Università degli Studi di Milano - Bicocca
Scuola di Scienze
Dipartimento di Informatica, Sistemistica e Comunicazione
Corso di laurea in Informatica

Strategie di automatizzazione di liquidità nella finanza decentralizzata

Relatore: Prof. Alberto Leporati

Correlatore: Ing. Paolo Antonio Rossi

Relazione della prova finale di:
Christian Kobril
Matricola 856448

Anno Accademico 2021-2022

Sommario

Qui scrivo il sommario

Indice

Capitolo 1

Introduzione

Il tema centrale di questa Tesi di Laurea è la **finanza decentralizzata**.
In particolare, si approfondirà il ruolo che la blockchain ha in questo settore, fornendo esempi di un prodotto software concretamente sviluppato durante il mio *Project Work*, svolto presso l'azienda *Five Elements Labs Srl*.

Inoltre, verranno approfonditi i concetti di *pool di liquidità*, *automated market maker(AMM)* e *strategie multiple* su Uniswap v3.

1.1 Un breve sguardo alla tecnologia

Alla base di tutti i protocolli e prodotti software analizzati in questa Tesi, vi è un minimo comune denominatore che prende il nome di **blockchain**[?].
La blockchain altro non è che un database (o in termini contabili, *libro mastro*) **condiviso** tra più nodi di una rete, **validati** dalla rete stessa e strutturato come una catena di blocchi contenenti delle **transazioni**.

Sostanzialmente possiamo riassumere le funzionalità di questa rete di blocchi in 3 importanti concetti:

- **tracciabilità** - tutti i partecipanti alla rete e le transazioni registrate devono poter essere rintracciabili
- **immutabilità** - una volta che una transazione viene salvata sulla blockchain, nessun partecipante alla rete può in alcun modo alterarne il contenuto
- **sicurezza** - la rete è realizzata mediante sistemi crittografici che ne garantiscono la sicurezza

Questa recente tecnologia è oggi in **piena crescita**: nonostante il mercato finanziario digitale non sia nelle migliori condizioni, lascia spazio a molte aziende e sviluppatori indipendenti per costruire prodotti innovativi e ad altissimo potenziale.

1.2 Cos'è la Finanza Decentralizzata

Il termine **finanza decentralizzata** viene usato per classificare tutti quei servizi finanziari che avvengono direttamente tra due entità su una blockchain.

1.2.1 Differenze tra Finanza Decentralizzata e Tradizionale

Per comprendere il concetto su cui si basa la finanza decentralizzata (da ora in poi *DeFi*), è bene dare un rapido sguardo alla sua controparte: la finanza tradizionale (o centralizzata).

Nella finanza centralizzata, ogni singola operazione finanziaria tra due persone (bonifici, prestiti, scambio di risorse, mutui) richiede l'interazione con soggetti di terze parti (tipicamente banche o altri enti).
Ciò incrementa le già prolisse tempistiche burocratiche, oltre ad aggiungere i costi dovuti al servizio fornito dagli enti che permettono l'operazione.

Invece, la finanza decentralizzata permette l'interazione tra due soggetti senza l'intermediazione di un sistema centralizzato, bensì mediante un applicativo software costruito sopra la tecnologia blockchain, rendendo le operazioni rapide, pubbliche e sicure.

1.3 Caratteristiche e vantaggi della DeFi

1.3.1 Applicazioni decentralizzate

Gli applicativi software utilizzati in DeFi vengono chiamati *dApps (decentralized applications)*, ovvero particolari prodotti che interagiscono con diverse blockchain. Di tali reti noi considereremo solo quella di **Ethereum**[?], nota per la sua flessibilità e accessibilità.

La blockchain di Ethereum, a differenza di reti come quella di **Bitcoin**[?], è **programmabile**; ossia è possibile costruirci sopra e distribuire dApps, rendendo la rete una sorta di gigantesco marketplace in cui trovare servizi finanziari, videogiochi, social network e diverse altre applicazioni.
Su tale blockchain risiedono dei particolari programmi denominati *Smart Contracts*[?], i quali si occupano di garantire sicurezza, trasparenza e irreversibilità delle operazioni avvenute sulla blockchain.

1.3.2 Accesso alle dApps

Il concetto di login ideato nel web2, tipicamente caratterizzato dall'inserimento di un'e-mail e una password, viene sorpassato da una nuova autenticazione del **web3**[?]: attraverso il proprio *portafoglio digitale* (da ora in poi, wallet[?]) è possibile accedere al proprio account e gestire i propri assets digitali, eventualmente mettendoli a disposizione della dApp a cui si è connessi.

I wallet presentano il vantaggio di non dover fornire nomi, indirizzi fisici, email o altre informazioni personali, garantendo la riservatezza dei propri dati; basta creare un wallet per avere immediato accesso alle piattaforme, senza registrazioni.

1.3.3 Operazioni in DeFi

Ogni operazione nella DeFi viene detta **transazione**. Una transazione è permanentemente salvata sui registri della blockchain, rendendo ogni singola operazione, associata ad un identificativo, consultabile in qualsiasi momento da qualsiasi persona.
Tale trasparenza viene difficilmente concessa dalle banche, ponendo la DeFi come un sistema aperto e rintracciabile.

1.3.4 Flessibilità

L'ultima caratteristica della DeFi che ritengo importante citare è ciò che più la contraddistingue dalla finanza tradizionale: la sua flessibilità.
In qualsiasi momento un utente può trasferire i propri assets digitali, senza dover chiedere il permesso a soggetti di terze parti, evitando costose commissioni e con un'attesa che va da pochi secondi a pochi minuti.

Capitolo 2

Uniswap, piattaforma di Liquidity Providing

Avendo approfondito cosa è la DeFi, quali sono le sue caratteristiche e i principali vantaggi, ritengo necessario concentrarsi su quali sono i prodotti "dApps" che hanno messo le basi per il lavoro svolto durante il mio Project Work.

2.1 Uniswap

Innanzitutto, è bene distinguere la piattaforma Uniswap[?] dall'omonimo protocollo.
La dApp di Uniswap (conosciuta come Uniswap Interface) è una piattaforma che permette agli utenti l'interazione con il protocollo Uniswap.

Quest'ultimo è una suite di Smart Contracts, per definizione persistenti e non aggiornabili, che insieme formano un **Automated Market Maker** (da ora in poi AMM)[?].

2.2 Scambi nei mercati tradizionali

La maggior parte dei mercati tradizionali ad accesso pubblico utilizza ciò che viene definito *Order Book*[?], ossia un elenco degli ordini di acquisto e vendita attualmente aperti per un asset, organizzati per prezzo.
Sostanzialmente, un *sistema di corrispondenza*[?] si occupa di abbinare gli ordini di acquisto con quelli di vendita, usando l'order book per eseguire le operazioni tra i partecipanti dello scambio.

2.3 Automated Market Maker

La rivoluzione introdotta dalla blockchain sta nella possibilità di creare nuovi tipi di scambi che abbinano algoritmicamente ordini di acquisto e vendita utilizzando gli smart contracts.
Tali scambi vengono detti *Scambi Decentralizzati (DEX).*

Un AMM è un protocollo DEX che si basa su un algoritmo di valutazione per prezzare

gli asset mediante una formula matematica.
Il citato order book viene rimpiazzato con una pool di liquidità[?], contenente due asset, entrambi valutati l'uno rispetto all'altro.
Quando un asset viene scambiato per un altro, i prezzi relativi dei due asset cambiano, e viene determinato un nuovo tasso di mercato per entrambi. In questo modo acquirenti e venditori interagiscono direttamente con la pool (e di conseguenza gli smart contracts), senza dover interagire tra di loro in modo diretto.

2.3.1 Esempio pratico di AMM

Un esempio pratico che mi ha aiutato a comprendere il funzionamento degli AMM è quello dei contadini di mele e patate.
Immaginiamo di essere un contadino e di avere solo patate, senza la possibilità di coltivare, e di conseguenza mangiare, nient'altro.
Un giorno ci viene proposto di effettuare degli scambi con un venditore di mele attraverso un messaggero, il quale decide di custodire mele e patate in un contenitore magico, in modo tale che rimangano a disposizione senza marcire (e dunque perdere di valore).

La regola fondamentale per questo scambio è una sola: *il contenitore magico dovrà sempre contenere lo stesso valore di mele e patate.*
Tale regola è in realtà la formula alla base dell'AMM di Uniswap, conosciuta come **Constant Product Formula**:

$$x * y = k$$

dove x corrisponde al numero di mele e y al numero di patate nel contenitore; in genere k è detta **liquidità** del contenitore.
Inizialmente il contenitore sarà perfettamente bilanciato, per esempio con 500 mele e 500 patate, entrambe prezzate ad €1 per un valore totale (che in questo caso corrisponde alla liquidità) di €1.000.

Tuttavia, se un contadino volesse scambiare le sue patate grazie al contenitore, potrebbe ricevere in cambio meno mele rispetto alle patate inviate. Questo perché il prezzo della mela potrebbe aumentare, e dunque per bilanciare il contenitore il prezzo delle patate dovrà di conseguenza diminuire.
Allo stesso modo, se un contadino volesse scambiare le proprie mele, riceverebbe più patate di quelle che avrebbe ricevuto inizialmente, considerato l'aumento del prezzo della mela rispetto alle patate.

Nella realtà dei fatti, questo contenitore magico è conosciuto come *pool di liquidità.*

2.4 Pool di liquidità

È possibile vedere una pool di liquidità come uno spazio in cui i contandini (trader) possono mettere a disposizione la propria liquidità di mele e patate (criptovalute, nello specifico token ERC-20[?]); tali utenti vengono definiti fornitori di liquidità (*liquidity*

providers, da ora in poi LP).

Come ricompensa per la liquidità fornita, gli **LP** ricevono commissioni sulle operazioni che avvengono nella pool a cui partecipano. Tali commissioni si applicano sulle singole transazioni effettuate con la liquidità fornita da un LP, e possono variare di percentuale dal 0.01% fino all'1%.

Nel caso di Uniswap, gli LP depositano un valore equivalente di due token; per esempio, 50% ETH e 50% USDC nella pool ETH/USDC.

2.5 Protocollo Uniswap v3

Uniswap v3 è l'ultima versione del protocollo rilasciata da Uniswap nel maggio 2021. Tale protocollo definisce le funzionalità della suite di smart contracts con cui gli utenti interagiscono, introducendo importanti novità rispetto al suo predecessore, v2.

2.5.1 Posizioni

Utilizzando l'interfaccia Uniswap, gli utenti possono connettere il loro wallet personale per mettere a disposizione un certo ammontare di liquidità all'interno di una pool. Tale liquidità, come spiegato in precedenza, dovrà mantenere un'equa proporzione tra i due asset messi a disposizione: tale operazione viene definita come *apertura di una* ***posizione*** (o in inglese, position minting).

Su Uniswap v3, le posizioni vengono rappresentate mediante NFT (ERC-721[?]), i quali certificano un determinato wallet, in questo caso chi effettua il minting, come proprietario della posizione.

2.5.2 Complicazioni di Uniswap v2

In precedenza, nella v2 di Uniswap, i LP potevano mettere a disposizione i propri asset per scambi **a qualsiasi intervallo di prezzo**.

Consideriamo che io, trader che utilizza Uniswap, voglia mettere a disposizione due miei asset chiamati token A e token B. Ricordando che l'intervallo di prezzo che scelgo per aprire una posizione è sempre il prezzo di A rispetto al prezzo di B (A **su** B), decido di scegliere un range che copra tutti i prezzi possibili.
In questo modo non vi è alcuna perdita di liquidità, portando però un importante svantaggio: la maggior parte della liquidità non viene mai utilizzata negli scambi.

Provando a considerare una pool contenente una coppia di due stable coins, ossia token il cui prezzo rimane relativamente costante nel tempo, possiamo assumere che la liquidità al di fuori del tipico intervallo di prezzo dei suddetti stable coins non verrebbe mai toccata.

Per esempio in Uniswap v2 la coppia DAI/USDC (**entrambi stable coins dal valore**

di circa \$1) utilizza circa il 0.50% del capitale totale disponibile per gli scambi all'interno del range tra \$0.99 e \$1.01[?]. Il resto della liquidità è distribuito nella restante fascia di prezzo tra 0 e ∞ (escluso il range sopra citato), rendendo quel capitale inutilizzabile (e dunque, non consentendo agli LP di guadagnare commissioni).

Ciò è dovuto al fatto che la liquidità sia uniformemente distribuita in un range di prezzo da 0 a ∞, senza che sia *concentrata* nel giusto intervallo: per tale ragione è stato introdotto **Uniswap v3**.

2.5.3 Liquidità concentrata

Ciò che rende Uniswap v3 un protocollo davvero valido è l'idea della *Liquidità Concentrata*[?]. Tale liquidità viene distribuita in un intervallo di prezzo personalizzabile, a scelta dell'utente.

Riprendendo l'esempio sopra citato, un trader potrebbe decidere di investire nella pool DAI/USDC, scegliendo come range il più proficuo, ossia quello compreso tra \$0.99 e \$1.01. In tal modo, la liquidità concentrata garantirà un guadagno superiori di commissioni (da ora in poi fees) sfruttando il capitale messo a disposizione dai LPs.

2.5.4 Tick di prezzo

Per rendere la liquidità concentrata funzionale, lo spazio continuo del prezzo è stato partizionato in **tick**.

I tick sono i limiti di aree discrete nello spazio del prezzo. Tali limiti sono posizionati in modo tale che il diminuire o aumentare di 1 tick rappresenti l'aumento o la diminuzione percentuale del 0.01% del prezzo in ogni punto dello spazio.
Dunque quando una posizione viene creata, un LP non può scegliere qualsiasi valore per il range di prezzo: è necessario che il limite inferiore (**lower tick**) e il limite superiore (**upper tick**) corrispondano a dei tick di prezzo validi.

2.5.5 Swap e Fees

Il modo più utilizzato per interagire con Uniswap v3 è tramite gli scambi (da ora in poi **swap**). Uno swap è relativamente semplice: un utente seleziona un token ERC-20 del quale è proprietario e un token che vorrebbe scambiare per esso. Uniswap venderà il token attualmente in possesso dell'utente, restituendo una quantità proporzionale del token desiderato, sottraendo una **swap fee**, ossia quella percentuale riconosciuta ai LPs per aver messo a disposizione la loro liquidità con la quale è avvenuto lo scambio.

Tuttavia, la transazione potrebbe richiedere alcuni minuti, a seconda della rete su cui avviene, rilevando un fenomeno conosciuto come **slippage**.
Lo slippage è l'alterazione di prezzo di un token che avviene mentre la transazione è in attesa di essere completata. Tale alterazione ha una soglia di tolleranza dell'1% superata tale soglia l'operazione viene rifiutata e lo swap annullato, onde evitare grosse perdite per l'utente.

2.6 Complicazioni di Uniswap v3

Qualora il prezzo di un token dovesse muoversi verso una direzione (di discesa o di salita), il proprietario della posizione si ritroverebbe con un ammontare superiore di uno dei due token rispetto all'altro, in quanto il prezzo dell'uno sull'altro cambierà, fino a quando l'intera liquidità sarà relativa a solo uno dei due asset.

Per esempio, se in una pool ETH/USDC il prezzo di ETH dovesse diminuire, la percentuale di liquidità relativa a ETH aumenterebbe, per bilanciare il valore immesso all'interno della posizione stessa. Allo stesso modo, se ETH dovesse aumentare di valore, la percentuale di USDC aumenterebbe a sua volta.

Con l'aumentare o il diminuire del prezzo di un asset nella pool, tale prezzo potrebbe uscire dall'intervallo che un LP ha impostato per una certa posizione. Nel momento in cui una posizione si dovesse trovare fuori dall'intervallo scelto (**Out Of Range position**) la liquidità diventerebbe inattiva (**idle liquidity**) e l'utente proprietario di tale liquidità non guadagnerebbe più fees.
Tuttavia, nel momento in cui il valore del primo token sul secondo dovesse tornare nell'intervallo di prezzo iniziale, il LP tornerebbe a guadagnare fee.

È proprio dal problema della liquidità inattiva che nasce **Orbit DeFi**, il prodotto sviluppato durante il mio project work.

Capitolo 3

Orbit, piattaforma per automatizzare strategie DeFi

Ho avuto l'opportunità di svolgere il mio Project Work per Five Elements Labs, azienda specializzata nella produzione di software nel mondo blockchain; in particolare nei settori DeFi e NFT.
Durante tale esperienza, mi sono unito allo sviluppo del loro principale prodotto: **Orbit**[?].

3.1 Introduzione ad Orbit

Orbit è una piattaforma di gestione di liquidità nel settore DeFi: ossia un **layer** che permetta ai propri utenti di automatizzare strategie e di ottimizzare posizioni di liquidità su Uniswap v3.
Può essere interpretato come **un'estensione** del proprio wallet, in grado di gestire gli assets degli utenti per fornire la possibilità di ottenere un ritorno aggiuntivo dalla liquidità concentrata.

3.2 Perché nasce Orbit

Con il rapido crescere del settore DeFi e del corrispondente ecosistema di protocolli, ognuno con le proprie caratteristiche e logiche, risulta sempre più complesso gestire delle strategie di liquidità al passo con i tempi.

3.2.1 Gas fee

Ogni transazione avvenuta sulla blockchain ha un *"costo"* chiamato **gas fee**[?].
È possibile affermare che il gas sta alla blockchain come la **benzina** sta alla macchina; è necessario per far funzionare i nodi che compongono la rete.
Sostanzialmente è l'unità di misura dello *sforzo computazionale* fatto dai nodi per sostenere una transazione, la quale può prevedere diverse complesse operazioni al suo interno.
Tale benzina deve essere in qualche modo pagata, per questo esistono commissioni sul gas (gas fee).

Diverse blockchain con basse gas fee (come **Polygon**[?]) stanno diventando sempre più popolari, spostando l'attenzione degli sviluppatori e degli utenti verso la possibilità di costruire piattaforme e strumenti veloci ed efficaci.

Orbit dunque cavalca quest'onda di innovazione e di creatività, portando la liquidità concentrata verso un nuovo livello, costruendo uno strumento efficiente e facile da utilizzare per professionisti e novizi del mondo DeFi.

3.3 Vantaggi per gli utenti

Automatizzare strategie riguardo posizioni su Uniswap v3 ha un diretto impatto sui **ritorni generati** da queste ultime.
La maggior parte dei protocolli presenti sul mercato forniscono ai trader **strategie attive**, sulle quali è necessario compiere delle complesse scelte conosciute perlopiù da utenti professionisti.

Le funzionalità fornite da Orbit permettono agli utenti di creare **strategie passive**, senza il necessario bisogno di rimanere aggiornati sui protocolli, bensì lasciando alla piattaforma il compito di gestire la propria liquidità.
Inoltre, nella prima versione del protocollo, sarà Orbit ad occuparsi dei costi di gas dovuti alle transazioni rivolte agli smart contract dell'applicazione.

3.4 Modelli di gestione di liquidità

Nello stato attuale della DeFi, vi sono concretamente due modelli ben distinti di gestione di liquidità: **Aggregatori**[?] e **Smart Vaults**[?].

3.4.1 Aggregatori

Gli Aggregatori sono delle particolari piattaforme DeFi, le quali permettono ai propri utenti di effettuare transazioni verso diverse dApps **in un unico posto**. Ciò permette di risparmiare tempo e aumentare l'efficienza delle transazioni.

Tali Aggregatori permettono ai trader di **replicare** strategie di utenti esperti e di applicarle al proprio portfolio. Per esempio, possono confrontare i prezzi degli assets su diverse piattaforme, proponendo lo scambio più conveniente all'utente.

Tuttavia, protocolli utilizzatori di Aggregatori come *Yearn Finance*, *Beefy* o *Idle* consentono all'utente l'utilizzo di strategie singole, tipicamente con un modello "*Black Box*", ovvero senza verificarne l'effettivo funzionamento interno.

3.4.2 Smarts vaults

Contrariamente agli Aggregatori, gli **Smart Vaults** garantiscono un'alta **personalizzazione** delle strategie scelta direttamente dagli utenti.
Il funzionamento è relativamente semplice: un utente diventa *proprietario* di un particolare Smart Contract effettuando una transazione che ne crea un'istanza contenente l'indirizzo del suo creatore.[1].
Successivamente, il contratto creato viene utilizzato come *un'estensione* del proprio wallet per gestire assets e per interagire con altri protocolli per consentire allocazioni automatiche di liquidità: tale Smart Contract è chiamato Smart Vault.

Il modello a Smart Vault è stato poco utilizzato in passato, principalmente a causa delle alte gas fee richieste dalle reti per attivare strategie multiple.
Tuttavia, con l'avvento di blockchain sempre meno costose in termini di gas, è ora possibile utilizzare gli Smart Vaults per integrare facilmente nuovi protocolli e fornire all'utente la possibilità di avere un totale controllo sulle interazioni con essi.

Per tali ragioni, il modello a Smart Vault è stato scelto per la realizzazione di Orbit.

[1] Approfondimenti tecnici riguardanti la creazione dello Smart Vault e dei relativi Smart Contracts verranno trattati nel capitolo 5

Capitolo 4

Tecnologie utilizzate nello sviluppo di Orbit

Orbit è una piattaforma che prende forma nel mondo del Web3, pertanto per la sua realizzazione sono state richieste tecnologie specifiche di questo emergente settore.
Possiamo logicamente suddividere Orbit in due macro parti: il **frontend**, ossia l'interfaccia della dApp[?] con la quale l'utente interagisce direttamente, ed il **backend** composto da una suite di contratti che racchiudono le logiche e meccanismi su cui Orbit si basa.

4.1 Tecnologie Frontend e librerie utilizzate

La scelta del linguaggio utilizzato per la dApp Orbit, trattandosi di un'applicazione web, è ricaduta sul linguaggio **Javascript**[?].
Le caratteristiche del linguaggio, quali *versatilità, leggerezza e facilità d'apprendimento* hanno condizionato questa scelta.

Inoltre, a seguito di uno studio di mercato riguardante le librerie utilizzate dalle moderne dApps, incrociato con le competenze degli sviluppatori del team di Five Elements Labs, sono state selezionate una serie di librerie coerenti con lo il linguaggio scelto.

Lo sviluppo del frontend di Orbit e la relativa interazione con wallet e smart contracts è stato il principale compito affidatomi durante il mio Project Work.

4.1.1 Elementi dell'interfaccia

Per la creazione degli elementi che formano l'interfaccia di Orbit, è stata scelto **ReactJS**[?], una libreria Javascript per la sviluppo di **Single Page Application** (SPA[?]), che permette la creazione di componenti flessibili e riutilizzabili, nonché la più utilizzata nel 2021 tra le librerie web secondo *StackOverflow*[?].
Per quanto riguarda invece lo stile dei componenti è stata utilizzata **tailwind**[?] un framework CSS che permette di utilizzare classi per il layout, colori, tipografia e altre caratteristiche stilistiche dell'interfaccia grafica.

4.1.2 Interazione con gli Smart Contracts

Ciò che più differenzia un'applicazione web2 da una dApp in web3 è l'interazione con gli Smart Contracts.
La libreria scelta per tale interazione è stata **ethers.js**[?]: come suggerisce il nome, si tratta di una libreria Javascript che mira a facilitare l'interazione con la Blockchain di Ethereum (nel nostro caso, Polygon).
Tra le sue molteplici funzionalità, vi sono la *connessione di un wallet* alla dApp, la *creazione di istanze di oggetti che rappresentano Smart Contracts* utilizzabili direttamente nel Frontend ed infine la *firma dei messaggi relativi alle transazioni.*

4.2 Connessione wallet to wallet

Analogamente all'interazione *"Client-Server"* del web tradizionale, vi è un'interazione *Wallet-to-Wallet* nel Web3. Ogni utente più avere uno o più wallet digitali, ciascuno dei quali viene identificato da un indirizzo (**Ethereum Address**) a **42 caratteri alfanumerici**. Tra questi, 40 sono in formato numerico esadecimale e i restanti 2 compongono il prefisso **0x**.

A questo punto, è lecito chiedersi come possa un wallet interagire con gli Smart Contracts di Orbit.
Nello stesso modo in cui i wallet possono comunicare tra loro, ossia attraverso l'invio e ricezione di transazioni, anche uno Smart Contract può ricevere delle transazioni, essendo identificato sulla blockchain con un proprio indirizzo esadecimale.

Per fare un esempio, lo Smart Contract del token **WETH** presente sulla rete Polygon, si trova all'indirizzo:

$$0x7ceb23fd6bc0add59e62ac25578270cff1b9f619$$

Ciò significa che se un wallet volesse interagire con il contratto, magari per controllare l'ammontare di WETH che possiede, dovrebbe inviare una transazione a tale indirizzo. Dunque, a seconda dei parametri impostati, verrà eseguirà una specifica funzione presente nello Smart Contract, che potrebbe utilizzare con i dati presenti in esso, modificandoli o inviandoli al mittente della transazione.

4.2.1 Metamask e Wallet Connect

Per dare la possibilità ad un utente di connettere il proprio wallet ad Orbit ed effettuare le operazioni appena discusse, è stato scelto di implementare due sistemi: il primo, forse più noto nel Web3, è il wallet per criptovalute **Metamask**[?] mentre il secondo è il protocollo **WalletConnect**[?].

Metamask non è solo un wallet digitale, ma un vero e proprio *gateway* per interagire con applicazioni decentralizzate. Come si può vedere nella figura **??**, si presenta come un'estensione del browser, dalla quale si può gestire più wallet personali presenti

su blockchain diverse, visualizzare lo storico delle transazioni effettuate, l'ammontare dei token ERC-20 posseduti dall'utente ed effettuare transazioni.

Figura 4.1: Screenshot di uno dei miei personali wallet su Metamask, estensione browser

WalletConnect è invece un protocollo che permette di integrare all'interno della propria dApp più tipologie di wallet (tra i quali anche Metamask).
Le principali caratteristiche di WalletConnect sono **adattabilità** e **facilità di integrazione** con un qualsiasi portafoglio digitale: basterà che l'utente selezioni la propria wallet app o che inquadri un QR Code per effettuare immediatamente la connessione alla dApp di Orbit.

Inoltre, è una valida soluzione per facilitare la connessione ad una dApp da dispositivi mobili, che allo stato attuale risulta ostica per la maggior parte dei prodotti Web3.

4.2.2 Usabilità del sistema

Nei mesi precedenti al lancio del prodotto, è stato svolto un importante lavoro di analisi di usabilità per rendere Orbit una piattaforma quanto più accessibile per i propri utenti, svolgendo delle sessione di **Beta Testing**, attraverso le quali sono sorti dubbi e feedback che hanno estremamente semplificato il flusso utente del prodotto finale.

In quest'ultima parte, sono riuscito a dare il mio contributo attraverso nozioni imparate nel mio percorso di studi, in particolare in *Interazione Uomo-Macchina*, quali principi di design dell'usabilità, scelta di palette cromatiche inclusive per persone con anomalie visive quali il daltonismo ed euristiche per facilitare l'esperienza utente.

4.3 Tecnologie Backend

Il Backend è ciò che l'utente non vede concretamente, ma che contiene la logica applicativa di Orbit, costruita sopra una suite di Smart Contracts che ricoprono le principali funzionalità della piattaforma.

4.3.1 Solidity

Per lo sviluppo degli Smart Contracts di Orbit è stato scelto **Solidity**[?], il linguaggio di programmazione che viene utilizzato per sviluppare la maggior parte delle tecnologie blockchain presenti sul mercato. Nonostante la sua iniziale difficoltà di apprendimento, l'ecosistema di Solidity permette di avere una vasta scelta di librerie e contratti consolidati su cui basare i propri progetti.

Si tratta di un linguaggio *orientato agli oggetti*, di alto livello e che supporta *complessi tipi di oggetti* definiti dallo sviluppatore, ideale per ospitare ogni tipo di strutture dati.

Tuttavia, come già accennato nel capitolo precedente, le blockchain **Ethereum-based** hanno un costo computazionale rilevante, misurato in gas. Per questa ragione bisogna prestare particolare attenzione quando si utilizzano linguaggi come Solidity, in quanto una semplice **implementazione di una funzione** mal realizzata potrebbe richiedere un costo extra che, a seconda della rete in cui ci si trova, potrebbe risultare in un'importante spesa per il il mittente della transazione.

4.3.2 Hardhat

Data la delicatezza e fragilità dello sviluppo di una suite di Smart Contracts, è necessario avere uno strumento per testarne le principali funzioni, possibilmente prima che essi vengano rilasciati sulla blockchain. Proprio per questo è stato scelto di utilizzare **Hardhat**[?], un ambiente di sviluppo per costruire software sulla blockchain di Ethereum, il quale comprende diverse librerie che sinergizzano alla perfezione con lo sviluppo di Smart Contracts.

Hardhat Runner è una tra queste: viene utilizzato per costruire dei **task**, ossia azioni eseguibili tramite linea di comando. Per esempio, sono state realizzate specifiche azioni per compilare il codice degli Smart Contracts, eseguire dei test su di essi e distribuirli (*deploy*) sulla rete locale di Hardhat.
Infatti, uno dei tanti vantaggi di Hardhat è quello di testare i propri contratti su una rete locale, contenente 20 indirizzi utilizzabili per effettuare transazioni verso i propri Smart Contracts, al fine di avere un ambiente che preceda quello di produzione.
Inoltre, sono stati realizzati decine di test per ogni funzione dei contratti di Orbit, realizzati nel linguaggio **Typescript** che, a differenza di Javascript, ha un forte controllo sulla tipizzazione delle variabili, costruzione di interfacce e fornisce una sicurezza maggiore nello sviluppo.

Per la costruzione delle Test Suite sono state utili delle nozioni imparate durante il corso di *Sicurezza e Affidabilità*, che mi ha permesso di dare un contribuito nel lavoro svolto dai miei colleghi nello sviluppo dei contratti.

Capitolo 5

Design Patterns dell'Architettura di Orbit

Durante i mesi di sviluppo di Orbit, sono state effettuate numerose ricerche da parte della squadra di sviluppo di Five Elements Labs, in modo tale da scegliere un'architettura e dei **design patterns** quanto più sicuri e aggiornati con le ultime tecnologie del mondo DeFi e Smart Contracts.

5.1 Audit

5.2 Architettura

5.2.1 Position Manager

5.2.2 Actions

5.2.3 Moduli

5.2.4 Helpers

5.3 Sicurezza

5.3.1 Registry patterns

5.3.2 Timelock

Capitolo 6

Future implementazione all'interno di Orbit