UNIVERSITÀ
DI TORINO

Università degli Studi di Torino

Corso di Laurea in Informatica

La blockchain nell'industria musicale: un modello di tokenizzazione e distribuzione delle royalties

Tesi di Laurea Triennale in Informatica

Relatore
Bracciali Andrea

Candidato
Floridia Giorgio Francesco
1054584

Anno Accademico 2024/2025

Indice

DICHIARAZIONE DI ORIGINALITÀ

”Dichiaro di essere responsabile del contenuto dell’elaborato che presento al fine del conseguimento del titolo, di non avere plagiato in tutto o in parte il lavoro prodotto da altri e di aver citato le fonti originali in modo congruente alle normative vigenti in materia di plagio e di diritto d’autore. Sono inoltre consapevole che nel caso la mia dichiarazione risultasse mendace, potrei incorrere nelle sanzioni previste dalla legge e la mia ammissione alla prova finale potrebbe essere negata.”

Abstract

La distribuzione delle royalties musicali è storicamente gestita da intermediari e sistemi centralizzati che comportano ritardi nei pagamenti, mancanza di trasparenza e una ripartizione economica spesso sfavorevole per gli artisti. In questo contesto, le tecnologie blockchain offrono un'alternativa credibile per innovare profondamente il settore. Questa tesi propone un modello decentralizzato per la tokenizzazione delle royalties musicali, in cui l'intera carriera dell'artista viene rappresentata da un numero fisso di token su blockchain. Il valore di tali token è aggiornato dinamicamente in base ai guadagni mensili dell'artista derivanti da piattaforme come Spotify tramite smart contract, offrendo così un meccanismo trasparente, automatico e verificabile per la redistribuzione del valore. Il sistema integra anche un marketplace per la compravendita dei token e funzionalità di engagement che rafforzano il legame artista-fan. La soluzione proposta si configura come un ecosistema sostenibile, scalabile e innovativo che mira a ridefinire il ruolo degli artisti, dei fan e degli investitori nel panorama musicale contemporaneo.

1 Introduzione

Il settore musicale, pur avendo subito una profonda trasformazione nell'era dello streaming, continua a presentare numerose criticità legate alla gestione delle royalties. Gli artisti ricevono i propri compensi con mesi di ritardo, le informazioni sui flussi economici sono spesso opache e la dipendenza da etichette, distributori e piattaforme centralizzate limita l'autonomia decisionale degli autori. In questo scenario, la tecnologia blockchain si propone come un'alternativa concreta per ridisegnare i meccanismi economici e relazionali del mondo musicale.

Questa tesi esplora un modello innovativo per la gestione e la monetizzazione delle royalties musicali basato su smart contract e token non fungibili (NFT). Gli *smart contract* sono programmi informatici auto-eseguibili che risiedono sulla blockchain e che si attivano al verificarsi di determinate condizioni, garantendo trasparenza, sicurezza e automazione dei processi senza necessità di intermediari. Essi costituiscono l'elemento tecnologico fondamentale per la realizzazione del modello proposto e verranno analizzati più nel dettaglio nel paragrafo *4.2 Cosa sono gli smart contracts*.

L'idea centrale è quella di rappresentare, attraverso un numero fisso di token, l'intera carriera musicale di un artista.

Il valore dei token viene aggiornato periodicamente in base ai guadagni generati dell'artista su piattaforme di streaming musicale come, in questo caso, Spotify. Questo avviene mediante formule matematiche eseguite da smart contract e alimentate da dati ottenuti tramite fonti certificate. I possessori dei token, che possono essere fan, collaboratori o investitori, possono beneficiare del loro apprezzamento nel tempo o accedere a vantaggi esclusivi come eventi privati, contenuti personalizzati e merchandising numerato. La presenza di un marketplace interno consente inoltre la compravendita dei token, favorendo la liquidità e la partecipazione attiva.

Il modello proposto si basa su alcuni principi fondamentali: trasparenza nei flussi economici, automazione nei pagamenti, riduzione degli intermediari e rafforzamento del legame diretto tra artista e fan. L'adozione di architetture decentralizzate consente di superare molti limiti del sistema attuale, offrendo una maggiore equità nella distribuzione dei ricavi e nuovi strumenti di coinvolgimento per il pubblico.

La tesi include un'analisi tecnica della proposta, un confronto con due piattaforme esistenti, NUSIC e Royal, e una riflessione sulle implicazioni economiche, sociali e culturali dell'introduzione della blockchain nel settore musicale. L'obiettivo è fornire una visione concreta, scalabile e realistica di un nuovo paradigma distributivo e partecipativo, in grado di restituire centralità economica e creativa agli artisti.

2 Problemi del sistema attuale e soluzione blockchain

Le criticità legate al sistema tradizionale delle royalties musicali sono molteplici. In questa sezione vengono analizzati quattro problemi principali, suggerendo per ciascuno una possibile risposta basata sulle potenzialità della blockchain.

2.1 Il problema della centralizzazione

Uno degli elementi fondamentali per il successo di una piattaforma decentralizzata di gestione delle royalties è l'integrazione diretta con i servizi di streaming come Spotify. Tuttavia, è ragionevole chiedersi perché una piattaforma consolidata e di enorme portata come Spotify dovrebbe abbandonare i meccanismi tradizionali di pagamento a favore di uno basato su blockchain.

L'attuale industria musicale è caratterizzata da una marcata **centralizzazione** del controllo sui flussi economici e decisionali: poche grandi etichette, collecting societies e piattaforme di streaming detengono una posizione dominante che condiziona modalità di compenso, accesso ai dati e criteri di rendicontazione. Questa concentrazione genera *asimmetrie di potere*, tali per cui molti aspetti essenziali della remunerazione degli autori, dalla corretta attribuzione dei diritti alla tempistica delle liquidazioni, vengono gestiti attraverso canali opachi e soggetti a discrezionalità, con conseguenti inefficienze e conflitti di interesse.

Le complessità burocratiche e amministrative di questo modello non sono prive di conseguenze. In diversi casi, tali frizioni si sono tradotte in azioni legali e accuse pubbliche che mettono in luce le tensioni insite in un sistema fortemente centralizzato. Ad esempio, come riportato da *Reuters*, *"Spotify sued over millions allegedly unpaid in music royalties"*[1], una causa intentata nel 2024 evidenzia come il sistema attuale lasci margini di errore e contestazione difficilmente risolvibili senza costosi procedimenti legali. Analogamente, la *Recording Academy* ha denunciato che *"Spotify [is] under fire for failing to properly license music again"*[2], sottolineando la persistenza di problematiche nella gestione delle licenze. Inoltre, un'inchiesta del *The Times* ha smascherato la presenza di *"ghost artists"*[3], cioè profili e contenuti commissionati direttamente dalla piattaforma, con l'effetto di ridurre i ricavi destinati agli artisti indipendenti e di alimentare dubbi sulla trasparenza dei meccanismi di redistribuzione.

In questo contesto, adottare una piattaforma decentralizzata basata su blockchain consentirebbe a Spotify di **alleggerirsi di una parte significativa del lavoro burocratico e amministrativo**. La gestione delle royalties non richiederebbe più verifiche contabili ripetute, processi di riconciliazione con etichette e collecting societies, né lunghi contatti legali per dirimere controversie, poiché gli smart contracts garantireb-

bero in modo automatico la corretta redistribuzione delle entrate. In altre parole, la piattaforma potrebbe concentrarsi esclusivamente sul suo ruolo primario, ossia fornire un servizio di streaming efficiente e competitivo, demandando la complessità della ripartizione economica a un'infrastruttura digitale trasparente e condivisa.

La **decentralizzazione** non sarebbe dunque solo un beneficio per artisti ed etichette, ma rappresenterebbe anche un vantaggio operativo per le piattaforme stesse. Ridurre la dipendenza dagli intermediari come unica fonte di verità significherebbe abbassare la probabilità di contenziosi dovuti a discrepanze nei rendiconti e diminuire i costi legali e amministrativi che oggi gravano pesantemente sull'ecosistema. Smart contract e registri pubblici consentono infatti di automatizzare la contabilizzazione, di distribuire pagamenti in modo immediato e trasparente e di mettere a disposizione delle parti interessate uno **storico immodificabile delle transazioni**.

È importante sottolineare che la blockchain non rappresenta una soluzione definitiva. Per essere efficace richiede standard condivisi per i metadati, oracoli affidabili per collegare dati *off-chain* e *on-chain*, e un graduale allineamento degli incentivi tra piattaforme, editori e titolari dei diritti. Tuttavia, rispetto all'attuale architettura centralizzata, presenta vantaggi concreti in termini di riduzione delle asimmetrie informative, migliore tracciabilità dei flussi economici e potenziale contenimento delle controversie legali che oggi rallentano e appesantiscono il sistema.

Per questi motivi, l'integrazione tra servizi di streaming e piattaforme decentralizzate può essere considerata una delle vie più promettenti per restituire agli autori un controllo più diretto e trasparente sulle proprie opere e, allo stesso tempo, per liberare le stesse piattaforme da un carico gestionale e burocratico che rischia di comprometterne la sostenibilità nel lungo periodo.

2.2 Indipendenza economica dell'artista

Molti artisti oggi operano sotto contratti discografici che prevedono una notevole perdita di controllo sulla propria produzione artistica, oltre a una divisione poco favorevole dei proventi economici. Etichette, manager e intermediari si riservano spesso percentuali elevate delle royalties, lasciando all'artista una parte residuale del guadagno effettivo.

La proposta di una piattaforma basata su blockchain, in cui l'artista può decidere autonomamente **quante quote tokenizzare delle proprie royalties**, rappresenta un modello che promuove l'indipendenza artistica ed economica. In questo scenario, l'artista diventa il primo responsabile della propria economia musicale, con la possibilità di vendere direttamente le quote ai fan o investitori, senza dover cedere il controllo decisionale a terzi.

È noto che in media gli artisti percepiscono solo una parte marginale dei proventi generati e una quota significativa che finisce nelle mani di distributori e case discografiche. In questo contesto, la proposta blockchain consente una **maggiore trasparenza** e un **maggiore margine operativo** per l'artista, riducendo la dipendenza da intermediari e permettendo una gestione diretta delle proprie royalties.

2.3 Tempistiche di pagamento e ottimizzazione economica

Una criticità strutturale del sistema delle royalties streaming riguarda i **lunghi ritardi tra l'ascolto e l'effettivo pagamento all'artista**. Tali ritardi sono dovuti ai molteplici attori coinvolti e all'impostazione stessa del flusso economico.

Intermediari nella filiera delle royalties I fondi generati dalle piattaforme di streaming (Spotify, Apple Music, YouTube, ecc.) vengono inizialmente destinati ai *rights holders*, ovvero i detentori dei diritti di registrazione (master) o di composizione, che possono essere etichette discografiche, distributori, editori oppure, nel caso di artisti indipendenti, l'artista stesso.[4] In media, le piattaforme redistribuiscono circa il 70% dei ricavi complessivi a questi soggetti, che successivamente trattengono una quota significativa prima di liquidare l'importo spettante agli artisti. Secondo diversi studi, le etichette possono arrivare a trattenere fino al 73% delle royalties, mentre gli autori ed editori ricevono circa il 16% e agli artisti rimane soltanto l'11%.[5] A questo si aggiungono ulteriori ritardi imputabili alle collecting societies (Performance Rights Organizations – PROs), come SoundExchange negli Stati Uniti, che hanno il compito di raccogliere, elaborare e distribuire le royalties, applicando però commissioni amministrative che riducono ulteriormente i margini.

Quantificazione dei ritardi Nel sistema tradizionale di distribuzione musicale, le tempistiche di pagamento delle royalties non sono immediate. Gli artisti possono dover attendere diversi mesi prima di ricevere i compensi derivanti dallo streaming delle proprie opere, a causa dei complessi processi di rendicontazione e intermediazione tra le varie piattaforme e gli enti di gestione dei diritti.
Un discorso analogo vale per i distributori digitali come DistroKid, i quali dichiarano che i rapporti mensili riflettono vendite avvenute circa tre mesi prima.

> "Streaming services usually deliver these reports monthly, and they reflect sales from about 3 months ago. So, if someone streamed your song yesterday, royalties for it won't show up in your DistroKid bank until about 3 months from now."

Analogamente, gli stessi distributori confermano che:

> "Royalties are always 2 to 3 months behind (Spotify and Apple Music) [...] payments are not made immediately the following month."

Perché questi ritardi sono problematici Queste dinamiche assumono un peso ancora maggiore se si considera che lo streaming rappresenta oggi la principale fonte di guadagno dell'industria musicale. Nel 2022, ad esempio, la quota di mercato dello streaming oscillava tra il 67% e l'84% dei ricavi globali, rendendo quindi cruciale la rapidità dei pagamenti.[6]

Secondo il *Global Music Report 2025* dell'IFPI, lo streaming rappresenta ormai il 69% dei ricavi globali dell'industria musicale, con entrate da streaming in crescita del +4,8% e un totale di 752 milioni di abbonamenti pagati nel 2024, un quadro che rende i ritardi nei pagamenti economicamente rilevanti e fortemente avvertiti dagli artisti.[7]

Nonostante le grandi cifre in gioco, basti pensare che Spotify ha dichiarato di aver versato oltre 10 miliardi di dollari ai rights holders solo nel 2024, gli artisti spesso percepiscono una quota irrisoria, che in molti casi equivale a una frazione di centesimo per stream. La distribuzione dei compensi rimane inoltre opaca e poco verificabile da parte degli stessi musicisti.

Come può intervenire la tecnologia blockchain Implementando un modello basato su smart contract e tokenizzazione, sarebbe possibile introdurre un meccanismo di pagamento molto più rapido ed equo. Gli importi maturati potrebbero essere trasferiti direttamente a uno smart contract, che redistribuisce istantaneamente le somme secondo regole prefissate. In questo modo si ridurrebbe drasticamente il numero di intermediari, limitando tempi morti e costi di gestione. Inoltre, la natura pubblica e immutabile della blockchain garantirebbe maggiore trasparenza, poiché ogni transazione risulterebbe registrata on-chain e verificabile da tutte le parti coinvolte. Ciò renderebbe anche possibile l'introduzione di pagamenti più frequenti, con liquidazioni mensili o addirittura in tempo quasi reale.

Modello tradizionale	Modello blockchain
Il flusso dei pagamenti coinvolge numerosi intermediari (etichette, aggregatori, PROs, distributori), ognuno dei quali trattiene una percentuale e introduce ritardi.	I pagamenti confluiscono direttamente in uno smart contract, che ridistribuisce automaticamente le somme secondo criteri prestabiliti, senza passaggi intermedi.
I tempi medi di liquidazione vanno da due a sei mesi, a seconda della piattaforma e del distributore.	La liquidazione può avvenire in maniera quasi istantanea, man mano che i dati di ascolto vengono resi disponibili.
Il sistema è poco trasparente: gli artisti non hanno visibilità chiara sulle percentuali trattenute dagli intermediari né sulla distribuzione effettiva dei flussi economici.	Ogni transazione è registrata on-chain, risultando tracciabile, verificabile e accessibile a tutte le parti coinvolte.
Gli artisti spesso perdono margini significativi a causa di commissioni e trattenute, ricevendo solo una piccola parte dei ricavi complessivi.	La riduzione del numero di intermediari e l'automazione tramite smart contract permette di ridurre i costi e migliorare i margini per gli artisti.

2.4 Coinvolgimento e relazione artista-fan

Un ulteriore aspetto innovativo introdotto dalla piattaforma proposta è la capacità di trasformare la relazione tra artista e fan in un **legame attivo e partecipativo**, superando la tradizionale dinamica passiva di fruizione musicale. Attraverso la tokenizzazione delle royalties, il fan non è più un semplice ascoltatore: diventa un **partecipante economico** al successo dell'artista.

Il possesso dei token legati alla carriera dell'artista incentiva il fan a promuoverne la musica, a condividerne i contenuti e a rimanere coinvolto nel tempo. Ogni ascolto, ogni visualizzazione o condivisione ha infatti un impatto diretto sul valore economico dei token detenuti. In questo modo, si instaura una **relazione di tipo cooperativo**, dove il successo dell'artista è anche il successo dei fan-investitori.

Oltre all'aspetto economico, la piattaforma può offrire una serie di **vantaggi esclusivi** per i possessori di token, favorendo la creazione di una **fan base solida, attiva e fidelizzata**.

Questo aspetto di coinvolgimento e fidelizzazione verrà approfondito nel paragrafo *"4.10 Incentivi e soulbound token"*, dove vengono illustrati i meccanismi concreti per premiare i possessori a lungo termine.

3 Casi di studio: piattaforme blockchain musicali

Vediamo ora come le problematiche evidenziate nei capitoli precedenti vengono affrontate da soluzioni già operative sul mercato, che rappresentano esempi concreti di applicazione della blockchain all'industria musicale.

3.1 AnotherBlock

AnotherBlock è una piattaforma Web3 lanciata nel 2022 che consente di acquistare e possedere una quota dei diritti di royalties legati a brani musicali già pubblicati. L'obiettivo è quello di democratizzare l'accesso ai proventi derivanti dallo streaming, permettendo a fan e investitori di partecipare direttamente al successo economico di un'opera.

Ogni brano disponibile su AnotherBlock viene tokenizzato in una serie limitata di NFT, ciascuno dei quali rappresenta una frazione reale dei diritti di royalties associati. Gli acquirenti degli NFT diventano così titolari di un diritto economico effettivo, e ricevono periodicamente le loro quote di guadagno.

- **Distribuzione diretta delle royalties** – Gli artisti e i detentori dei diritti possono scegliere di mettere in vendita una percentuale delle proprie royalties. Gli acquirenti ricevono automaticamente la loro parte dei proventi derivanti da piattaforme come Spotify, Apple Music e YouTube, in base ai dati certificati dai distributori musicali. I pagamenti avvengono in ETH o USDC e vengono distribuiti più volte l'anno, in maniera trasparente e tracciabile on-chain.

- **Trasparenza dei flussi economici** – Ogni transazione, dall'acquisto dell'NFT fino alla distribuzione delle royalties, è registrata sulla blockchain, rendendo verificabile l'intero processo. Questo elimina le asimmetrie informative tipiche dell'industria musicale tradizionale e offre agli artisti una rendicontazione chiara e verificabile.

In sintesi, AnotherBlock rappresenta una piattaforma che combina la **tokenizzazione economica dei diritti musicali** con una **esperienza utente accessibile**, favorendo la partecipazione diretta dei fan nella distribuzione dei proventi e nella crescita del valore dell'artista.

3.2 Royal

Royal, fondata dall'artista e produttore Justin Blau (3LAU), adotta una filosofia simile, ponendosi anch'essa come intermediario decentralizzato per la distribuzione diretta delle royalties musicali. Tramite NFT, Royal permette agli utenti di acquistare quote effettive dei guadagni futuri di una canzone o di un intero catalogo.

- **Distribuzione automatica e trasparente** – Gli artisti scelgono la percentuale delle royalties da tokenizzare e vendere. Gli acquirenti ricevono periodicamente i propri proventi in USDC o altra stablecoin, sulla base dei dati provenienti da distributori digitali certificati (come Stem o DistroKid). Tutte le operazioni avvengono on-chain, garantendo tracciabilità e sicurezza.
- **Efficienza grazie a Layer 2** – Royal opera su *Polygon*, una sidechain di Ethereum a basso costo, che consente di ridurre le gas fees e rendere sostenibili microtransazioni e pagamenti frequenti. Questo approccio è essenziale per gestire una distribuzione economica su larga scala senza gravare sugli utenti.

Royal affronta in modo diretto le problematiche legate a **ritardi nei pagamenti**, **mancanza di trasparenza** e **scarsa partecipazione dei fan**, proponendo un sistema decentralizzato che redistribuisce in modo immediato e verificabile i ricavi dello streaming. Gli artisti mantengono comunque pieno controllo sulle percentuali cedute, preservando la propria indipendenza economica.

3.3 Confronto tra approcci

Sia AnotherBlock che Royal rappresentano esempi concreti di come la blockchain possa ridefinire la relazione tra artista, fan e flussi economici della musica. Pur condividendo elementi comuni, come l'uso di NFT, smart contract e stablecoin, le due piattaforme si differenziano per impostazione e obiettivi:

- **AnotherBlock** adotta un approccio fortemente orientato alla *tokenizzazione reale dei diritti*, consentendo di acquistare effettive quote di royalties di brani già pubblicati. Il modello privilegia la trasparenza e la partecipazione diretta, rendendo accessibile a chiunque un mercato prima riservato a investitori professionali.
- **Royal** propone invece una struttura più ampia e istituzionale, puntando su partnership con artisti di rilievo e su un'infrastruttura tecnica ottimizzata (Layer 2), che rende scalabile il sistema di redistribuzione delle royalties e adatto a un'adozione di massa.

La proposta di questa tesi si colloca in posizione intermedia tra questi due modelli: unisce la **trasparenza e partecipazione economica** di AnotherBlock alla **struttura automatizzata e scalabile** di Royal, con l'obiettivo di costruire un ecosistema che valorizzi la carriera dell'artista, riduca i tempi di pagamento e rafforzi la relazione diretta con la propria community.

4 La proposta tecnica: tokenizzazione delle royalties

Dopo aver analizzato criticamente le soluzioni esistenti e le problematiche strutturali del settore, il presente capitolo fornisce una descrizione approfondita del modello tecnico sviluppato: vengono delineati i meccanismi di tokenizzazione delle royalties, la logica di aggiornamento del valore e della distribuzione economica, i processi di emissione, la logica di rilascio dei token, le strategie di incentivazione attraverso soulbound token, nonché l'architettura tecnica del marketplace integrato e il quadro delle principali soluzioni di sicurezza adottate a tutela dell'ecosistema.

4.1 Le royalties su Spotify

Il dato principale di nostro interesse riguarda la quota di royalties che Spotify versa mensilmente a un artista. Questo valore non è calcolabile in maniera precisa, poiché le royalties non vengono determinate in base a una tariffa fissa per ascolto. La stessa piattaforma chiarisce infatti:

> "Distribuiamo il guadagno netto derivante dalle tariffe degli abbonamenti Premium e dalle pubblicità ai detentori dei diritti. Per calcolare il guadagno netto, sottraiamo il denaro che raccogliamo ma che non spetta a noi. Ciò include pagamenti per elementi come tasse, commissioni per le transazioni con carte di credito e fatturazione, insieme ad altri aspetti come le commissioni di vendita. Da qui, la quota di guadagno netto spettante al detentore dei diritti viene determinata dalla quota di ascolti. Calcoliamo la quota di ascolti contando il numero totale di ascolti in un dato mese e determinando quale quota di tali ascolti rappresentava persone che ascoltavano musica di proprietà o gestita da un particolare titolare dei diritti. Contrariamente a quanto potresti aver sentito dire, Spotify non paga le royalties degli artisti in base a una tariffa per ascolto o per riproduzione; i pagamenti delle royalties ricevuti dagli artisti potrebbero variare in base alla modalità di ascolto della propria musica o secondo gli accordi tra loro e i distributori o le etichette."[8]

Questa precisazione permette di comprendere perché, nel modello proposto, non si utilizzi un valore fisso per riproduzione, ma piuttosto le royalties complessive dichiarate dalla piattaforma come base di calcolo.

Tuttavia, diversi studi hanno cercato di stimare un intervallo indicativo del guadagno per stream, individuandolo attorno a **0,003–0,005 dollari per riproduzione**. Si tratta di valori medi, che possono variare in base alla provenienza geografica degli ascolti, al tipo di abbonamento dell'utente (gratuito o premium) e agli accordi contrattuali specifici tra artisti, etichette e distributori.

Per semplicità, in questa tesi si farà riferimento a tali stime come valori di base negli esempi numerici. In particolare, l'utilizzo della formula

$$Entrate \approx MonthlyListeners \times PayPerStream$$

consente di ottenere una misura approssimativa delle entrate di un artista. Gli ascoltatori mensili rappresentano infatti il dato più immediato e intuitivo per valutare la popolarità di un artista e stimare l'impatto economico della sua musica.

4.2 Cosa sono gli smart contracts

Prima di entrare nel dettaglio del progetto, è utile introdurre brevemente il concetto di smart contract. Si tratta infatti di uno degli strumenti principali utilizzati in questa tesi, ed è quindi importante fornire fin da subito una spiegazione di base per comprendere meglio di cosa si sta parlando. Anche se il funzionamento e le logiche di questi contratti digitali verranno approfonditi nei capitoli successivi, una panoramica iniziale aiuta a capire il loro ruolo all'interno del sistema proposto e il motivo per cui rappresentano un elemento chiave della soluzione sviluppata. Gli smart contract sono programmi eseguiti in modo deterministico su una blockchain e rappresentano un veicolo per codificare regole che governano la creazione, lo scambio e la gestione di beni digitali. A differenza di un software tradizionale, un smart contract opera in un contesto distribuito, immutabile e trasparente: le sue istruzioni vengono registrate on-chain e le modifiche di stato risultano verificabili da qualunque partecipante alla rete.

4.3 Struttura e dinamiche dei token

Il cuore del progetto consiste nella tokenizzazione delle royalties generate dal repertorio musicale di un artista. L'idea è quella di emettere, tramite una piattaforma basata su blockchain Ethereum, un numero fisso di token che rappresentino quote delle entrate complessive derivanti dagli ascolti su *Spotify*.

A differenza di modelli che operano sul singolo brano, l'approccio proposto mira a valorizzare l'intera carriera musicale dell'artista, semplificando il processo e creando un unico strumento digitale che riflette la popolarità complessiva. Ogni token diventa quindi una rappresentazione digitale e scambiabile della carriera musicale, capace di condensarne l'andamento in un valore economico aggiornato nel tempo.

I token possono essere acquistati da fan, collaboratori o etichette discografiche, creando un legame diretto tra l'artista e la sua comunità. Il loro valore non rimane statico, ma viene aggiornato dinamicamente con cadenza regolare in base al numero di royalties mensili guadagnate dall'artista su *Spotify*. In questo modo, l'evoluzione della popolarità dell'artista si riflette in modo immediato e trasparente sull'andamento economico dei token.

Il meccanismo di aggiornamento è governato da uno *smart contract*, che applica una formula matematica in grado di trasformare le variazioni nei dati di ascolto in corrispondenti variazioni di valore. In tal modo, il guadagno dei possessori non è astratto, ma ancorato a informazioni reali e verificabili, strettamente connesse al successo musicale dell'artista.

L'intero sistema mira a creare un mercato dove i token possono essere scambiati liberamente, generando al tempo stesso benefici sia per l'artista, che rafforza il rapporto con i propri sostenitori e guadagna dalla vendita dei suoi token, sia per i fan e gli investitori, che vedono concretizzata la possibilità di partecipare in maniera attiva e diretta all'andamento della carriera musicale.

4.4 Formula di aggiornamento del valore

Il meccanismo di tokenizzazione prevede che l'artista destini una percentuale β delle proprie royalties alla creazione di token digitali. Ogni token rappresenta quindi una quota proporzionale della parte di guadagni messa a disposizione dall'artista e potrà essere acquistata da fan, etichette o investitori. Il valore dei token non è fisso, ma viene aggiornato ogni 28 giorni, in corrispondenza con l'aggiornamento dei dati sugli ascoltatori mensili di Spotify.

L'aggiornamento del valore riflette le royalties effettivamente generate nel periodo, consentendo di rappresentare in modo diretto e trasparente l'andamento della carriera musicale dell'artista. La formula utilizzata è la seguente:

$$V_t = V_{t-1} \times \frac{R_t}{R_{t-1}},$$

dove:

- V_t è il valore del token al mese t;

- V_{t-1} è il valore del token al mese precedente;
- R_t rappresenta le royalties complessive del mese t, calcolate come:

$$R_t = \text{royalties}_t \times \beta$$

  cioè il totale delle royalties destinate ai token (con β che indica la percentuale scelta dall'artista).

Questa formulazione garantisce che il valore del token aumenti quando l'artista ottiene più ascolti e genera più royalties, e diminuisca quando le performance calano. In questo modo, chi investe nei token partecipa realmente all'andamento del successo musicale dell'artista, beneficiando dei periodi di crescita e condividendo i rischi nei momenti di contrazione.

Per rendere operativo un progetto come questo, occorrerebbe l'appoggio di una piattaforma di streaming consolidata, come Spotify o simili. Idealmente, la piattaforma dovrebbe inviare direttamente in uno smart contract l'ammontare dei guadagni netti generati dall'artista in un determinato mese. Lo smart contract sarebbe quindi in grado di applicare le formule sopra descritte per aggiornare il valore del token in modo trasparente e verificabile, riflettendo le performance effettive dell'artista.

Questa integrazione tra piattaforma di streaming e smart contract rappresenta il nucleo operativo del progetto.

4.5 Attribuzione del valore economico ai token

Un passaggio fondamentale nello sviluppo del progetto consiste nell'attribuire un valore economico ai token, così da renderne possibile non solo l'acquisto iniziale, ma anche la futura rivendita su un *marketplace interno* dedicato. Senza un'ancora di valore stabile, infatti, il token resterebbe un mero indicatore astratto della quota di royalties, privo di un corrispettivo monetario diretto utilizzabile dagli utenti.

In questo contesto, la scelta della valuta di riferimento è cruciale. Una possibilità sarebbe ancorare il valore direttamente a **ETH**, l'asset nativo di Ethereum. Tuttavia, ETH è caratterizzato da un'elevata volatilità, difficilmente compatibile con il sistema di royalties musicali che sono calcolate in valuta fiat.

Per questo motivo, una soluzione più stabile consiste nell'utilizzo di una **stablecoin**, ovvero una criptovaluta ancorata 1:1 a una valuta reale. In particolare:

- **USDC (USD Coin)**, molto diffusa, trasparente e auditata, rappresenta una scelta solida per chi vuole ancorare a un valore in dollari;

- **EURC (Euro Coin)**, analoga a USDC ma ancorata all'euro, risulterebbe ancora più coerente nel caso in cui si voglia lavorare direttamente con la moneta europea. Questa sarà la valuta di riferimento del progetto.

L'adozione di una stablecoin consente di minimizzare gli effetti della volatilità tipica del mercato cripto, garantendo un legame più chiaro tra royalties, valore del token e prezzo di scambio.

4.6 Esempi numerici

Per illustrare in modo esaustivo la dinamica dei token, consideriamo un esempio ipotetico di un artista X con un numero fisso di $N_{\text{token}} = 500.000$ token emessi. L'artista decide di destinare $\beta = 0{,}30$ delle proprie royalties mensili ai token.

Calcolo del valore iniziale del token Poiché Spotify non utilizza un pay-per-stream fisso, per ottenere un valore numerico iniziale dei token facciamo un'assunzione realistica: consideriamo che il guadagno medio per ascolto sia di 0,003 €.

Sia il numero di ascolti del primo mese pari a $20.000.000$, quindi le royalties totali dell'artista saranno:

$$\text{Royalties totali} = 20.000.000 \times 0,003\,€ = 60.000\,€$$

La quota destinata ai token sarà:

$$R_0 = 60.000 \times 0,30 = 18.000\,€$$

Da cui il valore iniziale di ciascun token:

$$V_0 = \frac{R_0}{N_{\text{token}}} = \frac{18.000}{500.000} = 0,036\,€ \text{ per token}$$

Aggiornamento mensile Il valore dei token nei mesi successivi viene aggiornato secondo la formula già introdotta:

$$V_t = V_{t-1} \times \frac{R_t}{R_{t-1}}, \quad \text{con } R_t = \text{royalties}_t \times \beta = \text{ascolti}_t \times 0{,}003 \times \beta$$

Dati ipotetici sugli ascolti mensili (12 mesi)

Mese	Ascoltatori mensili
Gennaio	20 000 000
Febbraio	21 500 000
Marzo	19 800 000
Aprile	22 000 000
Maggio	20 500 000
Giugno	23 000 000
Luglio	21 200 000
Agosto	24 000 000
Settembre	23 500 000
Ottobre	25 000 000
Novembre	24 200 000
Dicembre	26 000 000

Calcolo delle royalties destinate e del valore token

Mese	R_t (€)	V_t (€)	Var. %	Osservazioni
Gennaio	18 000	0,036	—	Valore iniziale
Febbraio	19 350	0,0387	+7,5%	Crescita ascolti + royalties
Marzo	17 820	0,0356	-7,9%	Calo ascolti, valore diminuisce
Aprile	19 800	0,0395	+11,2%	Aumento significativo ascolti
Maggio	18 450	0,0368	-6,8%	Leggero calo
Giugno	20 700	0,0414	+12,2%	Nuovo picco di ascolti
Luglio	19 080	0,0381	-7,8%	Riduzione momentanea
Agosto	21 600	0,0430	+13,2%	Picco massimo estivo
Settembre	21 150	0,0421	-2,1%	Leggera flessione
Ottobre	22 500	0,0448	+6,4%	Nuovo incremento
Novembre	21 780	0,0434	-3,2%	Calo moderato
Dicembre	23 400	0,0466	+7,4%	Chiusura anno in crescita

Osservazioni

- Il valore del token segue direttamente l'andamento delle royalties destinate (R_t), quindi degli ascolti mensili.
- L'investitore partecipa sia ai picchi di crescita sia alle flessioni, evidenziando dinamiche realistiche di rischio/guadagno.
- La correlazione tra performance musicale e valore economico dei token è immediata e verificabile.

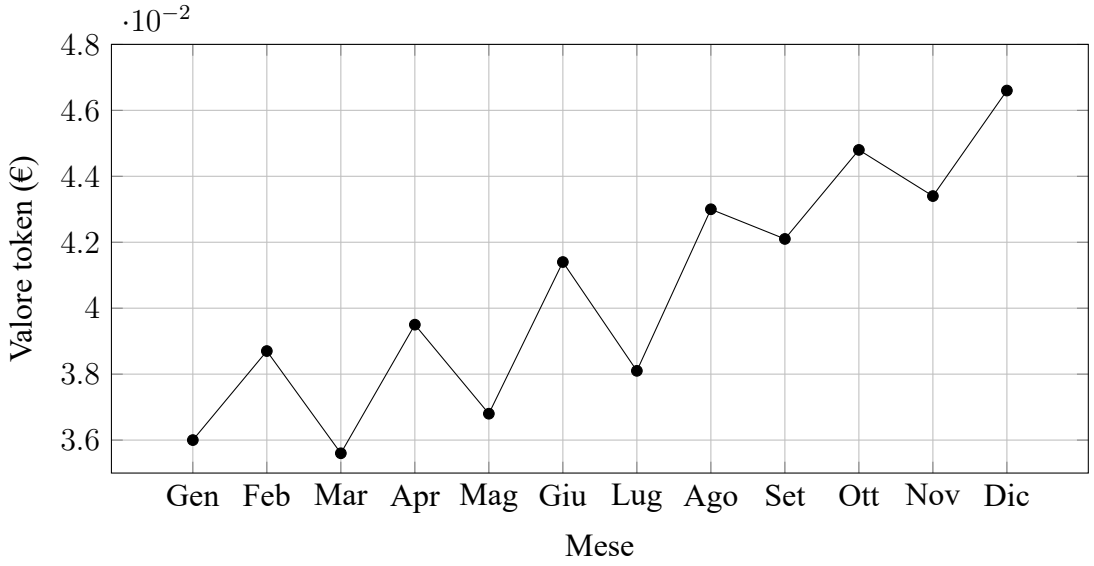
- Nel corso dei 12 mesi, il token varia tra 0,0356 € e 0,0466 €, mostrando come flessioni temporanee siano compensate dai periodi di crescita.

Esempio di investimento Supponiamo che un utente acquisti 3.000 token a gennaio al prezzo iniziale di 0,036 € ciascuno, per un investimento totale di:

$$3.000 \times 0,036 = 108\,€$$

- Dopo 6 mesi (giugno), il valore del token è 0,0414 €, quindi il portafoglio dell'utente vale $3.000 \times 0,0414 = 124,2\,€$.
- A dicembre, il valore sale a 0,0466 €, portando il valore del suo portafoglio a $3.000 \times 0,0466 = 139,8\,€$.

Grafico di evoluzione del valore del token



4.7 Architettura tecnica della piattaforma

Per sviluppare e testare gli smart contracts è stato necessario scegliere con attenzione la blockchain di riferimento. In questo lavoro è stata adottata **Ethereum**, oggi lo standard principale per la creazione di applicazioni decentralizzate (dApp). Ethereum si distingue per la sua maturità, per la sicurezza offerta dalla rete e per un ecosistema ampio di strumenti, documentazione e librerie. È stata inoltre la prima blockchain a introdurre un linguaggio di programmazione completo per la scrittura di smart contract, permettendo di implementare logiche complesse oltre al semplice trasferimento di denaro digitale. La sua diffusione e apertura garantiscono interoperabilità e fiducia sia tra gli sviluppatori che tra gli utenti.

Gli smart contract sono stati realizzati in **Solidity**, il linguaggio di programmazione più usato su Ethereum. Solidity è stato scelto per la sua popolarità, per l'ampio supporto della community e per la disponibilità di strumenti che semplificano le fasi di test e distribuzione.

Durante lo sviluppo iniziale è stato utilizzato **Ganache**, un ambiente di simulazione locale che permette di creare una blockchain Ethereum privata sul proprio computer. Con Ganache è possibile disporre di account già dotati di Ether virtuale, generare transazioni e blocchi in modo immediato e monitorare lo stato della blockchain. Questo rende i cicli di sviluppo e debug rapidi, senza costi e senza vincoli legati alla disponibilità di risorse.

Dopo aver validato i contratti in locale, si è passati ai test su una **testnet pubblica**, nello specifico **Sepolia**. Le testnet consentono di verificare il funzionamento degli smart contract in un ambiente distribuito e realistico, simile alla mainnet, ma senza costi reali. Sepolia è stata scelta perché è una delle testnet ufficiali più stabili e supportate dalla community.

L'utilizzo delle testnet presenta però alcuni limiti. Per eseguire transazioni è necessario disporre di Ether della rete di test (SepoliaETH), distribuito gratuitamente tramite *faucet*. I faucet rilasciano però solo piccole quantità di token per volta, così da evitare abusi. Questo può rendere più difficoltoso il testing intensivo, poiché la disponibilità di fondi virtuali non è sempre immediata. Di conseguenza, i cicli di sviluppo possono risultare più lenti rispetto a quelli su Ganache. Si tratta comunque di un compromesso inevitabile, dato che le testnet hanno lo scopo di simulare in modo realistico la mainnet senza sprechi di risorse.

In sintesi, l'approccio seguito è stato quello di utilizzare Ganache nelle fasi di prototipazione e debug veloce, per poi validare i contratti su Sepolia. Questa combinazione ha permesso di unire rapidità nello sviluppo con l'affidabilità dei test in un ambiente distribuito, riducendo i costi e i rischi legati al deploy finale sulla mainnet Ethereum.

4.8 Marketplace e compravendita dei token

Un elemento centrale della piattaforma è il **marketplace interno**, che consente di acquistare e vendere i token degli artisti esclusivamente tramite lo **smart contract principale**. Questo contratto funge da unico intermediario autorizzato per tutte le operazioni di compravendita, garantendo la piena **trasparenza, sicurezza e tracciabilità** di ogni transazione. In questo modo, non è previsto alcuno scambio diretto tra utenti: tutte le operazioni vengono gestite dal contratto stesso, che aggiorna in modo automatico i saldi e i valori in base alle regole economiche definite nel modello.

Il marketplace assicura la **liquidità dei token**, consentendo il continuo ricircolo tra fan e investitori anche dopo che tutti i token iniziali sono stati distribuiti. I possessori possono realizzare guadagni o perdite in base alle variazioni del valore dei token, mentre il contratto mantiene in ogni momento la copertura necessaria per riacquistare quelli messi in vendita, preservando la stabilità complessiva del sistema.

L'acquisto dei token può avvenire attraverso diversi strumenti: è possibile pagare con **stablecoin** ancorate a valuta reale, come EURC o USDC, utilizzando wallet integrati nella piattaforma o connettendosi a portafogli esterni.

Quando un possessore decide di vendere i propri token, la richiesta viene inviata allo **smart contract principale**, che esegue la transazione riacquistando direttamente i token e trasferendo all'utente la corrispondente quantità di stablecoin. Tutte le operazioni sono registrate on-chain e rimangono pubblicamente verificabili, garantendo un sistema **sicuro, trasparente e autosufficiente** in cui ogni scambio avviene in modo automatico, senza la necessità di intermediari o scambi peer-to-peer.

4.9 Analisi della sostenibilità economica del modello

La sostenibilità economica rappresenta il pilastro fondamentale su cui si regge l'intero modello di tokenizzazione. Dimostrare che il sistema sia stabile e al contempo una fonte di guadagno per l'artista costituisce la prova definitiva della sua validità e affidabilità. L'obiettivo di questa sezione è mostrare, attraverso un'analisi quantitativa chiara e coerente, come lo smart contract sia progettato per restare sempre solvibile, anche in presenza di scenari sfavorevoli o di fluttuazioni di mercato.

Un aspetto chiave per comprendere la sostenibilità del modello è la distinzione tra i flussi finanziari in entrata. Le **royalties** mensili vengono utilizzate per aggiornare il valore dei token e rappresentano la base del meccanismo di valorizzazione. I **proventi delle vendite dei token**, invece, costituiscono la riserva di liquidità che permette al contratto di riacquistare token dagli utenti e di mantenere la piena solvibilità in ogni momento.

Questa separazione funzionale genera una struttura finanziaria robusta: le royalties alimentano la crescita del valore, mentre i proventi delle vendite assicurano la stabilità operativa. L'artista, inoltre, può prelevare progressivamente la parte di fondi che eccede il valore totale dei token in circolazione, ottenendo così un guadagno aggiuntivo senza compromettere la sicurezza del sistema. Questo aspetto, ovvero il potenziale incremento dei guadagni per l'artista derivante dall'adozione di questo modello, si aggiunge ai numerosi vantaggi già previsti e discussi in precedenza, come una maggiore indipendenza artistica ed economica, una gestione più trasparente delle entrate e tempi di pagamento significativamente più brevi.

Il modello opera in modo cumulativo: ogni mese, le royalties si sommano ai fondi esistenti, generando un effetto di accumulo. La *Tabella 1* mostra un esempio dell'evoluzione dei primi dodici mesi, evidenziando come i fondi totali e l'eccedenza prelevabile aumentino progressivamente.

Tabella 1: Evoluzione dei fondi e dell'eccedenza prelevabile nell'arco di 12 mesi

Mese	**Royalties** (€)	**Valore Token** (€)	**Fondi Totali** (€)	**Eccedenza** (€)
1	18.000	0,0360	18.000	0
2	19.350	0,0387	37.350	18.000
3	17.820	0,0356	55.170	37.350
4	19.800	0,0396	74.970	55.170
5	18.450	0,0369	93.420	74.970
6	20.700	0,0414	114.120	93.420
7	19.080	0,0381	133.200	114.120
8	21.600	0,0432	154.800	133.200
9	21.150	0,0423	175.950	154.800
10	22.500	0,0450	198.450	175.950
11	21.780	0,0436	220.230	198.450
12	23.400	0,0468	243.630	220.230

Dopo dodici mesi di attività, il sistema ha accumulato un'eccedenza di oltre **220.000 €**, mantenendo sempre la copertura totale del valore dei token. Questa crescita dimostra come il modello sia in grado di autofinanziarsi nel tempo, generando margini progressivi a favore dell'artista e garantendo un equilibrio sostenibile tra remunerazione e stabilità.

Per verificare la solidità strutturale del sistema, sono stati simulati due scenari estremi.

Scenario 1 — Vendita massiva dei token Si ipotizza che tutti i detentori decidano di vendere i token contemporaneamente al termine del sesto mese, quando il valore unitario è pari a 0,0414 €.

Anche in questa condizione estrema, il contratto dispone di risorse più che sufficienti per riacquistare tutti i token, confermando la piena solvibilità del modello. L'eccedenza residua, pari a 93.420 €, rappresenta il margine che l'artista può liberamente ritirare come ulteriore guadagno.

Tabella 2: Scenario di vendita massiva al sesto mese

Descrizione	**Valore (€)**
Fondi disponibili nel contratto	114.120
Valore complessivo dei token (500.000 × 0,0414 €)	20.700
Fondi residui dopo il riacquisto totale	**93.420**
Risultato	**Sistema pienamente solvibile**

Scenario 2 — Crisi prolungata delle royalties Si considera un caso in cui le entrate mensili diminuiscano progressivamente fino a stabilizzarsi su livelli minimi, simulando una perdita di popolarità dell'artista.

Tabella 3: Scenario di crisi con calo progressivo delle royalties

Mese	**Royalties** (€)	**Valore Token** (€)	**Fondi Totali** (€)	**Eccedenza** (€)
1	18.000	0,0360	18.000	0
2	19.350	0,0387	37.350	18.000
3	9.000	0,0180	46.350	37.350
4	4.500	0,0090	50.850	46.350
5	2.250	0,0045	53.100	50.850
6	1.125	0,0023	54.225	53.100

Nonostante un crollo del 94% delle royalties rispetto al picco, il sistema resta pienamente solvibile grazie ai fondi accumulati in precedenza. Questo conferma la capacità del modello di assorbire shock economici anche severi, mantenendo la copertura totale dei token e garantendo all'artista la possibilità di continuare a prelevare parte dell'eccedenza accumulata.

L'artista può in qualsiasi momento ritirare l'eccedenza maturata, parzialmente o in toto, come forma di remunerazione diretta del proprio lavoro creativo. Nel caso illustrato nella *Tabella 1*, dopo dodici mesi, l'artista dispone di un guadagno netto di 220.230 €, ottenuto senza compromettere in alcun modo la stabilità complessiva del sistema.

Per garantire un ulteriore margine di sicurezza, il modello prevede che una piccola parte dei fondi, pari al **10% del valore complessivo dei token**, resti sempre depositata nel contratto.

La quantità di fondi ritirabili in ogni istante può quindi essere formalizzata dalla seguente relazione:

$$E_t = F_t - (1{,}1 \times N \times V_t)$$

dove:

- E_t rappresenta l'eccedenza prelevabile,
- F_t indica i fondi totali presenti nel contratto,
- $N \times V_t$ è il valore complessivo dei token in circolazione,
- il coefficiente $1{,}1$ introduce un margine del 10% a tutela della stabilità.

In pratica, l'artista può prelevare soltanto l'importo che eccede del 10% il valore complessivo dei token, lasciando questa quota di sicurezza all'interno del contratto. Ad esempio, se al termine del dodicesimo mese i fondi totali ammontano a 243.630 €, e il valore complessivo dei token in circolazione è pari a 23.400 €, la quantità prelevabile sarà:

$$E_{12} = 243.630 - (1{,}1 \times 23.400) = 243.630 - 25.740 = 217.890\,€$$

Questo significa che, pur mantenendo una riserva di sicurezza, l'artista può comunque ritirare oltre 217.000 €, assicurando contemporaneamente la piena copertura dei token e una stabilità economica di lungo periodo.

Questo equilibrio tra redditività per l'artista e sicurezza per gli investitori costituisce il cuore del modello: ogni parte beneficia in modo proporzionale al successo dell'artista, in un sistema trasparente, tracciabile e sostenibile nel tempo.

L'analisi quantitativa conferma che il modello è **economicamente solido, autosufficiente e resiliente**. La combinazione tra proventi delle vendite dei token e flussi periodici di royalties crea un'architettura finanziaria in grado di mantenere la solvibilità, accumulare riserve nel tempo e garantire la remunerazione sostenibile dell'artista.

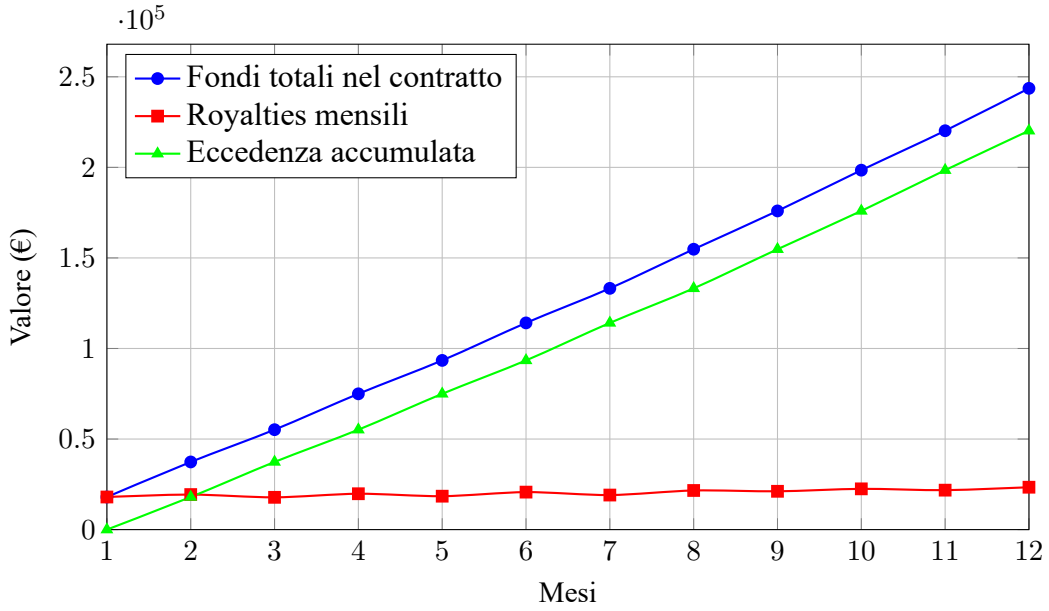


Figura 1: Andamento comparato dei fondi totali, royalties mensili ed eccedenza accumulata

La *Figura 1* mostra la dinamica di crescita dei tre elementi fondamentali: i fondi totali (linea blu), le royalties mensili (linea rossa) e l'eccedenza accumulata (linea verde), che rappresenta la quota prelevabile dall'artista.

4.10 Incentivi e soulbound token

Per evitare che l'acquisto dei token si riduca a un mero strumento di speculazione finanziaria, la piattaforma introduce una serie di **meccanismi di fidelizzazione** destinati a premiare chi mantiene i token per periodi prolungati. L'obiettivo non è soltanto quello di incentivare la detenzione, ma di trasformare il possesso dei token in un'esperienza capace di generare valore emotivo e senso di appartenenza, rafforzando nel lungo periodo il sostegno all'artista e consolidando una **community** attiva e coesa.

In questo quadro si inserisce l'utilizzo dei *Soulbound Token* (SBT), introdotti da Buterin, Weyl e Ohlhaver (2022) come nuova categoria di asset digitali non trasferibili. A differenza dei token fungibili e degli NFT tradizionali, gli SBT sono permanentemente vincolati a un wallet e non possono essere scambiati sul mercato. La loro funzione non è rappresentare un valore economico immediatamente liquidabile, ma attestare in modo duraturo identità, reputazione e impegni all'interno dell'ecosistema. Essi costituiscono quindi lo strumento ideale per certificare la fedeltà e la partecipazione degli utenti, senza rischiare derive speculative.

Gli incentivi previsti dalla piattaforma vengono quindi tradotti in SBT: la detenzione continuativa di token per almeno sei mesi può generare un SBT che garantisce

l'accesso a **merchandising esclusivo** o ad esperienze riservate, come concerti privati o sessioni dedicate ai fan più fedeli. Allo stesso modo, chi mantiene una quota significativa di token per dodici mesi riceve un SBT che attesta lo status di *long-term holder*, badge digitale inalienabile che abilita vantaggi ulteriori, quali **commissioni ridotte** sugli acquisti futuri, accesso anticipato a nuove emissioni o partecipazione a eventi esclusivi. Anche la presenza a concerti o iniziative speciali può essere registrata tramite SBT dedicati, che diventano vere e proprie medaglie digitali della carriera dell'artista e della relazione con la sua community.

Inoltre, il raggiungimento di traguardi artistici come il superamento di una certa soglia di ascolti mensili o l'ottenimento di riconoscimenti pubblici può essere certificato attraverso la distribuzione di SBT celebrativi, che attestano in maniera permanente tali milestone e rafforzano la trasparenza del modello.

Nel complesso, gli SBT diventano l'elemento cardine di un sistema che integra incentivi economici e identitari, garantendo da un lato **liquidità e sostenibilità**, dall'altro un legame simbolico e reputazionale tra artista e fan. L'adozione di questo strumento permette di superare le logiche speculative a breve termine, costruendo un ecosistema partecipativo, trasparente e orientato alla crescita di lungo periodo.

4.11 Emissione dei token e meccanismi di vesting

Per rendere operativo il **modello economico** descritto nei paragrafi precedenti, è necessario stabilire come i token vengano emessi e distribuiti nel tempo. Non basta infatti definirne la quantità complessiva: occorre anche regolare il rilascio progressivo, così da garantire equilibrio e coerenza nella crescita del progetto.

Il termine *vesting* indica un meccanismo, implementato tramite **smart contract**, che gestisce la messa a disposizione graduale di una certa quantità di token. In pratica, i token vengono creati fin dall'inizio (*pre-minted*), ma una parte di essi rimane temporaneamente bloccata, diventando disponibile solo dopo un determinato periodo o secondo una pianificazione temporale prestabilita. Generalmente, il vesting prevede una prima fase detta *cliff*, durante la quale nessun token può essere rilasciato, seguita da una distribuzione lineare o per tranche fino al completamento del periodo stabilito.

Questo meccanismo è ampiamente utilizzato nei progetti basati su blockchain perché consente di allineare gli incentivi tra i partecipanti e di garantire una crescita sostenibile e ordinata dell'ecosistema. Inoltre, la distribuzione scaglionata dei token può avere un effetto positivo sul coinvolgimento della community: sapere che nuove tranche saranno rese disponibili in futuro spinge gli utenti a mantenere alta l'attenzione sul progetto e sull'evoluzione dell'artista. Chi non acquista nella prima fase può attendere le successive, seguendo con interesse l'andamento del prezzo dei token e i risultati

dell'artista nel tempo. In questo modo, il vesting contribuisce anche a mantenere vivo l'interesse e la partecipazione della fanbase.

Nel modello proposto da questa tesi, la fornitura totale di token è fissata a $N_{\text{token}} = 500.000$. Tutti i token vengono generati al momento della creazione del contratto, ma solo una parte è resa immediatamente accessibile per la vendita e la circolazione sul marketplace. La quota restante è soggetta a un **periodo di vesting**, gestito da uno smart contract dedicato, che ne regola la disponibilità secondo una *schedule* predefinita. Questo approccio consente di mantenere un buon livello di liquidità iniziale, garantendo al contempo una distribuzione ordinata e progressiva delle risorse, coerente con la crescita del progetto e la partecipazione dell'artista e della community.

A titolo di esempio, si può prevedere una configurazione in cui il 50% dei token, pari a 250.000 unità, sia disponibile fin da subito al momento del lancio, mentre i restanti 250.000 vengano rilasciati gradualmente in tranche da 50.000 unità ogni tre mesi, fino al completamento dopo quindici mesi. Questo semplice schema assicura un equilibrio tra accessibilità iniziale e distribuzione programmata nel tempo.

Esempio di rilascio graduale: 50% iniziale e tranche da 50.000 token ogni 3 mesi

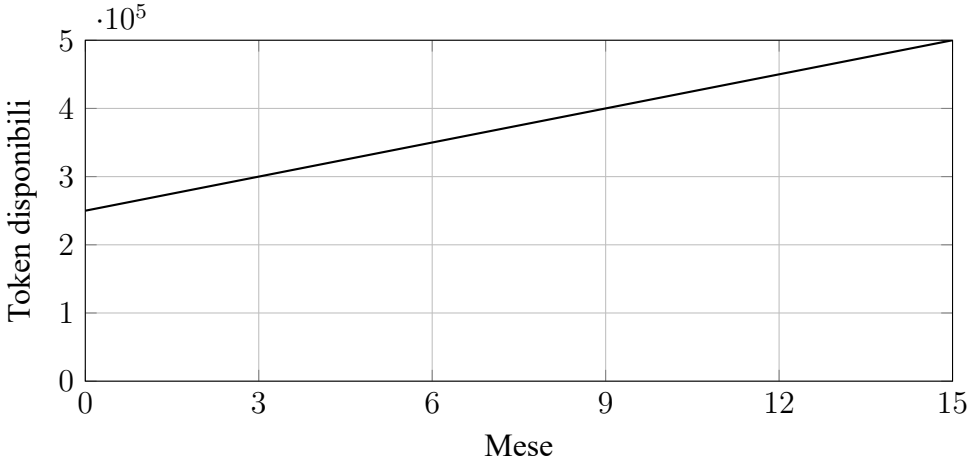


Figura 2: Curva illustrativa del rilascio progressivo dei token secondo una schedule di vesting predefinita.

In sintesi, il vesting costituisce uno strumento essenziale per assicurare una gestione trasparente e prevedibile dei token, contribuendo a rafforzare la stabilità del sistema, a mantenere viva l'attenzione della community e a favorire una crescita coerente e sostenibile del progetto nel tempo.

4.12 Analisi dei rischi e strategie di sicurezza del modello

L'adozione della **blockchain** e dei meccanismi di **tokenizzazione** applicati all'industria musicale porta con sé vantaggi significativi in termini di trasparenza, tracciabilità e automazione dei flussi economici. Allo stesso tempo, però, la natura distribuita e programmabile di tali sistemi apre scenari di rischio che non possono essere ignorati. Ogni progetto basato su infrastrutture decentralizzate deve essere analizzato alla luce di un adeguato **threat model**, ovvero un modello delle minacce potenziali in grado di evidenziare le vulnerabilità tecniche, organizzative ed economiche a cui il sistema è esposto. Nel contesto della distribuzione delle royalties musicali, l'elaborazione di un simile quadro risulta essenziale per garantire la fiducia degli utenti e la sostenibilità nel tempo.

Uno dei rischi più discussi riguarda gli attacchi alla blockchain stessa. Sebbene le principali reti pubbliche siano considerate altamente sicure grazie alla decentralizzazione e ai meccanismi di consenso, esistono scenari in cui un soggetto malintenzionato potrebbe ottenere un controllo eccessivo delle risorse computazionali o della quota di validazione, realizzando il cosiddetto *51% attack*. Nel caso della blockchain di **Bitcoin Gold** nel maggio 2018, gli aggressori riuscirono a controllare più della metà della potenza di calcolo della rete. Questo permise loro di riscrivere transazioni confermate, effettuare *double-spending* e sottrarre circa 18 milioni di dollari in criptovaluta. L'attacco dimostrò come anche reti decentralizzate possano essere vulnerabili se la distribuzione dei miner non è sufficientemente bilanciata e se la potenza di calcolo concentrata in poche mani diventa eccessiva.

Un altro esempio significativo riguarda la blockchain di **Ethereum Classic**, vittima di una serie di attacchi 51% tra il gennaio e il giugno 2019. In questo caso, gli aggressori sfruttarono la bassa difficoltà di mining rispetto a Ethereum, concentrando potenza di calcolo sufficiente a riscrivere transazioni e a prelevare fondi già confermati. Tali eventi hanno causato perdite per decine di milioni di dollari e generarono confusione tra gli utenti, mettendo in evidenza la criticità di fork malevoli e della gestione delle conferme delle transazioni.

Oltre ai 51% attack, le reti blockchain possono essere bersaglio di **attacchi denial-of-service** (DoS) mirati a saturare la rete. Nel caso del gioco Ethereum **Fomo3D** nel 2018, un alto numero di transazioni fu inviato simultaneamente per congestionare la rete e manipolare le condizioni di gioco, rallentando notevolmente le conferme delle transazioni legittime e aumentando i costi del gas per tutti gli utenti. Questo episodio evidenzia come la congestione della rete possa diventare una minaccia concreta anche per sistemi economici decentralizzati, con possibili ripercussioni sulla distribuzione regolare delle royalties.

Accanto agli attacchi rivolti alla rete sottostante, un ulteriore livello critico è rappresentato dagli **smart contract**. Questi ultimi costituiscono l'infrastruttura logica del modello e regolano automaticamente la distribuzione delle risorse e, di conseguenza, eventuali errori o vulnerabilità nel codice possono avere conseguenze dirette e immediate. L'esempio più famoso è rappresentato dalla *DAO attack* del 2016 su Ethereum. Un contratto denominato DAO, progettato come fondo di investimento decentralizzato, conteneva una vulnerabilità di **reentrancy**. Gli aggressori riuscirono a chiamare ripetutamente la funzione di prelievo prima che lo stato del contratto venisse aggiornato, riuscendo a sottrarre oltre 60 milioni di dollari in Ether. Questo incidente portò a un fork di Ethereum, generando Ethereum e Ethereum Classic come due reti separate.

Problemi più sottili ma ugualmente dannosi derivano da errori matematici nei contratti. Nel 2018, il protocollo DeFi **bZx** subì due attacchi quasi consecutivi, sfruttando errori di overflow e logiche di prestito flash. Gli aggressori manipolarono il prezzo di token su exchange decentralizzati, ottenendo profitti di circa 950.000 dollari in totale. Questo episodio dimostra come anche piccoli errori logici possano essere sfruttati per compromettere la distribuzione delle risorse.

Un'altra vulnerabilità comune riguarda il **controllo degli accessi**. Nel 2017, un bug nel **Parity Wallet** permise il congelamento permanente di oltre 150 milioni di dollari in Ethereum. Il contratto multisig non prevedeva correttamente i diritti di ownership e alcuni utenti malintenzionati poterono bloccare i fondi, rendendoli irreversibili. Anche errori di configurazione o mancanza di meccanismi di sicurezza possono avere conseguenze drammatiche nel contesto di royalties musicali tokenizzate, dove fondi e crediti devono essere distribuiti automaticamente a più stakeholder.

Alla luce di questi scenari, risulta evidente come un modello di distribuzione basato su blockchain debba essere sviluppato adottando un approccio di **security by design**. Comprendere i rischi è soltanto il primo passo: il passo successivo consiste nel definire e implementare una serie di misure di protezione che possano rendere il sistema resiliente anche in condizioni avverse. È proprio su questo secondo aspetto che si concentrano le seguenti considerazioni.

Dopo aver analizzato le principali minacce, infatti, è necessario valutare in che modo esse possano essere mitigate attraverso accorgimenti tecnici e organizzativi. Le contromisure non eliminano completamente i rischi, ma riducono la superficie di attacco e rafforzano la fiducia degli stakeholder nel modello. Nella sezione seguente vengono quindi illustrate le principali strategie di mitigazione, che spaziano dalla protezione della rete e degli smart contract fino alla salvaguardia della privacy e alla gestione operativa dei processi.

Contromisure tecniche e approcci di sicurezza

L'approccio non deve essere limitato a singoli accorgimenti puntuali, ma deve tradursi in una vera e propria architettura di **sicurezza multilivello**, capace di combinare tecniche crittografiche, pratiche di sviluppo sicuro e strumenti di monitoraggio.

Per contrastare i rischi legati ai *51% attack*, la misura più efficace consiste nell'appoggiarsi a blockchain pubbliche consolidate, caratterizzate da un alto livello di decentralizzazione e da una distribuzione equa della potenza di calcolo o della quota di staking. In alternativa, per progetti che operano su reti di dimensioni ridotte, possono essere adottati algoritmi di consenso differenti dal tradizionale Proof-of-Work, come il **Proof-of-Stake** o il **Delegated Proof-of-Stake**, i quali rendono più difficile concentrare il controllo della rete. Inoltre, meccanismi di checkpointing e sistemi di rilevazione di fork malevoli possono contribuire a ridurre l'impatto di eventuali tentativi di riscrittura della catena.

Gli attacchi di tipo **denial-of-service**, che mirano a congestionare la rete e ad aumentare i costi di transazione, possono essere mitigati attraverso l'adozione di soluzioni di **scalabilità di secondo livello**. Sistemi come *rollup* o *state channels* permettono di eseguire parte delle transazioni off-chain, riducendo il carico sulla rete principale e garantendo tempi di conferma più rapidi. Parallelamente, la definizione di politiche dinamiche per le commissioni di transazione (*gas fees*) e l'implementazione di limiti al numero massimo di transazioni per indirizzo in un dato intervallo temporale costituiscono ulteriori strumenti per prevenire situazioni di congestione artificiale.

Per quanto riguarda gli **smart contracts**, le misure di protezione devono essere ancora più stringenti. È ormai prassi consolidata affidarsi ad **audit indipendenti** da parte di società specializzate, capaci di individuare vulnerabilità logiche come quelle di *reentrancy* o di *integer overflow*. L'utilizzo di librerie standardizzate, come quelle fornite da **OpenZeppelin**, riduce sensibilmente la possibilità di errori nei moduli di base. Inoltre, la progettazione deve prevedere meccanismi di **fail-safe**, come le funzioni di pausa (*circuit breaker*), che consentono di sospendere temporaneamente l'esecuzione dei contratti in caso di comportamenti anomali o di attacchi in corso.

Gli incidenti legati al **controllo degli accessi**, come il caso del Parity Wallet, mostrano l'importanza di un modello di gestione delle autorizzazioni chiaro e ben definito. In questo contesto, l'adozione di contratti **multisig** rappresenta una delle strategie più efficaci per ridurre il rischio di errori o abusi. Un sistema multisig prevede che più soggetti, ciascuno in possesso di una propria chiave crittografica, debbano approvare congiuntamente le operazioni più sensibili, come il trasferimento di fondi o la modifica delle logiche di distribuzione. Questo approccio non solo previene azioni malevole da parte di un singolo individuo, ma riduce anche l'impatto di eventuali compromissioni

delle chiavi, introducendo un livello aggiuntivo di resilienza.

Infine, la resilienza complessiva del sistema dipende dall'adozione di strumenti di **monitoraggio continuo**. Dashboard di analisi delle transazioni, sistemi di allerta in caso di anomalie e procedure di risposta rapida a incidenti rappresentano componenti indispensabili per garantire la stabilità e la fiducia nel modello. Tali strumenti devono essere integrati in maniera trasparente, così da non ostacolare l'esperienza dell'utente finale ma al tempo stesso assicurare un controllo costante dell'infrastruttura.

In sintesi, la sicurezza di un modello basato su blockchain non può essere affidata a una singola misura, ma deve derivare dalla combinazione di più livelli di protezione. L'adozione di pratiche consolidate, l'integrazione di tecnologie avanzate e la predisposizione di procedure operative chiare costituiscono la base per costruire un sistema solido, capace di resistere agli attacchi più comuni e di tutelare gli interessi degli stakeholder coinvolti.
È importante sottolineare che le considerazioni presentate in questa sezione hanno un carattere prevalentemente teorico. La complessità delle tematiche affrontate, unita al livello di competenze tecniche richieste per l'implementazione di soluzioni di sicurezza avanzate, rende difficile una loro piena applicazione pratica nell'ambito di questo progetto di tesi. L'attenzione sarà quindi rivolta soprattutto alla gestione dei rischi più comuni e facilmente controllabili, dando per assodato che le misure di sicurezza di livello più elevato richiedano risorse, tempo e conoscenze altamente specialistiche che vanno al di fuori dagli obiettivi del presente lavoro.

4.13 Deploy finale degli smart contracts

In questo capitolo si descrivono in dettaglio le scelte e le pratiche adottate per il deploy finale degli smart contracts sviluppati nel corso del progetto. L'obiettivo è fornire una trattazione che sia al tempo stesso tecnica e applicativa: prima viene introdotto il concetto di gas e la metrica economica che ne deriva, successivamente viene analizzato il confronto pratico fra Ethereum e soluzioni Layer 2 (con particolare attenzione a Polygon), valutandone costi, tempi e implicazioni di sicurezza. Infine si propone una stima ragionata dei costi di deploy e delle strategie per contenere le spese operative in un progetto di questa portata.

Gas: definizione operativa e formula di calcolo Per comprendere le scelte di deploy è necessario partire dalla definizione operativa di *gas*. Il gas è l'unità di misura del lavoro computazionale richiesto per eseguire un'istruzione sulla macchina virtuale della blockchain (EVM). Ogni operazione in Solidity consuma una quantità prefissata di gas; il costo monetario di una transazione è quindi il prodotto fra le unità di gas con-

sumate e il prezzo del gas scelto dall'utente (espresso in Gwei, ovvero 10^{-9} ETH). In formula:

costo (ETH) $=$ gas_units $\times$ gas_price_gwei $\times 10^{-9}$

Questa relazione evidenzia due leve su cui il deploy e l'esercizio dei contratti impattano sui costi: le unità di gas, dipendenti dalla complessità del codice, e il prezzo del gas, variabile e determinato dalla domanda di spazio di blocco nel momento della trasmissione.

Caratteristiche operative: Ethereum mainnet vs Polygon Dal punto di vista architetturale Ethereum mainnet offre la massima garanzia di sicurezza e di decentralizzazione: la finalità degli stati, la profondità della rete di validatori e l'ampiezza dell'ecosistema costituiscono un livello di trust difficile da eguagliare. Questo primato di sicurezza si paga storicamente con costi di transazione maggiori e con latenza di conferma più alta nei periodi di congestione. Al contrario, soluzioni Layer 2 e sidechain come Polygon sono progettate per alleggerire il carico sul mainnet aggregando o elaborando off-chain gran parte dell'attività transazionale: il risultato pratico è una riduzione marcata del costo per singola transazione e, nella maggior parte dei casi, una conferma più rapida, a fronte però di un livello di garanzia differente rispetto al mainnet.

La scelta fra mainnet e Layer 2 impatta dunque su tre dimensioni essenziali: costo (spesa monetaria per operazione), velocità (tempo di conferma e esperienza utente) e sicurezza (modello di finalità e rischio di rollbacks o vulnerabilità infrastrutturali).

Quando usare Ethereum, quando Polygon e quando entrambi Non esiste una regola unica: la strategia più robusta è spesso ibrida. La pubblicazione di artefatti contrattuali che richiedono massima immutabilità e visibilità pubblica (ownership centrale, contratti di governance, o registri di proprietà che servono come fonte di verità legale) è consigliabile sul mainnet. Le operazioni che invece richiedono elevata frequenza e bassa latenza come interazioni utente, micro-pagamenti e minting massivo, sono tipicamente più sostenibili su Layer 2 come Polygon. L'architettura pattern che ne deriva prevede che il mainnet rappresenti lo strato di *settlement* (stato di verità a lungo termine), mentre il Layer 2 si occupi delle transazioni quotidiane, con possibili checkpoint verso il mainnet quando necessario.

Dal punto di vista tecnico, è utile adottare pattern che riducano il costo di deploy ripetuti. Un'altra pratica frequentemente adottata consiste nel ridurre la footprint di storage (evitando array dinamici non necessari e privilegiando mapping) e nell'ottimizzazione delle funzioni di inizializzazione in modo che siano chiamate una sola volta al deploy.

Analisi di costo Nel caso specifico di questo progetto, il deploy dei contratti principali ha richiesto i seguenti consumi di gas: circa 3.037.019 unità per `RoyaltyToken.sol`, circa 1.443.839 unità per `MockCoin.sol` e circa 3.975.968 unità per `SoulboundBadge.sol`. Considerando che alla data del 19 settembre 2025 il gas price su Ethereum mainnet era pari a 0.168 gwei, il costo effettivo può essere stimato come prodotto tra unità di gas e gas price, rapportato al valore dell'ether. Con questa configurazione, il deploy di `RoyaltyToken.sol` corrisponde a circa 0.00051 ETH, quello di `MockCoin.sol` a circa 0.00024 ETH, mentre il deploy di `SoulboundBadge.sol` si attesta intorno a 0.00067 ETH. In termini monetari, assumendo il prezzo di 3.524,97 € per ETH alla stessa data, il costo equivale rispettivamente a circa 1.80 €, 0.86 € e 2.36 €. Si osserva quindi che anche contratti complessi come `SoulboundBadge.sol`, pur richiedendo un numero elevato di unità di gas, comportano oggi un esborso contenuto grazie all'abbassamento medio del gas price.

Su Polygon (Layer 2) lo stesso stack architetturale porta a un ordine di grandezza di costo molto più basso: le singole operazioni di deploy e le chiamate ripetute sono spesso frazionali in termini di MATIC o di corrispettivo in dollari, rendendo economicamente sostenibile la strategia di effettuare la maggior parte delle interazioni lato Layer 2.

È fondamentale distinguere il costo one-time del deploy da quello operativo: il deploy è un costo singolo che, anche se non trascurabile, può essere ammortizzato quando l'applicazione scala, sono invece le operazioni ricorrenti (esecuzioni di payout, minting, trasferimenti) a determinare la sostenibilità economica dell'intero sistema nel lungo periodo.

Per contenere i costi e migliorare l'esperienza utente si raccomandano alcune pratiche consolidate. In fase di sviluppo è essenziale testare e profilare il consumo di gas con strumenti come i profiler di Hardhat/Foundry e i gas reporter, così da identificare funzioni particolarmente dispendiose in termini di storage o computazione.

Il deploy finale dei contratti va quindi valutato non solo come un'operazione tecnica, ma come una scelta architetturale che implica compromessi economici e giuridici. Ethereum rimane la scelta di riferimento quando la priorità è la massima sicurezza e la piena visibilità pubblica, mentre Polygon e altre soluzioni Layer 2 diventano indispensabili quando la necessità è processare un elevato numero di operazioni a costi contenuti.

5 Smart contract e librerie utilizzate

All'interno del progetto sono stati implementati diversi meccanismi tipici degli smart contract, come la gestione dei saldi, il controllo degli accessi, la possibilità di sospen-

dere temporaneamente alcune funzionalità critiche (*pause*) e le interazioni con token esterni tramite interfacce standardizzate. Per queste componenti si è fatto ampio uso di implementazioni consolidate e verificate, in particolare quelle messe a disposizione dalla libreria **OpenZeppelin**.

L'adozione di librerie standardizzate nasce da esigenze pratiche e di sicurezza: scrivere da zero ogni singolo componente espone a rischi noti, come errori di implementazione, logiche vulnerabili o casi limite non considerati. Le librerie pubbliche e ampiamente testate riducono sensibilmente la probabilità di bug critici e migliorano la qualità del codice. OpenZeppelin rappresenta oggi uno standard di riferimento per lo sviluppo di contratti *Solidity*, grazie alla modularità, ai continui audit e al supporto della comunità, ed è ampiamente utilizzata nella realizzazione di token conformi agli standard **ERC20** ed **ERC721**.

Nel scrittura dei contratti sono stati impiegati sei moduli principali provenienti da OpenZeppelin:

- l'implementazione completa dello standard **ERC20**;
- l'interfaccia **IERC20**, necessaria per le interazioni con token esterni;
- il modulo **Ownable**, per la gestione dei privilegi amministrativi;
- **Pausable**, che introduce un meccanismo di sospensione temporanea delle funzionalità critiche;
- l'implementazione dello standard **ERC721**, utilizzata per la gestione dei *soulbound token*;
- e infine **ReentrancyGuard**, impiegato per prevenire attacchi di tipo *reentrancy*.

Nei paragrafi successivi verrà descritto più nel dettaglio il ruolo di ciascun modulo e il modo in cui questi sono stati integrati nei contratti del progetto.

5.1 Librerie esterne

ERC20

Lo standard ERC20 definisce un set di funzioni e eventi che rendono un token fungibile e compatibile con l'ecosistema Ethereum. L'importazione

```
import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
```

fornisce un'implementazione completa delle primitive token, incluse le funzioni per leggere il bilancio (`balanceOf`), trasferire token (`transfer`), approvare spender (`approve`) e trasferire per conto di terzi (`transferFrom`), oltre ai meccanismi interni di minting (`_mint`) e burning (`_burn`). Queste funzioni interne consentono di creare o distruggere token in modo sicuro, aggiornando automaticamente il totale in circolazione e notificando gli eventi corrispondenti. Nel contratto `MockStablecoin` l'ereditarietà da `ERC20` ha permesso di definire in modo semplice il nome, il simbolo e il comportamento dei decimali, mentre nel contratto `RoyaltyToken` l'ereditarietà è stata sfruttata per modellare la totalità dell'offerta e integrare logiche addizionali come pool automatici di mercato e gestione dei dividendi senza dover reimplementare le primitive base della tokenomics.

L'utilizzo di `ERC20` semplifica inoltre la compatibilità con tool esterni (wallet, exchange, interfacce di terze parti) poiché tali strumenti si aspettano metodi e eventi conformi allo standard, garantendo interoperabilità immediata.

IERC20

L'interfaccia `IERC20` è la versione minimale che descrive il contratto attraverso la sola firma delle funzioni rilevanti, senza fornire un'implementazione. Nel codice essa viene impiegata per trattare un token esterno (nel nostro caso la stablecoin che funge da mezzo di pagamento), e l'import è il seguente:

```
import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
```

L'uso di un'interfaccia permette al contratto `ArtistCareerToken` di interagire con qualunque token che rispetti lo standard ERC20, indipendentemente dalla sua implementazione concreta. In particolare, le funzioni `transferFrom`, `transfer` e `balanceOf` sono chiamate sul riferimento di tipo `IERC20` per prelevare fondi, distribuire dividendi o verificare la liquidità disponibile. Questo approccio aumenta la modularità: la stablecoin utilizzata può essere sostituita tramite una semplice chiamata amministrativa (`setStablecoin`) senza richiedere modifiche alla logica di business.

Si noti come, nel codice, ogni trasferimento verso o dal contratto venga verificato controllando il valore booleano di ritorno (ad esempio assegnando il risultato di `transferFrom` a una variabile `ok` e richiamando `require(ok, "...")`), pratica difensiva che previene l'assunzione implicita che tutte le implementazioni aderiscano alle stesse convenzioni di revert. Questa tecnica garantisce robustezza anche in presenza di token ERC20 non conformi in modo completo.

Ownable

Il pattern dell'ownership è fondamentale per circoscrivere le operazioni sensibili ad un singolo attore amministrativo. L'import di OpenZeppelin

```
import "@openzeppelin/contracts/access/Ownable.sol";
```

fornisce la proprietà del contratto attraverso la funzione `owner()` e il modificatore `onlyOwner`, che abilita controlli di accesso semplici ed espressivi. Nel contesto del progetto, `onlyOwner` è applicato a funzioni quali la coniazione di nuova stablecoin (`mint` in `MockStablecoin`), il prelievo amministrativo di fondi (`ownerWithdrawStablecoin`) e le operazioni di configurazione.

L'uso di `Ownable` centralizza le responsabilità di gestione e semplifica i workflow di amministrazione in fase di sviluppo e testing. Allo stesso tempo, in ambienti production si raccomanda di combinare questo pattern con controlli di governance più articolati (multi-sig, timelock, meccanismi di voting) qualora si voglia ridurre il rischio legato ad un singolo custode delle chiavi. Dal punto di vista tecnico, la classe `Ownable` gestisce un indirizzo di proprietario e fornisce metodi per trasferirne la proprietà (`transferOwnership`) o rinunciarvi (`renounceOwnership`), permettendo un controllo completo sul ciclo di vita dell'ownership.

Pausable

Il modulo Pausable introduce un meccanismo di interruzione operativa che può essere attivato dall'operatore per rispondere a condizioni di emergenza. L'import utilizzato è

```
import "@openzeppelin/contracts/security/Pausable.sol";
```

e fornisce il modificatore `whenNotPaused` (e la controparte `whenPaused`) insieme alle funzioni interne per mettere in pausa (`_pause`) e ripristinare (`_unpause`) il contratto. Queste funzioni aggiornano uno stato booleano interno e generano eventi (`Paused`, `Unpaused`) per notificare i watcher esterni. Nel progetto le funzioni critiche come distribuzione di royalties e acquisto e vendita dei token sono protette da `whenNotPaused` in modo che, in caso di comportamento anomalo o vulnerabilità identificate, sia possibile sospendere temporaneamente le operazioni e procedere ad un'analisi e correzione senza esporre immediatamente i fondi a rischi addizionali.

La combinazione di `Ownable` e `Pausable` consente dunque una procedura amministrativa per la mitigazione rapida di incidenti: il proprietario può mettere il sistema in stato di pausa e, successivamente, coordinarne il ripristino dopo aver implementato le

contromisure. Dal punto di vista informatico, il pattern sfrutta uno stato interno `paused` e modificatori che bloccano l'esecuzione di funzioni critiche fino alla rimozione dello stato di pausa.

ERC721

Lo standard ERC721 rappresenta la specifica di riferimento per i token non fungibili (NFT), introdotti per consentire la gestione su blockchain di asset unici e non intercambiabili. A differenza dei token fungibili ERC20, in cui ogni unità è indistinguibile dalle altre, un token ERC721 incapsula un identificatore univoco che lo rende distinto da tutti gli altri esemplari. Questa caratteristica lo rende ideale per modellare beni digitali collezionabili, certificati di proprietà, attestati di identità o credenziali permanenti.

L'importazione avviene attraverso

```
import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
```

e fornisce un'implementazione completa delle funzioni e degli eventi richiesti dallo standard. Tra questi vi sono `balanceOf`, che indica il numero di NFT posseduti da un indirizzo, `ownerOf`, che restituisce il titolare di un determinato token, e le funzioni di trasferimento `transferFrom` e `safeTransferFrom`, che permettono lo spostamento dei token tra indirizzi mantenendo compatibilità con contratti esterni. Sono inoltre definiti gli eventi `Transfer` e `Approval`, indispensabili per tracciare e monitorare la circolazione dei token all'interno della rete. La classe implementa anche logiche di approvazione (`approve`, `setApprovalForAll`) e verifica di compatibilità con contratti destinatari (`_checkOnERC721Received`), garantendo sicurezza nei trasferimenti automatizzati.

Nel progetto lo standard ERC721 è stato utilizzato come base per l'implementazione dei *Soulbound Token*, una particolare tipologia di NFT caratterizzata dalla non trasferibilità. L'ereditarietà da `ERC721` ha permesso di sfruttare tutte le primitive standard, integrandole poi con una logica personalizzata che blocca le funzioni di trasferimento e di approvazione. In questo modo ogni token emesso diventa strettamente legato all'indirizzo che lo ha ricevuto, assumendo il ruolo di credenziale digitale permanente e non cedibile.

ReentrancyGuard

Il modulo `ReentrancyGuard` rappresenta un meccanismo di protezione essenziale contro una delle vulnerabilità più diffuse e pericolose nei sistemi basati su smart con-

tract: l'attacco di tipo *reentrancy*. Tale attacco si verifica quando una funzione esterna viene richiamata ricorsivamente prima che l'esecuzione precedente sia completata, permettendo a un attore malevolo di manipolare il flusso logico del contratto e di aggirare i controlli di stato. Il caso più celebre è quello legato al *DAO hack* del 2016, dove la mancanza di una protezione contro il rientro consentì il drenaggio sistematico dei fondi.

L'importazione da OpenZeppelin avviene tramite

```
import "@openzeppelin/contracts/security/ReentrancyGuard.sol";
```

e l'ereditarietà fornisce al contratto il modificatore `nonReentrant`, applicabile alle funzioni critiche che coinvolgono trasferimenti di token o ether. Tale modificatore utilizza una variabile di stato interna che blocca l'esecuzione ricorsiva della funzione fino al termine della chiamata corrente, impedendo che una funzione esterna richiamata durante l'esecuzione possa alterare lo stato interno.

Nel progetto questo meccanismo è stato applicato alle funzioni di movimentazione di fondi e di distribuzione dei dividendi, cioè quelle più sensibili a potenziali exploit. In combinazione con controlli addizionali sullo stato e con la logica di `Pausable`, `ReentrancyGuard` contribuisce a rafforzare la resilienza dell'intero sistema, riducendo in modo significativo la superficie di attacco. La sua adozione è quindi non solo una best practice, ma una misura imprescindibile ogni volta che un contratto interagisce con attori esterni e deve preservare l'integrità delle risorse economiche che gestisce.

5.2 MockStablecoin (MockCoin.sol)

Il contratto `MockCoin.sol` definisce un token di test che funge da "stablecoin" interno al progetto e ne costituisce il mezzo di pagamento per tutte le operazioni economiche simulate come l'acquisto di token o la distribuzione di nuove royalties. Questa scelta pratica consente di separare la logica applicativa (pricing, distribuzione, mercati) dall'asset utilizzato per i pagamenti, rendendo immediato il testing in ambienti locali senza la necessità di ricorrere a un token reale o a infrastrutture off-chain.

Per chiarezza espositiva riportiamo qui i frammenti di codice più significativi e, a seguire, la spiegazione dettagliata delle scelte implementative.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/security/Pausable.sol";
```

```
contract MockStablecoin is ERC20, Ownable, Pausable {
constructor(uint256 initialSupply) ERC20("MockCoin", "MC")
    Ownable(msg.sender) {
\_mint(msg.sender, initialSupply \* 10 \*\* decimals());
}
```

La funzione costruttore svolge due compiti principali: inizializzare i metadati del token (nome e simbolo) e coniare l'offerta iniziale destinata all'address di deploy. L'istruzione `ERC20("MockCoin", "MC")` definisce le stringhe che identificano il token, mentre la chiamata a `_mint(msg.sender, initialSupply * 10 ** decimals())` crea effettivamente la quantità iniziale, moltiplicando il valore numerico fornito in unità umane per il fattore di scala determinato da `decimals()`. Questo pattern è comune per mantenere la rappresentazione interna in unità intere e convenienti per la contabilità on-chain: l'unità minima del token (l'unità indivisibile registrata nei saldi) è pari a $10^{-decimals}$ dell'unità nominale.

Nel caso specifico il contratto definisce i decimali come due, come mostrato nel frammento seguente:

```
function decimals() public pure override returns (uint8) {
return 2;
}
```

Impostare i decimali a due significa che il token rappresenta valori con precisione ai centesimi, quindi 1 MockCoin corrisponde a 100 unità interne. Questa scelta è intenzionale: nel contesto di una stablecoin che simula l'euro, avere due decimali è più leggibile e coerente con la rappresentazione monetaria tradizionale. Dal punto di vista computazionale, Solidity lavora esclusivamente con interi dunque definire esplicitamente il numero di decimali evita ambiguità nelle operazioni aritmetiche e semplifica il calcolo dei prezzi e delle commissioni.

La funzione di `mint` fornisce la possibilità di emettere nuova moneta in modo controllato ed è protetta da due vincoli essenziali:

```
function mint(address to, uint256 amount) external onlyOwner
    whenNotPaused {
\_mint(to, amount \* 10 \*\* decimals());
}
```

`onlyOwner` limita questa operazione all'account amministrativo, mentre `whenNotPaused` impedisce la coniazione quando il contratto è stato posto in pausa. Queste due condizioni rappresentano meccanismi di difesa: la prima evita che attori non autorizzati aumentino arbitrariamente l'offerta, la seconda permette di bloccare le emissioni nel caso

in cui venga identificata una vulnerabilità o un comportamento anomalo. È importante sottolineare che, per un token utilizzato come mezzo di pagamento all'interno di un prototipo o di un ambiente di test, la disponibilità di una funzione di mint è pratica; in ambienti di produzione tuttavia si preferiscono limiti di emissione più stringenti o meccanismi di governance per mitigare il rischio di inflazione incontrollata.

Il contratto espone inoltre due funzioni amministrative per la gestione dello stato operativo:

```
function pause() external onlyOwner {
\_pause();
}

function unpause() external onlyOwner {
\_unpause();
}
```

Queste routine attivano e disattivano lo stato di pausa ereditato da `Pausable`. L'uso congiunto di `Ownable` e `Pausable` permette al deployer di intervenire rapidamente in caso di anomalie, sospendendo temporaneamente le operazioni critiche e riducendo l'esposizione ai rischi mentre si indaga e si applicano correzioni.

Dal punto di vista funzionale il `MockStablecoin` svolge le funzioni di riserva di valore e di mezzo di scambio all'interno dell'architettura: quando un utente compra token (nel contratto principale RoyaltyToken) trasferisce MockCoin al contratto di royalties tramite `transferFrom`, allo stesso modo i dividendi e le ricompense vengono distribuiti in MockCoin. Poiché il token non è effettivamente ancorato ad asset reali, la sua utilità è circoscritta all'ambiente applicativo del progetto: è una rappresentazione digitale che facilita il testing, la simulazione di flussi di cassa e la validazione delle logiche economiche del sistema.

Particolare attenzione viene dedicata alla fase di deploy e al comportamento del costruttore, dispositivo fondamentale sia dal punto di vista operativo che della sicurezza. Il costruttore è così definito:

```
constructor(uint256 initialSupply) ERC20("MockCoin", "MC")
    Ownable(msg.sender) {
     _mint(msg.sender, initialSupply * 10 ** decimals());
}
```

La scelta di parametrizzare la quantità iniziale di supply in fase di deploy deriva dall'esigenza, sottolineata nella trattazione architetturale, di flessibilizzare il modello secondo differenti scenari di test e simulazione. La funzione costruttore si aspetta un singolo parametro `initialSupply`, espresso in valore nominale privo di decimali (ad

esempio, 10000 per 100,00 MockCoin). Questo approccio semplifica la sintassi di deployment e riduce la probabilità di errori interpretativi relativi all'unità di misura. Internamente, si applica l'espressione `initialSupply * 10 ** decimals()`, aderendo alla convenzione di rappresentare i centesimi come frazioni distintive, con `decimals()` ridefinito per restituire 2: ciò consente di mantenere la divisibilità tipica delle valute fiat ed evitare problematiche di arrotondamento in contesti di micro-pagamenti e redistribuzioni frazionate.

Il costruttore stabilisce inoltre il possesso amministrativo del contratto tramite `Ownable(msg.sender)`, assicurando che la totalità delle funzioni di minting sia controllata dal wallet che ha eseguito il deploy.

In sintesi, `MockStablecoin` è un componente essenziale per la sperimentazione e la dimostrazione del modello economico: fornisce un mezzo stabile e controllabile per effettuare operazioni di mercato e simulare l'acquisto o vendita dei token dell'artista.

5.3 ArtistCareerToken (RoyaltyToken.sol)

Il contratto `RoyaltyToken.sol` implementa il token di carriera dell'artista e l'insieme delle logiche economiche ad esso associate: vesting dei token riservati, incasso e distribuzione delle royalties, l'aggiornamento del valore del token e un mercato interno per comprare e vendere token direttamente contro la riserva del contratto. Il contratto eredita `ERC20` per lo standard token fungibile, `Ownable` per la proprietà amministrativa, `Pausable` per il meccanismo di sospensione delle operazioni e `ReentrancyGuard` per mitigare attacchi di reentrancy. Le scelte implementative perseguono due obiettivi: modellare il ciclo economico dell'artista in modo riproducibile on-chain e introdurre salvaguardie operative che riducano la superficie di attacco nelle operazioni critiche.

Per chiarezza espositiva riportiamo qui i frammenti di codice più significativi e spieghiamo le scelte progettuali.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/security/ReentrancyGuard.sol";
import "@openzeppelin/contracts/security/Pausable.sol";

contract ArtistCareerToken is ERC20, Ownable, Pausable,
   ReentrancyGuard {
```

```
IERC20 public stablecoin;
uint256 public lastRoyalties;
uint256 public pricePerToken;

uint256 public constant PRICE_SCALE = 1e2;

// Vesting variables
uint256 public vestingAmount;
uint256 public vestingStartTime;
uint256 public vestingDuration;
uint256 public numTranches;
uint256 public currentTranche;
uint256 public releasedAmount;
uint256 public trancheDuration;

// Indirizzo autorizzato a distribuire royalties
address public immutable royaltyDistributor;

event RoyaltiesDistributed(address indexed from, uint256
   amount, uint256 newPrice);
event PriceUpdated(uint256 oldPrice, uint256 newPrice, uint256
    lastRoyalties);
event TokensBought(address indexed buyer, uint256 tokenAmount,
    uint256 cost);
event TokensSold(address indexed seller, uint256 tokenAmount,
   uint256 revenue);
event VestingReleased(uint256 amount, uint256 tranche);
```

Il contratto definisce alcune variabili di stato chiave: `stablecoin` è l'ERC-20 utilizzato per ricevere e pagare royalty, `lastRoyalties` memorizza l'ultimo ammontare di royalties ricevute, `pricePerToken` è il valore unitario del token, memorizzato con uno scaling definito da `PRICE_SCALE`. Le variabili di vesting tengono traccia dello stock bloccato nel contratto, delle tempistiche e dei token già rilasciati.

Segue il costruttore, che concentra le principali decisioni di configurazione.

```
constructor(
IERC20 _stablecoin,
string memory name,
string memory symbol,
uint256 totalSupply,
uint256 initialRoyalties,
```

```
address _owner,
address _royaltyDistributor,
uint256 _vestingPercentage,
uint256 _vestingDuration,
uint256 _numTranches
) ERC20(name, symbol) Ownable(_owner) {
require(_royaltyDistributor != address(0), "Invalid
   distributor address");
require(totalSupply > 0, "totalSupply>0");
require(_vestingPercentage <= 100, "Vesting percentage too
   high");

stablecoin = _stablecoin;
royaltyDistributor = _royaltyDistributor;
lastRoyalties = initialRoyalties;

vestingAmount = (totalSupply * _vestingPercentage) / 100;
vestingStartTime = block.timestamp;
vestingDuration = _vestingDuration;
numTranches = _numTranches;
currentTranche = 0;
releasedAmount = 0;
trancheDuration = _vestingDuration / _numTranches;

uint256 immediateAmount = totalSupply - vestingAmount;
_mint(_owner, immediateAmount);
_mint(address(this), vestingAmount);

pricePerToken = (initialRoyalties > 0) ? (initialRoyalties *
   PRICE_SCALE)
    / totalSupply : 0;
}
```

Il costruttore assegna lo stablecoin di riferimento, imposta l'indirizzo immutabile del distributore di royalties e registra l'ammontare iniziale di royalties ricevute, utile alla definizione del prezzo iniziale. La logica di vesting calcola la quota di supply soggetta a rilascio progressivo, imposta il tempo di inizio a `block.timestamp` e calcola la durata di ciascuna tranche. Il contratto conia immediatamente due "porzioni" distinte: la parte disponibile immediatamente all'owner e la parte soggetta a vesting che resta nel contratto fino al rilascio progressivo.

```
// ---------------- Vesting ----------------
function releaseVesting() external nonReentrant {
require(block.timestamp >= vestingStartTime, "Vesting not
   started");
require(currentTranche < numTranches, "Vesting completed");

uint256 timeSinceStart = block.timestamp - vestingStartTime;
uint256 tranchesPassed = timeSinceStart / trancheDuration;

if (tranchesPassed > numTranches) {
    tranchesPassed = numTranches;
}

require(tranchesPassed > currentTranche, "No new tranches to
   release");

uint256 tranchesToRelease = tranchesPassed - currentTranche;
uint256 amountPerTranche = vestingAmount / numTranches;
uint256 amountToRelease = tranchesToRelease * amountPerTranche
   ;

if (currentTranche + tranchesToRelease == numTranches) {
    amountToRelease = vestingAmount - releasedAmount;
}

releasedAmount += amountToRelease;
currentTranche += tranchesToRelease;

_transfer(address(this), owner(), amountToRelease);
emit VestingReleased(amountToRelease, currentTranche);
}

// ---------------- Royalties ----------------
function distributeRoyalties(uint256 royaltyAmount) external
    whenNotPaused nonReentrant {
require(msg.sender == royaltyDistributor, "Not authorized");
require(royaltyAmount > 0, "royaltyAmount>0");
bool ok = stablecoin.transferFrom(msg.sender, address(this),
   royaltyAmount);
require(ok, "stablecoin transferFrom failed");
```

```
uint256 totalTok = totalSupply();

uint256 oldPrice = pricePerToken;
uint256 newPrice = (lastRoyalties == 0)
    ? (totalTok > 0 ? (royaltyAmount * PRICE_SCALE) / totalTok
        : 0)
    : (pricePerToken * royaltyAmount) / lastRoyalties;

pricePerToken = newPrice;
lastRoyalties = royaltyAmount;

emit RoyaltiesDistributed(msg.sender, royaltyAmount, newPrice)
   ;
emit PriceUpdated(oldPrice, newPrice, lastRoyalties);
}

// ---------------- Marketplace ----------------
function buyFromContract(uint256 tokenAmount) external
    whenNotPaused nonReentrant { ... }
function sellToContract(uint256 tokenAmount) external
    whenNotPaused nonReentrant { ... }

// ---------------- Admin ----------------
function ownerWithdrawStablecoin(uint256 amount) external
   onlyOwner
    nonReentrant { ... }
function setStablecoin(IERC20 _stablecoin) external onlyOwner
   { ... }
function pause() external onlyOwner { _pause(); }
function unpause() external onlyOwner { _unpause(); }
function viewPricePerToken() external view returns (uint256)
    { return pricePerToken; }
function viewLastRoyalties() external view returns (uint256)
    { return lastRoyalties; }
```

Parametri del costruttore: significato e linee guida per il deploy

- `_stablecoin (IERC20)`: indirizzo dello smart contract ERC-20 utilizzato per pagare e ricevere royalties (nel contesto di questo progetto, MockCoin.sol).

- `name, symbol`: stringhe descrittive del token, usate per identificazione e interazione UI/metamask.
- `totalSupply (uint256)`: quantità complessiva di token emessi. Determina sia la porzione disponibile immediatamente sia quella sottoposta a vesting.
- `initialRoyalties (uint256)`: ammontare iniziale delle royaltiesregistrate, usato per il calcolo del prezzo iniziale del token.
- `_owner (address)`: indirizzo che riceve la porzione immediata dei token e ha i privilegi amministrativi.
- `_royaltyDistributor (address)`: indirizzo autorizzato a chiamare `distribute Royalties`.
- `_vestingPercentage (uint256)`: percentuale della supply totale soggetta a vesting.
- `_vestingDuration (uint256)`: durata complessiva del vesting, in secondi.
- `_numTranches (uint256)`: numero di tranche in cui il periodo di vesting viene suddiviso.

5.4 SoulboundBadge (SoulBoundToken.sol)

Il contratto `SoulBoundToken.sol` implementa i badge non trasferibili (soulbound) che vengono assegnati agli utenti sulla base di criteri di *holding amount* e *holding duration* relativi al possesso di token `ArtistCareerToken`. Il contratto eredita `ERC721` per lo standard NFT, `Ownable` per la gestione amministrativa e utilizza `Counters` di OpenZeppelin per gestire gli ID dei token. L'assegnazione dei badge si appoggia su un meccanismo di claim basato sul tempo e sul possesso minimo richiesto. L'utilizzo pratico di questi token è già stato discusso nel paragrafo *4.10 Incentivi e Soulbond Token*.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/utils/Counters.sol";

interface IRoyaltyToken {
```

```
    function balanceOf(address account) external view returns
       (uint256);
}

contract SoulboundBadge is ERC721, Ownable {
    using Counters for Counters.Counter;
    Counters.Counter private _tokenIds;

    IRoyaltyToken public royaltyToken;

    struct BadgeType {
        uint256 id;
        string name;
        uint256 minHolding;
        uint256 holdingDuration;
        bool active;
    }

    mapping(uint256 => BadgeType) public badgeTypes;
    uint256 public badgeTypeCount;

    mapping(uint256 => mapping(address => uint256)) public
       holdingStartTimestamp;
    mapping(uint256 => uint256) public tokenIdToBadgeType;
```

Il contratto definisce i badge come strutture dati con identificativo, nome, requisito minimo di token posseduti, durata minima di possesso e stato di attivazione. La mappatura `holdingStartTimestamp` tiene traccia del momento in cui un utente inizia a soddisfare i requisiti di possesso, mentre `tokenIdToBadgeType` collega ciascun NFT al tipo di badge corrispondente.

```
event BadgeTypeCreated(uint256 indexed badgeTypeId, string
   name,
    uint256 minHolding, uint256 holdingDuration);
event BadgeTypeUpdated(uint256 indexed badgeTypeId, string
   name,
    uint256 minHolding, uint256 holdingDuration, bool active);
event HoldingProgressUpdated(uint256 indexed badgeTypeId,
   address indexed user,
```

```
    uint256 startTimestamp);
event BadgeClaimed(uint256 indexed badgeTypeId, address
   indexed user,
    uint256 tokenId);
event BadgeAwardedByAdmin(uint256 indexed badgeTypeId, address
    indexed user,
    uint256 tokenId);
event BadgeRevoked(uint256 indexed tokenId, address indexed
   user);
```

Gli eventi tracciano la creazione e aggiornamento dei tipi di badge, il progresso del possesso degli utenti, la rivendicazione dei badge, l'assegnazione da parte dell'amministratore e la revoca.

```
constructor(address _royaltyToken, string memory name_, string
    memory symbol_)
    ERC721(name_, symbol_)
    Ownable(msg.sender)
{
    require(_royaltyToken != address(0), "Royalty token
       address zero");
    royaltyToken = IRoyaltyToken(_royaltyToken);
}
```

Il costruttore inizializza l'NFT con nome e simbolo e associa il contratto `RoyaltyToken` che fornirà le informazioni sul saldo dei possessori.

```
// Creazione e aggiornamento badge
function createBadgeType(string calldata name, uint256
   minHolding,
    uint256 holdingDurationSeconds) external onlyOwner returns
        (uint256);
function updateBadgeType(uint256 badgeTypeId, string calldata
   name,
    uint256 minHolding, uint256 holdingDurationSeconds, bool
       active)
    external onlyOwner;
```

L'amministratore può creare nuovi tipi di badge definendo nome, quantità minima di token posseduti e durata minima di possesso. I badge possono essere aggiornati o disattivati.

```
// Aggiornamento progresso
function updateHoldingProgress(uint256 badgeTypeId, address
   user) public;
function secondsHeldSoFar(uint256 badgeTypeId, address user)
   public
    view returns (uint256);
```

Il contratto registra il timestamp di inizio possesso di ogni utente per ciascun tipo di badge. Se l'utente soddisfa i requisiti di `minHolding`, il timestamp viene impostato, in caso contrario viene azzerato.

```
// Claim da utente
function claimBadge(uint256 badgeTypeId) external returns (
   uint256);
function claimBadgeFor(uint256 badgeTypeId, address user)
   external
    returns (uint256);
function _claimBadgeFor(uint256 badgeTypeId, address user)
   internal
    returns (uint256);

// Assegnazione amministrativa e revoca
function awardBadgeByAdmin(uint256 badgeTypeId, address user)
   external
    onlyOwner returns (uint256);
function revokeBadge(uint256 tokenId) external onlyOwner;
```

Gli utenti possono rivendicare i badge solo se soddisfano i requisiti di possesso e durata. L'amministratore può assegnare badge direttamente o revocarli, garantendo flessibilità di gestione.

```
// Blocca trasferimenti e approvazioni
```

```
function _update(address to, uint256 tokenId, address auth)
    internal
     virtual override returns (address);
function approve(address, uint256) public pure override;
function setApprovalForAll(address, bool) public pure override
    ;
```

I badge sono non trasferibili: i trasferimenti e le approvazioni sono bloccati. Solo minting (da zero) e burning (verso zero) sono consentiti.

```
// Aggiornamento contratto RoyaltyToken
function setRoyaltyToken(address _royaltyToken) external
    onlyOwner;

// Recupero informazioni badge
function getBadgeType(uint256 badgeTypeId) external view
     returns (BadgeType memory);

// Imposta manualmente timestamp possesso
function adminSetHoldingStart(uint256 badgeTypeId, address
    user,
     uint256 timestamp) external onlyOwner;

// Burn utente
function burn(uint256 tokenId) external;
```

Queste funzioni consentono di aggiornare l'indirizzo del token di riferimento, leggere le informazioni sui badge, impostare manualmente i timestamp di possesso e bruciare i propri badge.

Il flusso operativo tipico prevede che un utente mantenga un saldo minimo di token `ArtistCareerToken` per un certo periodo. Il contratto registra il timestamp di inizio possesso. Quando l'utente soddisfa sia il requisito di quantità che quello temporale, può rivendicare il badge, generando un NFT non trasferibile. L'amministratore può intervenire per assegnare badge direttamente o revocarli.

6 Introduzione al prototipo

Il prototipo realizzato per questa tesi concentra in un'unica interfaccia web le funzionalità necessarie a sperimentare e validare le logiche implementate dagli smart contract sviluppati. La parte client è stata implementata con `React` per sfruttare la sua modularità, la separazione tra componenti presentazionali e logiche di stato e la vasta disponibilità di strumenti per il testing e il debugging. Per l'interazione con la blockchain si è scelto `ethers.js`, una libreria che offre un'interfaccia diretta e sicura per la costruzione, la firma e l'invio delle transazioni, nonché per l'ascolto degli eventi emessi dai contratti. Questa scelta ha permesso di mantenere chiaro il confine tra la logica di business eseguita on-chain e la logica di presentazione gestita lato client.

Al fine di garantire ripetibilità sperimentale, rapidità nello sviluppo e assenza di costi legati al gas, tutte le transazioni descritte nel capitolo sono state eseguite su una blockchain locale fornita da `Ganache`, con il wallet (`MetaMask`) configurato per connettersi alla rete di test locale. Questo setup ha reso possibile simulare scenari reali inclusi errori di transazione e tempi di mining senza dipendere dalla rete pubblica.

In questo capitolo verranno mostrate le schermate principali del prototipo, accompagnate da una descrizione delle logiche sottostanti e delle modalità di interazione con gli smart contract. I dati rappresentati nelle interfacce non hanno alcuna validità reale, ma sono stati generati unicamente per scopi dimostrativi e di testing, così da illustrare il funzionamento delle componenti sviluppate in un contesto controllato.

6.1 Connessione al wallet

La prima schermata che l'utente visualizza all'apertura del prototipo è quella dedicata alla connessione del wallet, passaggio preliminare e indispensabile per poter interagire con tutte le funzionalità disponibili. L'interfaccia invita l'utente a scegliere il portafoglio digitale da utilizzare, specificando chiaramente che esso rappresenta l'identità attraverso la quale verranno firmate ed eseguite le operazioni sulla blockchain. Nel caso del prototipo sviluppato, la scelta è ricaduta su `MetaMask`, uno degli strumenti più diffusi e di semplice utilizzo, che ben si presta a scopi sperimentali e didattici.

Dal punto di vista dell'esperienza utente, la schermata è pensata per essere immediata e intuitiva: un pulsante consente di avviare la connessione e, una volta autorizzata l'operazione, l'interfaccia mostra l'indirizzo del wallet connesso. In questo modo l'utente ha sempre la consapevolezza di operare in un contesto sicuro e coerente, evitando possibili ambiguità o fraintendimenti.

Questa fase di connessione è cruciale anche per trasmettere all'utente il principio di responsabilità individuale che caratterizza l'interazione con la tecnologia blockchain. Ogni operazione che verrà eseguita nelle schermate successive sarà infatti legata all'identità digitale del wallet selezionato e richiederà una conferma esplicita, garantendo trasparenza e controllo personale sulle azioni svolte.

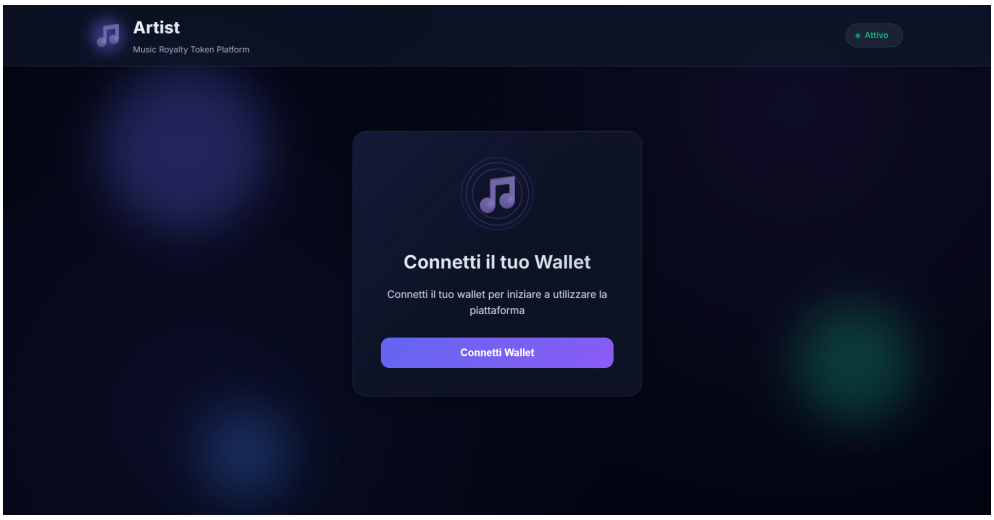


Figura 3: Schermata di connessione al wallet.

6.2 Pagina iniziale e sezione "Wallet"

La pagina iniziale del prototipo si apre direttamente sulla sezione *Wallet*, cuore dell'interfaccia utente, dalla quale è possibile avere una visione immediata e complessiva della propria situazione. In alto compaiono anche le altre sezioni principali dell'applicazione *Operazioni*, *Badge* e *Log* che consentono di esplorare in maniera più approfondita le diverse funzionalità disponibili.

Nel caso qui riportato, il wallet connesso è identificato come *Admin*. Questo status particolare comporta privilegi aggiuntivi rispetto a un utente ordinario: in particolare, la possibilità di sospendere o riattivare l'operatività del contratto tramite i pulsanti *Pausa* e *Riprendi*, visibili nella parte superiore dell'interfaccia. Questi controlli permettono di testare concretamente le logiche di gestione amministrativa previste negli smart contract, garantendo che sia sempre possibile bloccare temporaneamente l'attività in caso di necessità e successivamente ripristinarla in modo sicuro.

Al centro della schermata si trovano le informazioni principali legate al portafoglio: il prezzo attuale del token, determinato secondo la formula già illustrata nei capitoli precedenti della tesi, il numero di token posseduti dall'utente, il saldo in `MockCoin`, la

moneta di scambio creata ad hoc (`MockCoin.sol`) per simulare gli scambi e le operazioni economiche, e infine il valore complessivo del portafoglio, calcolato moltiplicando il numero di token posseduti per il prezzo unitario corrente. In questo modo, l'utente dispone in tempo reale di una rappresentazione sintetica ma esaustiva della propria posizione.

La parte inferiore della schermata è dedicata al *Vesting Schedule*, ovvero al piano di rilascio graduale dei token. Nel caso riportato il processo di vesting risulta già completato, e l'interfaccia lo evidenzia mostrando sia il totale dei token originariamente sottoposti a vincolo sia le tranche già rilasciate, con l'indicazione che tutti i token previsti sono ormai stati resi disponibili. Questa sezione rende tangibile e facilmente comprensibile una delle logiche centrali del contratto, cioè il rilascio progressivo delle risorse nel tempo, garantendo trasparenza e chiarezza sullo stato attuale.

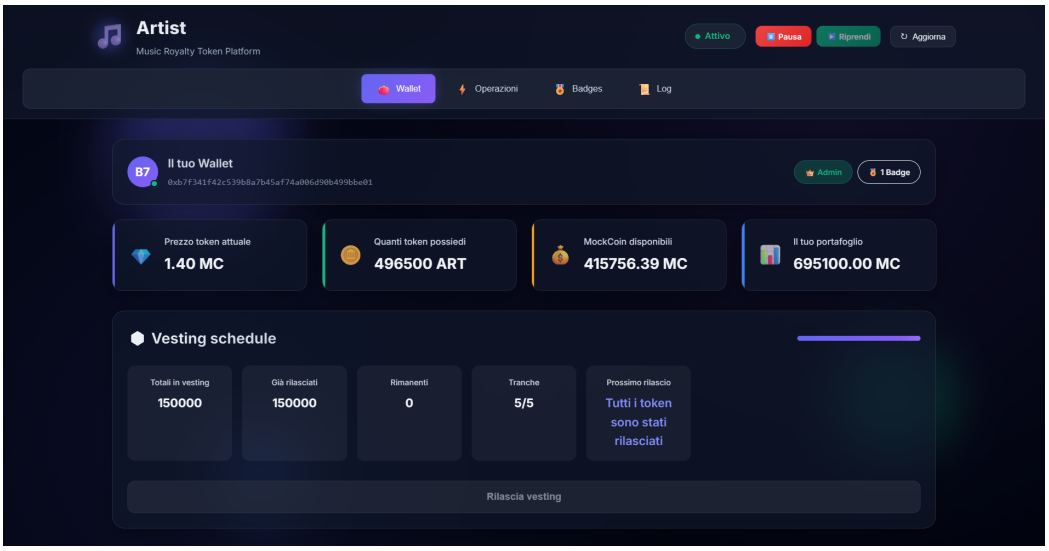


Figura 4: Pagina iniziale del prototipo: sezione *Wallet* con riepilogo dei saldi, del prezzo del token e del piano di vesting.

6.3 Sezione "Operazioni"

La sezione *Operazioni* rappresenta uno dei punti chiave del prototipo, in quanto consente all'utente di interagire attivamente con i token e con la moneta di progetto `MC` (MockCoin). L'interfaccia è stata progettata per essere immediata e intuitiva, offrendo tre principali funzionalità: acquisto di token, vendita di token e trasferimento di `MC` tra wallet.

Nella parte dedicata all'acquisto e alla vendita, l'utente può selezionare il numero di token desiderato. I pulsanti per confermare le transazioni sono evidenziati in modo chiaro, richiedendo una conferma esplicita per ogni operazione, così da trasmettere all'utente il concetto di responsabilità individuale nell'interazione con la blockchain.

Un'altra funzionalità importante presente in questa sezione è la possibilità di trasferire `MockCoin` tra wallet. Pur trattandosi di una funzione pensata principalmente a scopo di testing, essa consente di simulare facilmente scambi tra diversi account, rendendo possibile la verifica delle logiche di saldo e di trasferimento implementate nello smart contract. Grazie a questo strumento, è possibile testare scenari realistici di scambio e verifica dei saldi senza dipendere da contesti esterni, assicurando un controllo completo sulle operazioni all'interno della rete locale.

L'interfaccia della sezione *Operazioni* integra inoltre un feedback immediato per ogni transazione: al completamento di un acquisto, di una vendita o di un trasferimento, l'utente riceve una notifica visiva che conferma l'esito dell'operazione e aggiorna automaticamente il saldo dei token e dei `MockCoin` disponibili. Questa scelta di design aumenta la chiarezza e la fiducia dell'utente, rendendo ogni interazione trasparente e facilmente monitorabile.

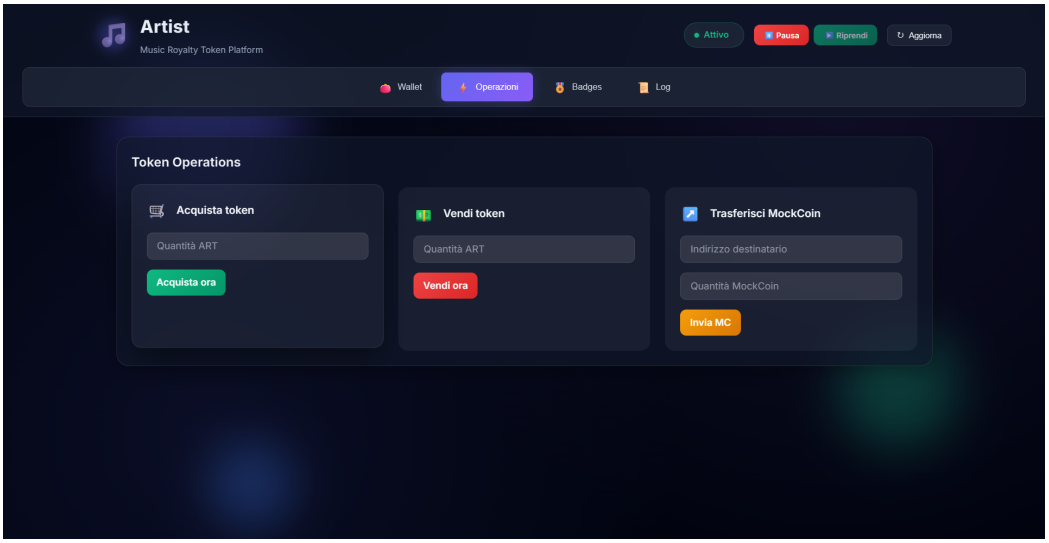


Figura 5: Sezione *Operazioni*: acquisto, vendita e trasferimento di token e `MC`.

6.4 Sezione "Badge"

La sezione *Badge* del prototipo introduce un concetto centrale della sperimentazione: i *soulbound token*.

Nel prototipo attuale, i badge non possiedono un valore economico diretto né una funzione operativa concreta, essi servono principalmente a illustrare le logiche di assegnazione e gestione dei token non trasferibili. Tuttavia, l'implementazione è stata pensata per consentire futuri sviluppi in cui i badge possano acquisire funzioni pratiche e incentivi reali, ad esempio: sconti nell'acquisto di token, accesso anticipato a nuove emissioni, partecipazione a eventi privati dell'artista, oppure accesso a merchandising esclusivo. In questo modo, i badge diventano strumenti di engagement e fidelizzazione, andando a premiare gli utenti più attivi e coinvolti.

La creazione di un badge è riservata all'*Admin*. Nell'immagine riportata, si vede un badge pronto per essere riscattato, evidenziando come l'admin possa istituire nuovi riconoscimenti da distribuire agli utenti.

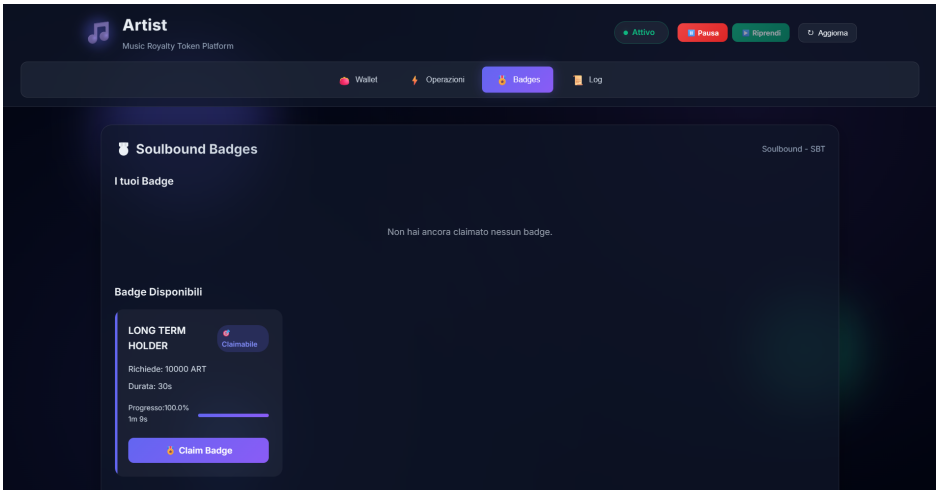


Figura 6: Creazione di un badge pronto per essere riscattato. Solo l'admin può creare nuovi badge.

Una volta che il badge viene effettivamente riscattato dall'utente che soddisfa i requisiti, la schermata si aggiorna per mostrare lo stato "posseduto", rendendo visibile l'avvenuta attribuzione. Questo passaggio permette di monitorare in modo immediato chi ha ottenuto ciascun badge e di confermare che le condizioni siano state rispettate, garantendo trasparenza e tracciabilità all'interno del sistema.

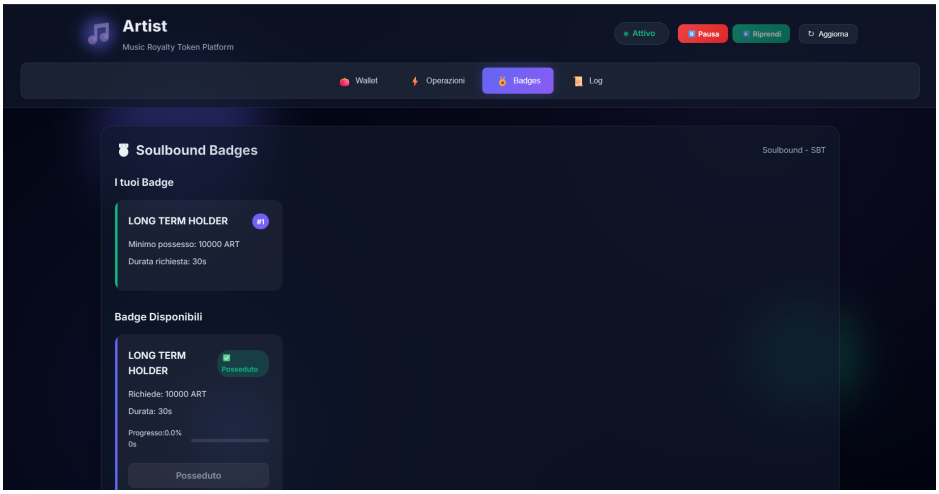


Figura 7: Badge riscattato dall'utente: lo stato viene aggiornato per riflettere l'avvenuta assegnazione.

6.5 Sezione "Log"

La sezione *Log* del prototipo funge da registro cronologico delle operazioni eseguite, offrendo all'utente una panoramica immediata delle attività svolte all'interno del sistema. In questa schermata vengono visualizzati eventi quali trasferimenti, aggiornamenti di stato e altre interazioni con gli smart contract, permettendo di monitorare in tempo reale l'evoluzione del portafoglio e delle azioni compiute.

Nell'immagine riportata, si mostra solamente una piccola parte dei log disponibili, non sono infatti visibili tutte le tipologie di eventi registrati, come la vendita o l'acquisto di token, il riscatto dei badge, i trasferimenti di `MockCoin` o il rilascio delle tranche di vesting. Questi log aggiuntivi vengono comunque generati e tracciati dal sistema, garantendo un monitoraggio completo e coerente delle operazioni.

L'obiettivo principale della sezione *Log* è di supporto implementativo: consente agli sviluppatori e ai tester di verificare che tutte le funzionalità dei contratti operino come previsto, che le transazioni vengano registrate correttamente e che le logiche di business siano rispettate. In questo senso, i log rappresentano uno strumento prezioso per la validazione del prototipo, permettendo di individuare rapidamente eventuali anomalie o comportamenti inattesi.

La visualizzazione cronologica e ordinata facilita inoltre la comprensione della sequenza delle operazioni, rendendo trasparente ogni azione eseguita e fornendo una sorta di "traccia digitale" che accompagna tutte le interazioni con la piattaforma. Questo

approccio rafforza l'affidabilità del prototipo e supporta una sperimentazione sicura e controllata, coerente con gli obiettivi della tesi.

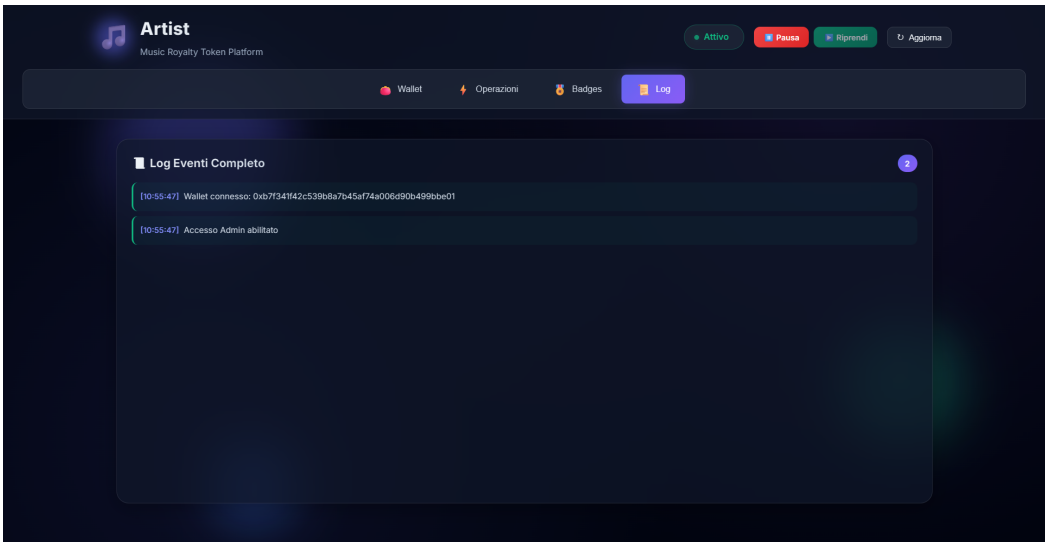


Figura 8: Sezione *Log*: registro cronologico delle operazioni eseguite nel prototipo.

6.6 Messaggi di conferma delle transazioni

Un elemento fondamentale dell'esperienza utente nel prototipo è rappresentato dai messaggi di conferma generati da *MetaMask* prima dell'esecuzione di qualsiasi transazione sulla blockchain. Questi pop-up svolgono un ruolo chiave nella sicurezza e nella trasparenza del sistema, richiedendo all'utente di autorizzare esplicitamente ogni operazione, e fornendo un feedback immediato su quanto sta per avvenire.

Nell'esempio riportato, relativo all'acquisto di token, una volta selezionato il numero desiderato e confermato l'importo in `MockCoin`, si apre un pop-up di MetaMask che mostra tutti i dettagli della transazione come l'indirizzo del contratto e le gas fees. Solo dopo aver esaminato e approvato questi dettagli, la transazione viene inviata alla blockchain.

Questa modalità di interazione rafforza il principio di responsabilità individuale, tipico della tecnologia blockchain: ogni azione è legata in modo univoco all'identità digitale del wallet e richiede una conferma esplicita da parte dell'utente. Inoltre, i messaggi di conferma permettono di prevenire errori accidentali, fornendo un'ulteriore occasione di controllo prima che una transazione diventi definitiva.

Dal punto di vista del prototipo, l'integrazione di questi pop-up ha anche una funzione didattica e illustra concretamente come le transazioni blockchain richiedano sempre

la firma digitale dell'utente, rendendo tangibile il concetto di consenso e autorizzazione che sottende ogni operazione on-chain. In questo modo, anche chi non ha familiarità con la blockchain può comprendere facilmente i meccanismi di sicurezza e verifica alla base delle interazioni.

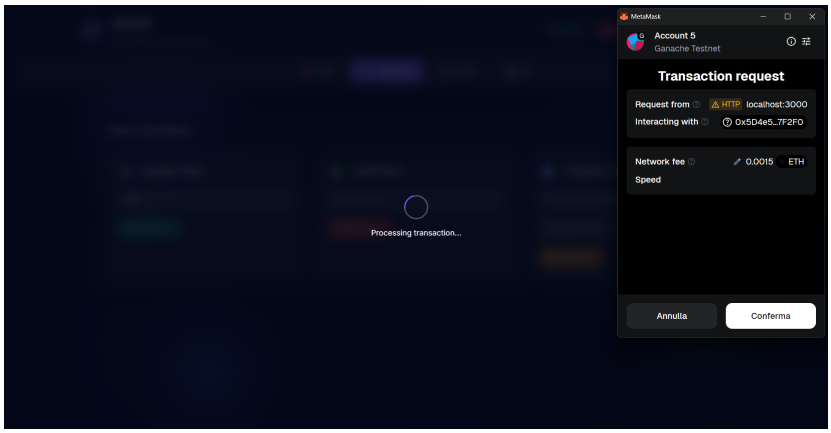


Figura 9: Pop-up di MetaMask per la conferma dell'acquisto di token. L'utente deve approvare la transazione prima che venga eseguita.

7 Conclusioni

Il lavoro sviluppato in questa tesi ha affrontato in modo concreto i problemi principali che caratterizzano la distribuzione delle royalties musicali. I punti critici individuati come la scarsa trasparenza nei flussi di pagamento, i lunghi tempi di attesa per la ricezione dei compensi, la disuguaglianza nella ripartizione tra i soggetti coinvolti e la mancanza di indipendenza per l'artista sono stati affrontati e superati attraverso la progettazione di un sistema basato su blockchain e smart contract, capace di ridurre drasticamente la complessità del processo e di rendere il modello economico più efficiente, trasparente e verificabile.

L'introduzione della **tokenizzazione delle royalties** rappresenta la prima risposta concreta a queste problematiche. Questo meccanismo consente di collegare in modo diretto e trasparente le performance economiche dell'artista con il valore dei token detenuti dagli utenti, eliminando la necessità di intermediari e riducendo i margini di errore o manipolazione. In questo scenario, la blockchain diventa il nuovo "registro contabile" dell'intero ecosistema musicale, in cui ogni flusso economico è registrato e verificabile da chiunque, senza possibilità di alterazione retroattiva.

Uno dei vantaggi più significativi introdotti da questa architettura riguarda i **tempi di pagamento delle royalties**. Nel sistema tradizionale, gli artisti ricevono i compensi con mesi di ritardo a causa delle procedure di verifica, dei passaggi tra distributori e delle rendicontazioni centralizzate. Con la soluzione proposta, l'intero processo diventa praticamente istantaneo: i dati relativi agli incassi vengono comunicati alla blockchain e lo smart contract aggiorna in modo automatico e trasparente il valore dei token. Questo azzera di fatto i tempi di attesa e offre all'artista un accesso immediato ai propri guadagni, con un flusso costante e verificabile in ogni momento. Un sistema simile non solo migliora la sostenibilità economica del musicista, ma contribuisce anche a stabilizzare il mercato, riducendo la dipendenza da cicli di pagamento lenti e poco trasparenti.

Un altro risultato fondamentale riguarda la **maggiore indipendenza economica dell'artista**. Nel modello tradizionale, le etichette discografiche e le piattaforme di distribuzione trattengono una percentuale significativa delle royalties, lasciando al musicista solo una parte ridotta dei ricavi totali. L'adozione di smart contract e di una logica decentralizzata consente invece di eliminare gran parte di questi intermediari, restituendo all'artista una percentuale nettamente superiore delle entrate generate. L'automatizzazione dei pagamenti e la registrazione diretta delle transazioni rendono superflua la mediazione di soggetti terzi, offrendo un modello più equo e sostenibile nel lungo periodo. Ciò produce anche un effetto culturale rilevante: l'artista non è più vincolato da contratti opachi o dipendente da entità centralizzate, ma diventa protagonista della propria economia creativa.

Dal punto di vista tecnico, i contratti sviluppati hanno dimostrato come gli **smart contract** possano essere sfruttati non solo per gestire la distribuzione economica, ma anche per rafforzare il legame tra artista e pubblico. `RoyaltyToken.sol` garantisce la gestione sicura e proporzionale dei valori economici associati ai token, mentre `SoulboundBadge.sol` introduce un sistema di badge non trasferibili per premiare la partecipazione e la fedeltà degli utenti. Questi elementi mostrano come la blockchain possa diventare una piattaforma per la costruzione di relazioni digitali verificabili e durature, non solo per lo scambio di valore, ma anche per la creazione di una vera comunità economica basata su fiducia e partecipazione.

L'uso di moduli standardizzati di OpenZeppelin, come `Ownable` e `ReentrancyGuard`, ha assicurato elevati livelli di sicurezza e affidabilità del codice, prevenendo vulnerabilità note e facilitando la manutenzione del sistema. Questa scelta dimostra come le architetture decentralizzate possano essere implementate con rigore tecnico e criteri di sicurezza conformi alle migliori pratiche del settore, rendendo la proposta non solo teoricamente valida, ma concretamente realizzabile.

Nel complesso, il progetto ha dimostrato che l'applicazione della tecnologia **blockchain** al settore musicale può non solo aumentare la fiducia e la trasparenza, ma anche trasformare radicalmente i tempi e le modalità di distribuzione del valore, rendendo l'artista più indipendente e la relazione con il pubblico più diretta. L'eliminazione dei ritardi nei pagamenti, la riduzione delle trattenute e la possibilità di controllo immediato sui propri introiti costituiscono i principali risultati di questa sperimentazione. Inoltre, la creazione di un sistema aperto e tracciabile genera nuove opportunità per il mondo musicale indipendente, che può finalmente accedere a un'infrastruttura economica equa e sostenibile, svincolata da logiche centralizzate.

7.1 Sviluppi futuri

Pur avendo raggiunto risultati significativi, il progetto apre la strada a ulteriori sviluppi di grande interesse. Una delle direzioni più promettenti riguarda l'introduzione di un meccanismo di **governance decentralizzata**, capace di coinvolgere attivamente la community nella gestione e nell'evoluzione del sistema. Nel contesto della blockchain, la *governance* rappresenta l'insieme dei processi attraverso cui una rete di utenti può proporre, discutere e approvare modifiche al protocollo o alle sue politiche operative. In questo progetto, essa potrebbe essere realizzata tramite un modello di voto basato sul possesso dei token: ogni detentore avrebbe la possibilità di partecipare alle decisioni relative a certi aspetti della carriera dell'artista, in proporzione alla propria partecipazione economica. Un sistema di questo tipo, ispirato al modello delle *Decentralized Autonomous Organizations* (DAO), renderebbe la piattaforma più partecipativa, favorendo la nascita di comunità coese, consapevoli e realmente coinvolte.

In prospettiva, la possibilità di integrare funzioni di **voto on-chain** rappresenterebbe un'evoluzione naturale del progetto, completando la visione di una piattaforma completamente decentralizzata in cui ogni attore, dall'artista ai fan, possa contribuire in modo verificabile alle scelte collettive. L'introduzione di tali strumenti di governance permetterebbe di regolare non solo le politiche di distribuzione delle royalties, ma anche l'allocazione dei fondi per la promozione, la produzione o l'organizzazione di eventi, creando un ecosistema in cui le decisioni economiche e creative vengono condivise.

Un ulteriore passo avanti potrebbe consistere nell'integrazione di **oracoli decentralizzati** in grado di collegare i dati off-chain relativi alle piattaforme di streaming (come Spotify) con il sistema on-chain. In questo modo, le informazioni sugli ascolti e sui guadagni dell'artista verrebbero validate automaticamente e trasmesse agli smart contract in maniera sicura e verificabile. L'utilizzo di oracoli affidabili, come Chain-

link, consentirebbe di garantire l'integrità dei dati e di ridurre ulteriormente il rischio di manipolazione, rafforzando la trasparenza complessiva dell'ecosistema.

In prospettiva, l'integrazione con strumenti di **intelligenza artificiale** potrebbe ampliare ulteriormente le potenzialità del modello. Algoritmi di analisi predittiva potrebbero, ad esempio, stimare in anticipo l'andamento delle royalties basandosi su trend di ascolto, engagement social e interazioni con i fan. Queste previsioni consentirebbero agli artisti e agli investitori di pianificare strategie più consapevoli, introducendo una dimensione di analisi avanzata che unisce dati off-chain e dinamiche di mercato on-chain.

Un ulteriore sviluppo futuro riguarda la creazione di un **marketplace** in cui gli utenti possano vendere e acquistare i token direttamente tra di loro. Attualmente, infatti, la compravendita dei token avviene esclusivamente tramite lo smart contract principale dell'artista. L'introduzione di questa nuova funzionalità permetterebbe di ampliare le possibilità di utilizzo del sistema, offrendo agli utenti uno spazio dedicato per effettuare transazioni in modo diretto, mantenendo comunque la stessa trasparenza e sicurezza garantite dalla blockchain. In questo modo, il progetto potrebbe evolvere verso un modello più completo e flessibile, in cui le interazioni tra gli utenti assumono un ruolo centrale nello scambio dei token e nella crescita dell'ecosistema.

Il futuro di questo modello risiede nella sua capacità di evolvere verso forme sempre più decentralizzate, collaborative e intelligenti, in cui la tecnologia non solo automatizza i processi, ma amplifica la libertà creativa ed economica dell'artista.

Riferimenti bibliografici

[1] Reuters (2024), *Spotify sued over millions allegedly unpaid in music royalties*, https://www.reuters.com/legal/litigation/spotify-sued-over-millions-allegedly-unpaid-music-royalties-2024-05-17/.

[2] Recording Academy (2024), *Spotify under fire for failing to properly license music — again*, https://www.recordingacademy.com/advocacy/news/spotify-under-fire-failing-properly-license-music--again.

[3] The Times (2024), *Spotify's shadowy ghost artists unmasked*, https://www.thetimes.co.uk/article/spotifys-shadowy-ghost-artists-unmasked-hstc9wcdb.

[4] The Pudding (2022), *The mechanics of streaming royalties*, https://pudding.cool/2022/06/streaming/.

[5] Wikipedia (2025), *Criticism of Spotify*, https://en.wikipedia.org/wiki/Criticism_of_Spotify.

[6] Umbrex (2023), *How the music industry works*, https://umbrex.com/resources/how-industries-work/media-entertainment/how-the-music-industry-works.

[7] IFPI (2025), *Global Music Report 2025*, https://www.ifpi.org/wp-content/uploads/2024/03/GMR2025_SOTI.pdf.

[8] Spotify, *Understanding Spotify royalties*, https://support.spotify.com/it/artists/article/understanding-spotify-royalties/.