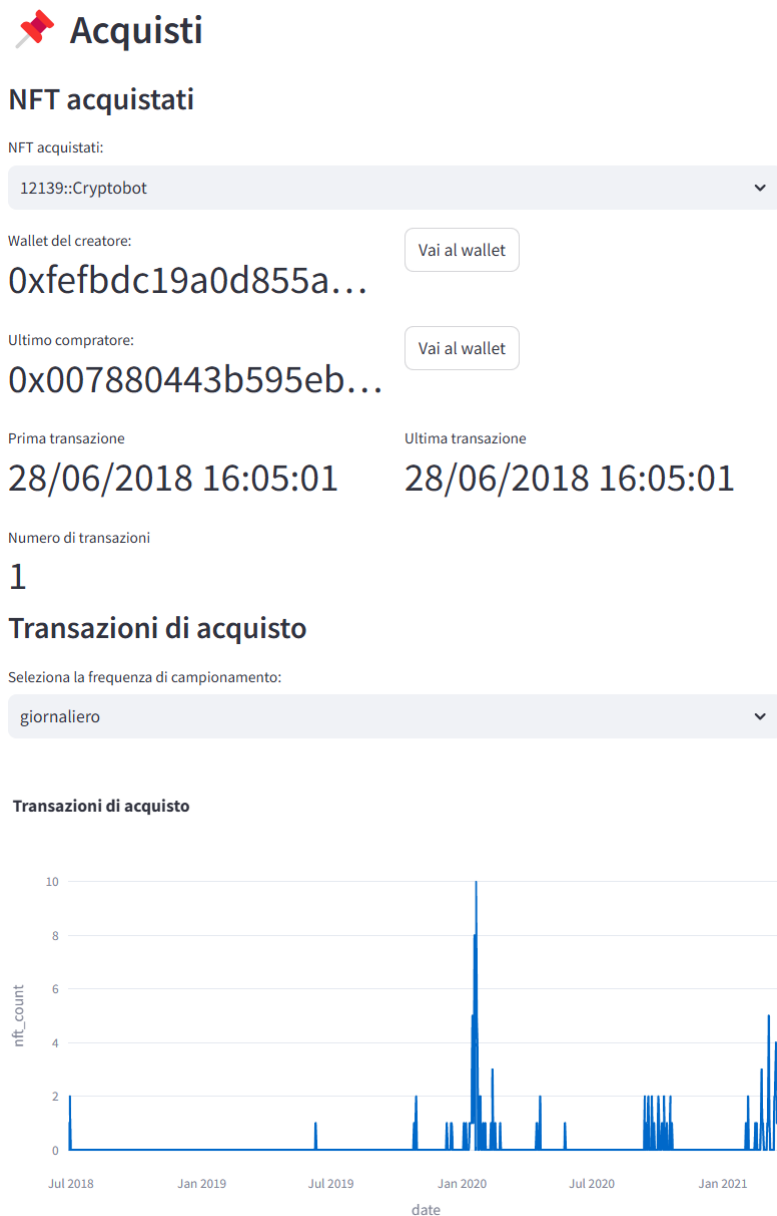


Figura 13: Scheda acquisti del wallet - dettagli e transazioni

Vendite

Sezione dedicata alle vendite, che include l'elenco degli NFT venduti, il loro riepilogo e il grafico delle transazioni.

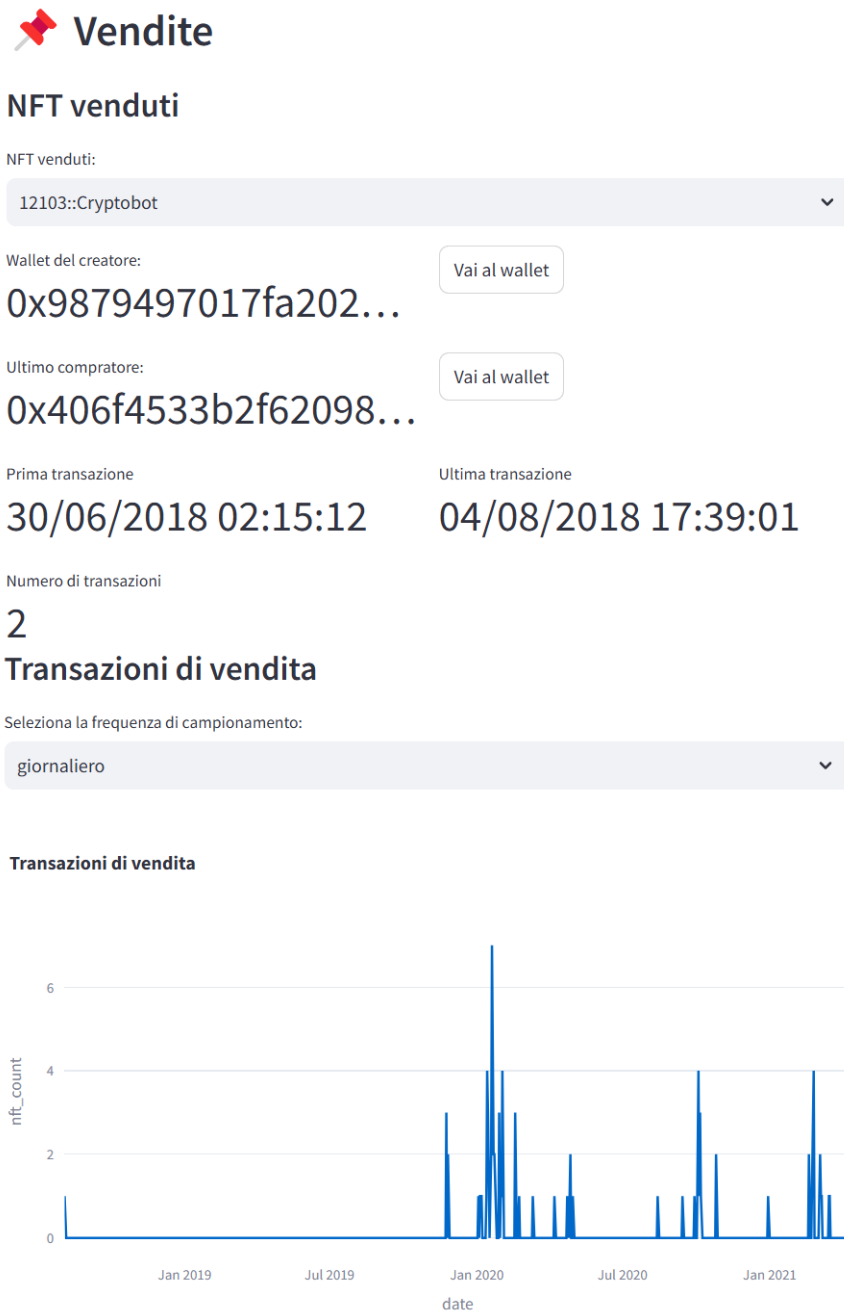



Figura 14: Scheda vendite del wallet - dettagli e transazioni

NFT posseduti e guadagni

Questa sezione elenca gli NFT attualmente posseduti (alla data dell'ultima registrazione), fornendo un riepilogo delle informazioni principali. Inoltre, include una tabella che mostra il prezzo di acquisto e di vendita per ciascun NFT, evidenziando l'eventuale guadagno o perdita associata.


Lista degli NFT attualmente posseduti

NFT attualmente posseduti:

12139::Cryptobot

Wallet del creatore:

0xfefbdc19a0d855a...

Vai al wallet

Ultimo compratore:

0x007880443b595eb...

Vai al wallet

Prima transazione

Ultima transazione

28/06/2018 16:05:01
28/06/2018 16:05:01

Numero di transazioni

1

Guadagni

	NFT	Prezzo Acquisto	Prezzo Vendita	Guadagno
0	12103::Cryptobot	\$1.22	\$2.79	\$1.57
1	510::Color	\$2.71	\$7.96	\$5.25
2	2983::Cryptovoxel	\$61.95	\$234.20	\$172.25
3	21::Chainface	\$8.38	\$82.97	\$74.59
4	173::Chainface	\$16.76	\$70.45	\$53.70
5	962::Bitsforai	\$3.43	\$5.58	\$2.15
6	1627007::Cryptokittie	\$7.45	\$13.13	\$5.69
7	1627007::Cryptokittie	\$7.45	\$13.13	\$5.69
8	5480::World	\$0.02	\$0.20	\$0.18
9	1137702::Cryptokittie	\$71.44	\$88.75	\$17.31

Figura 15: Scheda NFT posseduti e guadagni del wallet

4.2 Problematiche risolte

Le principali difficoltà affrontate e successivamente risolte nel corso del progetto riguardano principalmente il design dell'applicazione, al fine di garantire una navigazione fluida e intuitiva all'interno della dashboard, e l'implementazione del ricampionamento dinamico dei grafici temporali, basati su pandas e plotly. Questo ricampionamento consente di visualizzare i dati su diverse granularità temporali, come ora, giorno, settimana e mese, permettendo all'utente di modificare la visualizzazione del grafico senza la necessità di ricaricare la pagina. Tali soluzioni sono state adottate per soddisfare le varie esigenze di visualizzazione degli utenti, assicurando al contempo una gestione efficiente delle risorse.

4.2.1 Facilitare la navigazione

Una problematica di design relativa alla pagina del wallet riguarda la sua navigazione. Inizialmente, le sue schede **Dettagli del wallet**, **Acquisti**, **Vendite**, **NFT posseduti & Guadagni** erano parte di un'unica pagina. Tuttavia, questa soluzione rendeva la navigazione poco pratica: la pagina risultava eccessivamente lunga e difficoltosa da consultare, rendendo necessaria un'alternativa più efficiente. Ho inizialmente considerato una navigazione intra-pagina tramite un menu che consentisse di scorrere direttamente alla sezione desiderata. Tuttavia, **Streamlit** non supporta ancora questa funzionalità, sebbene potrebbe introdurla in futuro, dato che nei suoi forum le nuove feature vengono prioritarizzate in base alle richieste degli utenti [24]. Ovviamente, sarebbe stato possibile implementarla manualmente tramite **JavaScript**, ma ai fini di questo progetto ho preferito mantenere un certo grado di coerenza, utilizzando esclusivamente soluzioni già supportate da **Streamlit**.

Di conseguenza, ho scelto di adottare `st.tabs`, implementando una navigazione a schede. Questa soluzione non solo ha reso l'interfaccia più pulita e intuitiva, ma ha anche migliorato le prestazioni: mentre l'utente visualizza la scheda **Dettagli del wallet**, le chiamate API relative a **Acquisti**, **Vendite** e **NFT posseduti & Guadagni** possono essere eseguite in background, riducendo i tempi di caricamento e

migliorando l'esperienza utente.

Il seguente frammento di codice mostra come sono state definite le schede della pagina del wallet:

```
tab1, tab2, tab3, tab4 = st.tabs(["Dettagli del wallet", "
    Acquisti", "Vendite", "NFT Posseduti & Guadagni"])

with tab1:
    st.title("Dettagli del wallet")
    ...
with tab2:
    st.header("Acquisti")
    ...
with tab3:
    st.header("Vendita")
    ...
with tab4:
    st.header("Lista degli NFT attualmente posseduti &
        Guadagni")
    ...
```

4.2.2 Auto-completamento

Nella pagina principale della dashboard (4.1.2) è stata implementata la funzionalità di auto-completamento nella barra di ricerca delle collezioni NFT. Questo permette agli utenti di trovare rapidamente una collezione specifica digitando parte del suo nome. Questa funzionalità è stata implementata tramite il seguente codice:

```
collection_list = collections["collections"]

def search_callback(query):
    return [collection for collection in collection_list if
        query.lower() in collection.lower()]
```

```

selected_collection = st_searchbox(
    search_callback ,
    key="searchbox" ,
    label="Cerca una collezione" ,
    clear_on_submit=True
)

```

Inizialmente, la lista delle collezioni viene estratta dalla risposta ottenuta dalla chiamata API e memorizzata nella variabile `collection_list`. Successivamente, si definisce la funzione `search_callback`, che riceve una query (il testo inserito dall'utente) e restituisce le collezioni il cui nome contiene la stringa digitata, ignorando maiuscole e minuscole.

Infine, viene creata la barra di ricerca `st_searchbox`, a cui viene passata come parametro la funzione `search_callback`. L'opzione `clear_on_submit=True` consente di svuotare automaticamente la barra dopo la selezione di un'opzione, migliorando l'usabilità.

4.2.3 Creazione grafici e ricampionamento dinamico

Nel file contenente il codice condiviso tra più componenti della webapp, ho definito un dizionario denominato `RESAMPLING_MAP`. Questo dizionario funge da mappatura tra l'opzione selezionata dall'utente nella `selectbox` di `Streamlit` e la stringa di formato corrispondente utilizzata per specificare la frequenza di ricampionamento nei metodi di `pandas`.

```

RESAMPLING_MAP = {
    "orario": "h",
    "giornaliero": "D",
    "settimanale": "W",
    "mensile": "ME"
}

```

Questa struttura consente di tradurre in modo chiaro e modulare le scelte dell'utente in parametri compatibili con il metodo `resample()` di `pandas`, che verrà utilizzato per aggregare i dati temporali con la frequenza desiderata.

Nel file `wallet.py`, il ricampionamento dinamico del grafico viene implementato, ad esempio, all'interno della funzione `get_nft_transactions_and_graph`. Questa funzione permette di selezionare un NFT e definire un intervallo temporale entro cui analizzare le transazioni associate. La selezione della frequenza di aggregazione avviene tramite una `selectbox`, che popola le proprie opzioni utilizzando le chiavi di `RESAMPLING_MAP`.

Dopo aver selezionato la frequenza di campionamento, il dataset delle transazioni viene caricato e preprocessato. Per evitare chiamate API ridondanti e migliorare le prestazioni, i dati vengono memorizzati all'interno della `session_state` di `Streamlit`. Se il dataset non è ancora stato caricato, la funzione `get_nft_transactions` viene invocata con la frequenza di default (`'D'`, corrispondente a "giornaliero"), e il risultato viene trasformato in un `DataFrame` di `pandas`, indicizzando le date per consentire operazioni di aggregazione efficienti.

Una volta caricato il dataset, il `DataFrame` viene ricampionato sulla base della selezione dell'utente, sfruttando il valore corrispondente nel dizionario `RESAMPLING_MAP`. Questo consente di aggregare i dati con la granularità desiderata (oraria, giornaliera, settimanale, mensile). Infine, il grafico viene generato utilizzando `Plotly` e visualizzato all'interno della webapp. Sull'asse delle X vengono mostrate le date mentre sull'asse delle Y vengono mostrati i valori aggregati delle transazioni. Questa implementazione migliora l'esperienza utente e le prestazioni del sistema, riducendo il numero di chiamate API e garantendo una visualizzazione fluida e interattiva dei dati.

```
def get_nft_transactions_and_graph(selected_nft , min_date ,
    max_date):

    resampling = st.selectbox(
        "Seleziona la frequenza di campionamento:" ,
        list(RESAMPLING_MAP.keys()) ,
```

```

        index=1, # Default: "giornaliero"
        key="resampling_select"
    )

    if "nft_df" not in st.session_state:
        data = get_nft_transactions(selected_nft, min_date,
                                    max_date, 'D')
        df = pd.DataFrame(list(data.items()), columns=['
            timestamp', 'value'])
        df['timestamp'] = pd.to_datetime(df['timestamp'])
        df = df.set_index('timestamp').sort_index()
        st.session_state["nft_df"] = df

    df = st.session_state["nft_df"].resample(RESAMPLING_MAP
        [resampling]).sum()
    fig = px.line(df, x=df.index, y='value', title="title")
    st.plotly_chart(fig)

```

Capitolo 5

Conclusioni e sviluppi futuri

Questa tesi ha esplorato il contesto del Web3, con particolare attenzione alle blockchain e agli NFT, e ha portato alla realizzazione di una dashboard di analytics per l'analisi dei dati provenienti da questo ecosistema. Il progetto ha dimostrato come sia possibile rendere più accessibili e interpretabili le informazioni relative alle collezioni NFT, ai wallet e alle transazioni, grazie a un'interfaccia chiara e interattiva sviluppata con Python e Streamlit.

Durante lo sviluppo sono state affrontate e risolte diverse sfide, tra cui l'ottimizzazione della navigazione della webapp, l'implementazione di funzionalità avanzate come l'autocompletamento nella ricerca e il ricampionamento dinamico dei dati nei grafici. Questi miglioramenti hanno contribuito a rendere la dashboard uno strumento efficace per utenti interessati all'analisi del mercato NFT.

Nonostante i risultati ottenuti, la dashboard presenta svariate prospettive di miglioramento. Si potrebbero integrare dati provenienti da più blockchain, con la possibilità di scegliere tra una e l'altra e strumenti di analisi più sofisticati, ad esempio basati su machine learning. Inoltre future evoluzioni potrebbero includere funzionalità di monitoraggio in tempo reale delle transazioni e un maggior livello di personalizzazione dell'esperienza utente.

Infine, questo lavoro ha mostrato come l'utilizzo di strumenti di analisi ben progettati possa facilitare la fruizione e la comprensione dei dati e delle dinamiche del Web3.

Bibliografia

- [1] Web3 - young platform. <https://youngplatform.com/glossary/web3/>. [Accessed 08-03-2025].
- [2] Fastweb - web3, cos'è e come funziona. <https://www.fastweb.it/fastweb-plus/digital-dev-security/web3-cose-e-come-funziona/>. [Accessed 11-03-2025].
- [3] AWS - cos'è la tecnologia blockchain. <https://aws.amazon.com/it/what-is/blockchain/?aws-products-all.sort-by=item.additionalFields.productNameLowercase&aws-products-all.sort-order=asc>. [Accessed 12-03-2025].
- [4] IBM - cos'è la blockchain. <https://www.ibm.com/it-it/topics/blockchain>. [Accessed 12-03-2025].
- [5] Unicusano - NFT: cosa sono e come funzionano. <https://www.unicusano.it/blog/universita/nft-cosa-sono-e-come-funzionano/>. [Accessed 12-03-2025].
- [6] forbes- le migliori piattaforme dove comprare nft. https://www.forbes.com/advisor/it/investire/criptovalute/migliori-nft-marketplace/#scrollto_cose_un_marketplace_di_nft_section. [Accessed 12-03-2025].
- [7] Coindesk- what is magic eden? how to get started on the NFT marketplace. <https://www.coindesk.com/learn/what-is-magic-eden-how-to-get-started-on-the-nft-marketplace>. [Accessed 12-03-2025].

- [8] NFT news today - how magic eden became the top solana NFT marketplace. <https://nftnewstoday.com/2024/07/17/how-magic-eden-became-the-top-solana-nft-marketplace>. [Accessed 12-03-2025].
- [9] Dappradar - NFT winter returns? market shrinks 50% as crypto prices fall. <https://dappradar.com/blog/nft-winter-returns-market-shrinks-50-as-crypto-prices-fall/#Chapter-5>. [Accessed 13-03-2025].
- [10] Coinbase - quali sono le differenze tra gli NFT in edizione aperta e quelli in edizione limitata? <https://www.coinbase.com/it/learn/crypto-glossary/what-are-the-differences-between-open-edition-and-limited-edition-nfts>. [Accessed 12-03-2025].
- [11] CryptoPunks. <https://cryptopunks.app/>. [Accessed 14-03-2025].
- [12] Pudgy penguins. <https://pudgypenguins.com/>. [Accessed 14-03-2025].
- [13] Bit2me - le 10 migliori collezioni NFT sul mercato: Pudgy penguins supera bored ape. <https://news.bit2me.com/it/Le-10-collezioni-nft-pi%C3%B9-importanti>. [Accessed 14-03-2025].
- [14] Bored ape yacht club. <https://boredapeyachtclub.com/>. [Accessed 14-03-2025].
- [15] Kinsta blog. <https://kinsta.com/it/blog/tutorial-python/>. [Accessed 11-03-2025].
- [16] Python documentation. <https://docs.python.it/>. [Accessed 11-03-2025].
- [17] Libreria requests. <https://requests.readthedocs.io/projects/it/it/latest/>. [Accessed 11-03-2025].
- [18] Libreria datetime. <https://docs.python.org/3/library/datetime.html>. [Accessed 11-03-2025].

- [19] Libreria pandas. <https://pandas.pydata.org/about/>. [Accessed 11-03-2025].
- [20] Libreria plotly. <https://plotly.com/python/>. [Accessed 11-03-2025].
- [21] Tyler Richards. *Streamlit for Data Science*. Packt Publishing Ltd., 2021.
- [22] Streamlit Inc. Streamlit concepts. <https://docs.streamlit.io/develop/concepts>. [Accessed 08-03-2025].
- [23] Open vpn connect. <https://openvpn.net/connect-docs/user-guide.html>. [Accessed 09-03-2025].
- [24] Streamlit discuss - scroll to page section that is being rendered. <https://discuss.streamlit.io/t/scroll-to-page-section-that-is-being-rendered/77822>. [Accessed 15-03-2025].